République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
ECOLE NATIONALE POLYTECHNIQUE

DÉPARTEMENT D'ÉLECTRONIQUE

Mémoire de projet de fin d'études

pour l'obtention du diplôme d'Ingénieur d'Etat en Electronique

Thème :

Synthèse et implémentation d'un NOC Hermes sous VIVADO à partir du simulateur ATLAS

Amine OUDJEHANE

Sous la direction de : M. Rabah SADOUN

Présenté et soutenu publiquement le (14/06/2016)

Composition du Jury :

Président	M. Mohamed TRABELSI.	PR	ENP
Promoteur	M. Rabah SADOUN.	PR	ENP
Examinateur	M. Mohamed TAGHI.	PR	ENP

Promotion : Juin 2016

DEDICACES

Je dédie ce travail :

Aux personnes m'étant les plus chers au monde, mes parents, à qui je serai toujours redevable,

A mes frères qui ont toujours été un soutien,

A tous mes amis qui m'ont toujours soutenu, et particulièrement mlle Souad Boukli Hacene qui a su m'encourager quand j'en avait besoin.

C'est grâce à toutes ces personnes que j'ai pu évoluer et grandir et je leur serais toujours redevable.

REMERCIEMENTS

Aucun travail ne s’accomplit dans la solitude

J’ai tout au long de mon projet de fin d’études eu le soutien de plusieurs personnes et je tiens à leur exprimer ma gratitude.

Mes vifs remerciements s’adressent en premier lieu à Monsieur SADOUN pour ses précieux conseils, son soutien, sa disponibilité ainsi que ses qualités relationnelles indéniables.

Je tiens aussi à remercier tous les enseignants du Département Electronique à l’Ecole Nationale Polytechnique auxquels je dois ma formation d’ingénieur.

Enfin, je remercie les membres du jury qui me font l’honneur d’évaluer mon travail.

- Monsieur TRABELSI (Président du jury).
- Monsieur TAGHI (Examinateur).

ملخص

الشبكة على رقاقة هو مفهوم جديد من الترابط في النظم ذات رقاقة واحدة. هذه التشكيلة تسهل التكامل بين مكونات معقدة. ومع ذلك، مثل أي تكنولوجيا جديدة، فإنها تتطلب جهودا في مجال البحث، من حيث تسريع وتبسيط مراحل التصميم. جزء الاتصالات داخل شبكة الرقاقة لا يقل أهمية عن مرحلة التنفيذ. لذلك، من المهم فهم هذا النمط الجديد. الهدف الرئيسي من مشروعنا هو تبسيط عقدة المفهوم، وبالتحديد نظام هيرميس ومكوناته المختلفة وذلك بتحليل الرسائل المتبادلة بالداخل وبعد ذلك تحقيق التنفيذ على بطاقة زينك 7000 لزادبورد من التعليمات البرمجية ش د ل التي تم إنشاؤها بواسطة أطلس

الكلمات الدالة

الشبكة على الرقاقة، هيرمس، أطلس، فيفادو، عقدة هيرمس

Abstract

The network on chip is a new concept of interconnections in a single-chip systems. This architecture facilitates the integration of complex components. However, like any new technology, it requires research efforts in terms of acceleration and simplification of design phases. The communication architecture within the network on chip is as important as the implementation phase. Therefore, it is of a major interest to better understand this operation. The main goal of our project is to simplify the concept of the node, specifically the Hermes node and its different components using the exchanged signals followed by an implementation of the HDL codes generated by ATLAS on a ZedBoard card.

Key Words: Network on Chip, Hermes, Atlas, Vivado, Network's node.

Résumé

Le réseau sur puce est un nouveau concept d'interconnexions dans les systèmes mono-puce. Cette architecture facilite l'intégration des composants complexes. Cependant, comme toute nouvelle technologie, elle requiert des efforts en recherche, en ce qui concerne l'accélération et la simplification des phases de conception. La partie communication à l'intérieur du réseau sur puce est toute aussi importante que la phase d'implémentation. C'est pourquoi, il est très intéressant de comprendre au mieux son fonctionnement. Le but principal de notre projet est de simplifier le concept de nœud, et plus précisément celui d'un nœud Hermes, ainsi que ses différents composants aux moyens de signaux échangés pour ensuite aboutir à une implémentation sur carte ZedBoard à partir de codes HDL générés par ATLAS.

Mots clés : Réseau sur puce, Hermes, Atlas, Vivado, nœud du réseau.

Table des matières

Liste des figures
Liste des abréviations

INTRODUCTION GENERALE **10**

I. CHAPITRE I : ETAT DE L'ART **12**

I.1 Paradigme Shift **12**
I.2 Réseau sur puce (NOC) **15**
I.3 Approche OSI **18**
I.3.1 Couche Application 19
I.3.2 Couche Présentation 19
I.3.3 Couche Session 20
I.3.4 Couche Transport 20
I.3.5 Couche Réseau 20
I.3.6 Couche Liaison 21
I.3.7 Couche Physique 21
I.4 Avantages et Défis des NOC **22**
I.5 La recherche sur les NOC **25**

II. CHAPITRE II : TOPOLOGIES ET MODELISATION **27**

II.1 Topologies existantes **27**
II.1.1 Topologie régulière vs irrégulière 27
II.1.2 Topologie directe vs indirecte 29
II.1.3 Topologie 2D 29
II.1.4 Topologie 3D 30
II.2 Modélisation du trafic **31**
II.2.1 Modèles disponibles pour le trafic 31
II.2.1.1 *Trafic synthétique* 32
II.2.1.2 *Trafic réaliste* *34*
II.3 Synthèse de topologie **35**

III. CHAPITRE III : ARCHITECTURES DE COMMUNICATION **37**

III.1 Techniques de commutation **37**
III.1.1 Commutation de circuit 38
III.1.2 Commutation de paquets 38
III.1.3 Techniques de commutation hybrides 39
III.2 Routage de paquets **39**
III.2.1 Deadlocks 41
III.2.1.1 *Deadlock recovery* *41*
III.2.1.2 *Deadlock Avoidance* *42*

III.2.2 Livelocks 42
III.2.3 Algorithmes de routage pour les architectures régulières 42
III.2.4 Algorithmes de routage pour les architectures irrégulières 44
III.3 Contrôle de flux et qualité de service 46
III.3.1 Contrôle de flux 46
III.3.2 Qualité de service 49
III.3.2.1 *Best effort* *49*
III.3.2.2 *Service garantis* 49

IV. CHAPITRE IV : NOCS ET SIMULATEURS EXISTANTS 50

IV.1 Réseaux NoCs existants 50
IV.1.1 ARTERIS 50
IV.1.2 STNOC 51
IV.1.3 PROTEO 52
IV.1.4 HERMES 52
IV.1.5 MANGO 53
IV.2 Simulateurs existants 53
IV.2.1 WORMSIM 53
IV.2.2 TOPAZ 54
IV.2.3 NOXIM 55
IV.2.4 NIRGAM 55
IV.2.5 SICOSYS 55

IV.3 Comparaison des simulateurs 56
IV.4 Conclusion 57

V. CHAPITRE : MISE EN ŒUVRE DU NOC HERMES 58

V.1 Introduction 58
V.2 HERMES 58
V.2.1 Topologie 58
V.2.2 Modèle du nœud HERMES 59
V.2.3 Constituants 60
V.2.4 Influence de la profondeur des buffers 60
V.2.5 Protocoles d'HERMES 61
V.2.6 Ordonnancement 63
V.2.7 Algorithmes de routage 63
V.2.8 Fonctionnement du nœud 64
V.3 ATLAS 66
V.3.1 Structure 66
V.3.2 Fichiers produits 68
V.3.3 Evaluation de performances 68
V.4 Choix de l'environnement de synthèse 70
V.4.1 QUARTUS 70
V.4.2 XILINX ISE 71

V.4.3 VIVADO HLS 71
V.5 Synthèse et Implémentation sur carte 73
V.6 Conclusion 77

CONCLUSION GENERALE ET PERSPECTIVES 78

BIBLIOGRAPHIE 80

Liste des figures

Figure I-1 : Noc Based MPSoC…… 16
Figure I-2 : Modèle OSI d'un NOC…….… 19
Figure I-3 : Classes de recherches sur les NoCs……... 25
Figure II-1 : Exemple de topologie (a) régulière (b) irrégulière……. 28
Figure II-2 : Maillage 2D 3x3…… 30
Figure II-3 : Exemple d'empilage 3D maillé…….. 31
Figure II-4 : Etapes de la sélection de topologie……. 35
Figure III-1 : Illustration des algorithmes de routage……... 44
Figure III-2 : protocole de contrôle de flux ACK/NACK (a) signaux (b) chronogramme…..… 47
Figure III-3 : Protocole de contrôle de flux STALL / GO (a) signaux (b) chronogramme…….. 48
Figure III-4 : Protocole de contrôle de flux à base de crédit (a) signaux (b) chronogramme.…. 48
Figure III-5 : comparaison des protocoles de contrôle de flux……. 49
Figure IV-1 : Exemple de topologie Spidergon d'un STNOC……. 51
Figure IV-2 : Comparatif des différents NoCs……. 56
Figure V-1 : Exemple d'un NoC Hermes a topologie maillé 2D (3x3)……... 58
Figure V-2 : Modèle du nœud de HERMES…… 59
Figure V-3 : Illustration de l'algorithme X_Y……. 64
Figure V-4 : Diagramme partiel de ports (Local et non Local) du nœud HERMES……..... 65
Figure V-5 : Simulation de la connexion entre le deux ports (Local et non Local)…….. 66
Figure V-6 : Fenêtre principale de l'interface graphique…….. 67
Figure V-7 : Design flow de l'outils ATLAS……... 67
Figure V-8 : Graphe de consommation de puissance (a) distribution moyenne de puissance (b) consommation moyenne /nœud…….. 69
Figure V-9 : Les trois phases du flot de conception sous Quartus…… 70
Figure V-10 : Design flow de conception sous VIVADO…….. 72
Figure V-11 : carte ZedBoard ZynQ7Z020……..... 72
Figure V-12 : Interface graphique de VIVADO…….. 73
Figure V-13 : RTL Hiérarchie du NoC Hermes 2x2……... 74
Figure V-14 : Vue détaillé d'un nœud Hermes et de ses composants……. 74
Figure V-15 : Schématique RTL de notre NOC HERMES 2x2……. 75
Figure V-16 : design d'implémentation d'un réseau Hermes 2x2……... 75
Figure V-17 : illustration d'implémentation des codes HDL sur la carte ZynQ7000 (a) avant implémentation (b) après implémentation……... 76

Liste des abréviations

3-D	Trois dimensions
AXI	Advanced eXtensible Interface
BE	Best Effort ↔ Trafic au mieux
CAO	Conception Assistée par Ordinateur
CMOS	Complementary Metal Oxide Semiconductor
CPU	Central Processing Unit
DR	Distibuted Routing ↔ Routage distribué
DRAM	Dynamic Random Access Memory
DSP	Digital Signal Processor
EDA	Electronic Design Automation
EOM	End Of Message
FIFO	First In First Out
FLIT	FLow control unIT ↔ Contrôle du flux
FPGA	Field Programmable Gate Array
GALS	Globally Asynchronous Locally Synchronous
GS	Guaranteed Services ↔ Services Garantis
HDL	Hardware Description Language
ISO	International Organization for Standardization
LAN	Local Area Network
MpSoC	Multi-Processors System on Chip
NoC	Network-On-chip
NI	Network Interface ↔ Interface réseau
OCP	Open Core Protocol
OSI	Open Systems Interconnection
PE	Processing Element
PHIT	PHysical unIT
QoS	Quality Of Service ↔ Qualité de service
RTL	Register Transfer Level
SAF	Store And Forward
SDM	Spatial Division Multiplexing
SGML	Stream Gate Manager
SoC	System On a Chip
SR	Source Routing ↔ Routage source
TCP/IP	Transmission Control Protocol
TDM	Time Division Multiplexing
VC	Virtual Channel
VCT	Virtual Cut Through
VLSI	Very Large Scale Integration ↔ Intégration à très grande échelle
WS	Wormhole Switching

INTRODUCTION GENERALE

Dans le domaine des réseaux, on rencontre de plus en plus de fonctionnalités introduites dans un même système ce qui conduit à la mise en communication d'un grand nombre de blocs fonctionnels tout en sachant que les liens de communication n'évoluent pas à la même vitesse et engendrent donc un goulot d'étranglement.

Les solutions de communication actuelles telles que le bus partagé trouvent leurs limites en termes de bande passante et d'extension à mesure que le nombre d'éléments communicants augmente. Cette structure est la plus utilisé de nos jours mais ne semble pas s'adapter aux applications futures. C'est dans ce contexte que le concept des réseaux sur puce (NOC) a vu le jour.

Simplifier le concept de réseau sur puce aux moyens de signaux échangés dans ce réseau devient très important. D'où l'intérêt de ce présent travail. Pour ce faire, le document est organisé comme suit :

Le premier chapitre établit un état de l'art du concept de réseau sur puce et des motivations qui ont conduit à l'adoption de ce dernier avec une vue d'ensemble sur les classes de recherches qui sont menées.

Le second chapitre est consacré à la présentation des différentes topologies des réseaux sur puces et leurs modélisations.

Le troisième chapitre présente l'architecture de communication d'un réseau sur puce notamment en matière d'algorithme de routage ou encore, les protocoles de contrôle de flux avec une introduction de la qualité de service.

Le quatrième chapitre regroupe les différents réseaux sur puce, qu'ils soient académiques ou commerciaux ainsi que les simulateurs présents pour conclure avec une comparaison générale.

Enfin, le cinquième et dernier chapitre met en œuvre le réseau Hermes couplé avec le simulateur ATLAS pour mieux cerner le concept de nœud et aborde les points suivants :

- Présentation détaillée du réseau HERMES.
- Introduction au simulateur ATLAS et ses outils avec quelques exemples.

- Explication détaillée du fonctionnement du nœud et des signaux échangés à travers les différents composants de ce dernier.
- Choix du EDA avec la carte FPGA associée.
- Synthèse et implémentation des codes HDL.

CHAPITRE I : ETAT DE L'ART

Le premier chapitre présente les concepts fondamentaux des réseaux sur puce NoCs commençant par la motivation qui a provoqué la transition des bus traditionnels aux architectures à base de NoCs, ainsi que les similarités et différences entre NoCs et les réseaux informatiques.

Les composants des NoCs sont introduits et leur fonctionnalité est démontrée en termes de structure de la couche OSI.

Une grande partie de ce chapitre est consacrée à l'explication des avantages et des défis relatifs à l'adoption des NoCs comme infrastructure de communication des SoCs.

Finalement, le classement des secteurs de recherche actuels est établi.

I.1 Paradigme

La force motrice derrière la technologie des circuits intégrés a été la loi de Moore depuis près de cinq décennies [1]. Bien que ceci aboutisse à ce qu'un dédoublement se produise tous les 3 ans au cours des prochaines années pour des tailles fixes de puces [2], la tendance exponentielle est toujours en vigueur. Malgré le fait que l'évolution soit continue, la mise au point est au niveau du système, ou d'un système champ d'application. Quand une technologie mûrit pour une mise en œuvre donnée, elle conduit à un changement de paradigme. Dans les années 1990, cette tendance a permis l'introduction de Systèmes on-Chip (SoC), ainsi que l'intégration de nombreux composants tels que les microprocesseurs personnalisés et analogiques dans une seule puce. L'une des conséquences de la loi de Moore est l'augmentation de la fréquence de fonctionnement. Cependant, au cours des dernières années les circuits intégrés n'ont pas réussi à suivre les fréquences de fonctionnement prévues par Moore en raison de la consommation d'énergie et de la densité thermique. Depuis, la densité des transistors a continué d'augmenter, ce qui a poussé les concepteurs à travailler autour de ce problème en introduisant des architectures multi-cœurs afin d'augmenter les performances sans pour autant accroitre la fréquence de fonctionnement [3]. Cette tendance se poursuit aujourd'hui avec l'introduction d'architectures hétérogènes multi-cœurs [4].

Historiquement, le calcul fut coûteux à l'inverse de la communication. Cependant, la mise à l'échelle des technologies de puces a changé cela. Plus précisément, durant les dernières années, le calcul est de plus en plus économique, tandis que la communication rencontre des limitations

fondamentales physiques telles que le temps de vol des signaux électriques, la consommation d'énergie dans la conduite de longs fils / câbles, etc. En comparaison avec les systèmes standards, la communication sur puce est nettement moins chère et permet d'avoir de la place pour beaucoup de fils sur une puce. Ainsi, le passage à des systèmes mono-puces a assoupli les problèmes de communication du système. Toutefois, les fils sur puce ne sont pas sur la même échelle que les transistors, et l'écart du coût entre le calcul et la communication se creuse. Pendant ce temps, les différences entre bilan On-chipwires et hors-chipwires montrent clairement que les systèmes on-Chip sont plus optimaux. Les technologies de silicium font alors face à d'autres problèmes. La synchronisation des puces futures avec une source d'horloge unique et négligeable sera extrêmement difficile, voire impossible. Le paradigme de synchronisation le plus probable pour l'avenir des puces, globalement asynchrone et localement synchrone, implique l'utilisation de plusieurs horloges. En l'absence d'une seule référence de synchronisation, Les puces des SoCs deviennent distribuées sur un substrat de silicium monocristallin. Le contrôle global du trafic de l'information a peu de chances de réussir parce que le système a besoin de garder une trace de chaque composant. Ainsi, les composants vont initier des transferts de données de manière autonome, en fonction de leur besoin et le schéma global de communication sera entièrement distribué, avec peu ou sans coordination. Etant donné que les fils mondiaux couvrent de multiples domaines d'horloge, des échecs de synchronisation dans la communication entre domaines différents seront des événements rares mais inévitables [5]. En outre, l'énergie et le dispositif de fiabilité vont imposer de faibles niveaux de swing de la logique et de la tension d'alimentation, probablement moins d'un volt [2]. Le bruit électrique en raison de la diaphonie, interférences électromagnétiques, et l'injection de charge induite par le rayonnement vont probablement produire des données erronées, aussi appelés troubles. Ainsi, la transmission des valeurs numériques sur des fils (wires) va être intrinsèquement peu fiable et non-déterministe. Les autres causes de non-déterminisme comprennent la conception de composants avec un haut niveau d'abstraction et de granularité grossière et un contrôle de communication distribué.

L'intégration de nombreux blocs de traitement et de mémoire sur une seule puce introduit à son tour une surcharge de communication que les architectures à base de bus traditionnels ne peuvent traiter pour un certain nombre de raisons [6]. Plus précisément, l'infrastructure d'interconnexion a un impact significatif sur les coûts du SoC du fait que, parmi d'autres, elle influe sur quatre facteurs clés de conception Soc : la taille, la consommation d'énergie, temps de conception et de performance.

• La taille peut augmenter si l'acheminement des fils d'interconnexion classiques et les exigences de la gate area explosent en raison de l'augmentation du nombre de blocs IP dans un SoC.

• La consommation électrique peut se multiplier si l'interconnexion d'un SoC ne pas être configurée facilement pour les systèmes de gestion de l'alimentation avancées telles que la fréquence dynamique et la mise à l'échelle de tension.

• Le temps de conception du projet peut se prolonger si l'interconnexion du SoC devient difficile à configurer et à vérifier. Cela peut ralentir le modèle down Stream si une conception de base est utilisée pour la conception de SoC dérivés.

• La performance peut souffrir si une approche d'interconnexion ne peut pas s'adapter aux exigences de l'évolution sur le SoC. (La qualité de service (QoS), la bande passante, la latence, la sécurité, et la fréquence d'horloge).

Le bus partagé est une interface simple car il est construit sur un concept bien connu et il est facile à modéliser. Cependant, il introduit des inconvénients en particulier dans un système fortement interconnectés (multi-cœurs). Cela se produit principalement parce que les bus évoluent mal pour un grand nombre de composants, tandis que les longs fils et le comportement du système deviennent imprévisibles du point de vue de la composante. Plus précisément, le nombre de conducteurs intégrés à un système augmente, et la consommation d'énergie par événement de communication se développe en raison de plusieurs unités principales attachées à une charge capacitive élevée. En dehors de la dissipation de puissance, le problème de l'arbitrage en particulier pour les multi-bus master est également crucial.

Sur la base de cette analyse, nous pouvons conclure que les NOCS visent à surmonter un certain nombre de limitations trouvées dans les autobus. Cependant, même cette interconnexion paradigme ne peut être pensée finalement. Evolutive, elle représente une solution intermédiaire. Plus précisément, les liaisons spécialisées point à point sont optimales en termes de disponibilité de la bande passante, latence, et la consommation d'énergie, car ils sont généralement conçus avec une approche full-custom (Solution d'interconnexion spécifique à l'application). En outre, il est beaucoup plus simple de modéliser, concevoir, et de vérifier le fonctionnement correct des liaisons point-à-point, par rapport au réseau correspondant. D'autre part, l'accroissement du nombre de cœur, engendre une augmentation exponentielle du nombre de liens. Ainsi une zone et éventuellement, un problème se pose dans le calcul d'itinéraire.

Du point de vue de la conception d'effort, nous pouvons affirmer que dans les systèmes de petite taille (Consistant en quelques cœurs) ceci pourrait être efficace pour assurer la communication souhaitée à travers une structure de communication ad hoc. Mais, comme la taille du système augmente et le temps de cycle de conception diminue, il y a une demande croissante pour proposer des solutions plus généralisées. Pour plus de souplesse et d'évolutivité maximale, il est nécessaire d'opter pour un mouvement vers une structure de communication globale partagée et segmentée. Les longs fils segmentés dans cette approche évitent la dégradation du signal, et les bus sont mis en œuvre en tant que structures multiplexées afin de réduire la puissance et d'accroître la réactivité. Cette notion se traduit par un réseau d'acheminement de données comprenant des liens de communication et de nœuds de routage qui sont mis en œuvre sur la puce. Contrairement aux méthodes de communication SoC traditionnels décrites précédemment, un tel support de communication distribuée évolue parfaitement avec la taille de la puce et sa complexité. D'autres avantages comprennent une augmentation de la performance agrégée en exploitant les opérations en parallèle. En outre, les technologies d'intégration de la récente introduction de la 3-D [7] donnent une nouvelle dimension au problème d'interconnexion à base de bus.

Une solution connue consiste à adopter un paradigme bien établi, celui de réseaux informatiques. Ainsi, les réseaux sur puce (NoCs) ont été introduits. La technologie NoC est souvent appelé "une solution front-end à un problème d'arrière-plan". Beaucoup de SoCs d'aujourd'hui sont trop complexes et utilisent une hiérarchie traditionnelle à bus ou une approche d'interconnexion transversale. Le trafic du village d'hier s'est transformé en des autoroutes congestionnées d'aujourd'hui.

I.2 Réseau sur puce (NOC)

La technologie NoC est une approche relativement nouvelle qui permet non seulement des interconnexions efficaces mais aussi des processus de conception et de vérification plus optimaux pour les SoCs modernes. NoC est une approche de signalisation qui correspond aux besoins du signal à travers divers protocoles de communication d'une manière qui réduit la complexité de l'interconnexion des puces. Les signaux de bande passante lente ou faible peuvent être multiplexés sur une seule ligne avec d'autres signaux, tandis que seule la vitesse la plus élevée des signaux de forte bande passante communique directement sur des chemins parallèles volumineux. Un MPSoC typique à base de NoC est représentée sur la (figure I.1).

Il est composé d'un certain nombre de composants, appelés nœuds, y compris des éléments de traitement (PE), tels que les processeurs, blocs IPs, DSP et des éléments de stockage (blocs

de mémoire embarquée). Les PE sont fixés aux adaptateurs de réseau via les interfaces réseau (NI), tandis que leur fonctionnalité est de découpler le calcul (noyaux) et la communication (le réseau). La communication par le NoCs est effectuée en permettant au PE d'envoyer et de recevoir des paquets à travers la structure de réseau composée de commutateurs / routeurs connectés ensemble par des liens physiques, ou des canaux. En règle générale, chaque liaison est une paire opposée, unidirectionnelle, point à point, y. Sachant que les propriétés souhaitées pour les NoCs sont généralement orientées vers l'application, il existe différentes approches pour la conception de ces liaisons (par exemple, elles peuvent consister en une ou plusieurs logique ou canaux physiques). Souvent un interrupteur avec son PE hôte, ou de la mémoire, est appelé « tuiles ».

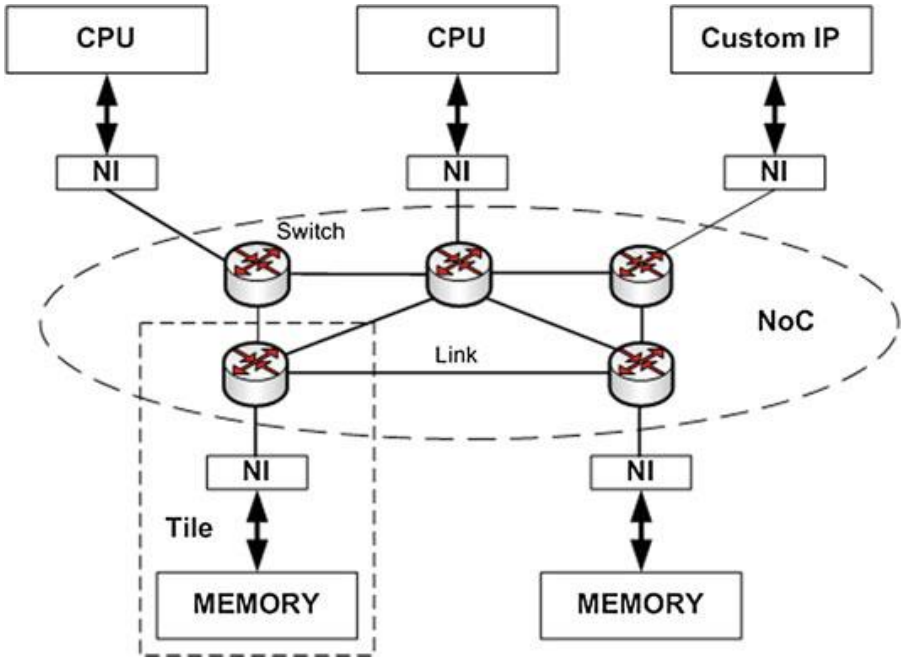


Figure I-1: Noc Based MPSoC

Un paramètre critique aux NoCs affecte la décision sur la façon dont les paquets doivent traverser le réseau. Etant donné que le NoC est un réseau distribué, ces décisions sont prises en fonction de la stratégie de routage utilisée habituellement à chaque routeur. Ce paramètre

affecte fortement les performances de la plate-forme à base de NoCs, jusqu'à maintenant, différents algorithmes de routage ont été proposés dans la littérature.

Pour de nombreux designers, l'utilisation d'un NoC se traduit par une plus grande flexibilité pour optimiser leur SoC. Avant d'utiliser les NoCs, ces concepteurs ont eu le temps de changer seulement leur interconnexion environ 5-6 × par projet. Aujourd'hui, ces mêmes concepteurs sont en mesure d'utiliser leur interconnexion 9-10 × par projet. Cela leur donne une plus grande flexibilité pour accueillir des changements à venir relatifs à l'ingénierie et aux clients, tout en offrant plus de temps pour la mise à niveau du système et optimisation des performances [8].

Le paradigme NoC, n'a bien évidemment pas été établi en une nuit. Dans un premier temps, ce n'était pas le choix le plus évident et offrant la meilleure architecture pour soulager le problème d'écart de productivité. *Jantch* et *Tenhunen* [9] ont introduit deux propriétés requises d'un processus de conception afin de devenir un véritable changement de paradigme :

- *Composabilité arbitraire :* un système de composants et combinateurs qui permettent d'être connectés et intégrés au plus grand assemblage de composants est arbitrairement composable si un assemblage de composants donné A peut être prolongé d'une composante en utilisant les combinateurs existants sans changer le comportement relatif de A.
- *Effort linéaire :* Un processus de conception qui construit un système à partir d'un ensemble des composants et de combinateurs possède cette propriété si l'effort de conception pour intégrer un ensemble de données de n assemblages au moyen des combinateurs dépend de n, mais pas de la taille des assemblages.

Même si les propriétés ci-dessus ne sont pas rigides mathématiquement, elles représentent des critères importants qui doivent être remplis par un processus de conception pour être considéré comme un nouveau paradigme. Par conséquent, la conception des NoCs doit être effectuée de manière appropriée afin de les satisfaire, sinon les garantir dans tous les cas.

Certaines des caractéristiques qui permettent aux plateformes NoCs à adhérer à la ci-dessus propriétés fondamentales sont résumées comme suit :

- Permet la réutilisation des conceptions.
- Sépare entre communication et calcul.
- Evite le contrôle centralisé mondial pour la communication.
- Permet un nombre arbitraire de terminaux.
- Permet l'ajout de futurs liens si la taille du système augmente (évolutivité).

• Permet la non utilisation des longs fils mondiaux couvrant toute la puce.

• Facilite la personnalisation (par exemple : la largeur de liaison, la taille du buffer, la topologie, etc.).

• Permet de multiples tensions et de fréquences domaines.

• Délivre des données en ordre soit naturellement, soit via le protocole en couches.

• Fournit des garanties variables pour les transferts.

• Offre un soutien pour les essais du système.

• S'adapte facilement aux plates-formes d'intégration 3-D.

Ensuite, nous décrivons plus en détail certains des défis les plus importants pour les architectures NoCs [10]. Bien sûr, il faut mentionner qu'à chaque conception de nouvelle architecture NoC, ces caractéristiques doivent être réglées de manière appropriée afin de mieux répondre aux spécifications du produit.

I.3 Approche OSI

Les NOCs sont des réseaux de communication à commutation de paquets provenant du domaine parallèle de l'informatique. En exploitant les enseignements tirés par la communauté des télécommunications, la communication globale sur la puce est décomposée en couches similaires au vu de ISO / OSI modèle de référence dans le réseau informatique.

Le modèle ISO / OSI ne définit pas exactement comment le système doit être construit, mais constitue plutôt des actes formant un guide conceptuel. Plus précisément, en permettant à chaque couche de cacher les couches au-dessus et à seulement communiquer avec les couches adjacentes, il est possible d'activer différents services en fournissant au programmeur une abstraction du cadre de la communication [11]. Une série de ces couches est connue comme une « pile protocole » et peuvent être conçus dans le matériel ou le logiciel.

Dans le modèle OSI, la fonctionnalité de réseau est divisée en sept couches différentes, considérant que les objectifs de cette classification peuvent être résumés comme suit :

• Les couches devraient échanger un minimum d'informations à travers les interfaces.

• il devrait y avoir suffisamment de couches de sorte que d'autres fonctions ne soient pas mises sur la même couche.

• chaque couche doit avoir des limites avec sa couche supérieure et inférieure seulement.

Le modèle OSI / ISO d'un NoC est présenté dans la figure I-2.

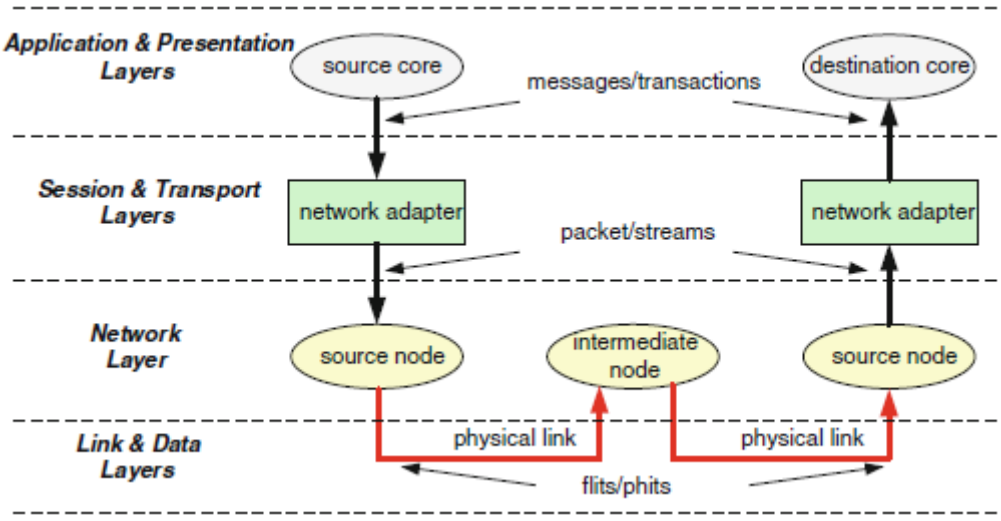


Figure I-2 : Modèle OSI d'un NOC

I.3.1 Couche application

Cette couche est celle où se trouvent les ressources. Notez qu'une ressource peut exécuter de nombreuses tâches différentes (par exemple, un microprocesseur peut exécuter plusieurs processus simultanément).

I.3.2 Couche présentation

Comme les ressources peuvent avoir différentes représentations pour les nombres, il doit y avoir conversion entre eux. Cette couche effectue la conversion entre les différentes représentations utilisées à partir des ressources matérielles. Des exemples typiques sont les représentations avec une résolution différente (des entiers dans big-endian et little-endian format). Un autre exemple pourrait être le cas d'un microprocesseur utilisant un processeur de signal pour accélérer les calculs.

Dans un tel cas, si le microprocesseur utilise une représentation à virgule flottante, tandis que le processeur de signal utilise un format de point fixe, la conversion entre ces formats est absolument nécessaire. Une telle conversion est en fait réalisée à l'intérieur de la couche de présentation.

I.3.3 Couche session

La couche session sera responsable de l'établissement de connexions entre les ressources. Deux protocoles différents sont utilisés pour ce champ d'application :

(i) Un protocole orienté connexion, qui est plus économe en énergie dans le trafic lourd en raison de retransmissions.

(ii) Un protocole sans connexion, pour soutenir la livraison de données désorientées.

I.3.4 Couche Transport

Cette couche intègre tous les mécanismes nécessaires pour contrôler et vérifier qu'aucun colis ne soit perdu dans les couches inférieures. La couche transport traite la segmentation du message en paquets, ainsi que leur réassemblage. Une autre tâche essentielle qui est traitée à l'intérieur de la couche de transport affecte le contrôle de flux, ce qui affecte fortement la performance des NoC. Spécifiquement, la congestion du réseau augmente le coût par bit transmis en raison de la résolution, tandis que la quantité des données qui entrent dans le réseau, peut être réglée, au prix de débit.

I.3.5 Couche Réseau

La couche réseau gère les questions liées à la topologie et le routage des chemins. Plus précisément, les deux architectures hiérarchiques et hétérogènes peuvent être soutenues par cette couche afin de mieux répondre aux besoins de communication des plates-formes modernes. Par exemple, une solution optimale implique généralement le regroupement des nœuds avec des exigences de haute bande passante et de leur connexion à travers des chaînes dédiées de longueur courtes. Cette sélection permet un gain d'énergie assez conséquent, au lieu d'utiliser le reste du réseau, sachant que la communication à travers l'inter cluster est de loin plus efficace par rapport aux liens inter cluster correspondants. En plus de cela, la couche réseau est également responsable des algorithmes d'acheminement employés. Les deux techniques les plus répandues pour les NoCs sont la commutation de circuit et la commutation de paquets. Les avantages et les inconvénients de l'emploi de ces algorithmes se résument comme suit :

• *La commutation de circuits :*

- Les frais généraux de contrôle de réseau encourent une seule fois.
- Meilleure dans le cas de la communication persistante.

• *La commutation de paquets* :

- Les frais généraux de contrôle de réseau sont distribués.
- Plus d'énergie efficace pour les communications irrégulières.

I.3.6 Couche Liaison

La couche de liaison de données assure un transfert fiable malgré le manque de fiabilité physique. A cet effet, elle traite les tâches de détection et de correction des erreurs générées au niveau de la couche physique. En outre, cette couche est également responsable des accès au support de commande pour le partage d'une ressource de canal commun, avec un accès basé sur la contention. Une approche typique pour corriger une erreur est par la retransmission de données dans le cas d'erreur. Cependant, cette approche peut être coûteuse en termes d'énergie et de performances. Bien entendu, le choix optimal pour ce paramètre est tiré après la prise en compte également des contraintes imposées par le système cible, ainsi que les caractéristiques des canaux physiques.

I.3.7 Couche Physique

La couche physique se réfère à tout ce qui concerne les caractéristiques électriques des fils, les circuits et les techniques pour l'acheminement de l'information (pilotes, répéteurs, mise en …etc.). Différentes techniques peuvent être utilisées lors de la conception de la couche physique pour optimiser le rendement et réduire la puissance / dissipation d'énergie. Parmi d'autres, sont l'utilisation de la signalisation low-swing à l'émetteur par la réduction de Vdd. L'utilisation de récepteurs différentiels, ainsi que l'emploi de techniques qui fournissent moins de données fiables (si cela est faisable par rapport aux spécifications du système cible).

En plus de cela, la conception de la couche physique comprend des tâches qui affectent la signalisation au niveau du récepteur pseudo-différentiel (par exemple, le partage de signal de référence, moins de signal transitions, et la marge de bruit réduit). Enfin, comme les horloges ne sont pas vraiment économes en énergie et que la synchronisation globale de l'architecture d'interconnexion n'est pas optimale, l'utilisation d'unités globalement asynchrones et localement synchrones (GALS) pourrait être une solution viable.

Bien qu'une telle structure hiérarchique dans la conception de réseaux est analogue à celle constatée dans des réseaux LAN, l'utilisation d'algorithmes et / ou des topologies existantes ne convient pas pour la réalisation de la communication sur puce en raison des nombreuses différences fondamentales.

- *L'Insertion dynamique et la suppression de nœuds* qui est essentiel dans les réseaux informatiques. Aussi, le nombre de nœuds est une donnée connue, la seule considération spéciale requise est quand un nœud ou un lien tombe en panne.
- *Coût des tampons et des liens* : En réseau informatique, alors que les tampons sont peu couteux, les liens eux le sont. L'inverse est vrai pour l'environnement NoC, où il est courant d'avoir 128 bits liens larges et routeurs, même sans tampon.
- *Fiabilité et non-déterminisme* : La conception d'une architecture NoC fiable est particulièrement difficile car l'infrastructure d'interconnexion est potentiellement affectée par l'échec d'un nœud, ou l'échec de toute communication devant traverser le nœud.
- *Performance* : En dépit de l'objectif commun pour tous les réseaux qui est d'utiliser le moins d'énergie possible, de faire de petits retards et de minimiser le temps de conception, l'utilisation des technologies de réseau traditionnels (tels que TCP / IP) aux NoCs, entraîne trop de latence. En plus de cela, les NoCs présentent des proximités locales et moins de non-déterminisme. Les premières tentatives de conception NoC étaient peut-être trop influencées par les réseaux informatiques, souvent sous-estimé et les différences fondamentales ignorées.

I.4 Avantages et défis des NOCs

Traditionnellement, les circuits intégrés ont été conçus avec des connexions dédiées point à point, avec un fil dédié à chaque signal. Pour les grands designs et architectures en particulier, on retrouve plusieurs limites de conception d'un point de vue physique. Les fils occupent une grande partie de la zone de la puce, et dans la technologie CMOS nanométriques, les interconnexions dominent à la fois la performance et de la dissipation de puissance dynamique, comme la propagation des signaux dans les fils à travers la puce nécessitent plusieurs cycles d'horloge.

Les liens NoCs peuvent réduire la complexité de la conception de fils pour une vitesse prévisible, énergie, le bruit, la fiabilité…etc. grâce à leur structure régulière bien contrôlée. D'un point de vue de la conception du système, avec l'avènement des systèmes de processeurs multi cœurs, un réseau est un choix d'architecture naturelle. Le NoC peut assurer la séparation entre le calcul et la communication, la modularité et la réutilisation de support IP via des interfaces standards, gérer les problèmes de synchronisation, servir de plate-forme pour le test du système, et, par conséquent, augmenter la productivité de l'ingénierie.

- *Réutilisation des IP cores*

Initialement, la réutilisation massive de blocs de propriété intellectuelle existants est une nécessité absolue. L'interconnexion de NoCs doit transmettre les transactions entre les prises de différents protocoles (OCP, AXI, propriétaires, etc.), chacun d'entre eux est très paramétré. Cette variété a un impact majeur du point de vue de la performance. Par exemple, l'IP peut être cadencé à des fréquences différentes, alors qu'en ce qui concerne leurs orbites elles peuvent présenter des largeurs différentes de données, ce qui conduit à une large gamme de débits de pointe qui doit être adaptée entre les initiateurs et les cibles. En outre, les initiateurs présentent généralement différents modes de circulation, lorsqu'il s'agit d'une transaction longueur, alignement d'adresses, ainsi que la régularité. Par conséquent, leur réaction, la résistance à la latence, ou contre-pression transitoire, pourraient différer ainsi.
En outre, répondant aux QoS requis par chaque initiateur, malgré la présence de l'autre, le trafic devient plus difficile quand le nombre des IPs augmente.

- *Gestion de puissance*

Dans la plupart des systèmes sur puce, la gestion agressive de puissance devient une exigence, non seulement pour les appareils de poche où il est essentiel pour l'autonomie du système, mais aussi pour le bureau ou les applications automobiles, où elle a des conséquences directes sur la puce l'emballage et le refroidissement du système.

- *Contraintes physiques et Timing Closure*

La plupart des IPs rétrécissent avec le procédé CMOS, mais ce n'est pas le cas du NoC. Il est, par définition, la propagation à travers la filière, ce qui fait de la phase d'arrière-plan un défi difficile à relever. D'un côté, en fonction des fréquences d'horloge, les étages de pipeline peuvent être insérés, il suffit de laisser une marge de retard de propagation spatiale. D'autre part, les fils longs sont des ressources coûteuses, donc la possibilité d'ajuster localement le nombre de fils dans la communication est la clé.

- *Observabilité et la sécurité du système*

Si la complexité du matériel augmente, la complexité des logiciels augment aussi avec des conséquences sur le matériel. Vers cette direction, de nombreuses couches logicielles collaborent sur un SoC : Firmware, système d'exploitation, les pilotes, des applications, sous-systèmes de données sensibles, etc., tous répartis sur les différentes adresses IP. L'infrastructure d'interconnexion doit prévoir des mécanismes afin d'accorder ou de refuser, le droit pour des opérations d'atteindre certains objectifs, selon les initiateurs et dans la bande de sécurité.

Pour des raisons de débogage logiciel, le NoC doit être observable dans une certaine mesure. La détection des erreurs et leur signalisation est un minimum, mais l'interdiction de transaction et parfois la mesure des performances sur le chip sont nécessaires. Sur puce, le trafic devient de plus en plus distribué, l'observation devient plus chère. Etant donné que l'on ne peut observer tout le trafic en temps réel, l'observation doit se concentrer sur le trafic stratégique comme le DRAM accesses. En fin de compte, le coût par rapport à l'observabilité est un compromis qui doit être laissé entre les mains du concepteur.

- *Vérification*

En général, la vérification de la qualité des IPs est un problème bien connu au quel les concepteurs ont été confrontés au cours du développement des nouveaux produits. En raison de son large espace de configuration (en milliers des paramètres est un bon ordre de grandeur), vérifier la mise en œuvre d'un NoC particulier est un problème beaucoup plus complexe. Pour compliquer les choses, la vérification de la conformité des prises à leur protocole respectif ne suffit pas. La bonne conversion des transactions entre le côté initiateur et le côté de la cible doit être vérifiée aussi. Afin de répondre à cette exigence, en fonction de l'initiateur et les configurations des prises ciblées, les transactions peuvent être divisées, réalignées, non emballées, etc. En plus de cette vérification fonctionnelle, la performance du NOC doit également être validée.

- *EDA flot de conception*

Le développement du NoC recouvre toute la durée de conception du SoC. De la phase d'architecture, à la dernière étape qui est (Place & Route P & R), en passant par la qualification de performance, l'intégration IP, la vérification, l'intégration de logiciels. On doit accorder une attention particulière aux décisions prises au cours d'une première phase, tels que la conception architecturale, pour que celles-ci ne mettent pas en péril la faisabilité d'une phase ultérieure, comme back-end. En outre, il est fréquent que les exigences de commercialisation pour un SoC donnée changent au fur et à mesure que la conception avance, ajoutant ou en supprimant certaines adresses IP, et affectant ainsi la spécification NoC.

De plus, lorsque la complexité et le coût d'une nouvelle architecture augmentent, l'organisation de projets autour de plates-formes réutilisables devient plus efficace. De petites variations de la plate-forme, ou des dérivés, peuvent ensuite être mises sur le marché dans des périodes très courtes.

La méthodologie de conception de système d'interconnexion doit se faire de sorte que les changements de spécifications de dernières minutes soient faciles à intégrer et à vérifier dans la conception.

I.5 La recherche sur les NOCs

Il y a quelques exigences auxquelles chaque mise en œuvre d'un NoC doit répondre. Entre autres, les exigences de performance comprennent : petite latence, débit garanti, diversité de trajet, la capacité de transfert suffisante, et la faible consommation de puissance. De même, en ce qui concerne les exigences architecturales, la nécessité pour l'évolutivité de la généralité, et de programmation. En dehors d'eux, en raison de mise à l'échelle de la technologie et de la différence entre les conditions d'exploitation, les NoCs doivent également aborder les questions liées à la tolérance aux pannes, ainsi qu'au fonctionnement valide en vertu des exigences de qualité de service. Dans cette section, nous présentons une analyse des approches des différents groupes de recherche. La Figure I-3 illustre une classification simplifiée de cette recherche. Bien que les NoCs puissent emprunter des concepts et des techniques bien établis du domaine des réseaux informatiques, il est impossible de réutiliser aveuglément ces caractéristiques « classiques » des réseaux informatiques. En particulier, les switches des NoCs devraient être petits, économes en énergie, et rapides. Les algorithmes de routage doivent être mis en œuvre par une logique simple, et le nombre des buffers de données devraient être minime. La topologie du réseau et les propriétés peuvent être spécifiques à l'application.

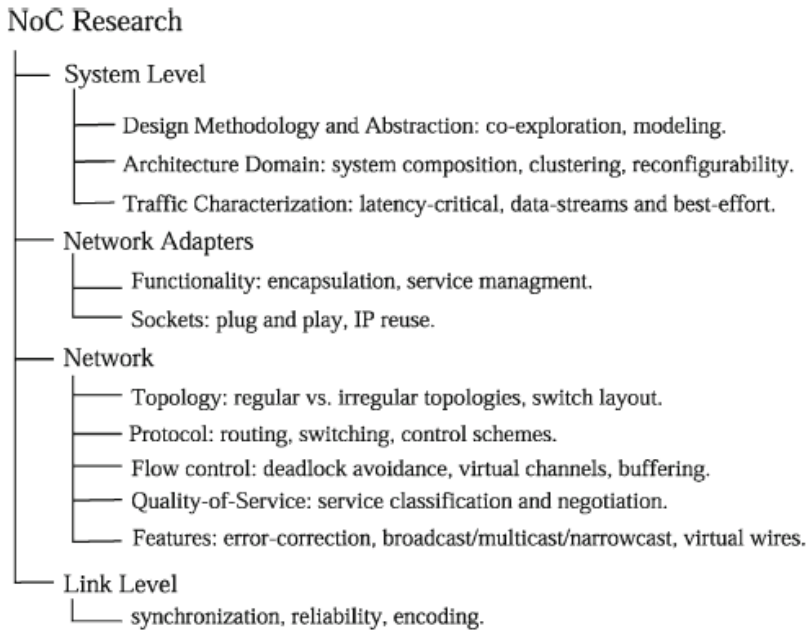


Figure I-3 : classes de recherches sur les NoCs

Certains chercheurs pensent que les NoCs doivent subvenir aux besoins de la QoS, à savoir atteindre les diverses exigences en termes de retard, débit, de bout en bout, et deadlines. Le calcul en temps réel, ainsi que la lecture audio et vidéo qui est une des raisons pour fournir la qualité de service. Cependant, les implémentations actuelles du système comme VxWorks, RTLinux, ou alors QNX sont en mesure d'atteindre des résultats de calcul en millisecondes sans matériel spécial. Cela peut indiquer que, pour de nombreuses applications en temps réel, une logique de matériel dédié serait nécessaire pour atteindre la précision d'une microseconde, un degré qui est rarement nécessaire pour les utilisateurs particuliers.

Une autre motivation pour la QoS des NoCs est de soutenir le partage des ressources d'un seul multiprocesseur à puce pour plusieurs utilisateurs simultanément dans une infrastructure de computation public cloud. Dans de tels cas, la qualité de service de matériel logique permet au fournisseur de services de faire des garanties contractuelles sur le niveau de service que l'utilisateur reçoit, une fonctionnalité qui peut être considérée comme souhaitable par certaines entreprises clients.

À ce jour, plusieurs NoCs prototypes ont été conçus et analysés dans l'industrie et les universités, mais seulement quelques-uns ont été mis en œuvre. Cependant, de nombreux défis et problèmes de recherche restent à résoudre à tous les niveaux, à partir du lien physique, passant par la couche réseau jusqu'au sommet de l'architecture du système (application).

En outre, un certain nombre de publications scientifiques dans des revues, des conférences et des ateliers peuvent être trouvées. Ces publications étudient divers sujets d'actualité de NoC, allant du logiciel jusqu'à l'architecture et la mise en œuvre du matériel.

CHAPITRE II : MODELISATION ET TOPOLOGIES

Ce chapitre décrit deux des tâches les plus importantes pour la conception des systèmes basés sur les NoCs traitant de la modélisation de NoC, ainsi que l'exploration de la topologie.
A cet effet, l'état de l'art des solutions architecturales est discuté et les sujets de recherche ouverts sont mis en évidence. En outre, ce chapitre fournit une description des modèles de circulation alternatifs utilisés comme entrée dans le domaine NoC pour évaluer l'efficacité de divers paramètres architecturaux.
Le dernier sujet abordé dans ce chapitre concerne la synthèse de la topologie et la cartographie de l'application sur l'architecture NoC dérivée sous diverses contraintes.

II.1 Topologies existantes

La topologie se réfère à la structure du réseau et à son organisation, car elle détermine les connexions entre les nœuds sur puce. Plus précisément, elle traite le nombre de nœuds (soit éléments de traitement ou des éléments de stockage), les routeurs, la communication liens, ainsi que leur interconnexion. De même, la topologie traite l'évaluation des topologies alternatives afin de quantifier l'efficacité pour chacun d'elles sous divers critères de conception. Basés sur ces résultats, les architectes de l'appareil effectuent la sélection de la topologie. Au cours de cette tâche, est sélectionnée, la topologie la plus souhaitable, qui satisfait les exigences de communication de l'application et impose le coût minimum (en termes de consommation de puissance / énergie, surface de silicium, etc.). Notez que lors de la sélection de la topologie, les contraintes au niveau physique sont également prises en considération. Enfin, il y a une étape d'évaluation, où l'efficacité des sélections précédentes est quantifiée.
Jusqu'à présent, les chercheurs proposent l'utilisation de diverses topologies de bases telles qu'un bus, étoile, maille, de point à point, ainsi que les topologies hiérarchiques.

II.1.1 Topologies régulières et irrégulières

Une première classification des topologies NoC concerne leur régularité. Plus précisément, en fonction de la structure de la régularité de la mise en page de réseau, le NoC est caractérisé soit en tant que structure régulière ou irrégulière. Ces deux topologies présentent des avantages selon le domaine d'application. La figure II-1 donne un exemple de ces dernières.

Plus précisément, une topologie régulière assume une répartition homogène des routeurs, ce qui conduit à réduire le temps de conception et son coût. Même s'il est possible d'incorporer

une telle topologie de conceptions à usage général, il convient surtout aux architectures maillées. En outre, elles sont très réutilisables et permettent d'imposer l'effort de re-conception minimale, au cas où elles sont utilisées pour différentes applications / architectures.
Malgré les avantages mentionnés ci-dessus, l'utilisation de topologies régulières n'est pas largement acceptée pour des produits commerciaux, car ils imposent un certain nombre de lacunes. Ces limitations se produisent principalement en raison de l'utilisation non optimale des réseaux d'interconnexion, ce qui entraîne à son tour l'augmentation du retard et de la consommation d'énergie.

Afin de pallier à ces limitations, l'adoption de topologies irrégulières, ou modifiées, est également proposée. Étant donné que ces topologies sont principalement orientées vers l'application, elles sont adaptées pour les systèmes sur puce formés à partir de noyaux hétérogènes. En outre, la conception de topologies irrégulières nécessite plusieurs algorithmes de routage avancés qui prennent en compte la non-uniformité du réseau sous-jacent. Par exemple, on suppose une architecture NoC où chacun des routeurs doit être attaché à un nombre différent de nœuds. Dans le cas d'une topologie régulière, certains des routeurs ne seront pas utilisés, ce qui entraîne à son tour un retard, une consommation importante de puissance. D'autre part, il est possible de concevoir une architecture NoC non hétérogène, où chaque routeur dispose d'un nombre différent de ports. Une telle modification des composants matériels résulte de meilleures performances, par rapport au cas où une topologie régulière est utilisée. Cependant, l'hétérogénéité introduite impose un effort supplémentaire de conception à la fois pour les routeurs, ainsi que pour tout le NoCs.

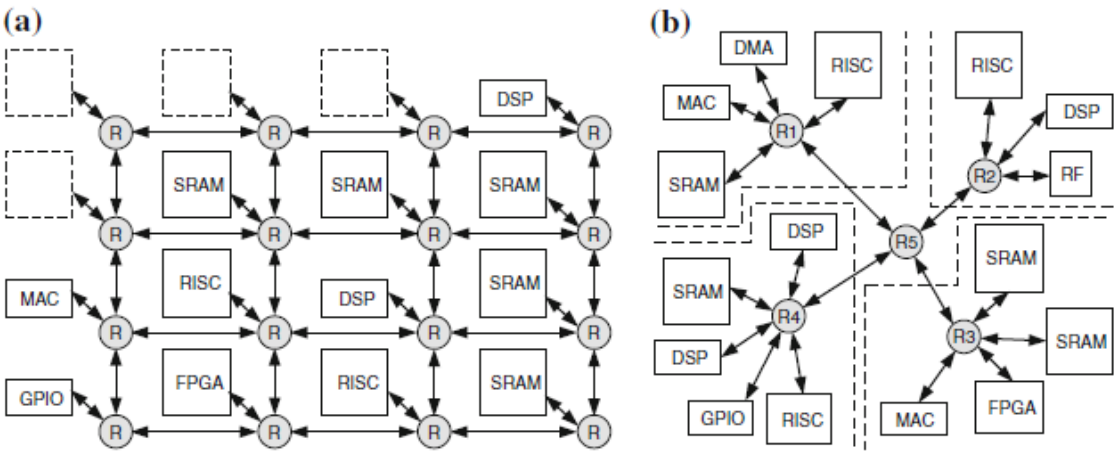


Figure II-1 Exemple de topologie (a) régulière (b) irrégulière

En conclusion, nous pouvons dire qu'il est impossible de concevoir une seule topologie de NoC qui convient à tous les domaines d'application, puisque chacune d'elles introduit des avantages et des contraintes qui doivent être exploités autant que possible afin de maximiser l'efficacité du réseau de communication sous-jacent. Dans ce but, mis à part les deux solutions extrêmes mentionnées précédemment, les architectes intègrent également à leurs conceptions hétérogènes, ou topologies par morceaux homogènes. Ces topologies sont constituées de blocs fortement configurables qui peuvent être personnalisés pour un domaine d'application spécifique (par exemple multimédia, télécommunications, etc.).

II.1.2 Topologies directes et indirectes

De même, il est possible de classer la topologie en réseau directe ou indirecte. Lors d'une topologie directe, chaque nœud a un lien direct de point à point avec un sous-ensemble de l'autre nœud dans un système, appelés nœuds voisins. Les topologies directes conduisent généralement à une plus grande disponibilité de la bande passante de communication, mais elles imposent une augmentation du nombre de nœuds dans le système. Par conséquent, il existe un compromis fondamental entre la connectivité et le coût. Un certain nombre de produits des NoCs existants incorporent une topologie directe, entre autres sont le Nostrum [12], Proteo [13]…etc.

La majorité des topologies de réseau ciblées ont une implémentation orthogonale et les nœuds intégrés sont généralement disposés dans un espace n-dimensionnel orthogonale. Un tel agencement architectural présente l'avantage compétitif de simplicité. Par conséquent, il est possible d'utiliser des algorithmes de routage assez légers et rapides qui permettent d'atteindre le routage des paquets avec des performances comparables, mais avec une complexité moins significative. Instanciations typiques de cette topologie sont la maille à n dimensions, le tore, tore croisés, l'hyper cube, et l'octogone.

II.1.3 Topologies 2D

Un maillage 2D, schématisé sur la figure II-2, est la plus simple et la plus populaire des topologies pour les NoCs. Il se compose d'un maillage M x N nœuds d'interconnexion de commutateurs (Par exemple, cœurs de traitement, mémoires, etc.) placés le long avec les commutateurs. Chaque commutateur, à l'exception de ceux sur les bords, est relié à quatre commutateurs voisins et un nœud. Dans ce cas, le nombre de commutateurs est égal au nombre de nœuds. Afin de réaliser la voie de communication physique entre les nœuds et les commutateurs, nous utilisons des canaux de communication, dont chacune se compose de deux

voies unidirectionnelles. La topologie de maillage 2D suppose que toutes les liaisons ont la même longueur, mais aussi, une régularité inhérente qui simplifie la conception physique considérablement. En outre, il est plus facile de prévoir les besoins en espace pour des topologies de maillage, car elle accroît presque linéairement avec l'augmentation du nombre de nœuds. En dehors de ces avantages, l'utilisation de topologie maillée a aussi quelques inconvénients. La pléthore de routeurs incorporés dans cette topologie conduit généralement à des régions congestionnées. A cet effet, une conception soignée et une cartographie de l'application doit être effectuée afin d'éviter le cumul du trafic, en particulier au centre de la maille.

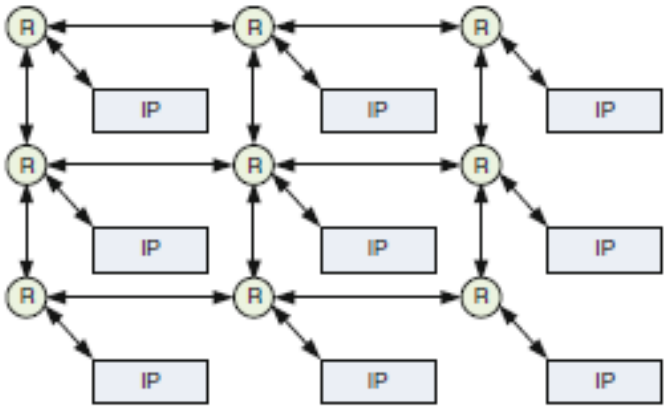


Figure II-2 : Maillage 2D 3x3

II.1.4 Topologies 3D

L'empilage en 3D sur puce (figure II-3) est présenté comme la technologie de solution miracle qui peut maintenir l'élan de Moore et alimenter la prochaine vague de produits électroniques grand public. En dehors de la flexibilité imposée par ce nouveau paradigme de la conception, l'un des principaux défis auxquelles les concepteurs font face aujourd'hui en matière d'intégration 3D est de savoir comment obtenir l'interconnexion à la fois entre les composants à l'intérieur d'une couche, ainsi que pour les couches de manière évolutive et efficace. Une solution fiable à ce problème est l'utilisation des NoCs.

En combinant les avantages offerts par l'intégration 3D avec l'évolutivité accrue trouvée dans les NoCs, ce nouveau paradigme de conception semble prometteur pour fournir les infrastructures de communication pour la prochaine génération de systèmes complexes. Plus précisément, la localisation le long de l'axe z conduit notamment à réduire considérablement le retard d'interconnexion, la structure d'interconnexion canonique, une flexibilité accrue, ainsi que l'intégration de systèmes et de technologies différents. Cependant, pour que ces

architectures puissent être acceptées, de nouvelles méthodologies et outils de conception, doivent prendre en considération les caractéristiques inhérentes offertes par cette technologie.

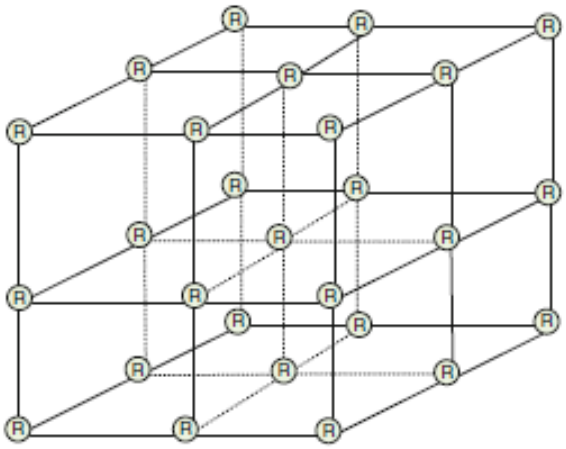


Figure II-3 : Exemple d'empilage 3D maillé

II.2 Modélisation du trafic

Habituellement, la description d'une architecture NoC est basée sur les états et les transitions entre les états qui sont causées par diverses actions dans le système. Ce modèle représente une abstraction du comportement du système, ce qui est suffisamment précis, mais traitable, pour l'analyse et la vérification. Mesurer et comparer la performance, le coût, et d'autres paramètres de l'architecture, est un défi important qui a à peine été abordé. Malheureusement, les benchmarks pour des systèmes à multiprocesseurs [14, 15] sont généralement orientés vers les applications, et donc ils ne peuvent pas être utilisés directement pour quantifier les architectures de communication à forte densité, tels que les NoCs. Et puisque le trafic varie considérablement au cours de l'exécution de l'application, plus de modèles de trafic doivent être intégrés afin d'étudier en détail le comportent du système cible. A cet effet, tout au long de cette section, on décrit un certain nombre bien établi de modèles de trafic utilisés à des fins d'évaluation.

II.2.1 Modèles disponibles pour le trafic

La conception d'architectures de NoCs exige des mécanismes permettant de prédire avec précision l'utilisation du réseau à l'avance, au lieu de le concevoir et ensuite évaluer son rendement. Afin de répondre à cette exigence, les architectes emploient divers modèles de trafic pour déterminer et quantifier l'impact des paramètres critiques du réseau sous-jacent. Plus précisément, un modèle de circulation qui est un graphique qui décrit à la fois la répartition

spatiale, ainsi que la répartition temporelle du trafic dans le réseau (communication de données et signalisation entre les nœuds).

En général, ce graphe se concentre sur la quantité de trafic qui doit être transmise au NoC et non sur les dépendances entre les nœuds. Même si cela implique que les conclusions dérivées en employant de tels graphiques ne correspondent pas aux exigences du système réel.

Cependant, leur utilisation est souhaitable en particulier pendant la phase de conception initiale afin de quantifier les caractéristiques spécifiques de l'architecture sous-jacente. En outre, avec ces modèles de trafic, il est possible d'avoir une vue d'ensemble abstraite de l'impact de divers modèles de trafic sur la performance de l'architecture NoC. Les modèles de circulation sont classés plus réalistes que synthétiques. Plus précisément, les modèles réalistes de trafic sont des traces d'exécution de l'application sur le NoC. D'autre part, les modèles de trafic synthétiques correspondent à des modèles abstraits de paquets échangés entre les nœuds des NoCs contrairement au trafic réaliste, le trafic synthétique est généré sur une base mathématique des modèles. Par conséquent, il couvre une large classe d'applications exécutées sur des plates-formes de NoC.

L'avantage concurrentiel de l'intégration des modèles synthétiques est qu'il permet à un réseau d'être souligné avec un motif régulier et prévisible. Cependant, ils ne représentent pas le trafic des applications réelles.

En dehors de la modélisation du trafic, les outils qui permettent la génération de trafic sont également importants pendant la phase d'exploration. En raison de l'importance de cette opération, jusqu'à présent, de nombreux générateurs de trafic ont été proposées. En outre, ces solutions sont capables de générer du trafic stochastique en fonction des résultats générées par les applications réelles.

II.2.1.1 Trafic synthétique

- *Distribution temporelle*

Une répartition temporelle détermine comment un nœud génère du trafic en fonction du temps et montre aussi la façon dont le trafic se propage vers l'architecture NoC. Cette distribution comprend une liste de propriétés de la circulation, comme la période (ou taux) de messages générés, tandis que les valeurs typiques de ce paramètre sont constantes (périodique), aléatoire, normal, etc. De même, il est possible d'effectuer une autre classification en fonction de leur temps d'information de synchronisation.

Nous allons décrire plus en détail les propriétés de trois différents modèles de distribution temporelle.

Modèle synchrone parfait : le modèle synchrone parfait est basé sur la "hypothèse de synchronisation parfaite " [16]. Cette approche suppose que ni la communication ni les tâches de calcul ne prennent du temps. En d'autres termes, chaque fois que nous utilisons le parfait modèle synchrone, le résultat d'un calcul est disponible en même temps que la donnée à l'entrée est appliquée. En outre, si plusieurs processus sont reliés entre eux, alors le résultat des calculs se répandra instantanément à travers le système. Même si le modèle synchrone parfait est facile à comprendre, il conduit à des résultats non réalistes, car il ne prend pas en considération toutes les informations de synchronisation. Par conséquent, les conclusions obtenues en considérant cette approche peuvent beaucoup différer par rapport à une approche plus précise, ou la mise en œuvre réelle du NoC.

Modèle synchrone cadencé : Cette approche suppose qu'un signal d'horloge global contrôle le début de chaque calcul dans le système. Plus précisément, dans ce modèle synchrone, la communication ne prend pas de temps, alors que le calcul est effectué à un seul cycle d'horloge. Par conséquent, cette approche est susceptible d'être utilisé à des modèles de cycle vrai. D'autre part, la limitation de ce modèle affecte le manque de sensibilisation à prendre en compte le temps physique, étant donné que la période de temps est exprimée en cycles d'horloge.

Modèle d'événement discret : Contrairement aux deux précédents modèles axés sur le temps les fonctions, celui-ci donne la possibilité d'exprimer le temps en virgule flottante, qui permet à son tour la simulation du temps physique. Cette fonctionnalité simplifie considérablement la procédure de simulation, tandis que la flexibilité correspondante devient beaucoup plus importante à chaque fois que le modèle se compose de plusieurs domaines temporels.

En outre, il existe différents modèles de trafics disponibles pour les topologies de NoCs, qui soutiennent la génération de paquets pour des taux différents.

- *Répartition spatiale*

Uniforme : A cette approche, chaque nœud envoie des paquets à un nœud de destination choisi au hasard. Même si ce dernier est l'un des modèles de trafic les plus simples, sa simplicité pourrait conduire à des résultats non acceptables pour l'évaluation de l'architecture NoC.

Bit Permutation : dans ce trafic de permutation de bits, chaque source envoie tout son trafic à une seule destination. Le nœud de destination dans cette approche est effectué sur la base

d'une fonction de l'adresse de source (ce qui est également connu comme motif de permutation de bits). En d'autres termes, un nœud source envoie ses données à un seul nœud de destination, alors que l'adresse du nœud de destination est une fonction de l'adresse du nœud source. Car ce type de trafic se concentre sur un couple individuel source-destination.

Transposition : Dans le trafic transposition, les coordonnées de destination sont la transposition des coordonnées de source. Sous cette charge, la bissectrice diagonale du réseau est un goulot d'étranglement que tous les paquets doivent traverser.

En incorporant la transposition de la circulation en conjonction à l'aide d'algorithmes de routage NoC appropriés, il peut conduire à une charge de réseau déséquilibrée.

Bit Complément : Un autre trafic de charge couramment utilisé est le bit complément dans lequel chaque nœud échange des paquets avec un nœud sur le côté opposé du réseau avec un taux d'uniformité aléatoirement distribué. Afin de calculer les coordonnées du nœud de destination dans ce modèle de trafic, on effectue une inversion bit par bit des coordonnées de source. Ceci offre une parfaite charge du réseau équilibré.

D'autres modèles de trafics existent aussi, mais se basent toujours sur des manipulations d'inversion de bit ou rotation de bit … etc.

II.2.1.2 Trafic Réaliste

Mis à part les modèles de trafic de synthèse mentionnés ci-dessus, il est courant d'évaluer les architectures NoCs avec l'utilisation du trafic capturé à partir des applications réelles. Ce trafic, aussi connu comme le trafic réaliste, permet au concepteur d'analyser avec une certaine précision un nombre de paramètres tels que le retard et la puissance de consommation. Il convient de noter, cependant, que les modèles de trafic générés par les différents modules dans un NoC dépendent fortement de l'application pour laquelle le NoC est conçu. Vu que les performances sont fonctions du profil du trafic, la façon la plus précise pour évaluer les caractéristiques des NoCs serait de générer des applications dépendantes du profil du trafic. Même si cela est le scénario optimal, généralement la conception de systèmes affecte les dispositifs qui peuvent satisfaire une large gamme d'applications. Diverses approches pour un trafic réaliste ont été proposées ces dernières années. Parmi d'autres sont la voix GSM CODEC [17], SPLASH-2 [18], MediaBench [19], et SPEC [20].
Malheureusement, une telle sélection pourrait conduire à un temps de période très élevé excessif pour le profilage, même si toutes les applications sont bien connues. Afin de pallier à cette

limitation, les concepteurs utilisent habituellement des profils de trafic de synthèse (comme on l'a vu dans les sections précédentes), qui peuvent représenter le trafic pour une classe d'applications. Ceci suggère l'utilisation des deux profils de trafic réalistes et synthétiques qui constituent un ensemble complet pour l'évaluation des techniques proposées pour les systèmes de NoC.

II.3 Synthèse de topologies

L'une des tâches les plus difficiles lors de la conception de NoC est la synthèse de la topologie. Plus précisément, le problème de la synthèse de la topologie NoC a pour but de générer au niveau physique l'infrastructure de communication basée sur le réseau sélectionné. Comme les contraintes à cette procédure sont les objectifs de conception multiples posés par les exigences des systèmes, tels que la performance, la puissance et la zone occupée. Les tâches qui ont lieu dans un cadre typique pour la conception d'architectures NoC sont représentées sur la figure II-4

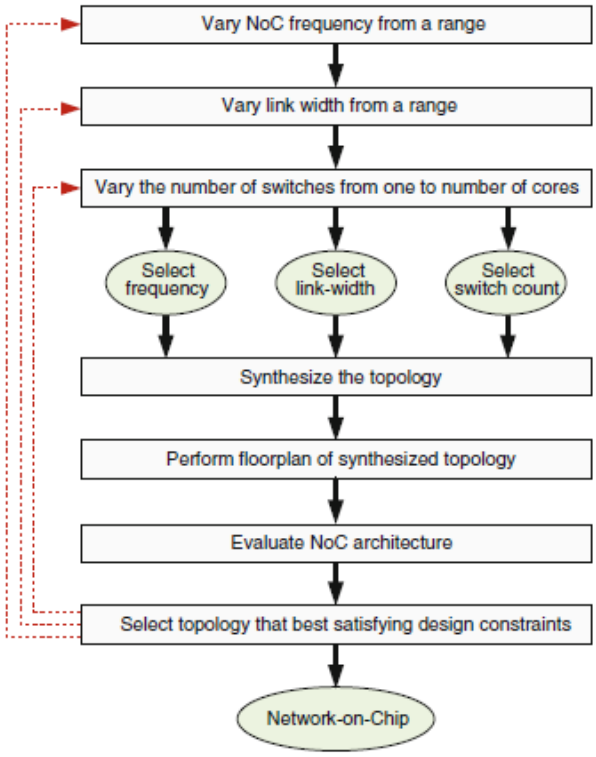


Figure II-4 : Etapes de la sélection de topologie

Dans les itérations extérieures, un certain nombre d'opérations (par exemple, fréquence) et architectures (Par exemple, largeur de liaison) sont modifiées dans un ensemble de valeurs appropriées. Les paramètres sélectionnés sont généralement employés lors de la conception du

NoC, étant donné que le produit de la fréquence et la largeur des liaisons correspond à la bande passante disponible. Dans ce cas, nous effectuons une exploration exhaustive, puis des topologies ayant des nombres de commutateurs différents, en partant d'une topologie dans laquelle tous les noyaux sont connectés à un commutateur, à un système où chaque noyau est relié à un commutateur séparé.

Pour chacun des paramètres architecturaux de l'espace de conception, nous réalisons la génération de topologie, afin de s'assurer que le trafic sur chaque liaison est inférieur ou égal à sa bande passante disponible. Ensuite, une étape d'analyse dans le but d'évaluer l'efficacité de l'architecture dérivée. Pour une analyse précise, une tâche de floorplaning pour déterminer la position 2D des noyaux et des composants du réseau doit également être entreprise.

Basé sur le point de fréquence et la longueur des fils obtenus, toute violation de synchronisation sur les fils est détectée et la consommation d'énergie sur les liens peut être récupérée.

Finalement, à partir de l'ensemble des topologies de synthèse et paramètres architecturaux de conception, la topologie et la configuration architecturale qui optimise les objectifs de l'utilisateur, tout en répondant à toutes les contraintes de conception, est choisie.

Le résultat de la synthèse de la topologie est soit une topologie régulière ou irrégulière. En ce qui concerne la topologie régulière, elle est adaptée pour les architectures classiques (par exemple, maille et tore). Cependant, dans le cas où une telle topologie est appliquée à un hétérogène MPSoC, le système de communication dérivée présente généralement une mauvaise performance et de grands frais généraux d'alimentation en raison de l'utilisation non optimale des ressources d'interconnexion. Afin de surmonter cette limitation, les topologies de NoC personnalisées peuvent également être générées. Même si ces architectures imposent un temps de conception plus important, parce que le système cible comprend des structures de tailles différentes, par rapport aux topologies régulières mentionnées précédemment, elles aboutissent à des performances supérieures pour les mêmes paramètres exigés.

En raison de l'importance du développement des NoCs spécifiques à l'application, un certain nombre de méthodologies de conception ont été proposées, dont chacun vise à lutter contre un problème d'optimisation.

CHAPITRE III : INFRASTRUCTURE DE COMMUNICATION

Après détermination de la topologie NoC pour l'application donnée, la conception de l'infrastructure de communication est la prochaine étape. L'algorithme de routage sélectionné est basé à la fois sur la topologie sélectionnée mais aussi sur les contraintes de conception. Après la sélection de l'algorithme de routage et du système de contrôle de flux, la conception du routeur et du lien peut commencer. Dans le sous chapitre de techniques de commutation, des algorithmes de routage et le débit des systèmes de contrôle sont discutés et comparés, tandis que la conception d'un routeur 2D et 3D générique est illustré et des améliorations proposées sont discutées.

Après la sélection de la topologie, la prochaine étape dans le flux de conception de NoC est de sélectionner la technique de commutation appropriée et l'algorithme de routage sur la base des contraintes de conception.

Sur la base de ces décisions, la conception du routeur et les canaux peuvent commencer.

III.1 Techniques de commutation

La technique de commutation définit quand et comment le canal de l'interrupteur d'entrée est connecté au canal de sortie sélectionné par l'algorithme de calcul d'itinéraire. Les données sont transmises sous forme de messages qui sont découpés en paquets, qui sont à leur tour divisés dans le contrôle en flux unités (flits) et enfin, en unités physiques (phits). La sélection de la technique de commutation implique la sélection de la granularité optimale pour les données ci-dessus.

• *Unités physiques (phits)* : l'unité de données transférées via le lien physique. Essentiellement, la largeur de bit ou longueur du canal, et par conséquent, le nombre de bits transmis entre les routeurs dans un seul cycle d'horloge.

• *Unités de contrôle de flux (flits)* : L'unité de synchronisation entre les routeurs. Au moins aussi grande qu'un phit et souvent égale.

• *Paquets* : Un ensemble successif de flits avec la même destination. Les routeurs peuvent stocker un paquet entier avant de le transmettre, ou transmettre les flits séparément.

• *Messages* : Un ensemble de paquets qui correspond typiquement à un transfert de données complet (Transaction) entre les nœuds. Un message pourrait être par exemple tout un bus d'opérations du processeur à la mémoire. Considérant que les protocoles de bus permettent à des salves de données longues, un message peut exiger le fractionnement de nombreux paquets. Les deux techniques de commutation utilisées couramment sont les suivantes :

(i) La commutation de paquets.

(ii) La commutation de circuits.

III.1.1 Commutation de circuits

Dans la commutation de circuits, le chemin de réseau entre deux nœuds qui doivent échanger des données est établi à l'avance (avant que les données ne soient envoyées) en affectant les bonnes ressources matérielles (liens). La commutation de circuits est réalisée en trois étapes : la mise en place du circuit (setup), la transmission de données, et la libération de circuits (démontage). La procédure de mise en place nécessite que l'entête du flit fasse son chemin de la source à la destination réservant des liens sur son chemin. Lorsque cette dernière atteint sa destination, un accusé de réception est retourné à l'expéditeur, sauf si un lien est réservé par un autre circuit, dans ce cas un accusé de réception négatif est envoyé. Au bout d'un accusé de réception réussie, le transfert de données commence et dure jusqu'à ce que le circuit soit libéré. La contention ne peut se produire que pendant la phase d'installation et la mise en mémoire tampon est uniquement nécessaire pour l'entête du flit, puisque les données sont câblées directement de la source à la destination.

L'avantage de la commutation de circuit réside dans le fait que, puisque la connexion sur laquelle toutes les données sont transportées est mise en place en premier lieu, la contention de résolution a lieu lors de la configuration à la granularité des connexions. Des garanties liées au temps pendant le transfert des données peuvent être données. D'autre part, quand un circuit entre deux nœuds est établi, toute autre communication sur les fils alloués est rejetée, jusqu'à ce que le transfert des données soit terminé, et par conséquent, d'autres messages peuvent être bloqués.

En conclusion, la commutation de circuit fonctionne bien lorsque de grandes quantités de données sont transférées, ce qui aide à amortir le temps d'installation.

III.1.2 Commutation de paquets

La commutation de paquets [21] permet aux paquets dans un message d'être transmis à travers les différents chemins. Afin d'assurer l'acheminement efficace, la commutation de paquets implique typiquement certaines restrictions. Plus précisément, dès qu'un flit (entête du flit) d'un paquet est envoyé sur un port de sortie, le port de sortie est réservé pour des flits appartenant seulement à ce même paquet. Puisque cette approche utilise un tampon de granularité plus fin et un canal de contrôle au niveau du flit au lieu du paquet, elle présente des performances accrues, en particulier lorsque les messages sont courts et fréquents. Cependant,

dans le cas où l'entête du flit d'un paquet est bloqué, les flits suivants peuvent donc être répartis sur plusieurs routeurs, bloquant les liens intermédiaires, provoquant un blocage communément appelé « Deadlock » (voir chap. III.2.1). La commutation de paquets comprend trois sous-catégories, à savoir : Store-and-Forward (SAF), Cut-Through-Virtuel (VCT). Plus précisément, l'approche store-and-forward est basée sur la réception et le stockage de l'ensemble du paquet entrant avant qu'il ne soit transmis au routeur suivant. Cela nécessite plus d'espace tampon puisque le paquet en entier doit être stocké, ce qui conduit, entre autres, a un temps de retard/routeur (le temps nécessaire pour que le routeur puisse recevoir le paquet).

D'autre part, la commutation de VCT est basée sur la transmission d'un paquet lorsque le routeur suivant garantit que le paquet complet sera accepté. Dans le cas où il n'y a aucune garantie, le routeur doit être capable de stocker l'ensemble du paquet. Par conséquent, même si une telle approche de routage nécessite un espace tampon pour un paquet complet, comme routage SAF, il permet néanmoins un retard moins important que plus faible.

Essentiellement, les flits peuvent aller à l'entrée du prochain routeur avant que le paquet ne soit complètement reçu dans le commutateur, et par conséquent le paquet est pipeliné à travers le commutateur.

III.1.3 Techniques de commutation hybrides

Une technique de commutation de paquets circuit hybride basé sur l'espace de multiplexage par répartition (SDM) a été proposée dans la littérature [22] afin de gérer efficacement la fois le streaming et le trafic généré dans des applications en temps réel, par conséquent en utilisant un sous-routeur à commutation de circuit à la fois avec SDM et TDM, et un sous-routeur à commutation de paquets.

III.2 Routage de paquets

Comme mentionné brièvement dans le Chap. I, les techniques de routage pour la communication sur puce déterminent le chemin choisi par un paquet pour atteindre sa destination. Il est évident que le routage est étroitement lié à la topologie cible.

La première tâche est d'identifier de manière appropriée et unique les ressources matérielles qui communiquent via le réseau (Nœuds) en utilisant une dénomination sans ambiguïté et un système d'adressage. Les approches habituelles sont :

• *Par nom* (par exemple l'objet X)

• *Par adresse* (objet, par exemple à destination X). Par exemple, pour un maximum de 8 nœuds 3 bits sont requis pour coder les adresses, pour un maximum de 16, 4 bits, etc.

• *avec un identificateur de groupe* (par exemple, tous les objets liés à X) utilisé pour identifier un groupe multidiffusion.

• *Topologie en maille ou en tore*, il est facile de répondre à un nœud en utilisant les coordonnées cartésiennes Les coordonnées (x, y et z dans le cas d'un 3-D NoC).

Les algorithmes de routage peuvent être classés en trois groupes principaux, à savoir :

• *déterministes* : l'itinéraire entre les paires de source-destination est fixe. Cette approche ne peut pas fournir une solution à la congestion dynamique ou à la tolérance aux pannes.

• *Adaptative* : le choix du chemin entre les paires de source-destination est dynamique.

• *Stochastique* : Dans cette approche le coût entre les différents chemins de routage varie selon des probabilités.

En termes de mise en œuvre, les algorithmes de routage peuvent être mis en œuvre par des mesures appropriées logiques ou communément à travers une table de routage qui contient essentiellement les informations de routage pour toutes les destinations possibles. Une autre technique associée à l'acheminement de calcul est la distinction entre le routage source et le routage distribué. Dans le routage source, le chemin du routage entier est calculé à la source (Typiquement NI connectée au nœud de source) et est attribué au paquet. Les routeurs ne prennent pas de décisions de routage, mais simplement mettent en œuvre le chemin de routage sur la base des informations codées dans le paquet. D'autre part, dans un routage distribué, le chemin de routage est décidé à chaque routeur, même pour des algorithmes de routage déterministe. La seule information qui doit être trouvée dans le paquet est l'adresse de destination. L'avantage d'un routage source est qu'il exige des routeurs simples qui peuvent facilement supporter des architectures irrégulières. Son inconvénient est qu'il ne fournit pas une solution d'adaptabilité et nécessite plus de NI et de paquets complexes.
Enfin, les algorithmes de routage peuvent être classés en fonction de la topologie cible (Régulier, irrégulier, hiérarchique, etc.).

Une des caractéristiques souhaitable d'un algorithme de routage est sa possibilité d'éviter les deadlocks et livelocks. Particulièrement le deadlock qui est critique pour les NoCs, puisque la mise en œuvre d'un mécanisme qui détecte automatiquement les deadlocks et procède à la

récupération peut ne pas être abordable en termes de ressources de silicium. Il peut aussi conduire à des retards imprévisibles.

III.2.1 Deadlocks

L'inter blocage est en général ce qui arrive quand dans le déroulement de deux processus, l'un attend l'autre qu'il termine en premier, et donc ne peut commencer. Dans l'environnement des réseaux, cet inter blocage peut se produire lorsque des dépendances circulaires existent, par exemple, un paquet p1 détient un canal c1 et demande un canal c2, qui est réservé à un paquet p2, qui à son tour demande le canal c3, qui est réservé à p3, qui demande le canal c4…etc. Enfin un paquet pN demande le canal c1, et par conséquent, tous les paquets restent bloqués pour une durée indéterminée, en attendant un événement qui ne se produira jamais. La commutation Wormhole Switching (WS) est particulièrement vulnérable à ce blocage.

Il existe deux solutions possibles au problème de blocage, à savoir la récupération et l'évitement [23]. La récupération nécessite moins de ressources et peut surpasser l'évitement si les blocages sont rares, mais conduit à des retards imprévisibles sachant que sa mise en œuvre reste difficile [24]. L'évitement nécessite généralement plus de ressources et peut limiter la flexibilité de routage et par conséquent les performances. Cependant, il est habituellement la solution préférée en raison de sa facilité de mise en œuvre.

III.2.1.1 La récupération (Deadlock Recovery)

La récupération nécessite un mécanisme de détection de blocage soutenu par un mécanisme de résolution de l'impasse. La détection de Deadlock reste difficile en raison de la nature distribuée des blocages. Une approche commune est d'employer des mécanismes conditionnels tel que des time-outs [25, 26]. Bien que cette approche soit sûre de détecter tous les blocages, elle peut aussi conduire à des paquets bloqués faussement marqués comme faux deadlock. La raison en est qu'il est difficile de déterminer la valeur de seuil optimale temporisation, car elle est fortement dépendante des paramètres du réseau (longueur de paquet, la taille du réseau, topologie, etc.) ainsi que les caractéristiques d'application (trafic spécifique).

En ce qui concerne les systèmes de récupération, ils peuvent être classés comme régressives ou progressives.

Un système de récupération régressive tue le paquet signalé comme un deadlock et réémet après un délai d'attente [27]. Cependant, la retransmission d'un paquet réduit la performance, et

généralement un système de récupération progressive utilise qui plus de matériel pour contourner les paquets suspects à leur destination [25] est préféré.

III.2.1.2 L'évitement (Deadlock Avoidance)

Une condition nécessaire et suffisante pour dire qu'un algorithme de routage est deadlock-free est essentiellement l'absence de dépendances circulaires.
Un graphe de dépendance de ressources cyclique peut être transformé en un graphe acyclique en y ajoutant simplement plus de ressources. Ceci est une approche commune pour assurer le non blocage en cas d'algorithmes non testé deadlock free.

Une autre approche est d'interdire certains virages, ce qui réduit les performances et cause des restrictions de routage. Cela n'est pas applicable aux architectures irrégulières.

III.2.2 Livelocks

Livelock est la situation où un flit ou un paquet est perpétuellement dévié mais jamais bloqué, qui n'atteint jamais sa destination. Il peut arriver dans les algorithmes de routage qu'il y-ait la déviation d'un paquet, et donc ces algorithmes doivent veiller par un mécanisme à ce que tous les paquets atteignent leur destination éventuellement. Il existe deux catégories de base des algorithmes sans livelocks, à savoir déterministe livelock-free déterministe et livelock-free probabiliste. Les premiers sont basés sur l'ajout à chaque paquet des informations supplémentaires utilisées pour prioriser les paquets et ne pas détourner ceux avec une priorité haute. Les paramètres typiques utilisés sont l'âge, le nombre de déviations, etc. Les algorithmes sans livelock probabilistes sont basés sur la garantie que plus le temps approche de l'infini plus la probabilité qu'un paquet puisse atteindre sa destination approche le 1 (100%), ce qui signifie que le paquet finira par atteindre sa destination. Cependant, le concepteur doit également veiller à ce que la solution choisie satisfasse les contraintes de performance.

III.2.3 Algorithmes de routage pour les architectures régulières

Les architectures régulières sont les plus couramment utilisées, et par conséquent un certain nombre d'algorithmes de routage a été proposé, en particulier pour les topologies maillées très appréciées.

En ce qui concerne les topologies de maillage, il est très facile de réaliser le chemin le plus court de routage dans une zone raisonnable, en utilisant une simple variation d'ordre de dimensions [28] tels que XY (ou XY Z pour 3-D NoC). Le routage XY routage assure un

deacklock-free et le livelock-free, mais il ne fournit pas une solution d'adaptabilité. Cet algorithme est une technique qui ne fait pas appel à une table de routage. C'est fait de sorte que chaque paquet soit acheminé d'abord dans la direction des X et après avoir atteint le même X de l'adresse de destination, il se déplace le long de la/les dimension (s) perpendiculaire (s). Une telle approche de routage fournit un certain nombre d'avantages sur d'autres implémentations alternatives :

- C'est le plus court chemin, et donc il minimise l'énergie dépensée par transfert d'unité d'information.
- Réduction des ressources d'algorithmes.

Un certain nombre d'algorithmes qui assurent le plus court chemin, partiellement adaptatifs sans inter blocage sont basés sur la restriction de certains tours comme mentionné dans la Sect. III.2.1.2. Ceux-ci sont West-First, Nord-Last, et Negative-First. Comme leurs noms l'indiquent l'indique, West-First exige qu'un paquet soit acheminé vers l'ouest comme première direction, si l'instruction aboutit à un résultat productif, sinon tout chemin plus court peut être pris. De même pour North-Last, et Negative-First. Tous ces régimes impliquent que certains tours soient interdits comme représenté sur la figure 6 ainsi qu'un exemple de possibles chemins de routage en utilisant les schémas ci-dessus.

Comme on peut le voir sur la figure III-1-a, le routage XY, en se limitant à quatre virages devient complètement déterministe. Les algorithmes de routage partiellement adaptatifs limitent seulement deux virages. Ouest d'abord, en exigeant que la direction de l'ouest soit prise d'abord si nécessaire, ceci rend tous les chemins les plus courts possibles quand un paquet doit se déplacer vers l'est, mais un seul chemin possible lorsqu'il va vers l'ouest (figure III-1-b), favorisant clairement la circulation vers la direction de l'Ouest.

De même pour Nord-Last (figure III-1-c), en exigeant que le trafic aille vers le nord en dernier, permet à tous les chemins les plus courts possibles en allant vers le sud, mais encore une fois, une seule voie pour la circulation en direction du nord est permise.
Et enfin, Negative-First (figure III-1-c) exige d'abord que dans le cas où la destination du paquet est vers un axe quelconque négatif, horizontal ou vertical, ainsi que toute autre direction, le paquet doit être acheminé en premier vers la direction de l'axe négatif et ensuite vers l'autre direction.

Toutes ces configurations impliquent que certains virages soient interdits.

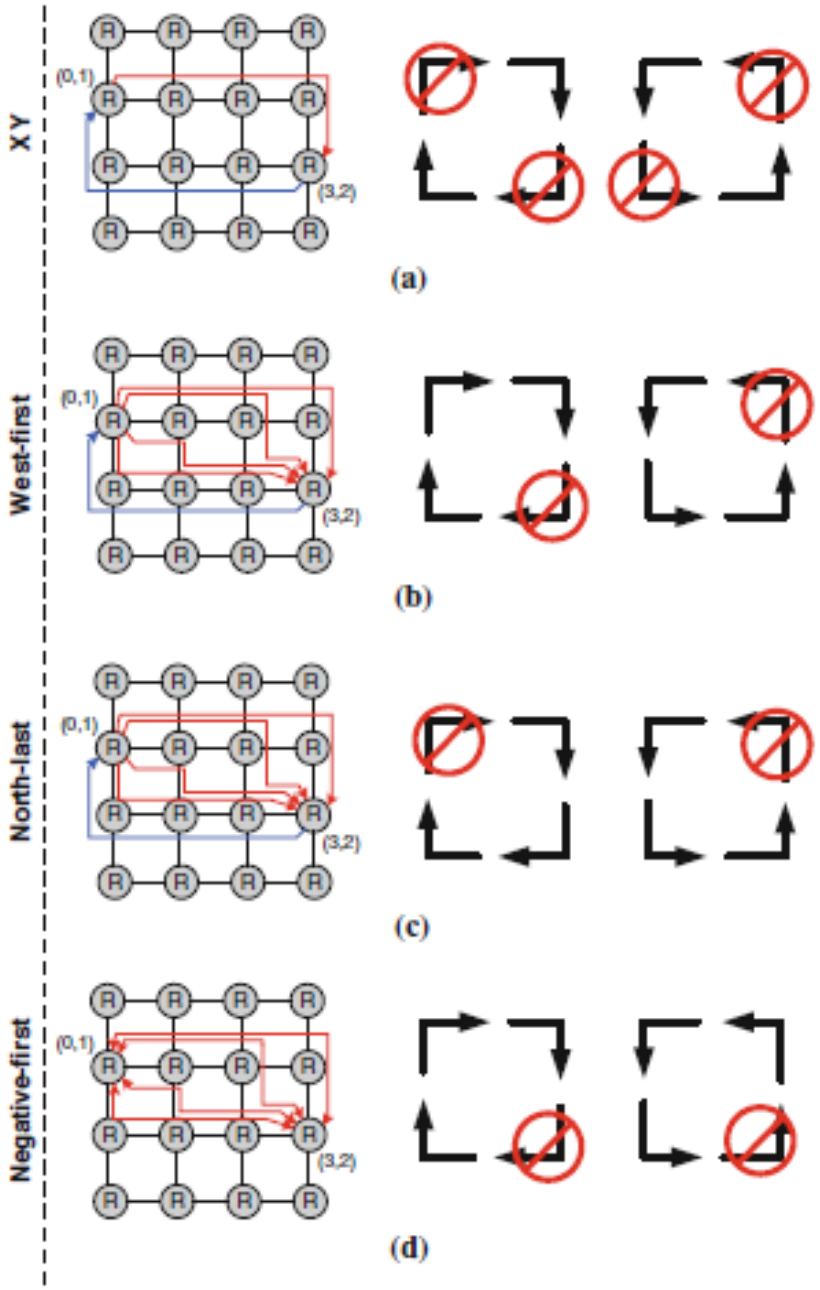


Figure III-1 : Illustration des algorithmes de routage

III.2.4 Algorithmes de routage pour les architectures irrégulières

Les algorithmes de routage décrits précédemment présentent deux inconvénients majeurs lorsqu'ils sont appliqués pour une architecture NoC. Plus précisément, certains d'entre eux (à savoir le routage XY) favorisent des chemins spécifiques de l'architecture NoC qui pourraient être plus utilisés, tandis que d'autres parties du dispositif peuvent être sous-utilisées.

En outre, le lay-out VLSI existantes sont irrégulières (en raison de la forme et la taille de la variabilité des modules VLSI lay-out), et donc une topologie de maillage ne peut être appliquée sans modifications significatives.

Afin de pallier à la limitation des algorithmes de routage courts en tournures, un nombre d'implémentations alternatives ont été étudiées.
On retrouve un algorithme de routage sans deadlocks pour des topologies irrégulières, appelé routage par couche. Cette approche regroupe des canaux virtuels en réseau couches et à chaque couche il assigne un ensemble limité d'adresses paires source / destination.
Une telle séparation du trafic donne une augmentation significative de l'efficacité de routage, mais aussi un équilibre de charge avec la garantie du plus court chemin sans pour autant exiger des caractéristiques spécifiques aux commutateurs autres que l'existence d'un bon nombre modeste de canaux virtuels.

La question de l'acheminement efficace des réseaux irréguliers a toujours posé problème. Une approche basée sur un acheminement avec des couches a été proposée [29]. Plus précisément, sur la base de cette approche, les paquets échappent les deadlocks possibles à travers une couche supérieure en faisant une transition à un niveau inférieur. Si la couche la plus basse est deadlock-free, alors l'ensemble du système le sera également.

Afin de soutenir les architectures NoCs avec des topologies irrégulières (à savoir, non maillée), le routage source (SR) et le routage distribué (DR) ont été employés. Comme mentionné, en ce qui concerne le routage source, chaque flit porte une séquence d'instructions d'acheminement, alors que dans le routage distribué la destination est recherchée à chaque routeur intermédiaire. En général, SR et DR nécessitent primordialement la table de routage. Pour la SR, les tables sont situées à chaque source (elles sont également indexées par les adresses de destination du paquet, mais elles contiennent des séquences de commandes d'acheminement, une pour chaque chemin de routage), tandis que pour la DR, les tables de routage se trouvent à chaque routeur (ils sont indexés par l'adresse de destination du paquet et contiennent des ports de sortie). Habituellement, les implémentations des tables de routage ont autant d'entrées que de nœuds dans le réseau. Toutefois, un tel scénario de communication, où un nœud peut communiquer à tout autre nœud est très peu probable, puisque l'ensemble réel de destinations utilisées à chaque source est généralement une petite fraction du nombre de nœuds. Cependant, même pour ces architectures avec des informations de routage limitées, la zone de tête des tables de routage n’est généralement pas négligeable.

Par conséquent, un des facteurs important pour optimiser dans de tels schémas de routage est la taille globale des tables de routage. Jusqu'à présent, un certain nombre d'approches ont été discutées afin de réaliser des économies de la région.

La performance des réseaux irréguliers peut encore être améliorée par des méthodes employant, comme le multiplexage de canal virtuel, adaptabilité, et le plus court chemin de routage associé à des chemins d'évacuation.

III.3 Contrôle de flux et Qualité de service (QoS)

Il est commun dans des applications réelles qu'on retrouve des contraintes de temps plus ou moins dures (délais) qui doivent être remplies pour que l'exécution de l'application soit appropriée. La garantie de ces contraintes est habituellement traitée comme étant la qualité de service (QoS) qui est une autre question importante dans la conception des architectures NoC [30]. La (QoS) garantie la conception et la validation indépendantes de toutes les parties du SoC en veillant à ce que exigences de l'application en temps réel soient et ce quel que soit les circonstances [31]. Des exemples typiques de garanties temporelles sont le débit, ainsi que les bornes de latence.

Sachant que l'architecture du NoC est essentiellement un support partagé, les protocoles de contention sont utilisés pour décider de la façon dont la bande passante est allouée à la demande. Si elle est mal gérée, l'encombrement peut se produire, un état dans lequel les tampons sont dépassés, des paquets sont refoulés et donc la QoS se voit dégradée.

Le contrôle de flux se réfère généralement aux politiques qui gèrent l'allocation des ressources à des paquets, et peut être considéré comme une solution pour les problèmes d'allocation de ressources ou de contention [23]. On retrouve une solution centralisée c.à.d. un contrôleur central prend les décisions. Dans un autre cas de figure, il existe une solution distribuée ou chaque routeur prend des décisions. Cette dernière est l'approche la plus commune dans les NOCs.

III.3.1 Contrôle de flux

Le contrôle de flux peut être intra-commutateur, commutateur vers commutateur.
Les systèmes de contrôle de flux commutateur à commutateur peuvent être classés en, avec tampon ou sans tampon. Dans le contrôle de flux sans tampon, les paquets sont déviés si nécessaire comme décrit dans la section des algorithmes de routage sans tampon.

Un protocole de contrôle de flux tamponné simple utilisé dans les NoCs est acknowledged/not acknowledged (ACK / NACK) [32]. L'approche ACK / NACK est fondamentalement un handshaking Protocole. Lorsqu'un émetteur met des données sur le lien, il active un signal VALID. Lorsque le récepteur est prêt à recevoir les données valides, il active le correspondant signal ACK. Si les données sont corrompues ou qu'il n'y a pas d'espace mémoire tampon pour les stocker, un signal NACK est activé à la place. À la réception d'un NACK, l'expéditeur commence la réémission des flits à partir de celui qui n'a pas été reconnu (Figure III-2). Ce système de contrôle de flux présente l'avantage qui réside dans le fait de supporter intrinsèquement une tolérance de panne, alors que son principal inconvénient est l'espace tampon supplémentaire nécessaire pour stocker les flits envoyés au cas où la retransmission est requise.

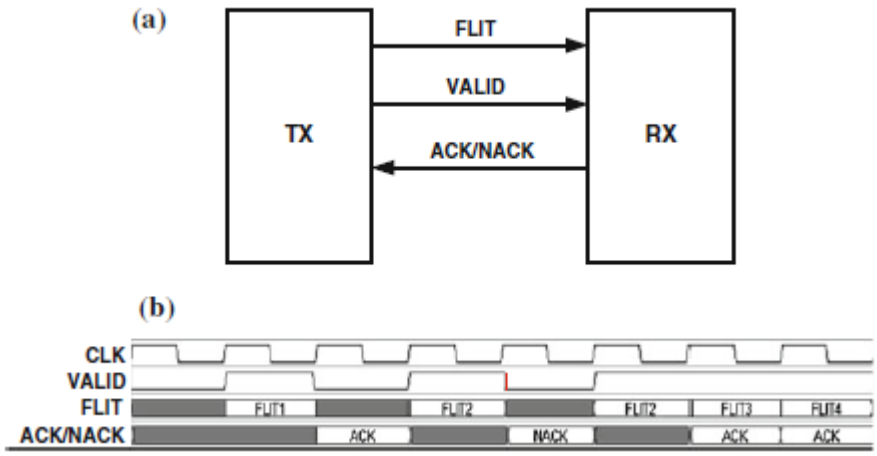


Figure III-2 : protocole de contrôle de flux ACK/NACK (a) signaux (b) chronogramme (Notez la retransmission du flit 2)

Lorsqu'un émetteur met des données sur le lien, il active le signal VALID lié. Quand le récepteur est prêt à recevoir les données validées, il active le signal ACK correspondant signal.

ON / OFF [23] est un autre protocole simple et STALL / GO est sa mise en œuvre la plus courante dans l'environnement NoC. Il nécessite seulement deux fils de commande : un Forward signifiant la disponibilité des données, et un Backward signalisant soit des tampons remplis « STALL » ou des tampons libres « GO » (Figure III-3).

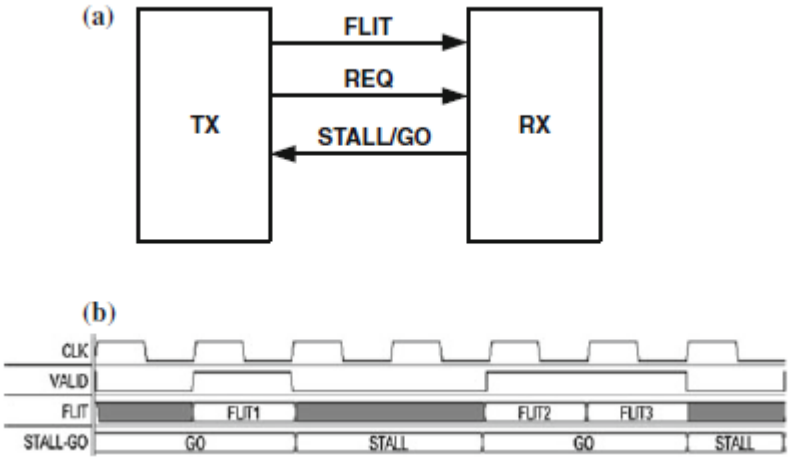


Figure III-3 : Protocole de contrôle de flux STALL / GO (a) signaux (b) chronogramme

T-Error [33] est un tout autre protocole beaucoup plus complexe visant à améliorer la performance.

Dans le contrôle de flux Credit-Based [23] l'émetteur a un compteur de « crédit » qui est initialisé à la valeur des intervalles de tampons vides du récepteur et décrémenté pour chaque flit envoyé. Le compteur de crédit doit être mis à jour au cas où le récepteur transmet un flit et augmente donc son espace tampon. Ceci est accompli quand une valeur de crédit est envoyée à l'émetteur afin d'être ajoutée à la valeur courante du compteur de crédit. L'émetteur suspen toute activité lorsque la valeur de crédit est égale à zéro et reprend lorsque sa valeur augmente à nouveau. Les signaux et le chronogramme du control de flux Credit-Based sont présentés sur la (Figure III-4).

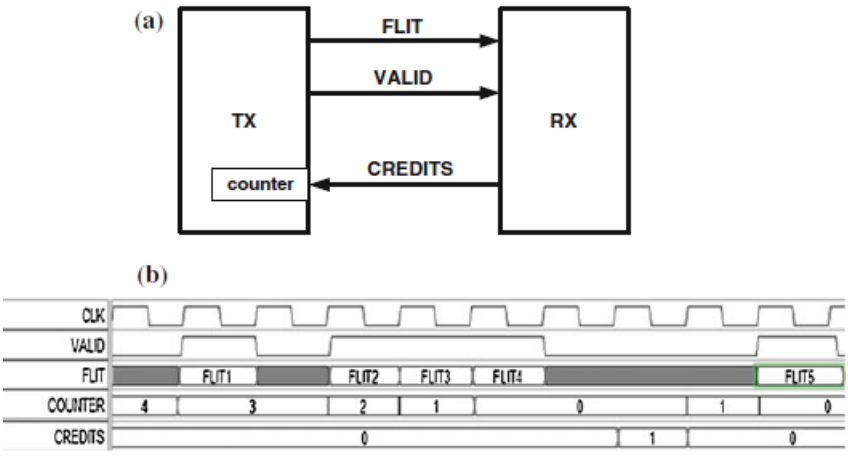


Figure III-4 : Protocole de contrôle de flux à base de crédit (a) signaux (b) chronogramme

Dans la figure III-5 on compare les protocoles de contrôle de flux ci-dessus en termes de performances, les exigences de mise en mémoire tampon, les exigences de la logique, et la tolérance aux pannes.

Flow control	Performance	Buffering	Logic	Fault tolerance
ACK/NACK	High for long messages	Very low	Low	Low
STALL/GO	Medium	High	Low	High
Credit-based	High	High	Low	High
T-error	High	High	Low	Low

Figure III-5 : comparaison des protocoles de contrôle de flux

III.3.2 Qualité de service (QoS)

Contrairement aux réseaux informatiques qui sont construits pour l'expansion continue, la croissance future et la compatibilité aux normes, les réseaux-sur-puces sont conçus et personnalisés pour un ensemble bien connu de ressources informatiques et de modèles de trafic. Par conséquent, les différents paramètres de conception de l'architecture du réseau (par exemple, taille du tampon, allocation de bande passante du lien, etc.) peuvent être personnalisés pour des applications spécifiques dans l'ordre de fournir une qualité de service requise (pour les modèles de trafic connus).

Les approches habituelles de QoS dans les architectures NoC sont Best Effort QoS et les services garantis QoS.

III.3.2.1 Best Effort (BE)

Les services BE (Best Effort) ne réservent pas de ressources, et donc ne fournissent aucune garantie, ce qui provoque également de petits retards. Même si cette approche utilise moins de ressources, il en résulte une communication très efficace du réseau puisque ce type de service est généralement conçu pour la moyenne des scénarios au lieu des pires et rares scénarios. Cependant, sa limitation réside dans son imprévisibilité.

III.3.2.2 Services garantis de QoS (GS)

Les NoCs avec les services garantis prennent des dispositions dans leurs architectures pour offrir des connexions avec des performances garanties, telles que l'absence de perte de données, un minimum de débit et une latence maximale.

CHAPITRE IV : NOCS ET SIMULATEURS EXISTANTS

Dans ce chapitre, on commencera par présenter quelques implémentations de réseaux sur puces et leurs caractéristiques. La deuxième section sera consacrée aux simulateurs les plus utilisés pour la conception avec une comparaison entre les points les plus importants.

IV.1 Les réseaux sur puces existants

Il existe actuellement un nombre important d'implémentations de réseaux-sur-puce aussi bien académiques destinés à la recherche, qu'industriels à vocation commerciale. Au-delà des détails d'implémentation de chaque réseau, il est tout aussi important de s'intéresser aux outils et méthodologies de conception qui y sont associés.

IV.1.1 ARTERIS

http://www.arteris.com/ ARTERIS ne représente pas une simple implémentation de réseau-sur-puce, c'est plutôt une suite d'outils développés par la compagnie (ARTERIS). Nous avons choisi d'exposer cet outil de conception de réseaux-sur-puce du fait que ce soit le premier à avoir été commercialisé, bien que cette vocation purement commerciale rende l'obtention de documentation technique quasiment impossible.

Cette suite comprend deux outils d'aide à la conception et à l'exploration architecturale «NOCexplorer » et « NOCcompiler ». Le premier outil va permettre la construction d'une topologie de réseau à partir d'un cahier des charges, où sont spécifiés les besoins en performances de l'application visée ainsi que les limites de coût et de surface pouvant être admis. Cet outil est basé sur un modèle de simulation SystemC du routeur ARTERIS reprenant les mécanismes de routage, de bufférisation ainsi que les mécanismes de contrôle de flux et de qualité de service. Les rapports de performances incluant les débits aux entrées/sorties ainsi que la latence sur le réseau sont produits à chaque simulation à partir des modèles de trafic synthétiques. Le concepteur a la possibilité de modifier ensuite manuellement la topologie donnée en entrée en fonction des performances obtenues afin d'arriver à la meilleure adéquation (besoins/couts) possible. L'intérêt majeur de ce genre d'outils est de permettre la modélisation et la simulation de plusieurs instances de réseaux-sur-puce de façon visuelle et très rapide, ce qui se traduit par un gain substantiel en temps de conception.

Il est à noter ici que la liberté donnée à l'utilisateur de choisir une topologie arbitraire écarte la possibilité d'utiliser des algorithmes de routage algébriques et impose l'utilisation de tables

de routage au niveau de chaque routeur. Une fois la topologie fixée « NoCcompiler » va permettre de générer une description matérielle du réseau.

IV.1.2 STNOC

Le réseau STNOC est un réseau à commutation de paquets, il appartient à la société STMicroelectronics. L'originalité du STNOC réside dans sa topologie en Spidergon (figure IV-1), topologie régulière supportant des algorithmes de routage minimalistes très simples nécessitant un minimum de ressources matérielles. Le point fort de la topologie Spidergon découle de l'encombrement minimal des interconnexions qu'elle engendre. En effet quelle que soit la taille du réseau STNOC il est toujours possible de le transformer en une implémentation planaire avec un minimum de croisements entre les liens. Des interconnexions fortement encombrées avec beaucoup de croisements nécessitent plus de surface sont à l'origine d'une surconsommation électrique. Une des autres caractéristiques de la topologie est qu'elle peut être adaptée aux besoins de l'application ciblée en retirant des liens inutilisés sans pour autant modifier les algorithmes de routage, ce qui ajoute un degré de flexibilité non négligeable. Le STNOC utilise une approche se basant sur une séparation stricte entre la logique interne des routeurs et les liens servant au transport des paquets. En plus du support implicite du mode GALS, cette approche donne la possibilité d'avoir plusieurs configurations de liens en fonction des besoins en performance. En effet, selon les contraintes physiques, il est possible d'utiliser des liens séries ou parallèles, et de choisir la fréquence de sérialisation/fonctionnent en fonction des débits souhaités indépendamment de la fréquence utilisée par la logique interne des routeurs.

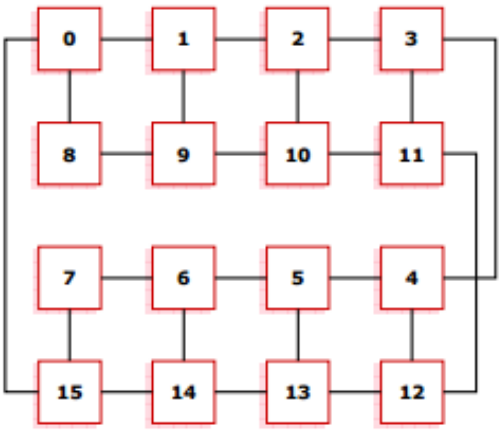


Figure IV-1 : Exemple de topologie Spidergon d'un STNOC

IV.1.3 PROTEO

À l'Université de Technologie de Tampere (TUT), la première architecture de NoC est appelée Proteo et a été conçue pour être simple et peu couteuse dans sa mise en œuvre. Dans la poursuite de la simplicité de la mise au point dans le développement de Proteo, une topologie en anneau a été préférée. Grace à sa simplicité architecturale, Proteo offre des performances plus élevées que les architectures les plus complexes.

La performance offerte par l'architecture Proteo a été obtenue au détriment de la programmabilité et la qualité de service (QoS). Pour compenser cela, des outils méthodologiques et des logiciels de conception ont été conçus afin d'aider à l'intégration de toutes les connaissances sur les exigences de l'application et de la communication dans la conception du réseau.

Ainsi, toutes les réalisations de Proteo sont biaisées vers les besoins de l'application cible. Cela fait de Proteo une application spécifique et moins appropriée pour la conception de la plate-forme à moins que toutes les applications cibles puissent être prévues au moment de la conception, ce qui est peu probable. Proteo a également été conçu uniquement pour la communication globale avec des bus de manipulation de la communication locale. L'approche découle de la conception de David Sigüenza-Tortosa, qui est le principal concepteur de l'architecture Proteo.

IV.1.4 HERMES

Hermes est un réseau avec stratégie de bufférisation Wormhole et un routage X-Y sur une grille 2D dont la taille peut être fixée en fonction des préférences de l'utilisateur. Divers paramètres peuvent être modifiés lors de la phase de conception, tels que la largeur des flits, la longueur des paquets, ou encore la profondeur des buffers internes. Le réseau est conçu pour supporter uniquement le mode de fonctionnement synchrone. C'est d'ailleurs l'une de ses principales limitations.

Le NoC Hermes est abordé avec plus de détails dans le chapitre suivant puisqu'il fera l'objet de notre étude.

IV.1.5 MANGO

Le concept MANGO développé à l'Université technique du Danemark (DTU) est peut-être l'approche académique la plus largement connue pour les NoCs nécessitant moins d'horloges.

Il promet un service garanti (GS) avec un fonctionnement asynchrone. MANGO fournit de multiples canaux virtuels (VCs) sur les liens, ce qui permet d'éviter les blocages inter-canaux, mais imposent un contrôle des frais généraux ainsi que l'établissement de la zone de commutation de manière drastique.

IV.2 Simulateurs existants

Au-delà des détails d'implémentation de chaque réseau, il est tout aussi important de s'intéresser aux outils et méthodologies de conception qui y sont associés.

IV.2.1 WORMSIM

WormSim [https://github.com/openworm/org.wormsim.frontend] est un simulateur à cycle précis développé avec flexibilité et modularité en C ++ en utilisant la bibliothèque de modèle standard. Il peut être utilisé pour simuler un large éventail d'architectures NoC (par exemple, des NoCs avec différentes topologies et différents algorithmes de routage, etc.), en utilisant les paramètres de performance contrôlables (taille du canal tampon, retard de routage, retard d'arbitrage de la barre transversale, etc.). Pour des raisons de flexibilité, WormSim peut être facilement étendu pour simuler des NoCs qu'il ne supporte pas au stade actuel.

Le générateur de trafic intégré de WormSim permet aux utilisateurs de simuler le système sous plusieurs modèles de trafic populaires, tels que le trafic uniforme, modèle de trafic de point chaud, modèle transpose, etc. Ce simulateur fournit également un moyen efficace pour permettre à l'utilisateur de spécifier les conditions de circulation arbitraire pour le Noc en simulation. Plus précisément, l'utilisateur a le contrôle sur le taux de génération de paquet au niveau de chaque nœud, la taille du paquet et de sa distribution, etc. Même plus, il permet à l'utilisateur d'attacher un fichier de traçage pour chaque nœud IP individuel de sorte que le système à la simulation puisse imiter l'état de la circulation exacte des applications réalistes en réutilisant les fichiers de traçage extraits de ces applications.

Mise à part la présentation des données de performance après simulation, WormSim permet également à l'utilisateur de collecter les données de consommation d'énergie de communication dans la simulation. Il supporte actuellement deux modèles de puissance. Le modèle Ebit et le modèle de puissance Orion.

WormSim utilise une interface de lignes de commande simple pour configurer les différents paramètres pour les systèmes sous simulation. Les paramètres configurés dans la ligne de commande sont, dans un sens global, à l'ensemble du NoC sous simulation. Une façon plus efficace pour la configuration du système est assurée en utilisant un fichier de configuration. Cela permet à l'utilisateur de contrôler les paramètres au niveau de chaque routeur ou bloc IP.

IV.2.2 TOPAZ

TOPAZ est un simulateur de réseau d'interconnexion à usage général qui permet la modélisation d'une grande variété de routeurs (de la barre transversale tamponnée au routeur rotatif), avec différents compromis entre la vitesse et la précision. La conception de l'outil est orientée objet et sa mise en œuvre est en langage C ++.

Dans TOPAZ, chaque réseau est construit hiérarchiquement. Le simulateur construit le réseau et les liaisons (topologie), le réseau construit tous les routeurs, et enfin les routeurs construisent leurs composantes intrinsèques et leurs interconnections. Chaque structure simulée est associée à deux classes C ++. Les composants et les flux. Les composants sont descriptifs, caractérisant chaque structure et sa relation avec les autres composants du système. Les flux établissent la façon dont le flux de l'information se déplace à l'intérieur du composant. A titre d'exemple, pour une structure de mémoire tampon, le composant détermine la taille et le nombre de ports ou encore le délai, alors que le flux détermine son comportement en fonction de la régulation du débit sélectionné. Au cours de la phase d'exécution, tous les composants du système sont itérativement visités.

TOPAZ est un outil de simulation à base de temps, certains flux peuvent être construits en interne comme des machines à états finis, ce qui rend ces composants dépendants d'évènements et non pas de temps (event-driven and not time-driven).

Le simulateur prend en charge l'exécution en parallèle.

IV.2.3 NOXIM

[https://github.com/davidepatti/noxim] Ce simulateur a été développé en utilisant le système C, il fournit une interface de lignes de commande pour définir plusieurs paramètres d'un NoC. En particulier, l'utilisateur peut personnaliser la taille du réseau, la taille de la mémoire tampon, la distribution de la taille des paquets, l'algorithme de routage, la stratégie de sélection, le taux

d'injection de paquets, la distribution temporelle du trafic, le motif de la circulation et la distribution du trafic.

Le simulateur permet d'évaluer le NoC en termes de débit, de retard, et de consommation d'énergie. Une telle information est fournie à l'utilisateur en termes de résultats moyens et par communication. Dans le détail, l'utilisateur est autorisé à recueillir différents paramètres d'évaluation, y compris le nombre total reçu de paquets/flits, débit global moyen, retard max / min, consommation totale d'énergie.

IV.2.4 NIRGAM

NIRGAM [http://nirgam.ecs.soton.ac.uk/] est un simulateur à base de system C à évènement discret et à simulateur de cycle précis. Il apporte un soutien important à expérimenter avec la conception du NoC en termes d'algorithmes de routage et d'applications sur diverses topologies. NIRGAM modélise un NoC comme une interconnexion à 2 dimensions de carreaux. Chaque carreau est composé d'un noyau connecté à un routeur / commutateur par un canal noyau bidirectionnel. Un carreau est relié aux autres voisins par des canaux bidirectionnels.

Les paramètres du NOC configurables sont la taille de la topologie (m x n), la fréquence d'horloge, la profondeur de la mémoire tampon, la taille de flit et les canaux virtuels. Les algorithmes de routage actuellement pris en charge par ce simulateur sont : XY – paire, impaire et le routage de source.

IV.2.5 SICOSYS

SICOSYS un simulateur de réseau d'interconnexion à usage général qui permet de modéliser une large variété de routeurs de messages d'une manière précise. Les résultats sont très proches de ceux obtenus en utilisant des simulateurs de matériel mais à un coût de calcul plus faible. Afin de rendre l'outil facilement compréhensible, extensible et réutilisable, la conception de l'outil est orientée objet et sa mise en œuvre est en langage C.

SICOSYS a été développé en tenant compte de la modularité, la polyvalence et la connectivité avec d'autres systèmes. En fait, il est maintenant possible de faire fonctionner SICOSYS avec des simulateurs de systèmes complets comme RSIM et SIMOS. Les modèles

utilisés sont conçus pour ressembler à des réalisations matérielles dans certains aspects, tout en maintenant la complexité aussi faible que possible.

De cette manière, le simulateur simule la structure matérielle des routeurs au lieu de simplement mettre en œuvre leur fonctionnalité. Afin de comparer les routeurs et les réseaux contre les autres alternatives et pour permettre au système de suivre les nouveaux développements tout en présentant une interface utilisateur homogène, la conception du simulateur donne beaucoup d'importance à son extensibilité et la simplicité de l'interface utilisateur. Ainsi, SICOSYS a une collection de composants à inspiration matérielle comme les multiplexeurs, des tampons ou des cross bars. Les routeurs peuvent être construits en connectant des composants les uns aux autres, car ils sont dans la description matérielle. Ensuite, les routeurs sont connectés d'une certaine façon pour former un réseau.

Tout cela est défini dans SGML qui peut être considéré comme un sur ensemble de HTML. Cela permet à SICOSYS de gérer des descriptions hiérarchiques de routeurs facilement tout en gardant une lisibilité des fichiers de configuration.

IV.3 Comparaison des simulateurs

Simulator	Framework	Topologies	Comments
Atlas	Java, SystemC, HDL	Hermes	Synthesizable
HNOCS	OMNeT++	All	Parallelism
Netmaker	SystemVerilog	-	Synthesizable
NiRGAM	SystemC	All	
NNSE	SystemC	Mesh, Torus	
Noxim	SystemC	Mesh	
OCCN	C++, SystemC	-	
ORION	C	-	power, area
SICOSYS	C++	Mesh with constant transfer time	
TOPAZ	C++	2-d/3-d mesh/torus, midimew, ring	

Figure IV-2 : Comparatif des différents NoCs

IV.4 Conclusion

Après avoir vu les différents Nocs et simulateurs existants, il nous semble judicieux de choisir le NoC HERMES pour deux raisons primordiales, sa disponibilité, mais aussi et surtout le fait que les codes utilisés soient synthétisables avec une particularité d'accès libre, il est ainsi possible d'ajouter et/ou modifier certaines fonctionnalités directement sur les sources HDL du réseau.

D'autre part, le seul choix possible de simulateur en vue d'assurer une certaine compatibilité avec le NoC réside dans l'utilisation d'ATLAS qui est à cet effet développé par la même équipe de conception d'HERMES.

CHAPITRE V : MISE EN ŒUVRE DU NOC HERMES

V.1 Introduction

Nous allons tout d'abord présenter le NoC Hermes et son environnement logiciel pour la génération de la structure du NoC, les plateformes d'émulation générées pour enfin réaliser quelques expérimentations. Ces expérimentations permettront de proposer une structure de NoC pour l'algorithme d'authentification d'œuvres d'art en fin de chapitre.

V.2 HERMES

Le NoC Hermes est développé par l'équipe de F. Moraes au PUCRS (Pontificale Université Catholique de Rio Grande do Sul, Brésil) depuis 2003. Hermes est un NOC flexible, simple, de faible coût et qui satisfait pleinement l'implémentation dans un circuit FPGA en assurant la communication entre les différents modules pour la conception de SoCs.

V.2.1 Topologie

Le NoC Hermes possède une topologie maillée à deux dimensions. Chaque nœud est identifié grâce à ses coordonnées XY, où X représente sa position horizontale et Y sa position verticale (Figure V-1). Les origines des coordonnées se trouvent en bas à gauche.

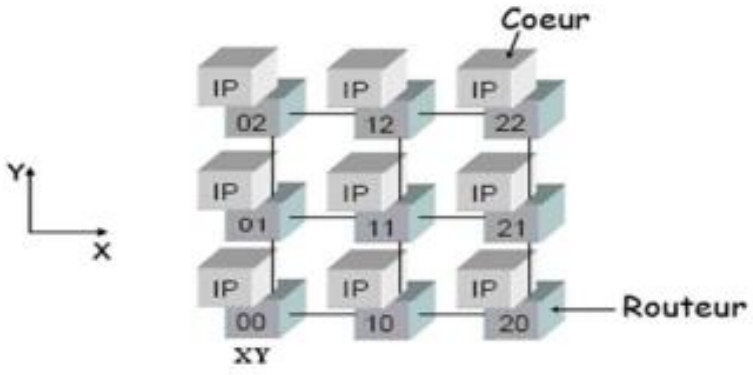


Figure V-1 : Exemple d'un NoC Hermes a topologie maillé 2D (3x3)

V.2.2 Modèle du nœud HERMES

Structure

Chaque nœud du réseau est constitué d'un routeur relié à un ou plusieurs blocs IP par l'intermédiaire de NI.

Le NOC Hermes intègre des routeurs, chaque routeur contient cinq ports (chaque port ayant un buffer d'entrée) bidirectionnels qui sont :

• Un port bidirectionnel qui relie chaque routeur au cœur, appelé le port « local ». Ce port sert à connecter les routeurs aux différents blocs IPs utilisés.

• Quatre ports bidirectionnels au maximum connectés vers les routeurs voisins appelés Nord, Sud, Est, Ouest pour assurer les échanges de données entre les routeurs voisins. L'architecture du routeur dépend de sa position dans le réseau.

Ceci est représenté dans la figure V-2

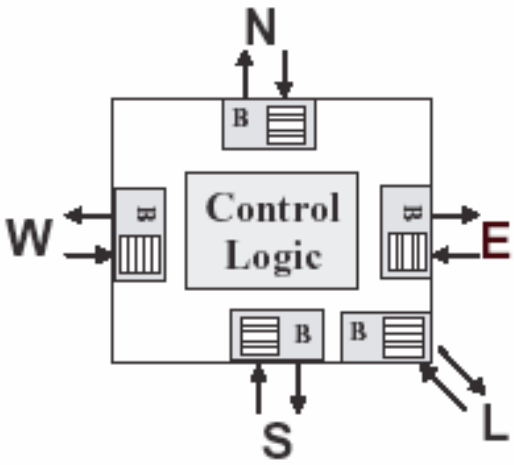


Figure V-2 : Modèle du nœud de HERMES

La communication entre les routeurs est asynchrone, alors que dans les routeurs les transactions sont synchrones (Globalement Asynchrone Localement Synchrone).

Le routeur est divisé en trois parties :

1. La logique de routage.
2. La logique d'arbitrage.
3. Les ports de communication.

Les messages arrivent sur les ports d'entrées alors une requête est envoyée à la logique d'arbitrage. Pour choisir le port d'entrée à servir, cette logique utilise un arbitrage à priorité tournante (round-robin) pour organiser l'accès aux ports. Puis, il passe une requête à la logique de routage en indiquant le port par lequel le message à router est arrivé.

Un flit reçu ou bloqué sera stocké dans un buffer (ceci limite la chute des performances). Des buffers à FIFO paramétrables sont utilisés pour chaque port d'entrée.

V.2.3 Influence de la profondeur des buffers

Tout comme pour la taille de l'unité de transfert, le choix judicieux des tailles de buffers optimise les ressources utilisées par le NoC car ce sont les organes nécessitant le plus de surface. En règle générale, il faut savoir que la taille des buffers influence la charge maximale pouvant être acceptée sur le réseau et à une influence moindre sur le débit maximal, car ce dernier paramètre dépend aussi des conditions de congestion. La latence quant à elle n'est pas influencée par la taille des buffers sous les conditions normales de trafic.

Il existe des méthodes plus ou moins formelles qui partant d'une description d'un graphe des tâches, permettent de calculer une distribution de l'espace de bufférisation, d'autres approches notamment considèrent un ensemble de métriques plus complet (énergie et latence en plus de la taille des buffers).

V.2.4 Constituants

Le modèle de référence OSI est une structure hiérarchique de sept couches qui définissent les exigences pour la communication entre les éléments de traitement. Chaque couche offre un ensemble de services à la couche supérieure, en utilisant les fonctions disponibles dans la même couche et dans celles encore plus basses. Ces couches sont décrites ci-dessous pour le contexte de notre NoC Hermes

Couche physique : Cette couche concerne les détails de transmission de données sur un support physique. Elle définit les spécifications électriques et physiques des données de connexion (Ex : nombre de fils / bits qui relient chaque commutateur).

Couche liaison : Cette couche est responsable de la transmission fiable des données sur la liaison physique. Il peut encapsuler la détection d'erreurs et la correction des codes.

Couche réseau : Cette couche fournit une vue de la communication bout-à-bout indépendamment de la topologie et inclut la stratégie de commutation et les algorithmes de routage. Elle fournit le bloc fonctionnel et procédural qui transfère la longueur arbitraire des séquences de données (Ex : commutateur, crossbar)

Couche de transport : Cette couche est responsable de cacher toutes les informations en dessous et d'assurer le contrôle de flux, la segmentation des paquets de segmentation et l'ordonnancement des messages. Elle agit comme un agent intermédiaire qui enveloppe et produit une abstraction des couches inférieures au bloc IP (Ex : Resource Network Interface (RNI)).

V.2.5 Protocole d'HERMES :

Il existe plusieurs variétés du NoC hermès et elles sont toutes supportées par notre simulateur. Donc, il serait plus judicieux de les présenter pour aborder la particularité de chacune.

HERMES TB

Le Hermes-TB est une infrastructure paramétrable utilisée pour mettre en œuvre un réseau 2D tore bidirectionnel. Pour le moment, Hermes-TB ne supporte pas les canaux virtuels et un seul algorithme de routage est disponible.

Le routeur Hermes-TB a une logique de commande de commutation centralisée et cinq ports bidirectionnels. Le port local établit une communication entre le routeur et un noyau local. Les autres ports du routeur sont connectés aux routeurs voisins. Chaque port d'entrée comporte un tampon de profondeur d, pour le stockage temporaire des flits. L'algorithme de routage est une version modifiée du modèle de West-First algorithme. L'implémentation actuelle est adaptative (comme prévu) et minimale.

Plusieurs paquets peuvent arriver simultanément dans un routeur donné. Une subvention d'arbitrage centralisée autorise l'accès aux paquets entrants. Si la demande de paquets entrants est accordée par l'arbitre, l'algorithme de routage est exécuté pour connecter le port d'entrée au port de sortie correct.

HERMES TU

Le Hermes-TU est une infrastructure paramétrable utilisée pour mettre en œuvre un réseau 2D tore unidirectionnel.

Celui-ci emploie deux canaux virtuels pour éviter une impasse (deadlocks) et un seul algorithme de routage est disponible.

Le routeur Hermes-TU dispose d'une logique de commande de commutation centralisée, quatre ports unidirectionnels et un port local bidirectionnel. Le port local établit une communication entre le routeur et un noyau local. Les ports unidirectionnels sont connectés aux routeurs voisins. Chaque port d'entrée comporte un tampon de profondeur d, pour le stockage temporaire de flit.

Plusieurs paquets peuvent arriver simultanément dans un routeur donné. Une subvention d'arbitrage centralisée autorise l'accès aux paquets entrants. Si la demande de paquets entrants est accordée par l'arbitre, l'algorithme de routage est exécuté pour connecter le port d'entrée au port de sortie correct.

HERMES SR

Le Hermes Source Routing (Hermes-SR) est une infrastructure paramétrable actuellement mise en œuvre dans l'Atlas comme un maillage 2D. Par rapport à la mise en œuvre de Hermes, celle-ci se différencie avec deux caractéristiques principales (l'approche de routage et le mécanisme d'arbitrage).

Un routeur Hermes-SR est essentiellement composé par un ensemble de ports d'entrée et de sortie. En plus de leurs fonctions de stockage et de contrôle du flux, le port d'entrée met en place un schéma de routage tandis que le port de sortie exécute l'algorithme d'arbitrage.

HERMES CRC

Le Hermes CRC est une extension d'Hermes qui utilise un code de bloc linéaire pour protéger le NoC contre les défauts transitoires. Trois options de code de bloc sont disponibles : Lien CRC, Source CRC et Hamming.

Ce NoC a une option de saboteur qui détermine si les défauts sont injectés lors de la simulation. Lorsque cette option est sélectionnée, l'utilisateur doit choisir au moins un type d'injection de défaut : Rising Delay, Falling Delay, Glitch négatif et / ou Glitch positif.

Pour le moment, Hermes CRC implémente seulement l'algorithme de routage XY, utilise le contrôle de flux à base de crédit et ne supporte pas les canaux virtuels.

V.2.6 L'ordonnancement

La commutation par ver de terre (Wormhole) caractérise le NOC Hermes. Elle offre une faible latence, une faible demande en mémoire, et le canal physique peut être divisé en plusieurs canaux logiques. Dans l'ordonnancement par ver de terre, chaque message est divisé en plusieurs morceaux appelés « flits » dont la taille est paramétrable. Seul le premier flit est routé et les autres flits le suivent. Pour indiquer la fin du message, deux solutions existent. La première est d'avoir un flit EOM (End Of Message) à la fin du message. La deuxième consiste à avoir un flit contenant le nombre de flit constituants le message. Le réseau HERMES utilise la deuxième solution.
Les flits reçus ou bloqués sont stockés dans un buffer (ceci limite la chute des performances). Le stockage des messages affecte tout le réseau. Des buffers à FIFO paramétrable sont utilisés pour chaque port d'entrée.

V.2.7 Algorithme de routage

L'algorithme de routage définit le chemin suivi par un paquet entre le routeur source et le routeur cible.

C'est un algorithme de routage statique de type *X_Y* signifiant que les paquets de données sont dirigés vers le *PE* destinateur à travers les nœuds du réseau d'abord selon l'axe des abscisses *X* du réseau, puis selon son axe des ordonnées *Y*.

La modélisation de l'algorithme de routage *X_Y* est réalisée en *VHDL* selon l'algorithme suivant :

- If X_routeur < X_destination, direction paquets = Direction_Est
- If X_routeur > X_destination, direction paquets = Direction_Ouest
- If X_routeur = X_destination et Y_router > Y_ destination, direction paquets = Direction_Sud
- If If X_routeur = X_destination et Y_router < Y_ destination, direction paquets = Direction_Nord
- If If X_routeur = X_destination et Y_router = Y_ destination, direction paquets = Local_PE

La figure V-3 illustre l'algorithme de routage *X_Y* d'un paquet entre les *PE_00* et *PE_22*. Une mauvaise description de cet algorithme peut faire apparaître au cours de la simulation du *NoC* des inter blocages (*Deadlock*) ou des bouclages (*Livelock*) de paquets.

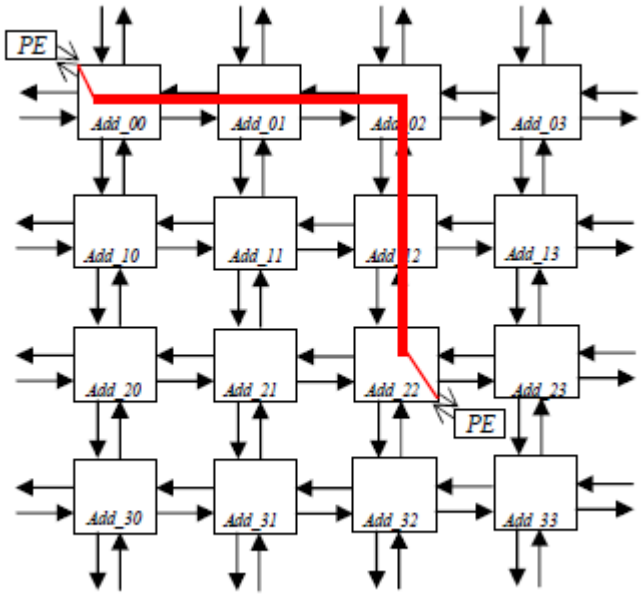


Figure V-3 : Illustration de l'algorithme X_Y

V.2.8 Fonctionnement du nœud

Le nœud Hermes a été décrit en VHDL et validé par simulation fonctionnelle. La figure V-4 présente quelques blocs internes du nœud et les signaux des deux ports (local et East). La Figure V-5 présente une simulation fonctionnelle pour les signaux les plus importants de la figure 4. Les étapes de simulation sont décrites ci-dessous, où la numérotation a des correspondances dans les Fig. 4 et 5.

1. Le commutateur (xLyL = 00) reçoit un flit par le port local (indice 4), le signal rx est affirmé et le contenu des données est dans le data_in.
2. Le flit est stocké dans la mémoire tampon et le signal ACK_rx est émis indiquant que le flit a bien été reçu.
3. Le port local demande le routage à la logique d'arbitrage en affirmant le signal h.
4. Après la sélection d'un port, la logique d'arbitrage émet une demande à la logique de routage. Ceci est accompli par l'envoi de l'entête du flit qui n'est autre que l'adresse cible (valeur 11) et la source de la demande d'entrée (signal *incoming*, la valeur 4, ce qui représente le port local), ainsi que la demande elle-même.
5. L'algorithme de routage XY est exécuté, la table de commutation est écrite, et le signal ack_rot est affirmé indiquant que la connexion a été établie.
6. La logique d'arbitrage informe le buffer que la connexion a été établie et que le flit peut maintenant être transmis.

7. Le commutateur confirme le signal Tx du port de sortie sélectionné et met les données du flit sur le signal de ce même port.

8. Une fois que le signal ack_tx est confirmé le flit est retiré du buffer et le prochain flit stocké peut être traité.

9. Ce deuxième flit démarre le compteur indiquant le nombre de cycles d'horloge après lequel la connexion doit être fermée.

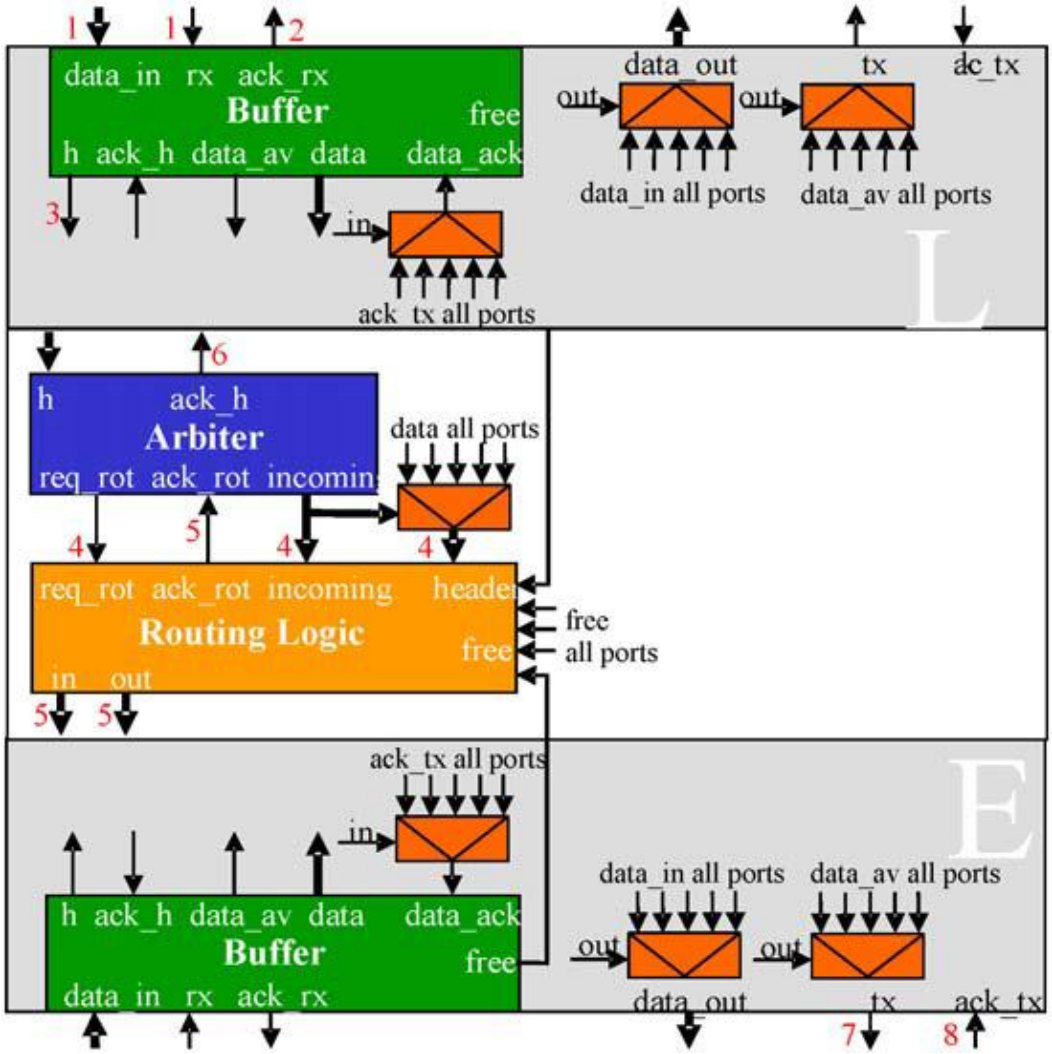


Figure V-4 : Diagramme partiel de deux ports (Local et Non Local) du nœud HERMES

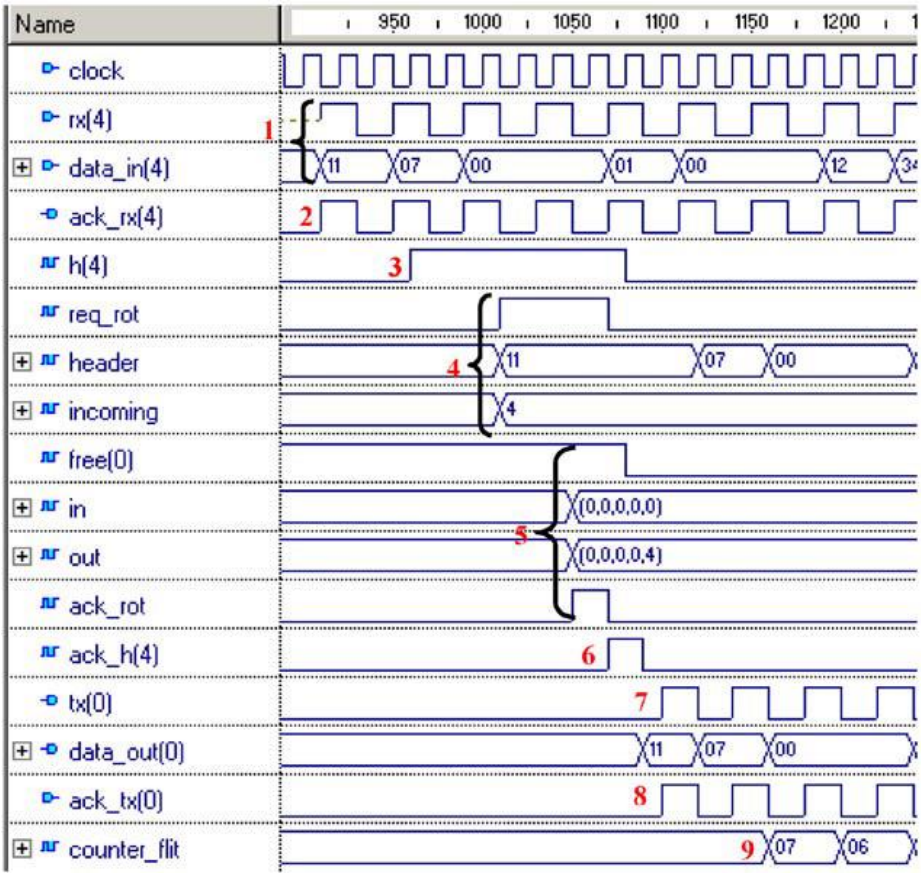


Figure V-5 : Simulation de la connexion entre deux ports (Local et non Local)

V.3 ATLAS

L'environnement ATLAS est entièrement dédié à la génération et à l'évaluation des performances du NoC Hermes. ATLAS permet la génération de l'architecture NoC en VHDL synthétisable ainsi que la simulation des performances en SystemC au moyen de blocs synthétiques de trafic. Il automatise les différents processus liés au flow de conception pour certains NoCs proposés par le Groupe GAPH.

V.3.1 Structure

Actuellement, le flux de conception est composé des outils suivants : génération du NoC - génération de trafic, simulation, évaluation de performance – évaluation de puissance). La figure V-6 suivante présente la fenêtre principale de l'interface graphique de l'environnement ATLAS qui permet de faire appel aux outils cités précédemment.

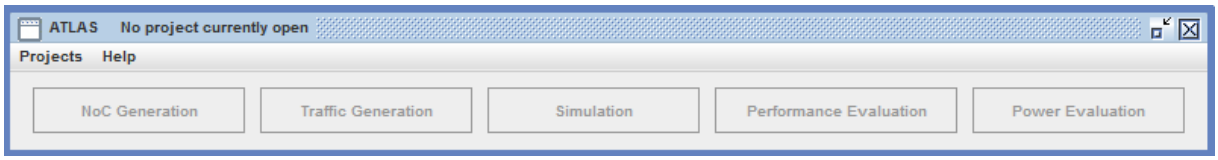


Figure V-6 : Fenêtre principale de l'interface graphique

Descriptif des outils d'ATLAS

Génération du NoC : ça concerne les paramètres de NoC (par exemple, la bande passante du canal, la profondeur de la mémoire tampon, le nombre de canaux virtuels, les stratégies de contrôle de flux).

Génération de trafic : relatant de la circulation pour caractériser les applications qui s'exécutent sur le NoC.

Simulation : L'outil de simulation NoC invoque un simulateur VHDL externe : ModelSim (simulation RTL). Tous les fichiers générés de la circulation sont interprétés et injectés dans le NoC. Pendant la simulation, les fichiers de sortie générés sont lus par le module d'analyse du trafic. Dans cette étape se produit la communication efficace entre les noyaux.

Evaluation des performances : Il est possible de générer des graphiques, des tableaux, des cartes et des rapports pour aider à l'analyse des résultats obtenus.

Evaluation de puissance : utilise un modèle d'estimation (à base d'équations mathématiques) pour générer des résultats de consommation de puissance.

La figure V-7 donne plus de détails sur la fonction de chaque outil et le design flow :

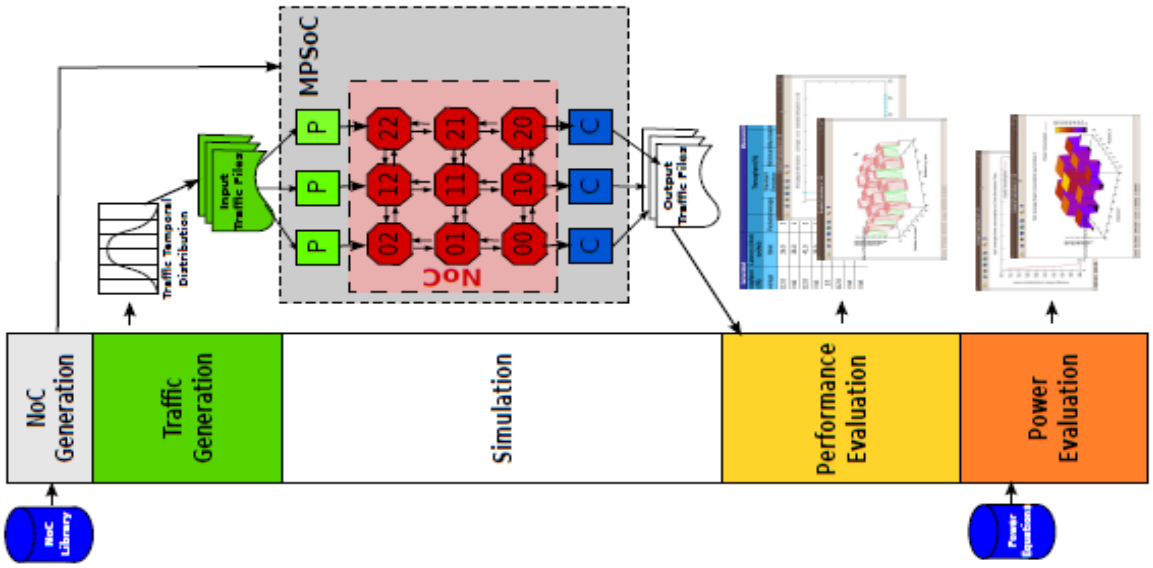


Figure V-7 : Design flow de l'outils ATLAS

V.3.2 Fichiers produits

Les fichiers VHDL synthétisables obtenus sont les suivants :

· Un package contenant tous les types et sous-types permettant de dimensionner le NoC ;

· Une entité décrivant le fonctionnement des ports d'entrée et de leurs buffers ;

· Une entité décrivant le fonctionnement du bloc logique (arbitre et aiguilleur) et donc l'algorithme de routage.

· Des entités pour chaque type de routeur : routeurs centraux, routeurs en bas à droite, à gauche, en haut à droite.

· L'entité globale du NoC.

On retrouve aussi des fichiers décrits en systemC destinés au benchmarking comprenant des stratégies de test et d'évaluation de performances.

V.3.3 Evaluation de performances

Débit de données

Le débit de données reflète les capacités d'écoulement de trafic du réseau en termes de quantité de données par unité de temps. Bien que plus ou moins lié à la notion de latence de données, il est important de bien distinguer les deux métriques.

Tout comme pour la latence, le débit sur un réseau sur puce comporte deux composantes, l'une statique et l'autre dynamique. Quant au débit statique (Octet/seconde) il correspond à la quantité de données maximale pouvant circuler entre deux nœuds par unité de temps, et dépend principalement des caractéristiques physiques des liens (fréquence de fonctionnement, largeur de données, schéma de sérialisation) mais prend en compte aussi les temps de routage et de bufférisation.

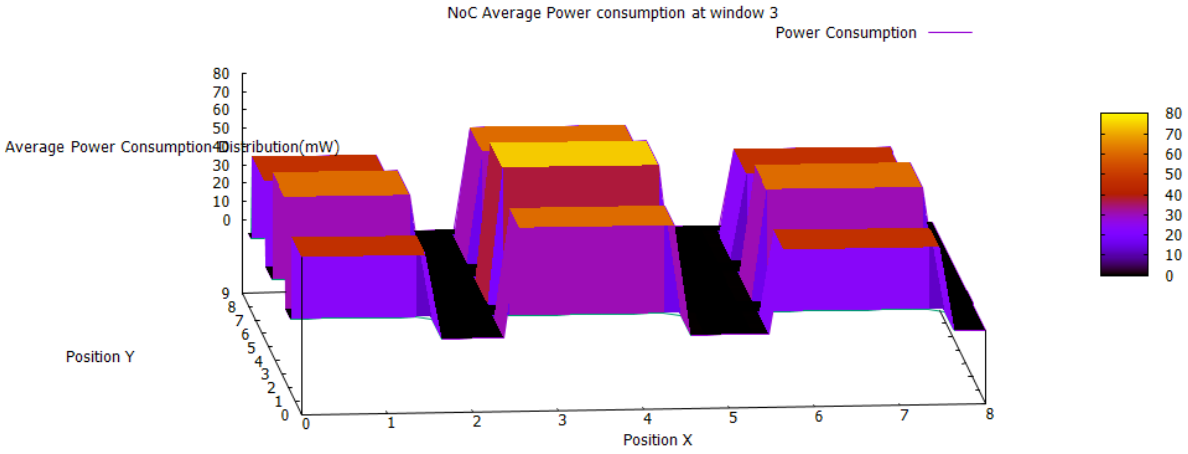
La composante dynamique du débit de signe négatif résulte des fluctuations des délais de transport induites par les conflits de ressources potentiels sur les chemins de transport. Ces fluctuations se traduisent par une variation de la charge pouvant être injectée au réseau. Dès lors que le débit comporte une composante dynamique fortement dépendante de l'état du réseau, une problématique intéressante consiste à définir une méthodologie de mesure de débit tenant compte justement de ces variations.

Une approche très simple consisterait à échantillonner la mesure de débit sur plusieurs intervalles de temps, donnant, plutôt qu'une valeur moyenne, une courbe caractéristique d'évolution du débit plus facilement exploitable.

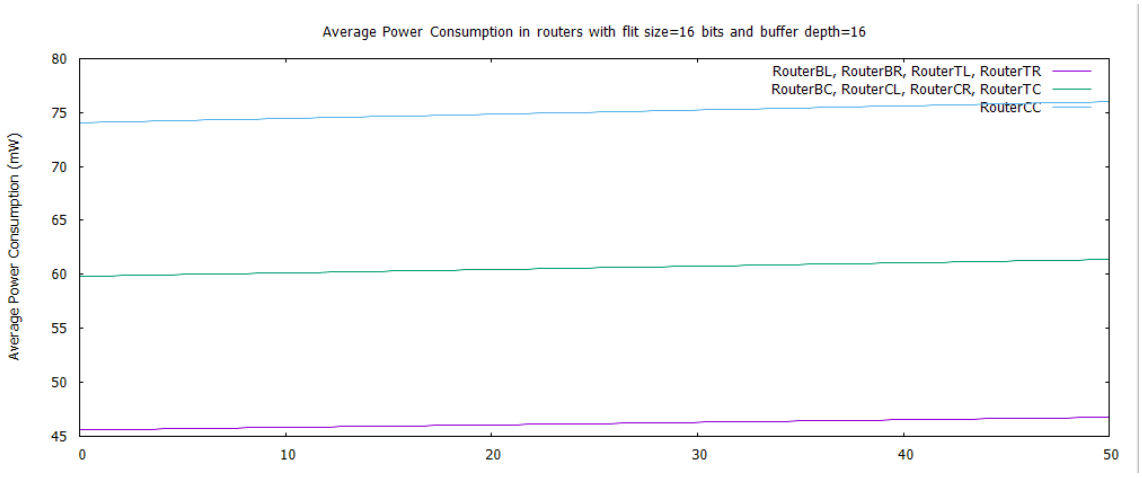
Cependant l'approche la plus couramment utilisée prenant en compte la composante dynamique consiste à exprimer le débit en termes de rapport entre le taux d'injection totale et le taux de charge effectivement acceptée.

Consommation d'énergie

On retrouve une distribution de la consommation moyenne du NoC (figure V-8a).
Mais aussi, la consommation moyenne de chaque nœud durant le temps de simulation (figure V-8b).



(a)



(b)

Figure V-8 : Graphe de consommation de puissance (a) distribution moyenne de puissance (b) consommation moyenne /nœud

V.4 Choix de l'environnement de synthèse

Après qu'une architecture de cartographie particulière soit sélectionnée, la phase finale vise à mettre en œuvre l'architecture de communication du NoC. Cela inclut la simulation et la vérification de la conception finale pour s'assurer que les contraintes de conception définies par l'utilisateur et les objectifs de conception générale sont correctement remplis.

Enfin, la synthèse, floorplanning et les étapes de génération de mise en page sont effectuées. Dans un premier lieu, on a eu à tester les fichiers produits par ATLAS avec 3 différents EDA. Dans ce qui suit, on décrira QUARTUS, XILINX ou encore VIVADO.

V.4.1 QUARTUS

Quartus est un logiciel développé par la société Altera, permettant la gestion complète d'un flot de conception CPLD ou FPGA. Ce logiciel permet de faire une saisie graphique ou une description HDL (VHDL ou verilog) d'architecture numérique, d'en réaliser une simulation, une synthèse et une implémentation sur cible reprogrammable. Il comprend une suite de fonctions de conception au niveau système, permettant d'accéder à la large bibliothèque d'IP d'Altera et un moteur de placement-routage intégrant la technologie d'optimisation de la synthèse physique et des solutions de vérification. De manière générale, un flot de conception ayant pour but la configuration de composants programmables se déroulent de la manière suivante :

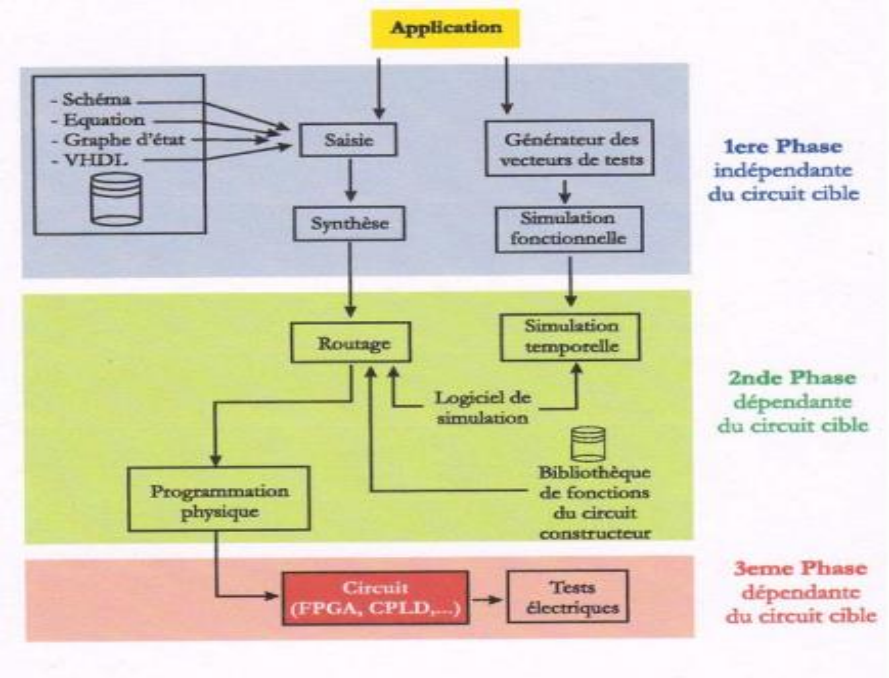


Figure V-9 : Les trois phases du flot de conception sous Quartus

V.4.2 XILINX ISE

Le logiciel Xilinx ISE est un logiciel de description, de simulation, et de programmation de circuits et systèmes numériques sur des composants programmables.

La suite ISE permet :

- La description de circuits numériques sous forme de schémas logiques, de machines à états finis ou en langages de description matériel (VHDL, Verilog, ABEL).
- La compilation, la simulation comportementale.
- La synthèse, le placement routage et l'implémentation.
- La simulation temporelle et l'analyse de timing.
- La programmation sur les circuits programmables de Xilinx (CPLD et FPGA).

V.4.3 VIVADO 2014.4

Vivado Design Suite est une suite logicielle produite par Xilinx pour la synthèse et l'analyse de conception HDL, remplaçant Xilinx ISE avec des fonctionnalités supplémentaires pour développement des systèmes sur puce.

Contrairement à ISE qui reposait sur ModelSim pour la simulation, Vivado comprend un simulateur logique intégré. Il introduit également la synthèse de haut niveau, avec un outil qui convertit le code C en logique programmable.

Vivado permet aux développeurs de synthétiser (compiler) leurs conceptions, effectuer une analyse de timing, d'examiner les diagrammes RTL, simuler la réaction d'une conception à différents paramètres, et configurer le périphérique cible avec le programmateur. Vivado est un environnement de conception pour des produits FPGA de Xilinx, et est étroitement couplé à l'architecture de ses puces, et ne peut être utilisé avec des produits FPGA autres que ceux de Xilinx.

Le design flow sous VIVADO est représenté sur la figure V-10.

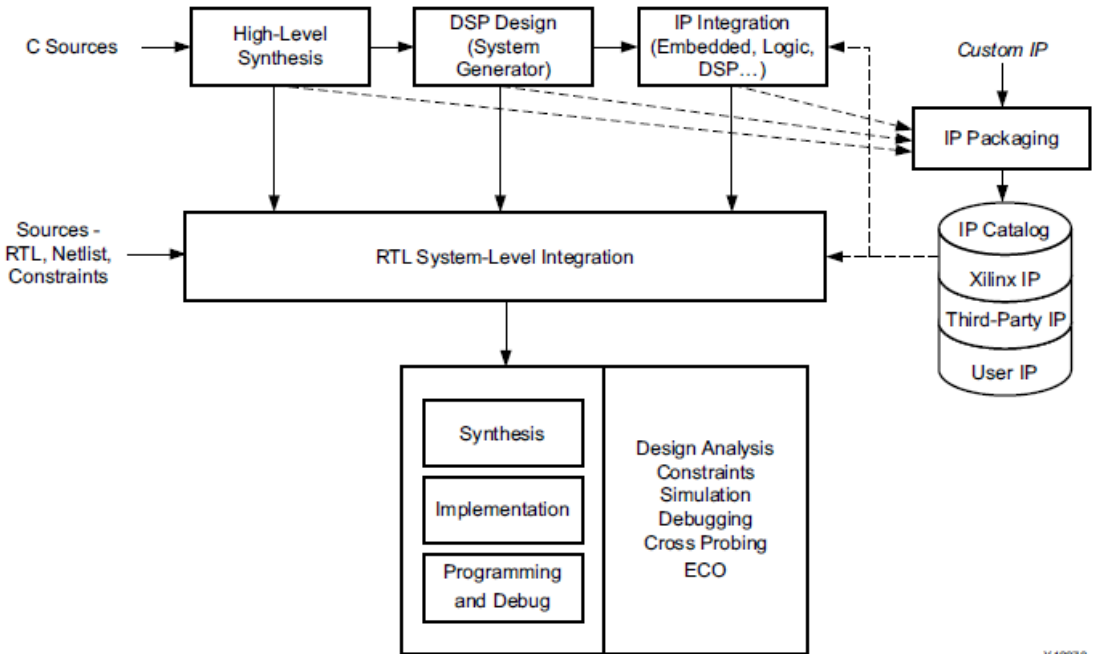


Figure V-10 : Design flow de conception sous VIVADO

La synthèse s'est faite sur VIVADO couplé à une carte ZynQ7000 de ZedBoard.

Carte ZedBoard ZynQ7000

http://zedboard.org/product/zedboard. La carte contient toutes les interfaces nécessaires et les fonctions de support pour permettre de palier à un large éventail d'applications. Les caractéristiques d'évolutivité de la carte la rendent idéale pour le prototypage rapide et le développement de concept.

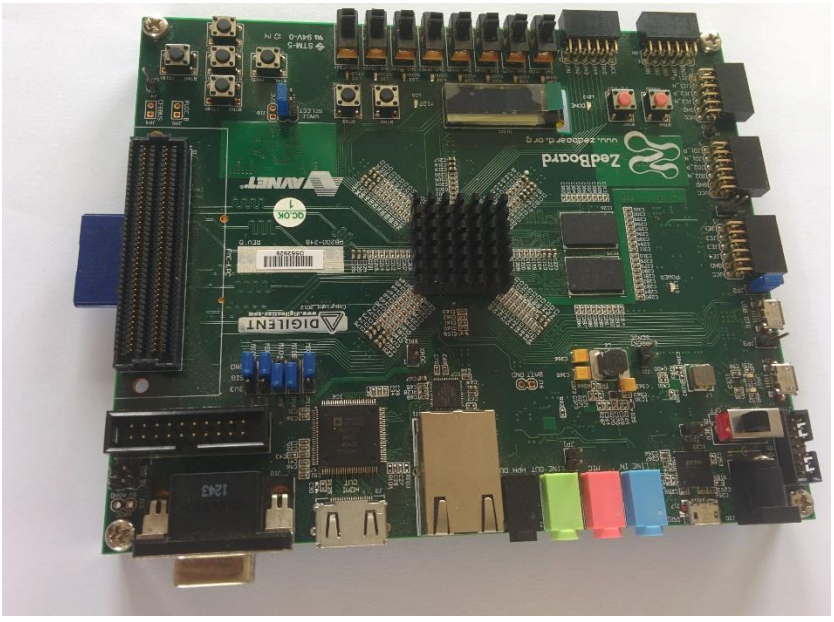


Figure V-11 : carte ZedBoard ZynQ7Z020

V-5 Synthèse et Implémentation sur carte

La synthèse est le processus de transformation d'une conception RTL-spécifiée en une représentation Gate-level et optimisée pour l'utilisation de mémoire pour de meilleures performances.

L'implémentation dans Vivado comprend toutes les mesures nécessaires pour le placement et le routage dans un niveau des ressources de la carte cible en termes de logiques, de contraintes temporelles et physiques de la conception.

Après avoir récupéré les fichiers VHDL générés par ATLAS et enlevé la partie systemC qui est destinée pour le benchmarking et l'évaluation, ces derniers ont été injectés dans VIVADO afin de procéder à l'implémentation sur la carte ZynQ.
L'interface graphique de VIVADO est présentée dans la figure V-12

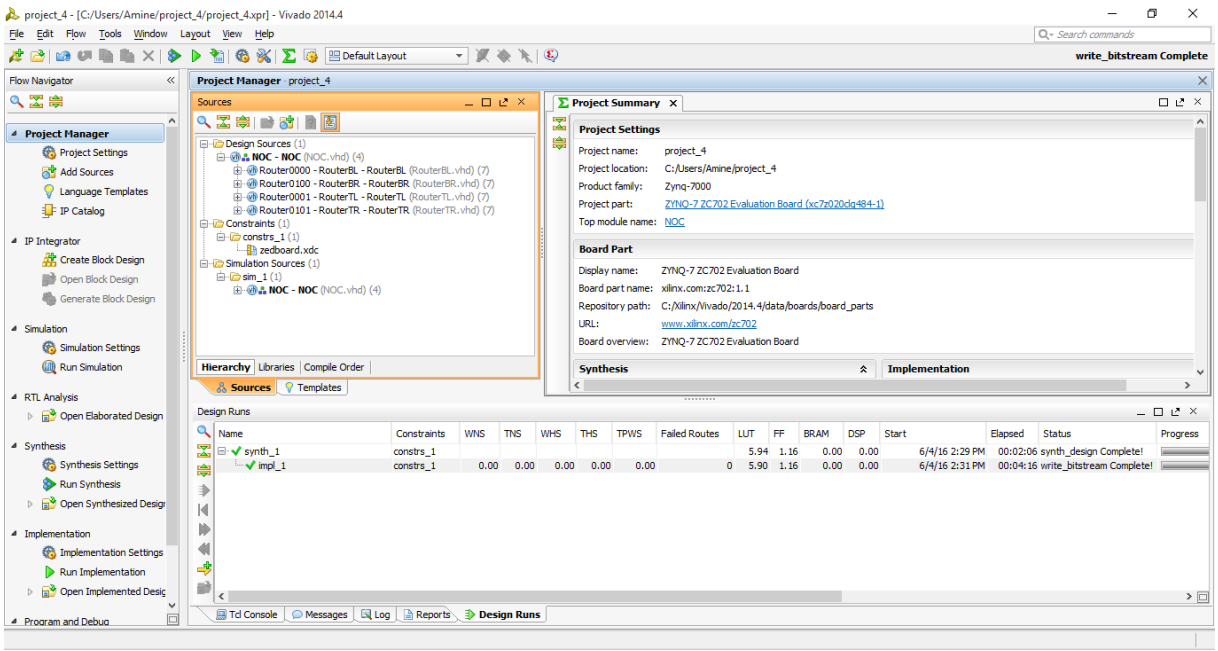


Figure V-12 : Interface graphique de VIVADO

Pour mieux comprendre la hiérarchie, nous avons généré un graphe (figure V-12) qui est beaucoup plus représentatif et qui permettra aux lecteurs de mieux visualiser le contenu des fichiers.

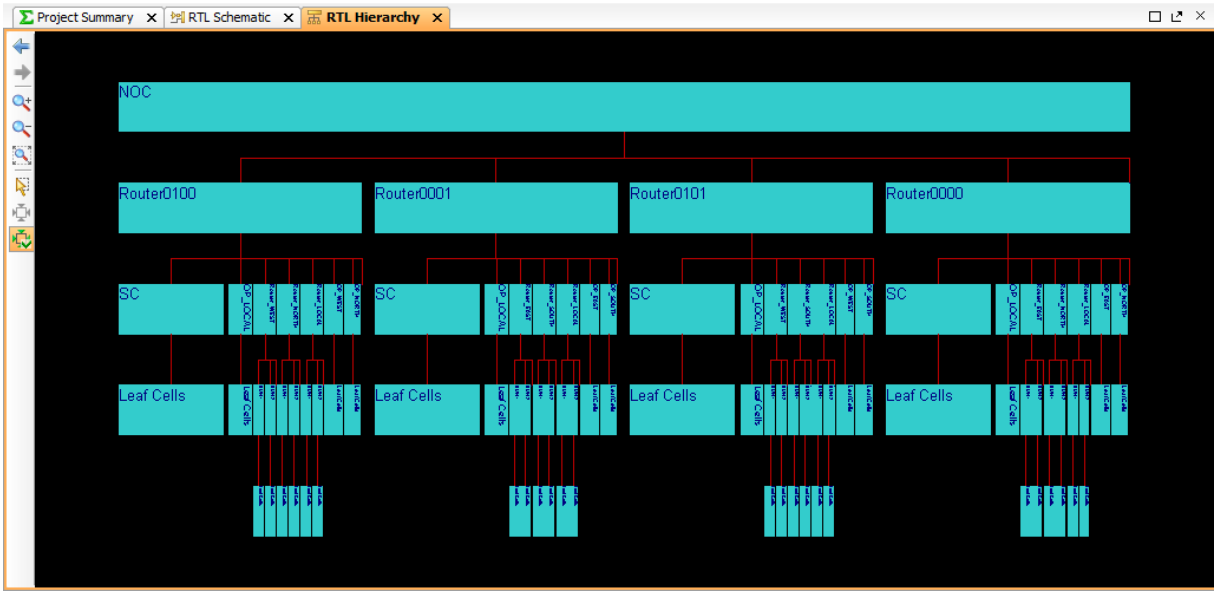


Figure V-13 : RTL Hiérarchie du NoC Hermes 2x2

Un zoom sur un des 4 blocs de nœuds, plus précisément celui de gauche, a permis d'obtenir la figure V-14 et qui montre bien les 5 ports (Nord – Est – Ouest - Sud et Local) ainsi que les buffers associés à chaque port.

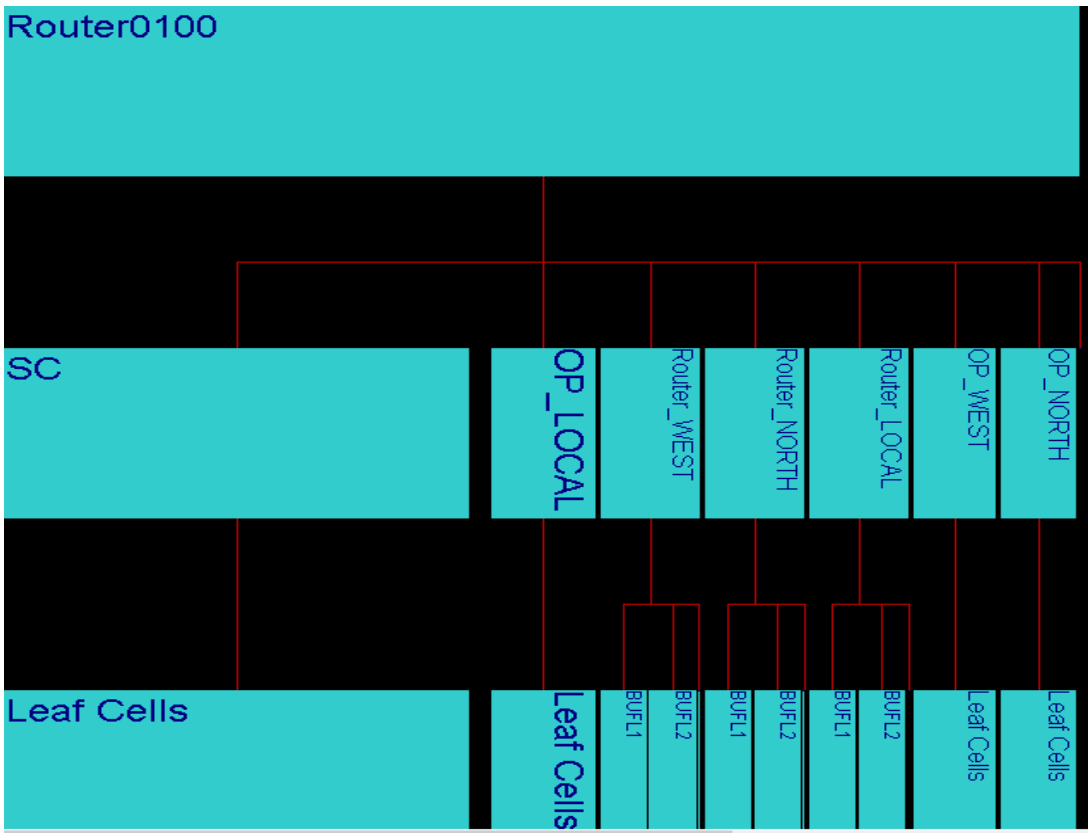


Figure V-14 : Vue détaillée d'un nœud Hermes et de ses composants

On remarque bien que les ports (Sud et Est) ne possèdent pas de buffers et ceci est dû au fait que ces deux ports ne sont pas utilisés par le réseau parce qu'ils ne trouvent pas de nœuds adjacents dans les directions mentionnées.

Si nous nous intéressons aux signaux d'entrées/sorties de notre schéma global du NoC Hermes 2x2, ceux-ci sont représentés sur la figure suivante ou les 4 blocs gris représentent les nœuds du réseau.

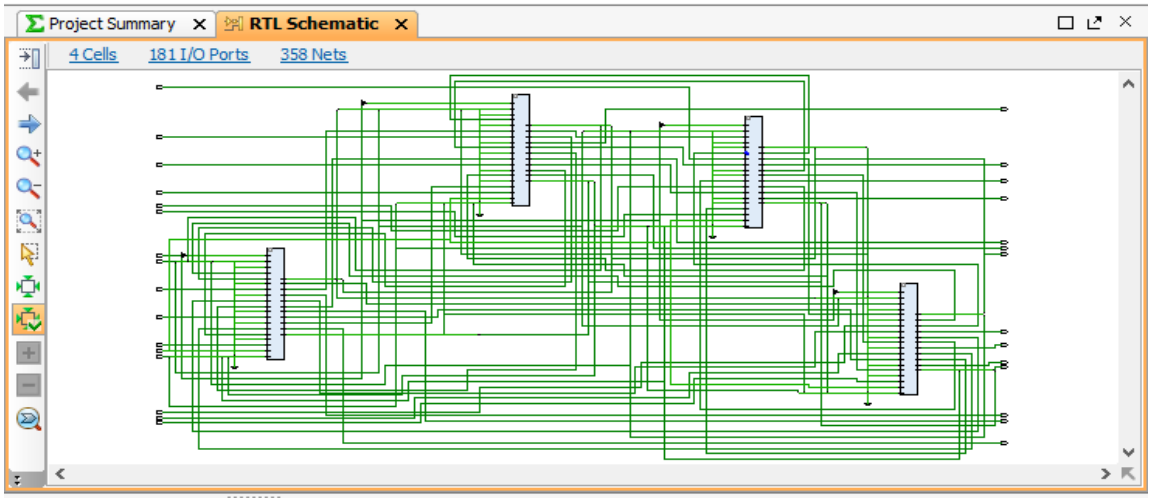


Figure V-15 : Schématique RTL de notre NOC HERMES 2x2

Design d'implémentation

Dans cette étape se produit une allocation de ressources sur notre cible ainsi qu'un mappage efficace après une optimisation du design et de la consommation d'énergie.

A la fin de l'implémentation, on retrouve un schéma de notre cible avec des connexions qui représentent la communication à l'intérieur du circuit (figure V-16).

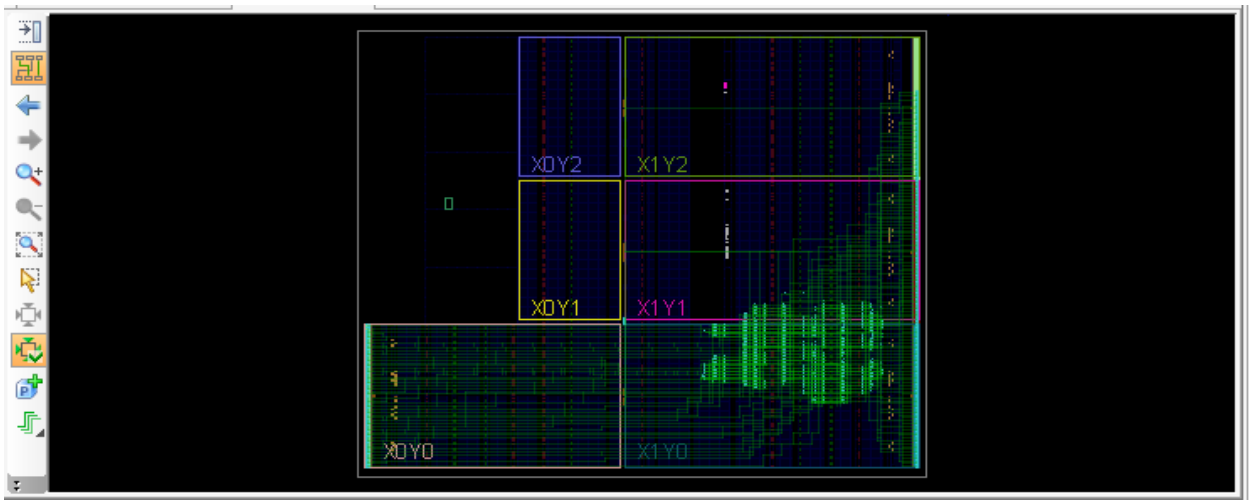
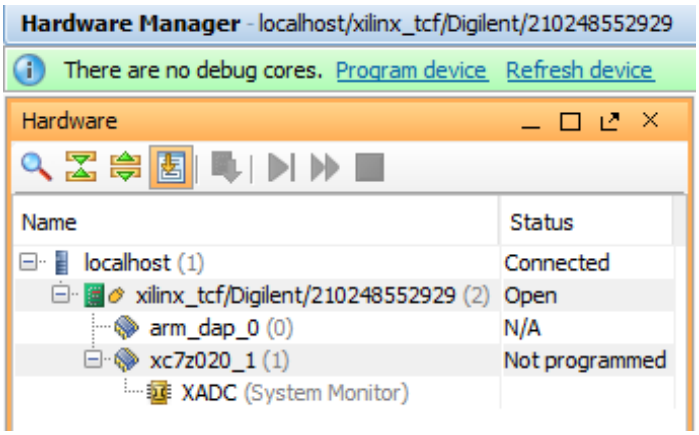


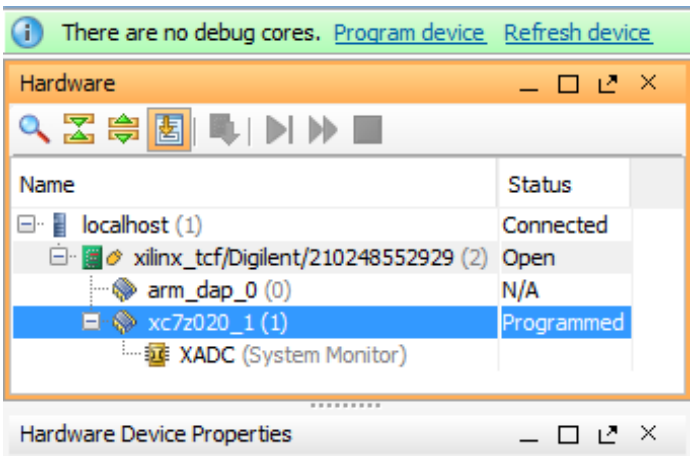
Figure V-16 : design d'implémentation d'un réseau Hermes 2x2

A partir de ce moment-là, on peut connecter notre carte ZynQ7000 via le port de programmation.

Un outil dédié à la connexion entre la machine locale et la carte cible permet de récupérer les fichiers d'implémentation (. Bit) et de les injecter sur la carte.
La figure V-17 prouve la réussite de l'implémentation sur la carte.



(a)



(b)

Figure V-17 : illustration d'implémentation des codes HDL sur la carte ZynQ7000 (a) avant implémentation (b) après implémentation.

V.6 Conclusion

Au cours de ce chapitre, nous avons abordé le NoC HERMES et son simulateur ATLAS pour concevoir, simuler, et implémenter le réseau. L'environnement de travail, constitué de plusieurs outils CAO, aide le concepteur du réseau à définir la structure du NoC, et les contraintes de communication, il permet également d'automatiser la génération de la description matérielle du NoC pour ainsi empêcher les erreurs dues à l'édition manuelle.

La communication dans le réseau est donc bien expliquée au moyen de signaux échangés entre les différents nœuds adjacents. Enfin, la dernière partie concernera la réalisation d'une synthèse et l'implémentation sur carte FPGA.

Conclusion Générale et Perspectives

Nous avons commencé cette étude en réalisant un état de l'art sur les NoCs à travers le chapitre I. Pour y parvenir, nous avons introduit le paradigme et identifié les paramètres des NoCs, en les classant conformément aux sept couches du modèle OSI. Cette classification a permis de déterminer deux types de paramètres. Les paramètres architecturaux qui permettent de choisir la topologie, le placement des communicants, et enfin la largeur des liens et la taille des mémoires. Le deuxième type de paramètre détermine les protocoles utilisés par le NoC pour acheminer les données, tels que l'algorithme de routage, la politique de mémorisation, et l'arbitrage. Cette étude, que nous avons réalisée sur une sélection de NoCs par rapport à leurs paramètres, a permis d'identifier les différentes classes relatives à ces derniers. On cite alors les NoCs avec une qualité de service qui garantissent le trafic (GT), en réservant certaines ressources du réseau à une communication pour garantir un débit ou/et une latence quel que soit la charge du réseau. Ou encore les NoCs où le trafic est acheminé au mieux (BE), en stipulant que cette classe ne réserve aucune ressource ce qui, par conséquent, n'offre aucune garantie de latence ou de débit. La dernière classe est celle où le réseau intègre les deux types de trafic GT et BE. La deuxième étape de notre démarche consiste à choisir le type de NoC, ses paramètres et son architecture. Après consultation avec l'encadrant, nous avons choisi de concevoir un NoC de type HERMES couplé avec le simulateur ATLAS pour deux raisons. La première raison est due au fait qu'il soit disponible par rapport aux autres NoCs. Quant à la deuxième, elle vise tout simplement à expliquer l'intérêt de l'implémentation du NoC sur carte FPGA, car le simulateur ATLAS, développé par la même équipe responsable de la conception d'Hermes, permet de récupérer des codes HDL synthétisables.

Ainsi dans le dernier chapitre, nous avons en premier lieu, exploré plus en détail notre NoC Hermes et tous ses paramètres. Pour ensuite simplifier le concept de réseau sur puce jusqu'à la plus petite entité qui compose le réseau qui est bien évidemment le nœud et ce qui se produit comme échange de signaux à l'intérieur de ce dernier. Eventuellement une synthèse d'un exemple de NoC Hermes 2x2 a été faite, suivie d'une implémentation sur carte ZynQ7000.

Les perspectives de ces travaux sont conséquentes en termes de compréhension de ce nouveau concept de réseau mais aussi de conception d'une électronique de système de vision embarquée sur carte FPGA.

Les évaluations architecturales proposées dans cette thèse n'ont été réalisées que pour des petits réseaux HERMES (Maximum 2x3) en raison de la limitation en espace mémoire sur la carte ZynQ. Une démonstration complète avec des réseaux de grande taille aurait permis d'aboutir à un comparatif entre ressources mémoires nécessaires et débit de données.

D'un point de vue de contrôle global des chargements de contexte, une unité programmable externe compléterait le système en se chargeant de la génération du trafic. Il serait alors pertinent d'étudier des extensions dans le cadre d'une utilisation d'unités de calcul de types processeurs élémentaires programmables.

Malencontreusement, le temps ne nous a pas permis l'exécution d'un travail aboutissant au même résultat mais en partant d'une synthèse de haut niveau (HLS) en exploitant cette fois-ci les codes en systemC au lieu de ceux en HDL.

Bibliographie

[1].G. Moore, No Exponential is Forever: But "Forever" Can Be Delayed! [semiconductor industry], Solid-State Circuits Conference (ISSCC), pp. 20–23, 2003

[2]. ITRS 2011.http://www.itrs.net

[3]. V. Lo, S. Rajopadhye, S. Gupta, D. Keldsen, M. Mohamed, B. Nitzberg, J. Telle, X. Zhong, OREGAMI: tools for mapping parallel computations to parallel architectures. Int. J. Parallel Programming **20**(3), 237–270 (1991)

[4]. S. Murali, G. De Micheli, SUNMAP: a tool for automatic topology selection and generation for NoCs, Design Automation Conference (DAC), pp. 914–919, 2004

[5]. B. Ackland, A. Anesko, D. Brinthaupt, S. Daubert, A. Kalavade, J. Knobloch, E. Micca, M. Moturi, C. Nicol, J. O'Neill, J. Othmer,E. Sackinger, K. Singh, J. Sweet,C.Terman, J.Williams, A single chip, 1.6-billion, 16-bMAC/s multiprocessor DSP. IEEE J. Solid-State Circuits **35**(3), 412–424 (2000)

[6]. A comparison of Network-on-Chip and Busses, white paper by Arteris Corp. http://www. design-reuse.com/articles/10496/a-comparison-of-network-on-chip-and-busses.html

[7]. V. Pavlidis, E. Friedman, *Three-Dimensional Integrated Circuit Design* (Morgan Kaufmann, San Francisco, 2010)

[8]. NoC Interconnect Improves SoC Economics, Objective Analysis - Semiconductor Market Research, 2011. http://www.objective-analysis.com/uploads/NoC_Interconnect_Improves_ SoC_Economics_-_Objective_Analysis.pdf

[9]. A. Jantch, H. Tenhunen, *Networks on Chip* (Springer, Berlin, 2003)

[10]. J.-J. Lecler, G. Baillieu, Application driven network-on-chip architecture exploration & refinement for a complex SoC. J. Des. Autom. Embed. Syst. **15**(2), 133–158 (2011)

[11]. G. De Micheli, L. Benini, *Networks on Chips: Technology and Tools (Systems on Silicon)* (Morgan Kaufmann, San Francisco, 2006)

[12]. R. Thid,M.Millberg, A. Jantsch, Evaluating NoC communication backbones with simulation, in *IEEE NorChip Conference*, pp. 27–30, 2003

[13]. I. Saastamoinen, M. Alho, J. Nurmi, Buffer implementation for Proteo network-on-chip. Int. Proc. Circuits Syst. **2**, 113–116 (May 2003)

[14]. The Standard Performance Evaluation Corporation, http://www.spec.org/hpg/

[15]. R. Dick, Embedded System SynthesisBenchmarks Suites (E3S), http://www.ece.northwestern.edu/dickrp/e3s/

[16]. N. Genko, D. Atienza, G. De Micheli, L. Benini, J.M. Mendias, R. Hermida, F. Catthoor, A novel approach for network on chip emulation', in *International Symposium on Circuits and Systems (ISCAS)*, pp. 2365–2368, 2005

[17]. W. Dong, B. Al-Hashimi, M. Schmitz, Improving routing efficiency for network-on-chip through contention-aware input selection, in*Asia and South Pacific Conference on Design Automation (ASP-DAC)*, pp. 36–41, 2006

[18]. S. Woo, M. Ohara, E. Torrie, J.P. Singh, A. Gupta, The Splash-2 Programs: characterization and methodological considerations, *International Symposium on Computer, Architecture*, pp. 24–36 (1995)

[19]. L. Chunho, M. Potkonjak,W.Mangione-Smith,Mediabench: a tool for evaluating and synthesizing multimedia and communications systems, *International Symposium on Microarchitecture*, pp. 330–335 (1997)

[20]. The Standard Performance Evaluation Corporation (2013). http://www.spec.org/

[21]. W.J. Dally, C.L. Seitz, The torus routing chip. J. Distrib. Comput. **1**(3), 187–196 (1986)

[22]. M. Modarressi, H. Sarbazi-Azad, M. Arjomand, A hybrid packet-circuit switched on-chip network based on SDM. in *Design, Automation and Test in Europe Conference and Exhibition (DATE '09)* (2009), pp. 566–569

[23]. W.J. Dally, B. Towles, *Principles and Practices of Interconnection Networks* (Morgan Kaufmann, San Francisco, 2004)

[24]. A. Lankes, T.Wild, A. Herkersdorf, S. Sonntag, H. Reinig, Comparison of deadlock recovery and avoidance mechanisms to approach message dependent deadlocks in on-chip networks. in *International Symposium on Networks-on-Chip (NOCS)* (2010), pp. 17–24

[25]. K. Anjan, T. Pinkston, DISHA: a deadlock recovery scheme for fully adaptive routing. In *Parallel Processing, Symposium* (1995), pp. 537–543

[26]. S. Lee, A deadlock detection mechanism for true fully adaptive routing in regular wormhole networks. Comput. Commun. **30**(8), 1826–1840 (2007)

[27]. J. Kim, L. Ziqiang, A. Chien, Compressionless routing: a framework for adaptive and faulttolerant routing. IEEE Trans. Parallel Distrib. Syst. **8**(3), 229–244 (1997)

[28]. K. Goossens, J. Dielissen, A. Radulescu, AEthereal network on chip: concepts, architectures, and implementations. Des. Test Comput. **22**(5), 414–421 (2005)

[29]. O. Lysne, T. Skeie, S. Reinemo, I. Theiss, Layered routing in irregular networks. IEEE Trans. Parallel Distrib. Syst. **17**(1), 51–65 (Jan. 2006)

[30]. R.Widyono, *The Design and Evaluation of Routing Algorithms for Real-Time Channels*, TR- 94-024, (University of California at Berkeley and Int'l Computer Science Institute, Berkeley, 1994)

[31]. F. Feliciian, S. Furber,An asynchronous on-chip network router with quality-of-service (QoS) support. in *International SOC Conference* (2004), pp. 274–277

[32]. D. Bertozzi, L. Benini, Xpipes: a network on chip architecture for gigascale systems-on-chip. IEEE Circ. Syst. Mag. **4**(2), 18–31 (2004)

[33]. R. Tamhankar, S. Murali, G. De Micheli, Performance driven reliable link design for networks on chips. Asia South Pacific Des. Autom. Conf. (ASP-DAC) **2**, 749–754 (2005)

Autres références bibliographiques

A.Kouadri-Mostefaoui. « Architectures Flexibles pour la Validation et L'exploration de Réseaux-sur-Puce ». Micro et nanotechnologies/Microélectronique. Institut National Polytechnique de Grenoble - INPG, 2009. Français. <Tel-00431799>.

A.Kouadri, Senouci, B., Pétrot, F. « Networks-In-Package: Performances management and design methodology », VLSI Design, Automation and Test, 2008. IEEE International Symposium on April 2008.

Amr Helmy. « Vérification formelle des communications dans un réseau sur puce : HERMES à l'aide d'ACL2 ». Huitièmes Journées Doctorales en Informatique et Réseaux (JDIR'07), Jan 2007, Marne-la-Vallée, France. pp.161-168, 2007. <Hal-01116655>.

Arteris. « A comparison of network-on-chip and busses ». white paper.

ATLAS main page. https://corfu.pucrs.br/redmine/projects/atlas.

B. M. Al-Hashimi: « System-On-Chip: Next Generation Electronics ». Circuits, Devices and Systems, 2006.

B.S, Feero, Friedman E.G, « 3-D Topologies for Networks-On-Chip", Very Large Scale Integration (VLSI) Systems », IEEE Transactions on, Vol. 15. 2007.

C. Hilton, B. Nelson: « PNoC: a flexible circuit-switched NoC for FPGA-based Systems », in Field Programmable Logic, Aug. 2005.

C. Tanougast. « Initiation à la modélisation et co-simulation comportementale C-VHDL d'un réseau de communication sur Puce (*Network on Chip*) ». Nancy, 2010.

F. ROUSSEAU. « Generation of emulation platforms for NoC exploration on FPGA ». Grenoble, FRANCE.

Farid Daneshgar, Majid Taghipoor. « Increase the efficiency of Network on Chip using buffer sharing mechanism ». Iran, 2014.

Fernando Moraes. « HERMES: an infrastructure for low area overhead packet-switching networks on chip ». Brazil, Elsevier, 2004.

G. De Micheli, Benini L. « Networks on chip: a new paradigm for systems on chip design », Design, Automation and Test in Europe Conference and Exhibition, 2002. Proceedings, 2002.

GAPH – Grupo de Apoio ao Projeto de Hardware. « Atlas – An environment for NoC Generation and Evaluation ». Available at

http://www.inf.pucrs.br/~gaph/AtlasHtml/AtlasIndex_us.html, captured in September, 2008.

J.Wu. « A fault-tolerant and deadlock-free routing protocol in 2d meshes based on odd-even turn model ». *Computers, IEEE Transactions on 52*, 9 (2003).

Jie Chen and Cheng Li, Paul gillard, « Network On-Chip (NoC) Topologies and performance: A Review ».

Junyan Tan. « Exploration d'architectures génériques sur FPGA pour des algorithmes d'imagerie multi spectrale. Micro et nanotechnologies/Microélectronique ». Université Jean Monnet - Saint- Etienne, 2012. Français. <NNT : 2012STET4028>. <Tel-00838037v2>.

K.Tatas, K.Siozios, D.Soudris. A.Jantsch. « Designing 2D and 3D Network-on-Chip Architectures ». NewYork, Springer 2014.

L. Benini: « Application Specific NoC Design », in DATE, 2006.

Mentor Graphics, « Modelsim SE User's Manuel, Sofware, Version 10.1 », 2011. http://www.mentor.com/.

Salvator Gkalea. « Fault-Tolerant Nostrum NoC on FPGA for the ForSyDe/NoC System Generator Tool Suite ». Royal Institute of Technology, 2014.

Soares R. « HardNoC: A Platform to Validate Networks on Chip through FPGA Prototyping». Brazil, IEEE, 2012.

V.Nagammai. S.B.Mohan. « An Efficient Topological Structure for Application Specific Network on Chip Synthesis ». En ligne, Vol.3, Issue 3, 2015

V. Puente, J.A. Gregorio and R. Beivide. « SICOSYS: An integrated framework for studying interconnection network performance in multiprocessor systems ». Span, IEEE, 2002.

Virginie Fresse, Catherine Combes, Hatem Belhassen. « Mathematical models applied to on chip network on FPGA for resource estimation ». International Conference on Computational Science and Engineering, Dec 2014, Chengdu, China. 2014, <http://umc.uestc.edu.cn/conference/CSE2014/>. <Hal-01098232>.

Xilinx, Inc. « Xilinx University Program Virtex-II Pro Development System ». Hardware Reference Manual, UG069, V1.0, March, 2005.

Y,Umit. Ogras. Radu Marculescu. « Modeling, Analysis and Optimization of Network-on-Chip Communication Architectures ». USA, Springer, 2013.