الجمهورية الجزائرية الديمقراطية الشعبية

وزارة التعليم العالي و البحث العلمي

**BADJI MOKHTAR ANNABA UNIVERSITY**
**UNIVERSITE BADJI MOKHTAR ANNABA**

جامعة باجي مختار- عنابة

Année 2018

Faculté des Sciences de l'Ingéniorat
Département d'Electronique

# THÈSE

**Présentée en vue de l'obtention du diplôme de DOCTORAT en Sciences**

# Mise en œuvre d'une architecture multiprocesseur autour d'un réseau sur puce (NoC) de type Wishbone

**Option : Télécommunications**

Par :

**MAYACHE Hichem**

**Directeur de Thèse:**
**Pr. TOUMI Salah** — **Université Badji Mokhtar d'Annaba**

**Co-Directeur de Thèse :**

**Pr. BOURENNANE El Bay** — **Université de Bourgogne Dijon France**

**Devant le Jury:**

**Président:**
**Pr. SAOUCHI Kaddour** — **Université Badji Mokhtar d'Annaba**

**Examinateurs:**
**Pr. CHAOUI Allaoua** — **Université Abdelhamid Mehri Constantine 2**
**Pr. KIMOUR Med Tahar** — **Université Badji Mokhtar d'Annaba**
**Dr. MESSAOUDI Kamel** — **Université Med Chérif Messaadia Souk-Ahras**


بسم الله الرحمن الرحيم

# Remerciements

En tout premier lieu, je remercie mon DIEU « **ALLAH, le tout puissant** » de m'avoir accordé santé, courage, patience, connaissance et tout ce qui m'a été nécessaire et utile pour achever ce travail.

J'adresse mes remerciements les plus chaleureux à l'ensemble des membres du Jury :

Professeur Kaddour SAOUCHI de l'Université Badji-Mokhtar-Annaba, d''avoir accepté de présider ce Jury,

Professeur Allaoua CHAOUI de l'Université Abdelhamid MEHRI – Constantine2, d'avoir accepté d'examiner ce travail,

Professeur KIMOUR de l'Université Badji-Mokhtar-Annaba, d''avoir accepté d'examiner ce travail,

Docteur Kamel MESSAOUDI de l'université Mohamed Chérif MESSAADIA-Souk Ahras, d'avoir accepté d'examiner ce travail,

Mes remerciements les plus sincères, à

***Professeur Salah TOUMI de l'Université Badji-Mokhtar-Annaba, et Professeur El-Bay BOURENNANE de l'Université de Bourgogne – Dijon,***

*deux personnes d'exception et sources d'inspiration, pour qui je dois beaucoup, avec qui j'ai appris énormément tant scientifiquement qu'humainement.*

*Merci de vos enseignements. Merci pour vos conseils. Merci pour votre engagement. Merci de m'avoir transmis cette passion pour la science et l'enseignement. Merci pour votre confiance, vos expertises, du temps que vous m'avez accordé.*
*Je vous en suis profondément reconnaissant....*

**Toute ma gratitude à**

***mes amis et compagnons de route : Atef, Kamel, pour votre aide, vos conseils, et vos encouragements... un grand merci à vous deux***

***mes amis et collègues de l'Université Larbi Tebessi – Tébessa : Hakim, Samir, Riad, Tarek, Hamza, Toufik, Chemss, Yassine, Abdallah, Rostom,***

# *Dédicace*

*A mes parents, mes sœurs, ma femme et ma petite princesse « Dania »…*

*vos sacrifices, votre soutien infaillible, vos prières, vos encouragements permanents.*

# الملخص

يُظهر التصميم الحديث للأنظمة على الشريحة (SoCs) اتجاهاً واضحاً نحو تكامل العديد من نوى المعالجات. لقد أصبحت هذه الأنظمة ضرورية لتلبية متطلبات الأداء الحالية. يتم ترحيل التطبيقات المضمنة الحالية من نظام واحد يعتمد على المعالج إلى أنظمة معالجة متعددة مع اتصالات بيانات مكثفة. يستلزم الأداء الذي تتطلبه هذه التطبيقات معماريات متعددة المعالجات في شريحة واحدة (MPSoCs) ، مما يتطلب ترابطًا معقدًا. يعد تطور الهواتف الذكية أفضل مثال على تطور وعدم تجانس MPSoCs.

من أجل استخراج معظم الأداء من MP-SoCs ، تم اقتراح شبكات على رقاقة (NoCs) كحل اتصال على الشريحة للتغلب على قيود الترابط التقليدي. تم إنجاز العمل الذي تم إنجازه على طول هذه الرسالة في الجزء الأول منها حول شبكات على شكل رقاقة (Wishbone) لأنظمة MPSoCs. تتعلق المساهمات باقتراح منبر للتوليد التلقائي لمراكز الرعاية الصحية الأولية على مستوى VHDL. تسمح واجهات الشبكة المقترحة بالتكيف السريع لمكونات عظم الترقب على الشبكة. تسمح لنا بنية الترقوة (نوع المصدر المفتوح) باستخدام عناوين IP مصممة بالفعل وتقليل وقت تصميم النظام. ويركز الجزء الثاني من العمل على اقتراح بنية شبكة عازلة على رقاقة ، تتكيف مع تطبيقات ترميز الفيديو. تتم محاكاة الهندسة المعمارية المقترحة في هذه الأطروحة بواسطة أداة Xilinx ISE Design Suite 14.1 ، والتي يتم تنفيذها على منصات Virtex 7 FPGA وتناقش النتائج.

الكلمات المفتاحية: نظم على رقاقة ، وشبكات على رقاقة ، بنية عظم الترقوة ، نظم المعالجات رقاقة على رقاقة.

# Abstract

Modern systems on Chip (SoCs) design shows a clear trend towards the integration of multiple processor cores. Such systems have become necessary to meet current performance requirements. Current embedded applications are migrating from a single processor-based system to multiprocessing data-intensive systems. The performance demanded by these applications requires multiprocessor architectures in a single chip (MPSoCs), requiring complex interconnections. The evolution of smartphones is the best example of the evolution and heterogeneity of MPSoCs. In order to extract more performance from these MP-SoCs, networks on a chip (NoCs) have been proposed as an On-Chip communication solution to overcome the limitations of conventional interconnections. The work done along this thesis in its first part is articulated around Wishbone networks on chip for MPSoC systems. The contributions relate to the proposal of a platform for the automatic generation of NoCs at the VHDL level. Added network interfaces allow for quick adaptation of Wishbone components on the network. The proposed network interfaces allow the quick adaptation of Wishbone components on the network. Wishbone architectures (open-source) allow us to use already designed IPs and to reduce the design time of the system. The second part of the work takes place on the proposal of an architecture of a bufferless network-on-chip, adapted to video encoding applications. The second part of the work is focused on the proposal of an architecture of a bufferless network-on-chip, adapted to video encoding applications. The architectures proposed in this thesis are simulated by the Xilinx ISE Design Suite 14.1 tool, implemented on Virtex 7 FPGA platforms and the results are discussed.

**Keywords:**

Systems on Chip, Networks on Chip, Wishbone Architecture, Multi-Processor Systems-on-Chip.

# Résumé

La conception moderne des systèmes sur puce (SoCs) montre une nette tendance vers l'intégration de plusieurs cœurs de processeurs. De tels systèmes sont devenus nécessaires pour répondre aux exigences de performance actuelles. Les applications embarquées actuelles migrent d'un système basé sur un processeur unique à des systèmes multitraitement à communication de données intensives. Les performances demandées par ces applications font appel à des architectures multiprocesseurs dans une seule puce (MPSoCs), nécessitant des interconnexions complexes. L'évolution des smartphones est le meilleur exemple de l'évolution et de l'hétérogénéité des MPSoCs. Afin d'extraire le plus de performance de ces MP-SoCs, les réseaux sur puce (NoCs) ont été proposés comme une solution communication sur puce afin de surmonter les limites des interconnexions conventionnelles. Le travail accompli le long de cette thèse dans sa première partie s'articule autour des réseaux sur puce de type Wishbone pour des systèmes MPSoCs. Les contributions portent sur la proposition d'une plateforme pour la génération automatique des NoCs au niveau VHDL. Les interfaces réseaux proposées permettent l'adaptation rapide des composants de type Wishbone sur le réseau. Les architectures de type Wishbone (de type open-source) nous permettent d'utiliser des IPs déjà conçus et de réduire le temps de conception du système. La deuxième partie du travail réalisé est axée sur la proposition d'une architecture d'un réseau sur puce sans buffer, adaptée aux applications d'encodage vidéo. Les architectures proposées dans cette thèse sont simulées par l'outil ISE Design Suite 14.1 de Xilinx, implémentées sur des plateformes FPGA Virtex 7 et les résultats sont discutés.

**Mots-clés:**
Systèmes sur puce, réseaux sur puce, architecture Wishbone, systèmes multiprocesseurs sur puce.

# LISTE DES ACRONYMES

**A**

**AMBA:** Advanced Microcontroller Bus Architecture
**ASIC:** Application Specific Integrated Circuit
**ASIF:** Application Specific Inflexible FPGA
**ASIP:** Application Specific Instruction set Processor
**AVC :** Advanced Video Coding

**B**

**BE :** Best Effort
**BLE :** Basic Logic Element
**BLR** : Buffer Less Router
**BRAM:** Block RAM
**BRW** : Block Read / Write

**C**

**CAO :** Conception Assistée par Ordinateur
**CB :** Connection Box
**CLB :** Configurable Logic Block
**CMOS:** Complementary Metal Oxide Semiconductor
**CMP :** Chip Multi-Processor
**CAIS :** Contention-Aware Input Selection
**CPU:** Central Processing Unit

**D**

**DAC:** Digital-to-Analog Converter
**DCT :** Discrete Cosine Transform
**DSP:** Digital Signal Processor
**DVB:** Digital Video Broadcasting
**DVD :** Digital Versatile Disc

**E**

**E/S :** Entrée / Sortie
**EDK :** Embedded Development Kit

**F**

**FBSME :** Fixed Block Size Motion Estimation
**FCFS** : First Come First Served
**FIFO:** First-In-First-Out
**FLIT :** FLow control unIT
**FME :** Fractional Motion Estimation
**FPGA:** Field Programmable Gate Array

**fps** : Frame Per Second
**Full HD :** Full High Definition

**G**

**GPP:** General Purpose Processor
**GS:** Guaranteed Service

**H**

**HPCA:** High Performance Computing Applications

**I**

**IBM :** International Business Machines
**IC:** Integrated Circuit
**IDCT:** Inverse Discrete Cosine Transform
**IME :** Integer Motion Estimation
**IOB:** Input/Output Block
**IP:** Intellect Property
**ISE :** International Synthesis Environment
**ISO :** International Organization of Standardization
**ITRS :** International Technology Roadmap for Semicondiconductors

**L**

**LSB:** Least Significant Bit
**LUT:** Look Up Table

**M**

**MIPS:** Million Instruction per Seconde
**MPEG :** Moving Picture Experts Group
**MPSoC :** Multi-Processor Systen-on-Chip
**MUX:** Multiplexer
**MV**: Motion Vector
**MVD :** Motion Vector Difference
**MVP :** Motion Vector Prediction

**N**

**NI :** Network Interface
**NoC:** Network-on-Chip
**NRE** : Non-Recurring Engineering

**P**

**PBD :** Platform Based Desgin
**PC :** Personal Computer
**PCI:** Peripheral Component Interconnect
**PDA :** Personal Digital Assistant

**PE:** Processing Element

**Q**

**QoS:** Quality-of-Service

**R**

**RC :** Routing Computation
**RGB:** Red Green Blue
**RMW :** Read Modify Write
**RTL**: Register Transfer Level

**S**

**SA :** Switch Allocator
**SB :** Switch Box
**SD :** Standard Definition
**SIMD :** Single Instruction Multiple Data
**SRAM :** Static Random Access Memory
**ST :** Switch Traversal
**SoC:** System-on-Chip
**SRW :** Single Read Write
**SW:** Software

**T**

**TSV:** Through Silicon Via
**TTM :** Time To Market

**U**

**ULSI:** Ultra Large Scale Integration

**V**

**VBSME :** Variable Block Size Motion Estimation
**VCA :** Virtual Channal Allocator
**VCEG :** Video Coding Experts Group
**VHDL:** Very High Speed Integrated Circuit Hardware Description Language
**VLIW :** Very Long Instruction Word
**VLSI:** Very-Large-Scale Integration
**VME:** VERSAModule Eurocard
**VPU :** Vector Processing Unit

**W**

**Wb_MNI:** Wishbone Master Network Interface
**Wb_SNI:** Wishbone Slave Network Interface

# LISTE DES FIGURES

# LISTE DES TABLEAUX

# TABLE DES MATIERES

# Introduction générale

# INTRODUCTION GÉNÉRALE

## 1. Introduction

Au fil de l'évolution technologique de ces dernières décennies, mémoires et processeurs doivent devenir plus rapides, plus petits, moins chers et plus économes en énergie, ce qui oblige les concepteurs de circuits intégrés à en intégrer davantage sur une seule puce. Il est désormais possible d'avoir des milliards de transistors, des millions de portes, des milliers de circuits, des centaines de designs sur une seule puce. Les concepteurs les plus performants surmontent tous ces défis pour assurer un fonctionnement correct et fiable des circuits intégrés. L'augmentation de la densité d'intégration rend la rentabilité au centre des préoccupations majeures dans les conceptions de circuits intégrés ; proposée par Gordon E Moore dans son article « Cramming more components on integrated circuits », 1965, article dans lequel il proposait la fameuse loi de Moore [1]. D'ici 2025, la dimension physique des transistors CMOS devrait franchir le seuil de 10 nm, selon un rapport publié en 2012 par « International Technology Roadmap for Semiconductors » [2].

Alors que la loi de Moore atteint sa limite ces dernières années et qu'on parle désormais de l'au-delà de Moore, et afin de suivre le rythme de ces niveaux complexes d'intégration que les concepteurs ont mis au point une nouvelle méthodologie de conception appelée System-on-Chip (SoC). Le SoC est une technologie où la technologie maximale est entassée dans le plus petit espace possible. La conception du système sur une puce est fortement influencée par ce que nous appelons « Intellectual Property (IP) Core », car elle permet la réutilisation des composants

préalablement développés afin de continuer à améliorer les performances tout en minimisant les coûts de fabrication [3].

Les blocs « IPs » sont les éléments de base de diverses conceptions de systèmes sur puce pour la mise en œuvre d'applications plus grandes et plus complexes destinées aux systèmes embarqués. Un système sur puce peut contenir des matériels tels que des processeurs, des périphériques de mémoire, des contrôleurs, des processeurs de signaux numériques et divers blocs logiques personnalisés et des logiciels pour contrôler les hardwares. Le principal avantage du système sur puce est sa faible consommation d'énergie, son faible coût et sa fiabilité supérieure à celle des systèmes multi-puces qu'il a remplacés. Néanmoins, la transition vers la technologie du système sur puce a été confrontée à de nombreux défis tels que l'évolutivité, la synchronisation, l'hétérogénéité du système, ainsi que les problèmes d'interconnexion au sein du système. Par conséquent, la performance est confrontée à un goulot d'étranglement, non seulement en termes de puissance de calcul et d'accès mémoire, mais aussi dans la communication des éléments d'une seule et même puce. Dans ce contexte, les concepteurs ont tenté de proposer une alternative aux bus partagés afin de fournir une nouvelle solution d'interconnexion plus rapide, rentable et évolutive. Une telle solution sera la clé du succès des systèmes numériques futurs, composés à la fois de multiprocesseurs de dizaines de cœurs identiques et de systèmes hétérogènes sur puce.

## 2. Contexte et objectifs de la thèse :

Habituellement, l'architecture d'interconnexion est basée sur des connexions point-à-point ou des bus partagés. Si le système dispose d'un nombre limité d'IP, l'architecture point-à-point peut s'avérer très efficace. À mesure que la complexité du système augmente, le nombre de connexions autour de l'IP augmente également. Par conséquent, les connexions point-à-point se caractérisent par une faible flexibilité et réutilisation. Un bus partagé est un ensemble de fils commun à plusieurs IPs. L'approche du bus partagé était très prometteuse car elle offrait de la flexibilité et beaucoup de réutilisabilité. Cependant, cette approche a montré certaines limites vu qu'elle ne permet qu'une seule opération de communication à la fois, étant donné que tous les IPs partagent la même bande passante de communication dans le système. De plus son évolutivité est limitée à quelques douzaines d'IPs, ce qui constitue un frein pour des applications du type HPCA (High Performance Computing applications). Ainsi, l'évolutivité est devenue un problème majeur avec l'architecture à base d'un bus partagé.

Ce sont les limites de cette solution d'interconnexion qui ont ouvert la voie à un nouveau paradigme de communication appelé Network-on-chip (NoC). L'architecture NoC a été proposée comme une alternative performante, évolutive et économe en énergie à l'architecture du bus partagé. Inspirés du domaine des réseaux en informatique, les NoCs sont basés sur une ensemble de protocoles bien défini similaire à l'ISO/OSI.

Au cours de la dernière décennie, de nombreux efforts de recherche ont été déployés pour optimiser les réseaux sur puce, depuis les aspects physiques de bas niveau jusqu'aux problèmes liés aux systèmes et aux applications. Désormais, ils ont atteint un niveau de développement avancé par leur intégration en tant que composant fondamental dans de nombreux produits commerciaux réussis.

Les travaux de cette thèse s'articulent autour des réseaux sur puce comme solution d'interconnexion pour les systèmes embarqués. L'objectif principal de notre travail est de mettre en place une plateforme à base de réseau sur puce de type Wishbone pour des architectures multiprocesseur. Cette solution est destinée à des applications modernes aux besoins élevés en performance : une bande passante large, une latence faible, une consommation énergétique basse, et une utilisation en ressource logique modérée. De cet objectif principal découlent de multiples objectifs secondaires qui consistent à faciliter la génération des réseaux sur puce pour l'interconnexion des composants de type Wishbone en fournissant un outil de génération automatique. De plus, nous nous proposons d'offrir un large choix d'interconnexion qui pourra servir pour diverses applications ayant différentes exigences.

Wishbone est une architecture de communication point-à-point pour les systèmes sur puces qui offre une interface commune et flexible pour la communication. Adapter le protocole Wishbone dans le domaine des réseaux sur puce est motivé par l'apport d'une telle architecture dans la conception des systèmes, car Wishbone est aussi une architecture « open-source ». Une telle architecture donne la possibilité aux concepteurs d'accéder directement à n'importe quel composant Wishbone ce qui conduit à une intégration facile et rapide des IPs en supprimant le temps de conception et de validation.

## 3. Contributions de la thèse

Les travaux réalisés dans cette thèse s'inscrivent dans le domaine du développement expérimental, de la conception architecturale et de l'implémentation

de solutions numériques avancées pour les communications de systèmes embarqués sur des cibles reconfigurables de type FPGA (Field Programmable Gate Array),

Afin d'atteindre les objectifs de cette thèse, le travail réalisé entreprend les tâches suivantes :

- Une analyse détaillée des solutions d'interconnexion sur puce en général et les réseaux sur puce en particulier. En outre, nous détaillerons aussi le protocole Wishbone qui sera utilisé ultérieurement.
- Une proposition pour l'adaptation de l'interface Wishbone aux communications par les réseaux sur puce.
- Proposition d'un flot de conception pour la génération des réseaux sur puce.
- Proposition d'une plateforme pour la génération des réseaux sur puce en langage de description matérielle VHDL.
- Une analyse détaillée basée sur la simulation des performances de la plateforme proposée pour la génération des réseaux sur puce.
- Proposition d'une architecture à base de réseaux sur puce sans buffer pour les applications d'encodage vidéo (cas de l'encodeur H.264/AVC) ayant pour objectif de réduire la consommation en ressources logiques et d'atteindre une plus grande fréquence de fonctionnement.
- Comparaison des performances de l'architecture proposée à des travaux similaires.

## 4. Cadre Conceptuel des travaux

Ce travail entre dans le cadre des thématiques de recherche abordées par le laboratoire LERICA de l'Université Badji Mokhtar-Annaba à travers les thèses proposées en collaboration avec le laboratoire LE2I de l'Université de Bourgogne-Dijon, France. Ces sujets de recherche s'articulent autour du développement d'architectures et de systèmes numérique pour les applications dédiées aux systèmes embarqués. L'équipe « Architecture des Systèmes Electroniques de vision » du laboratoire LE2I possède une longue expérience dans le domaine de la conception d'architectures purement numériques à base de technologie adaptative par reconfiguration dynamique pour les systèmes embarqués. Cette coopération a notamment été renforcée par un projet de recherche international PHC Tassili (CMEP) de 2011-2014 intitulé : Modélisation, Simulation et Génération d'une Architecture Multiprocesseur basée sur un NoC de type Whishbone. Cette thèse s'inscrit dans la continuité des travaux de recherche qui s'intéressent à la réalisation

de systèmes microélectroniques numériques embarqués selon une approche structurelle SoC pour les applications embarquées fortement communicantes où l'adaptabilité et la flexibilité sont des paramètres essentiels.

## 5. Plan du manuscrit

Ce manuscrit est divisé en deux parties principales à travers lesquelles les travaux effectués le long de la thèse sont présentés.

La première partie comportant deux chapitres est consacrée à l'état de l'art.

Chapitre 1 présente dans une première partie un rappel sur les concepts généraux concernant les systèmes multiprocesseurs sur puce. Une mise en évidence sur les défis des systèmes embarqués modernes et les intérêts de l'utilisation de la solution basée sur les MP-SoCs. La deuxième partie du chapitre est consacrée à l'introduction des différentes technologies pour les solutions logicielles ou matérielles des applications destinées aux systèmes sur puce.

Le deuxième chapitre présente un état de l'art sur les solutions d'interconnexion pour les systèmes sur puce. La problématique des interconnexions sur puce (On-Chip interconnection) est détaillée avant de mettre en avant la solution des réseaux sur puce. Cette approche est ensuite détaillée dans la deuxième partie de ce chapitre. L'architecture générique des routeurs pour NoC et les différents aspects tels que les algorithmes de routage, les mécanismes de commutation sont présentés. Ce chapitre est conclu par un tour d'horizon des métriques pour les architectures de réseaux sur puce.

La deuxième partie du manuscrit débute par le troisième chapitre qui s'intéresse premièrement au protocole Whishbone : son architecture, ses différents signaux et ses particularités. L'intérêt de l'adoption d'un tel protocole pour les systèmes sur puce est présenté. Le deuxième volet de ce chapitre est consacré au flot de conception d'une architecture de réseau sur puce et à l'adaptation du protocole Wishbone aux réseaux sur puce. Les interfaces réseau de type Maître/Esclave sont présentées. Un flot de conception pour la génération des réseaux sur puce de type Wishbone est proposé. En suite la plateforme de génération des codes VHDL des NoCs est validée et testée par simulation sur l'environnement ISE Suite Design de Xilinx.

Le quatrième chapitre consiste en une application de la plateforme pour la génération des NoCs pour les applications vidéo. Ce chapitre présente aussi une architecture à base de routeur sans buffer pour la connexion des blocs de l'encodeur H.264/AVC. Cette architecture est décrite en langage de description matérielle via l'outil ISE de

Xilinx, puis simulée sur l'environnement de simulation fournit par Modelsim de Mentor Graphics.

Enfin, la dernière partie du manuscrit est consacrée à la synthèse des différents travaux réalisés, suivie d'une discussion générale sur les contributions avant d'ouvrir les perspectives et les pistes à explorer pour les travaux de recherche futurs.

# Chapitre I

# CHAPITRE I

# Généralités sur les systèmes multiprocesseurs sur puce (MP-SoCs)

## 1. Introduction

Les systèmes multiprocesseurs sur puce sont la dernière incarnation de la technologie d'intégration à très grande échelle (VLSI). Un seul circuit intégré peut contenir des centaines de millions de transistors, et la feuille de route technologique internationale pour les semi-conducteurs (ITRS : International Technology Roadmap for Semi-conductors) prédit que des puces avec quelques milliards de transistors sont à portée de main. Exploiter toute cette puissance informatique brute incite les designers à aller au-delà la conception logique pour adopter une architecture informatique. Les exigences imposées à ces puces par les applications obligent les concepteurs à faire face à des problèmes qui ne sont pas confrontés à l'architecture informatique traditionnelle : les délais en temps réel, le fonctionnement à très faible puissance, etc. Ces opportunités et défis font du design MPSoC un domaine de recherche important [4].

## 2. Que sont les MPSoCs?

Commençons d'abord par définir le système sur puce (SoC). Un SoC est un circuit intégré qui met en œuvre la plupart ou la totalité des fonctions d'un système électronique complet. La caractéristique la plus fondamentale d'un SoC est la complexité. Une puce mémoire peut avoir plusieurs transistors, mais sa structure régulière en fait un composant et non un système [5].

De plus les composants assemblés sur le SoC varient en fonction des applications. De nombreux SoCs contiennent des circuits analogiques et à signaux mixtes pour l'entrée/sortie (E/S). Bien que certaines applications d'E/S hautes performances requièrent une puce d'interface analogique séparée qui accompagne un SoC numérique, la grande partie du SoC reste numérique, car c'est la seule façon de construire des fonctions aussi complexes de manière fiable. Le système peut contenir de la mémoire, des processeurs de jeu d'instructions (unités centrales de traitement [UC]), une logique spécialisée, des bus et d'autres fonctions numériques. L'architecture du système est généralement adaptée à l'application plutôt que d'être une puce polyvalente : nous discuterons des motivations pour les architectures personnalisées et hétérogènes dans la section suivante.

Les systèmes sur puce sont présents dans de nombreuses catégories de produits allant des appareils grand public aux systèmes industriels :

- Les téléphones cellulaires utilisent plusieurs processeurs programmables pour gérer les tâches de traitement du signal et de protocole requis par la téléphonie. Ces architectures doivent être conçues pour fonctionner au niveau de la faible puissance fournies par les batteries.
- Les télécommunications et la mise en réseau utilisent des systèmes sur puce spécialisés, tels que les processeurs de réseau, pour gérer les énormes débits de données présentés par les équipements de transmission modernes.
- L'équipement de production de télévision utilise des systèmes sur puce pour coder la vidéo. L'encodage de la vidéo haute définition en temps réel nécessite des taux de calcul extrêmement élevés.
- Les téléviseurs numériques et les boîtiers décodeurs utilisent des multiprocesseurs sophistiqués pour effectuer des fonctions de décodage vidéo et audio en temps réel et d'interface utilisateur.
- Les jeux vidéo utilisent plusieurs machines de traitement parallèles complexes pour rendre l'action de jeu en temps réel.

Ces applications n'utilisent pas d'architectures informatiques à usage général, soit parce qu'une machine à usage général n'est pas rentable ou parce qu'elle ne fournit tout simplement pas les performances nécessaires. Les appareils grand public doivent se vendre à des prix extrêmement bas. En haut de gamme, les machines universelles ne peuvent tout simplement pas suivre les débits de données pour la vidéo et le réseau haut de gamme ; ils ont également du mal à fournir des performances fiables en temps réel [6].

Alors, qu'est-ce qu'un MPSoC? C'est simplement un système sur puce qui contient plusieurs processeurs d'ensemble d'instructions. En pratique, la plupart des SoCs sont des MPSoCs car il est trop difficile de concevoir un système sur puce complexe sans avoir recours à plusieurs processeurs. Nous discuterons plus en détail la logique des multiprocesseurs dans la section suivante.

Prenons l'exemple illustré par la Figure 1.1 qui montre un schéma de principe pour un lecteur DVD / MP3, une puce qui contrôle un lecteur de DVD et qui décode les fichiers audio MP3. L'architecture d'un lecteur DVD est plus complexe mais présente de nombreuses caractéristiques similaires, en particulier dans les premières étapes du traitement. Ce diagramme de bloc résume l'interconnexion entre les différents éléments de traitement (PE : Processing Element). Bien que l'interconnexion soit un problème de mise en œuvre important, nous voulons d'abord nous concentrer sur la diversité des PEs utilisés dans un SoC.

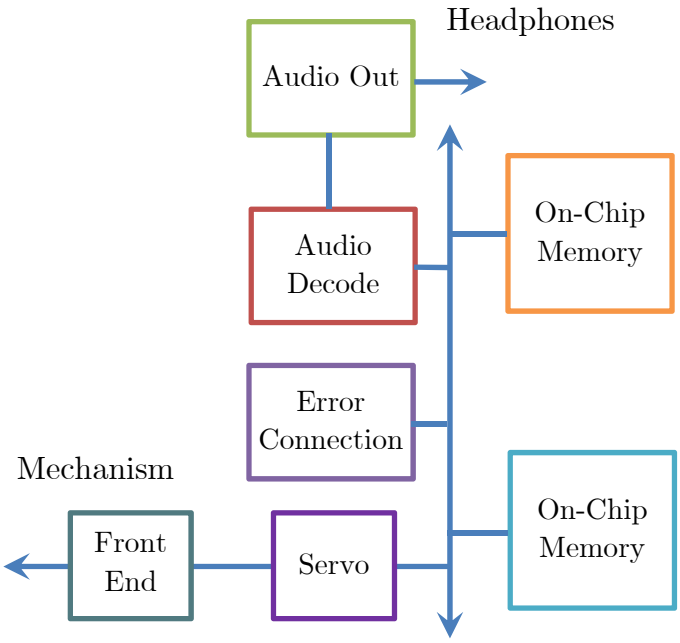



**Figure 1.1.** Architecture d'un lecteur DVD/MP3

Un petit nombre d'entrées analogiques provenant du capteur laser doit être décodé à la fois pour être sûr que le laser est sur la bonne voie et pour lire les données du disque. Un petit nombre de sorties analogiques contrôle l'objectif et le traîneau pour maintenir le laser sur la piste de données, qui est disposée en spirale autour du disque. Un pré-conditionnement du signal et un traitement de signal simple sont effectués dans des circuits analogiques, car c'est le seul moyen rentable de respecter les débits de données. Cependant, la plupart des circuits de commande pour le lecteur sont réalisés numériquement. Le lecteur de DVD est un triomphe du traitement du signal sur la

mécanique - un mécanisme très bon marché et de faible qualité est contrôlé par des algorithmes sophistiqués à des tolérances très fines. Plusieurs boucles de contrôle avec 16 prises ou plus sont généralement effectuées par un processeur de signal numérique (DSP) afin de contrôler le mécanisme d'entraînement du lecteur de DVD. Une fois que les bits bruts ont été lus sur le disque, une correction d'erreur doit être effectuée. Un algorithme de Reed-Solomon modifié est utilisé ; cette tâche est généralement effectuée par une unité spécialisée en raison des exigences de performance. Après la correction d'erreur, les bits de données MP3 doivent être décodés en données audio ; typiquement d'autres fonctions d'utilisateur telles que l'égalisation sont exécutées en même temps. Le décodage MP3 peut être effectué relativement bon marché, donc un CPU relativement peu sophistiqué est tout ce qui est requis pour cette phase finale. Un amplificateur analogique envoie l'audio aux écouteurs.

La Figure 1.2 montre l'architecture de la puce Emotion Engine de la Sony PlayStation 2 [7]. Le moteur d'émotion est un autre exemple de l'utilisation des SoCs au quotidien. Il est l'une des nombreuses puces complexes de la PlayStation 2. Il comprend un processeur universel qui exécute le jeu d'instructions de millions d'instructions par seconde (MIPS) et deux unités de traitement vectoriel, VPU0 et VPU1. Les deux unités de traitement vectoriel ont des architectures internes différentes. La puce contient 5,8 millions de transistors, fonctionne à 300 MHz.

Pourquoi nous soucions-nous de la performance ? Tout simplement car la plupart des applications pour lesquelles les SoCs sont utilisés ont des exigences de performance précises. Dans l'informatique interactive traditionnelle, nous nous soucions de la vitesse mais pas des délais. Les systèmes de contrôle, les protocoles et la plupart des systèmes du monde réel ne se soucient pas seulement de la performance moyenne, mais aussi du fait que les tâches sont effectuées dans un délai donné. La grande majorité des SoCs sont employés dans des applications qui ont au moins quelques échéances en temps réel. Les designers de matériel sont habitués à atteindre les objectifs de performance d'horloge, mais la plupart des délais s'étendent sur de nombreux cycles d'horloge [8].

Pourquoi nous soucions-nous de l'énergie ? Dans les appareils à piles, nous voulons prolonger la durée de vie de la batterie aussi longtemps que possible. Dans les appareils sans batterie, nous nous en soucions toujours car la consommation d'énergie est liée au coût. Si un périphérique utilise trop de puissance, il aura tendance à prendre plus de température et à surchauffer [9]. Au-delà d'une certaine température de fonctionnement, la puce doit être placée dans un emballage en céramique qui est beaucoup plus chers qu'un emballage en plastique.

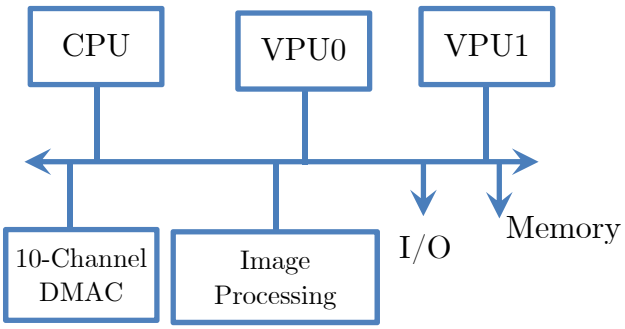



**Figure 1.2.** Architecture du moteur émotionnel Sony Playstation 2

Le fait qu'un MPSoC contienne plusieurs processeurs, signifie que la conception du logiciel fait partie intégrante de la conception globale de la puce. C'est un grand changement pour les designers de puces, qui sont habitués à proposer des solutions matérielles aux problèmes de conception sur puce. Dans un MPSoC, le matériel ou le logiciel peut être utilisé pour résoudre un problème ; ce qui dépend généralement de la performance, de la puissance et du temps de conception. Concevoir un logiciel pour un MPSoC est également un grand changement pour les concepteurs de logiciels. Les logiciels qui seront livrés avec une puce doivent être extrêmement fiables. Ce logiciel doit également être conçu pour répondre à de nombreuses contraintes de conception généralement réservées au matériel, telles que les contraintes de synchronisation et la consommation d'énergie. Cette fusion des disciplines de conception matérielle et logicielle est l'une des choses qui rendent le design de MP-SoC intéressant et stimulant [5].

Le fait que la plupart des MP-SoCs contiennent des multiprocesseurs hétérogènes les rend encore plus difficiles à programmer que les multiprocesseurs symétriques traditionnels. Les architectures régulières sont beaucoup plus faciles à programmer. Les multiprocesseurs scientifiques ont également convergé vers un modèle de mémoire partagée pour les programmeurs. Bien que ces architectures simples et régulières soient simples pour les programmeurs, elles sont souvent plus chères et moins énergivores que les architectures hétérogènes. La combinaison d'une fiabilité élevée, de performances en temps réel, d'un faible encombrement mémoire et de logiciels à faible consommation d'énergie sur un multiprocesseur hétérogène constitue un défi considérable dans la conception de logiciels destinés aux MP-SoCs [4].

Beaucoup de MP-SoCs ont besoin d'exécuter un logiciel qui n'a pas été développé par les concepteurs de puces. Comme les normes garantissent de grands marchés, les systèmes multi-puces sont souvent réduits aux SoCs seulement lorsque des normes

émergent pour l'application. Cependant, les utilisateurs de la puce doivent ajouter leurs propres caractéristiques au système pour différencier leurs produits des concurrents qui utilisent la même puce. Cela nécessite l'exécution d'un logiciel développé par le client et non par le concepteur de la puce. Les premiers systèmes VLSI avec des processeurs intégrés utilisaient généralement des environnements logiciels très rudimentaires qui auraient été impossibles à utiliser pour les concepteurs de logiciels externes. Les MP-SoCs modernes ont de meilleurs environnements de développement, mais créer un kit de développement logiciel différent pour chaque SoC est un défi en soi.

## 3. Pourquoi les systèmes multiprocesseurs sur puce ?

Un MP-SoC typique est un système multiprocesseur hétérogène : il peut y avoir différents types de PEs, le système de mémoire peut être distribué de manière hétérogène autour de la machine, et le réseau d'interconnexion entre les PEs et la mémoire peut également être hétérogène. Les MP-SoCs nécessitent souvent de grandes quantités de mémoire. Le périphérique peut disposer d'une mémoire embarquée sur puce, mais il peut aussi s'appuyer sur une mémoire standard hors puce.

Les exemples de SoCs vus précédemment implémentent, en fait, plusieurs processeurs hétérogènes. En revanche, la plupart des multiprocesseurs sont aujourd'hui beaucoup plus réguliers que les MP-SoCs typiques. La Figure 1.3 montre une architecture traditionnelle multiprocesseurs à mémoire partagée [11] ; un ensemble de processeurs et un autre ensemble de mémoires sont connectés par un réseau d'interconnexion. Chacun est généralement structuré régulièrement, et le programmeur reçoit un modèle de programmation régulier. Un modèle à mémoire partagée est souvent préféré car il simplifie la vie du programmeur. L'architecture brute [12] est un exemple récent d'architecture régulière conçue pour le calcul haute performance.

Plusieurs questions fondamentales se posent aux designers :

- Pourquoi ne pas utiliser une seule plate-forme pour toutes les applications ?
- Pourquoi ne pas construire des SoCs tels que des FPGAs (Field Programmable Gate Arrays), dans lesquels une seule architecture est construite dans une variété de tailles ?
- Et pourquoi utiliser une architecture multiprocesseur plutôt qu'une architecture monoprocesseur, qui a un modèle de programmation encore plus simple ?

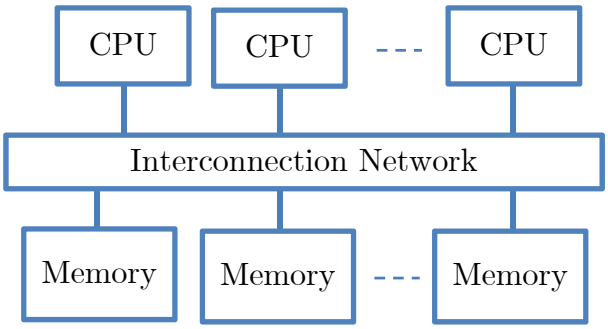



**Figure 1.3.** Architecture générique à base de multiprocesseurs et de mémoires partagées

Certains systèmes relativement simples sont, des systèmes monoprocesseurs et l'assistant numérique personnel (PDA) en est un excellent exemple. L'architecture du PDA typique ressemble à un PC, avec un processeur, des périphériques et de la mémoire attachée à un bus. Un PDA exécute beaucoup d'applications qui sont de petites versions d'applications de bureau, ainsi la ressemblance de la plate-forme de PDA à la plate-forme de PC est importante pour le développement de logiciel.

Cependant, les systèmes monoprocesseurs peuvent ne pas fournir assez de performances pour certaines applications. Les applications de base de données simples telles que les carnets d'adresses qui s'exécutent sur PDA peuvent facilement être traitées par des monoprocesseurs modernes. Mais lorsque nous passons à la vidéo ou aux communications en temps réel, les architectures multiprocesseurs sont généralement nécessaires pour suivre les débits de données entrants. Ces architectures multiprocesseurs fournissent la simultanéité computationnelle requise pour gérer les événements, du monde réel, simultanés en temps réel.

Les applications informatiques embarquées nécessitent généralement une véritable concurrence, et pas seulement la concurrence apparente d'un système d'exploitation multitâche fonctionnant sur un monoprocesseur. Le parallélisme au niveau des tâches est très important en informatique embarquée. La plupart des systèmes reposant sur des SoCs exécutent des tâches complexes composées de plusieurs phases. La Figure 1.4 montre le schéma fonctionnel du codage H.264/AVC [13]. Le codage vidéo nécessite plusieurs opérations simultanées : l'estimation de mouvement, la compensation de mouvement, la transformée en cosinus discrète (DCT) et la quantification entre autres. Les trames vidéo pénètrent généralement dans le système à 30 images / sec. Compte tenu de la grande quantité de calculs à effectuer sur chaque trame, ces étapes doivent être effectuées en parallèle pour respecter les délais. Ce type de parallélisme est relativement facile à exploiter car la spécification du système décompose naturellement le problème en tâches. Bien sûr, la décomposition qui convient le mieux à la spécification

peut ne pas être la meilleure façon de décomposer le calcul pour l'implémentation sur le SoC. C'est le travail des outils de conception de logiciels ou de matériel pour masser la décomposition en fonction des coûts de mise en œuvre.

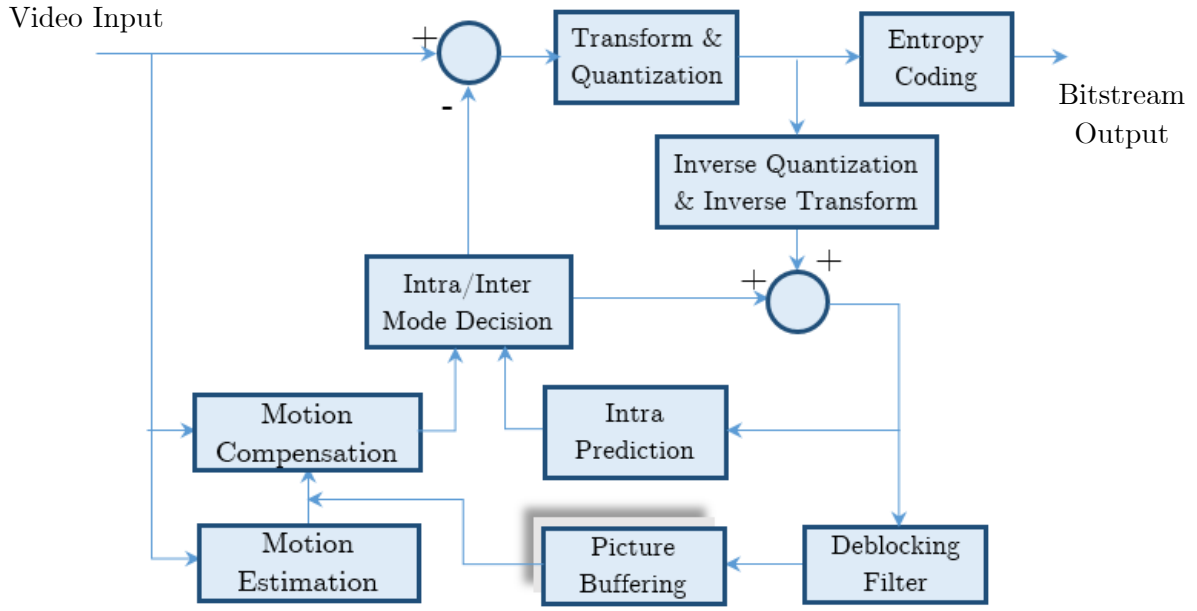



**Figure 1.4.** Schéma fonctionnel de l'encodeur H.264/AVC

Mais le fait d'avoir explicitement spécifié le parallélisme d'origine facilite la répartition de la fonctionnalité lors de la conception.

L'utilisation de processeurs génériques peut être envisagée pour atteindre les performances requises, car si nous pouvions utiliser la même architecture pour de nombreuses applications différentes, nous pourrions fabriquer les puces dans des volumes encore plus importants, permettant des prix plus bas. Les développeurs pourraient également développer plus facilement des logiciels car ils seraient familiers avec les plates-formes et disposeraient d'un ensemble d'outils plus riche. De plus, un processeur générique rendrait plus facile la cartographie d'une application sur l'architecture.

Cependant, nous ne pouvons pas appliquer directement le modèle informatique scientifique aux SoCs, car ces derniers doivent obéir à plusieurs contraintes qui ne s'appliquent pas au calcul scientifique :

- Ils doivent effectuer des calculs en temps réel.
- Ils doivent être efficaces dans la zone.
- Ils doivent être économes en énergie.
- Ils doivent fournir les bonnes connexions E/S.

Toutes ces contraintes poussent les concepteurs de SoC vers des systèmes multiprocesseurs hétérogènes.

L'informatique en temps réel est bien plus que de l'informatique haute performance. De nombreuses applications SoC nécessitent des performances très élevées, telles que le codage vidéo haute définition, mais ils exigent également que les résultats soient disponibles à un taux prévisible. Les variations de débit peuvent souvent être résolues en ajoutant une mémoire tampon, mais la mémoire entraîne des coûts de consommation d'énergie et de surface de silicium. S'assurer que le processeur peut produire des résultats à des moments prévisibles nécessite généralement une conception minutieuse de tous les aspects du matériel : jeu d'instructions, système de mémoire et bus système. Cela nécessite également une conception minutieuse du logiciel, à la fois pour tirer parti des caractéristiques du matériel et pour éviter les problèmes courants tels que le recours excessifs à la mise en mémoire tampon [14].

Les performances en temps réel reposent également sur un comportement prévisible du matériel. De nombreux mécanismes utilisés dans l'informatique générale, pour fournir des performances dans un modèle de programmation facile, rendent les performances du système moins prévisibles. La mise en cache de la surveillance, par exemple, gère dynamiquement la cohérence du cache mais au prix de retards moins prévisibles puisque le temps requis pour un accès mémoire dépend de l'état de plusieurs caches. Une façon de fournir des performances prévisibles et des performances élevées consiste à utiliser un mécanisme spécialisé pour les besoins de l'application : des systèmes de mémoire spécialisés ou des instructions spécifiques aux applications, entre autres. Et puisque différentes tâches dans une application ont souvent des caractéristiques différentes, différentes parties de l'architecture ont souvent besoin de structures matérielles différentes.

Les multiprocesseurs hétérogènes sont plus efficaces que les multiprocesseurs symétriques. De nombreux problèmes de calcul scientifique distribuent des données homogènes sur plusieurs processeurs ; notamment, ils peuvent décomposer une matrice en parallèle en utilisant plusieurs processeurs. Cependant, le parallélisme au niveau des tâches, que les applications informatiques intégrées affichent, est intrinsèquement hétérogène. Dans le diagramme de blocs H.264/AVC, comme avec d'autres applications, chaque bloc fait quelque chose de différent et a des exigences de calcul différentes.

Bien que l'hétérogénéité de l'application n'exige pas intrinsèquement l'utilisation d'un type de processeur différent pour chaque tâche, cela peut présenter des avantages significatifs. Un PE spécial peut être beaucoup plus rapide et plus petit qu'un processeur programmable ; ainsi, plusieurs machines d'estimation de mouvement très

petites et rapides ont été développées pour H.264/AVC. Même si un processeur programmable est utilisé pour une tâche, les processeurs spécialisés peuvent souvent améliorer les performances tout en économisant de l'espace. Par exemple, l'adaptation de la largeur du chemin de données du processeur aux tailles de données natives de l'application permet d'économiser une quantité considérable de surface. Le choix d'une taille de cache et d'une organisation correspondant aux caractéristiques de l'application peut grandement améliorer les performances.

La spécialisation de la mémoire est une technique importante pour concevoir des architectures efficaces. Un système de mémoire à usage général peut essayer de traiter des cas spéciaux à la volée en utilisant les informations recueillies pendant l'exécution, mais cela à un coût considérable en matériel. Si l'architecte du système peut prévoir certains aspects du comportement de la mémoire dans l'application, il est souvent possible de refléter ces caractéristiques dans l'architecture. La configuration du cache est un exemple idéal : un cache considérablement plus petit peut souvent être utilisé lorsque l'application dispose de modèles d'accès mémoire réguliers.

La plupart des conceptions de SoC sont sensibles à la puissance, qu'elles soient dues à des considérations environnementales (dissipation thermique) ou aux exigences du système (puissance de la batterie). Comme pour la zone, la spécialisation économise l'énergie. L'élimination des caractéristiques inutiles pour l'application réduit la consommation d'énergie ; ceci est particulièrement vrai pour la consommation d'énergie de fuite. Les multiprocesseurs scientifiques sont des équipements standards qui sont utilisés de différentes manières ; chaque installation d'un supercalculateur peut effectuer une tâche différente. En revanche, les SoCs sont des dispositifs de marché de masse en raison de l'économie de la fabrication VLSI. Les réglages de conception économisant de l'énergie pour une fonctionnalité d'architecture particulière peuvent donc être répliqués à plusieurs reprises au cours de la fabrication, ce qui réduit le coût de conception de ces fonctionnalités d'économie d'énergie.

Les SoCs nécessitent également des E/S spécialisées. Le but d'un SoC est de fournir un système complet. On espère que les dispositifs d'entrée et de sortie pourraient être mis en œuvre de manière générique avec suffisamment de transistors ; dans une certaine mesure, cela a été fait pour les blocs d'E/S des plateformes FPGA. Mais étant donné la variété des interfaces physiques existant, il peut être difficile de créer efficacement des périphériques d'E/S personnalisables.

On pourrait penser que l'augmentation du nombre de transistors pourrait plaider en faveur d'une tendance à s'éloigner des architectures hétérogènes et aller vers des

machines régulièrement structurées. Mais les applications continuent à absorber autant de puissance de calcul que peut fournir la loi de Moore. Les débits de données continuent d'augmenter dans la plupart des applications, par exemple, la communication de données, la vidéo et l'audio. De plus, les nouveaux appareils combinent de plus en plus ces applications. Un seul appareil peut effectuer une communication sans fil, une compression vidéo et une reconnaissance vocale. Les concepteurs de SoC ne commenceront à privilégier les architectures régulières que lorsque la pression de performance de l'application diminuera et que les performances des circuits intégrés seront rattrapées. Il ne semble pas que l'appétit des clients se rétrécisse de sitôt.

## 4. Les défis des MP-SoCs

Les réseaux sur puce sont apparus ces dernières années comme une approche architecturale de la conception de multiprocesseurs mono-puce. Un réseau sur puce utilise des réseaux de paquets pour interconnecter les processeurs dans le SoC. Bien que l'on sache beaucoup de choses sur les réseaux, la conception de réseau traditionnelle suppose relativement peu de caractéristiques du trafic sur le réseau. Les applications SoC peuvent souvent être bien caractérisées ; cette information devrait être utile pour spécialiser la conception du réseau afin d'améliorer le rapport coût / performance / puissance.

Les FPGAs sont apparus comme une alternative viable aux circuits intégrés spécifiques aux applications (ASIC) sur de nombreux marchés. Les tissus FPGA commencent également à être intégrés dans les SoCs. La logique FPGA peut être utilisée pour une logique personnalisée qui n'a pas pu être conçue avant la fabrication. Cette approche est un bon complément à la personnalisation basée sur le logiciel. Nous devons mieux comprendre où placer les tissus FPGA dans les architectures SoC afin qu'ils puissent être utilisés le plus efficacement possible. Nous avons également besoin d'outils pour aider les concepteurs à comprendre les performances et l'allocation à l'aide de FPGA en tant qu'éléments de traitement dans le multiprocesseur [15]. À mesure que les systèmes de navigation deviennent plus sophistiqués, et en particulier lorsqu'ils se connectent à Internet, la sécurité devient de plus en plus importante. Les failles de sécurité peuvent entraîner des dysfonctionnements, raison pour laquelle les architectures matérielles et logicielles doivent être conçues pour être sécurisées. De plus les méthodologies de conception doivent également être organisées pour s'assurer que les considérations de sécurité sont prises en compte et que les bugs menaçant la sécurité ne sont pas autorisés à se propager à travers la conception.

Enfin, les MP-SoCs seront de plus en plus connectés dans des réseaux de puces [16]. A titre d'exemple, les systèmes automobiles et avioniques utilisent depuis longtemps des réseaux pour connecter des puces physiquement séparées. Mais, plus important, les MP-SoCs structurés en réseaux sur puces n'ont pas un contrôle total sur le système. Lors de la conception d'une seule puce, l'équipe de conception a un contrôle total sur ce qui se passe sur la puce. Lorsque ces puces sont assemblées en réseaux, l'équipe de conception a moins de contrôle sur l'organisation du réseau dans son ensemble. Non seulement le SoC peut être utilisé de manière inattendue, mais le réseau de la configuration peut changer au fil du temps au fur et à mesure que des nœuds sont ajoutés et supprimés.

### 4.1. Temps de conception et temps de mise sur le marché

Les demandes des consommateurs ont forcé les sociétés de semi-conducteurs à introduire et à mettre à niveau régulièrement leurs produits. Prenons l'exemple des entreprises de téléphonie mobile qui doivent publier de nouveaux modèles dotés de fonctionnalités innovantes (telles que la détection de visages, des caméras à pixels plus élevés, etc.) tous les six mois environ pour maintenir leur clientèle. De plus, elles doivent publier plusieurs variantes d'un téléphone portable pour capturer les diverses attentes des utilisateurs et ainsi survivre à la concurrence. Ces facteurs de marché ont eu pour effet de raccourcir les délais de conception et les délais de commercialisation des MP-SoCs hétérogènes, ce qui démontre la nécessité de disposer de cadres d'automatisation de la conception.

### 4.2. Prix des produits

La conception des MP-SoCs hétérogènes complexes dans des délais très courts avec des contraintes de temps de mise sur le marché exigent un investissement dans de grandes équipes de conception talentueuses. Cependant, de tels investissements signifient que les prix des produits augmenteront, ce qui est inacceptable sur les marchés concurrentiels de consommation, car les utilisateurs préfèrent toujours acheter une technologie de pointe au prix le plus bas. Les techniques d'automatisation de la conception peuvent exécuter des phases lourdes du cycle de conception avec peu ou quasiment pas d'intervention du concepteur, ce qui raccourcit le délai de conception et le délai de commercialisation, réduisant ainsi les prix des produits. Par conséquent, une entreprise dotée d'un ou de plusieurs systèmes d'automatisation de la conception est plus susceptible de rivaliser sur le marché de la consommation en proposant des produits à la fois peu coûteux et innovants.

### 4.3. Complexité de conception

L'espace de conception des MP-SoCs hétérogènes explose en raison de la présence de diverses options telles que des éléments de traitement, des hiérarchies de mémoire, une infrastructure de communication et des modèles d'application / de programmation. Par exemple, un MP-SoC hétérogène devrait-il utiliser les ASIPs, les DSPs ou les deux, et combien de chaque type ? Dans une infrastructure de communication, entre autres, un concepteur doit choisir des buffers point à point, des buffers partagés, leurs tailles, ainsi que des réseaux sur puce. Les choix dans la hiérarchie de la mémoire comprennent le nombre de niveaux de cache, la configuration des caches et la taille des mémoires locales et partagées. L'exploration d'un espace de conception aussi diversifié ne peut être faite manuellement et nécessite donc des techniques d'exploration intelligemment conçues. En outre, l'exploration de l'espace de conception devrait être rapide et réalisable dans le cadre de l'automatisation de la conception.

### 4.4. Flexibilité et scalabilité

L'hétérogénéité d'un MP-SoC devrait être suffisamment souple pour permettre la mise en œuvre de plusieurs normes hétérogènes afin de pouvoir déployer rapidement plusieurs variantes d'un produit. En outre, il devrait permettre des correctifs et des mises à niveau rapides du produit après le déploiement. Ces exigences indiquent l'utilisation d'un traitement programmable des éléments tels que les DSPs et les ASICs comme éléments constitutifs d'un MP-SoC hétérogène. L'évolutivité au moment du design implique que le MPSoC devrait permettre d'ajouter facilement des composants à l'avenir pour gérer la complexité croissante des applications nouvelle génération sans effort majeur de refonte.

### 4.5. Contraintes de performance, de surface et d'énergie

Les applications modernes ont souvent des contraintes de performances qui doivent être remplies par des MP-SoCs hétérogènes. En outre, ces MP-SoCs sont déployés sur des périphériques embarqués fonctionnant avec des batteries standards, ce qui favorise l'encombrement minimal et la consommation électrique la plus faible possible. L'exploration de l'espace de conception, comme expliqué ci-dessus, doit être effectuée pour choisir le bon nombre et le bon type d'éléments de traitement, configurations de cache, tailles de mémoire, types de technique à faible consommation, etc.

### 4.6. Adaptabilité

Les exigences en termes de puissance de calcul d'une application exécutée évoluent avec le temps, nécessitant une adaptation des applications et des architectures au moment de l'exécution.

Prenons l'exemple d'un encodeur vidéo qui peut être saisi avec une vidéo contenant des mouvements faibles et des mouvements élevés. Les images vidéo à mouvement élevé nécessitent beaucoup plus de calculs que les images vidéo à faible mouvement. Par conséquent, une MP-SoC hétérogène devrait adapter son utilisation des ressources (éléments de traitement, mémoires, etc.) au moment de l'exécution en fonction de la charge de travail actuelle plutôt que de fonctionner dans le pire des cas (toutes les ressources sont actives). Une telle adaptation est nécessaire pour un fonctionnement à très basse consommation des MP-SoCs hétérogènes afin d'augmenter la durée de vie de la batterie dans les appareils portables. Les techniques de gestion du temps d'exécution doivent être utilisées pour gérer les ressources dans un système MP-SoC hétérogène, de sorte qu'il fonctionne toujours avec la consommation électrique la plus faible possible.

## 5. Les plateformes de mise en œuvre d'applications numériques

Les FPGAs sont des dispositifs reconfigurables qui peuvent exécuter diverses applications matérielles. Un flux CAO logiciel transforme une application matérielle en un flux binaire de programmation, qui peut être facilement et instantanément programmé sur un FPGA. Cette reprogrammabilité d'un périphérique FPGA peut être utilisée pour exécuter différentes applications matérielles à des instants mutuellement exclusifs. De même, toute erreur ou mise à jour du produit final peut être corrigée / mise à niveau en reprogrammant simplement le FPGA. Un FPGA permet également une reconfiguration partielle, c'est-à-dire qu'une partie d'un FPGA est configurée alors que d'autres parties sont encore en cours d'exécution. La reconfiguration partielle est utile dans la conception de systèmes qui nécessitent une adaptation fréquente en fonction des contraintes d'exécution. Comparées à d'autres technologies qui fabriquent des circuits d'application spécifiques sur silicium, les applications basées sur FPGA ont un coût d'ingénierie non récurrent (NRE : Non Recurring Engineering) moins élevé et un temps de mise sur le marché plus court. Ces avantages rendent les produits basés sur FPGA très efficaces et économiques pour une production de faible à moyenne production.

La flexibilité et la réutilisabilité d'un FPGA sont dues à ses blocs logiques configurables qui sont interconnectés via des ressources de routage configurables. Une conception d'application peut facilement être mappée sur ces ressources configurables d'un FPGA en utilisant un flux logiciel dédié. Cette application est initialement synthétisée en blocs logiques interconnectés (comprenant généralement des tables de correspondance et des bascules). Les instances de blocs logiques du circuit synthétisé sont ensuite placées sur des blocs logiques configurables de FPGA. Le placement est effectué de telle manière que des ressources de routage minimales sont requises pour les interconnecter. Les connexions entre ces blocs logiques sont ensuite routées à l'aide de ressources de routage configurables. Les blocs logiques et les ressources de routage sont programmés par des RAM statiques (SRAM) qui sont distribuées à travers le FPGA. Une fois qu'un circuit d'application est placé et routé sur le FPGA, les informations de bits SRAM de l'ensemble du FPGA sont rassemblées pour former un bitstream. Ce dernier est ensuite programmé sur les SRAM de la plateforme FPGA par un chargeur de bitstream intégré dans le FPGA.


**Figure 1.5.** Comparaison de différentes plateformes utilisées pour la mise en œuvre d'applications numériques

Le dispositif FPGA peut être comparé à d'autres dispositifs de calcul numériques. La section ci-dessous présente un aperçu général et leur comparaison avec les FPGAs.

### 5.1. Microprocesseurs

Un microprocesseur est un périphérique matériel polyvalent pouvant exécuter une tâche logicielle. La tâche logicielle est représentée sous la forme d'un flux d'instructions logicielles ; chaque instruction appartient à un ensemble d'instructions prédéfini. Les instructions d'une tâche sont exécutées sur le microprocesseur de manière sérielle. Ces instructions sont stockées dans la mémoire d'instructions, tandis que les données à traiter sont stockées dans la mémoire de données ou dans des registres à usage général. Un FPGA peut être comparé à un microprocesseur, car les deux sont flexibles et reprogrammables. Cependant, la configurabilité des FPGAs est fondamentalement

différente de la programmabilité d'un microprocesseur traditionnel. Un microprocesseur peut exécuter des applications logicielles, tandis qu'un FPGA peut exécuter des applications matérielles. Les blocs logiques fonctionnels d'un microprocesseur traditionnel ne peuvent pas être configurés pour différents programmes logiciels. Alors que les blocs logiques dans un FPGA peuvent être configurés différemment pour chaque application.

Le nombre limité d'unités fonctionnelles non reconfigurables dans un microprocesseur est soigneusement sélectionné par l'architecte du microprocesseur en fonction des besoins logiciels génériques. Ces unités fonctionnelles et leur intercommunication sont optimisées pour atteindre des fréquences d'horloge très élevées. Un microprocesseur prend un nombre variable de cycles d'horloge pour exécuter différentes instructions logicielles.

L'exécution d'une conception d'application matérielle bénéficie d'un parallélisme inhérent. Au contraire, un flux d'instructions logicielles est exécuté séquentiellement sur un microprocesseur. L'implémentation matérielle peut utiliser n'importe quelle fonctionnalité matérielle requise pour exécuter une tâche particulière. Alors qu'une implémentation logicielle peut nécessiter plusieurs instructions logicielles pour exécuter la même tâche. Pour ces raisons, la mise en œuvre matérielle d'un design particulier est généralement beaucoup plus rapide que son implémentation logicielle.

### 5.2. Processeurs à usage général (GPP : General Purpose Processors)

Les processeurs à usage général offrent une solution logicielle pure qui facilite le temps de conception et le temps de mise sur le marché grâce à la réutilisation du code, et permet de mettre à niveau et de corriger facilement les produits. En outre, la programmabilité des GPPs aide à réduire le temps de mise sur le marché et donc à réduire les coûts. Comme les GPPs ne peuvent pas être optimisés pour des applications spécifiques, ils offrent beaucoup moins de performances et consomment beaucoup plus d'énergie que les ASICs. L'analyse quantitative réalisée par [17] fait état d'une différence d'au moins cinq ordres de grandeur en matière d'efficacité énergétique et d'efficacité en surface logique entre les ASICs et les GPPs.

### 5.3. Processeurs de signaux numériques (DSP : Digital Signal Processors)

Les processeurs de signaux numériques, remplaçant des GPPs, sont des processeurs spécifiques aux domaines, conçus pour exécuter efficacement des applications appartenant à un certain domaine. Les DSPs offrent une meilleure efficacité en ressources logiques et énergétique que les GPPs [18, 19] en raison d'instructions

spécifiques au domaine, de multiples unités fonctionnelles spécifiques au domaine et de l'exploitation des instructions et des parallélismes au niveau des données. La technique « Very Long Instruction Word » (VLIW) permet à un DSP d'exécuter plusieurs opérations en parallèle, et le compilateur est responsable de l'encapsulation de plusieurs opérations dans une seule instruction. D'un autre côté, la technique SIMD (Single Instruction Multiple Data) permet à une instruction d'exécuter une opération sur plusieurs données en parallèle. VLIW et SIMD permettent aux DSPs d'exploiter le parallélisme au niveau des instructions et des données disponibles dans les applications multimédias [20].

### 5.4. Processeur à jeu d'instructions pour application spécifique (ASIPs)

Un ASIP (Application Specific Instruction-set Processor) est conçu en fonction des exigences d'une application particulière. L'ensemble d'instructions d'un ASIP est spécifiquement conçu pour accélérer les fonctions complexes les plus couramment utilisées requises par le domaine d'application particulier. Un microprocesseur à usage général est généralement conçu pour obtenir des performances maximales avec une flexibilité maximale. Considérant que, un ASIP réduit la flexibilité pour obtenir de meilleurs gains de performance pour un domaine d'application prédéfini. Les exemples ASIP incluent DSP, médias, réseau et autres processeurs spécifiques.

### 5.5. Circuits intégrés spécifiques pour application spécifique (ASIC)

Un ASIC (Application Specific Integrated Circuit) est un périphérique numérique personnalisé pour effectuer une tâche particulière. La fonctionnalité d'un ASIC est définie en utilisant un langage de description de matériel (HDL) tel que Verilog ou VHDL. Après la vérification fonctionnelle de la description matérielle, la fonctionnalité est synthétisée dans une liste de portes de bas niveau appelée cellules standards (standard-cells). Les cellules standards sont issues d'une bibliothèque de cellules standards pré-caractérisée qui comprend des portes logiques telles que : AND à 2-entrées, NOR 2-entrées, etc. Après la synthèse, un outil de placement place les cellules standards sur une région de l'ASIC. Les cellules standards sont placées en tenant compte de diverses contraintes de placement. Après le placement, l'outil de routage utilise les informations de placement et connecte les connexions électriques entre les cellules standards. La sortie finale est utilisée pour fabriquer un CI physique de l'ASIC. La bibliothèque de cellules standards peut être utilisée avec des systèmes de CAO modernes qui peuvent offrir des avantages considérables en termes de rapport « performances/coûts ».

La disposition peut être conçue en toute personnalisation en disposant chaque transistor individuel et leur interconnexion au lieu d'utiliser la bibliothèque de cellules standards. Une telle conception entièrement personnalisée peut améliorer la surface et les performances de l'ASIC, mais cette tâche est extrêmement laborieuse et nécessite des compétences supérieures. Elle augmente le temps de conception, le coût de l'ingénierie et le délai de mise sur le marché (Time-to-Market) du produit.

Quand un FPGA est comparé à un ASIC, il est révélé que tous les avantages d'un FPGA viennent avec un coût énorme. Un circuit implémenté sur un FPGA est généralement 35 fois plus grand, 4 fois plus lent et 14 fois plus puissant que le même circuit implémenté dans un ASIC à base de cellules standards [21]. Par conséquent, les FPGAs ne conviennent pas pour les applications nécessitant une production à haut volume, de hautes performances ou une faible consommation d'énergie. Cependant, les ASICs souffrent d'une inflexibilité, d'un coût d'ingénierie plus élevé et d'un temps de mise sur le marché plus élevé. Ainsi, les ASICs ne conviennent que s'ils sont produits en grandes quantités. De plus, toute correction ou mise à jour d'un produit basé sur ASIC nécessite un autre processus de production, ce qui augmente encore le temps de mise sur le marché.

## 6. Les FPGAs (Field programmable Gate Array)

Un réseau de portes programmable par l'utilisateur est un circuit intégré conçu pour être configuré après la fabrication. Les FPGAs peuvent être utilisés pour implémenter n'importe quelle fonction logique qu'un ASIC peut exécuter. Pour des besoins variables, une partie du FPGA peut également être partiellement reconfigurée alors que le reste d'un FPGA est toujours en cours d'exécution. Contrairement à d'autres technologies, qui implémentent le matériel directement dans le silicium, toute erreur dans le produit FPGA final peut être facilement corrigée, simplement en reprogrammant le FPGA. Toutes les futures mises à jour du produit final peuvent également être facilement appliquées en téléchargeant simplement un nouveau bitstream de l'application. La facilité de programmation et de débogage avec les FPGAs diminuent les coûts globaux d'ingénierie et le temps de la mise sur le marché des produits FPGA.

La reconfigurabilité des FPGAs est due à leurs composants reconfigurables appelés blocs logiques (Logic Blocs), qui sont interconnectés par un réseau de routage reconfigurable. Il existe deux principales topologies d'interconnexion de routage : le réseau de routage basé sur une structure en arbre [22] et le réseau de routage basé sur un maillage [23].

Une architecture FPGA basée sur une structure en arbre est créée en connectant des blocs logiques dans des groupes. Ces groupes sont connectés récursivement pour former une structure hiérarchique, au contraire d'une architecture FPGA à mailles qui interconnecte les blocs logiques via un maillage 2-D de réseau de routage. La topologie d'interconnexion en arbre souffre de problèmes d'évolutivité, mais elle occupe moins de surface que la topologie d'interconnexion en maillage [24]. La disposition d'un FPGA à base de mailles est évolutive et est donc couramment utilisée par les fournisseurs FPGA commerciaux tels que Xilinx et Altera.

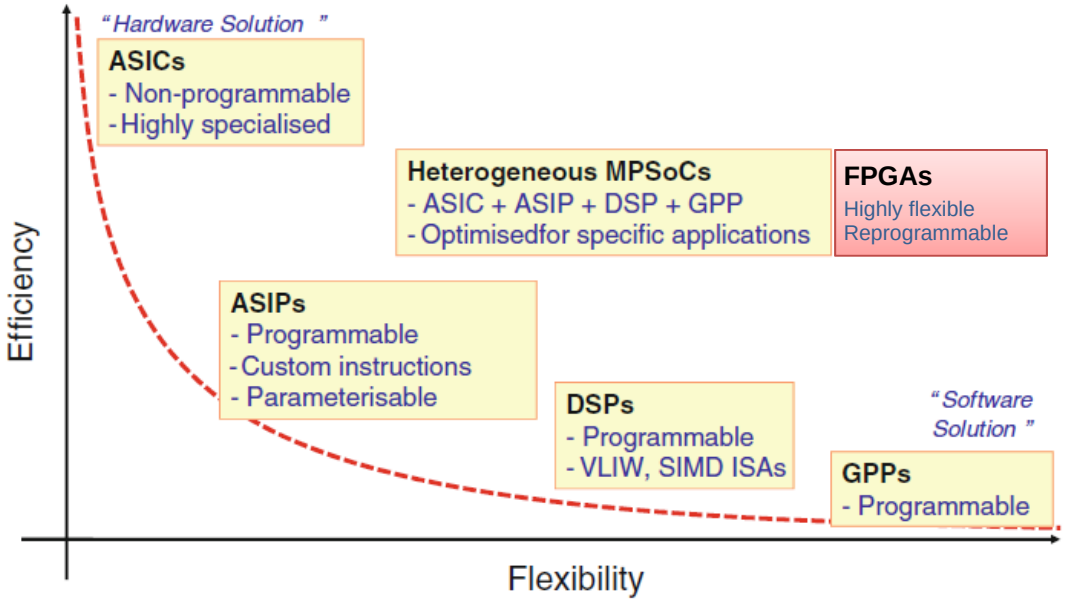



**Figure 1.6.** Comparaison entre les différentes solutions d'implémentation matérielle

La Figure 1.7 montre une architecture FPGA basée sur un maillage traditionnel. Les blocs logiques configurables (CLB) sont disposés sur une grille 2D et sont interconnectés par un réseau de routage programmable. Les blocs d'entrée/sortie (I/O B) sur la périphérie de la puce FPGA sont également connectés au réseau de routage programmable. Le réseau de routage comprend des pistes de canaux de routage horizontales et verticales.

Les boîtiers de commutation connectent les voies de routage horizontales et verticales du réseau de routage. Les boîtiers de connexion connectent les broches logiques et de blocs I/OB aux pistes de routage adjacentes. Un flux logiciel convertit un circuit matériel cible en CLB et entrée/sortie interconnectés, puis les mappe sur le FPGA. Le flux logiciel génère également un bitstream, qui est programmé sur le FPGA pour exécuter le circuit matériel cible.

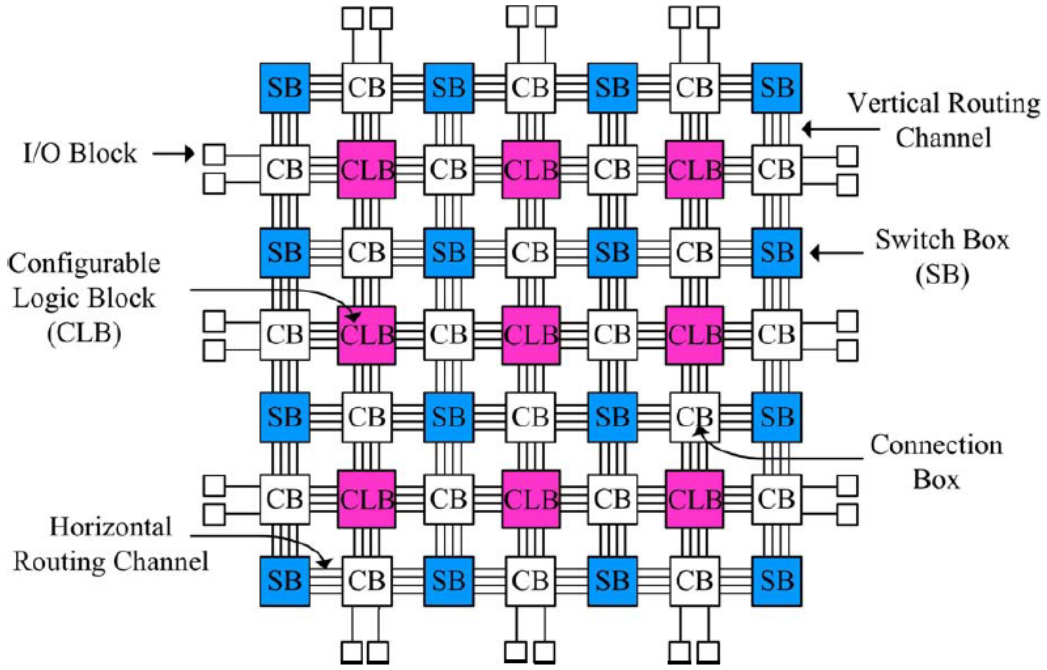



**Figure 1.7.** Vue d'ensemble de l'architecture FPGA

### 6.1. Bloc logique configurable

Le bloc logique configurable (CLB) est un composant de base d'un FPGA qui implémente la fonctionnalité logique d'une conception d'application cible. Un CLB peut comprendre un seul élément logique de base, ou un groupe de BLE interconnectés localement. Un BLE simple se compose d'une table de correspondance (LUT : Look Up Table) et d'une bascule. Une LUT avec k entrées (LUT-k) contient le nombre de « 2k bits » de configurations ; elle peut implémenter n'importe quelle fonction booléenne ayant k entrées. La Figure 1.8 montre un BLE simple comprenant une table de correspondance à 4 entrées (LUT-4) et une bascule de type D. La LUT-4 utilise une SRAM 16 bits (Static Random Access Memory) pour implémenter une fonction booléenne à 4 entrées. La sortie la LUT-4 est connectée à une bascule optionnelle. Un multiplexeur sélectionne la sortie BLE pour être soit la sortie d'une bascule ou de la LUT-4.

Une table de correspondance avec plus de nombre d'entrées réduit le nombre total de LUT requis pour mapper un circuit matériel. Plus de fonctionnalités logiques peuvent être mappées dans une seule LUT. Ceci réduit éventuellement l'intercommunication entre les LUTs, ainsi la vitesse du circuit matériel s'améliore. Cependant, une LUT avec plus de nombre d'entrées augmente l'occupation en surface de silicium de façon exponentielle. Des travaux ont permis de mesurer l'effet du nombre

d'entrées de LUT sur la surface, la vitesse et la routabilité des FPGAs [25]. Ils ont conclu que les LUTs à 4 entrées fournissent un bon compromis entre la vitesse et la densité des

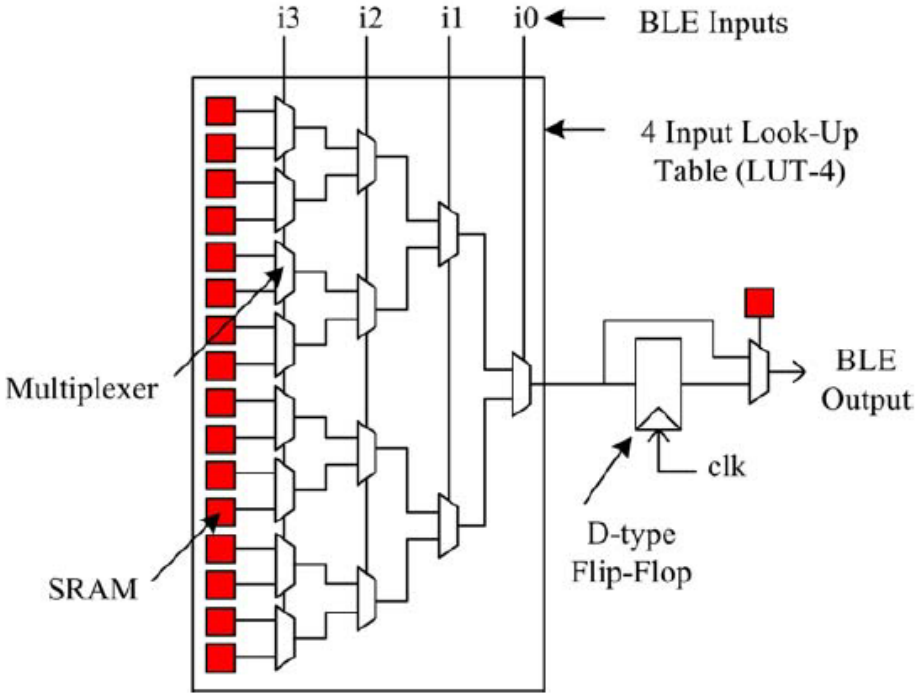



**Figure 1.8.** Element Logique de Base (BLE)

Un CLB peut contenir un groupe de BLE connectés via un réseau de routage local. La Figure 1.9 montre un groupe de 4 BLE; dont chaque BLE contient un LUT-4 et une bascule.

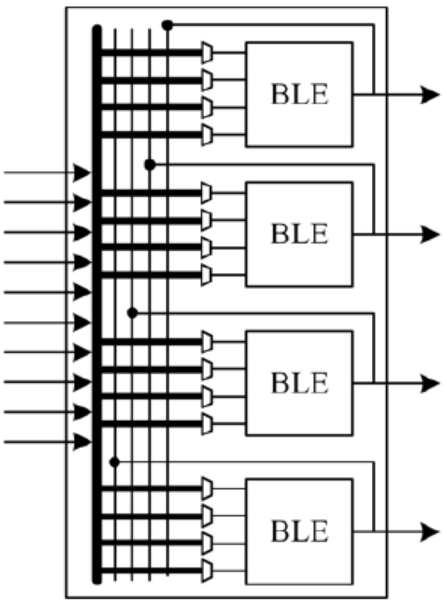



**Figure 1.9.** Bloc Logique Configurable ayant 4 BLEs

La sortie d'un BLE est accessible aux autres BLEs du même groupe via un réseau de routage local. Le nombre de broches de sortie d'un groupe est égal au nombre total de fichiers BLEs dans un groupe. Considérant que le nombre de broches d'entrée d'un groupe peut être inférieur ou égal à la somme des broches d'entrée requises par tous les BLEs dans le groupe. Les FPGAs modernes contiennent généralement de 4 à 10 BLEs dans un même groupe.

### 6.2. Réseau de routage

Le réseau de routage d'un FPGA occupe 80 à 90% de la surface de la puce FPGA, alors que la zone logique occupe seulement 10 à 20% de la surface [23]. La flexibilité d'un FPGA dépend principalement de son réseau de routage programmable. Un réseau de routage FPGA maillé est constitué de pistes de routage horizontales et verticales qui sont interconnectées via des boîtiers de commutation (SB :Switch Boxes). Les blocs logiques sont connectés au réseau de routage via des boîtiers de connexion (CB : connexion boxes).

La flexibilité d'un bloc de connexion (Fc) est le nombre de pistes de routage du canal adjacent qui sont connectées à la broche d'un bloc. La connectivité des broches d'entrée des blocs logiques avec le canal de routage adjacent est appelée « Fcin »; la connectivité des broches de sortie des blocs logiques avec le canal de routage adjacent est appelée « Fcout ». Un « Fcin » égal à 1,0 signifie que toutes les pistes du canal de routage adjacent sont connectées à la broche d'entrée du bloc logique. Un « Fcin » égal à 0,5 signifie que seulement 50% des pistes du canal de routage adjacent sont connectées à la broche d'entrée.

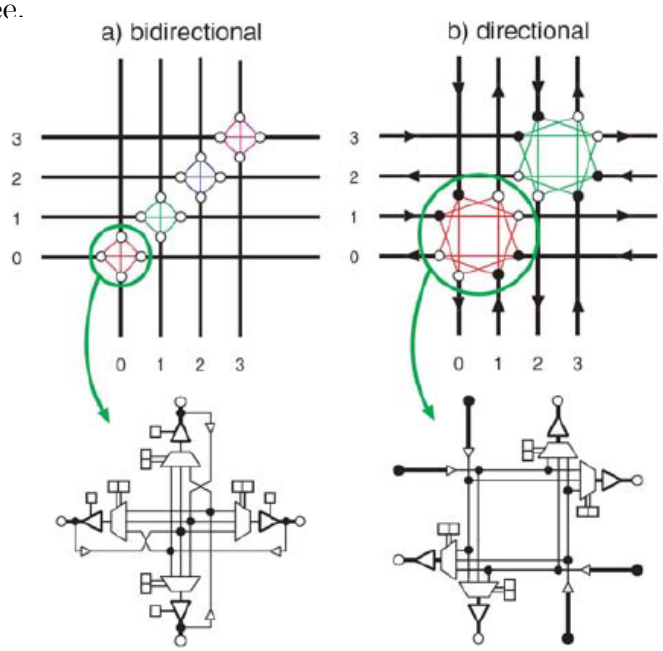



**Figure 1.10.** Bloc de commutation, connexion de longueur « 1 » [26]

La flexibilité de la boîte de commutation (Fs) est le nombre total de pistes avec lesquelles chaque piste entrant dans la boîte de commutation se connecte. Les pistes de routage connectées via une boîte de commutation peuvent être des pistes bidirectionnelles ou unidirectionnelles. La Figure 1.10 montre un boîtier de commutation bidirectionnel et unidirectionnel ayant Fs égal à 3. Les pistes d'entrée (ou fils) dans ces deux boîtiers de commutation se connectent à 3 autres pistes du même boîtier de commutation. La seule limitation de la boîte de commutation unidirectionnelle est que leur largeur de canal de routage doit être multiple de 1. Des fils multi-longueur sont créés pour réduire la consommation en surface et le latence. La Figure 1.11 montre un exemple de fils de longueurs différentes. Les segments de câble plus longs couvrent plusieurs blocs et nécessitent moins de commutateurs, ce qui réduit la surface de routage et la latence. Cependant, ils diminuent également la flexibilité de routage, ce qui réduit la probabilité d'acheminer un circuit matériel avec succès. Les FPGAs commerciaux modernes utilisent couramment une combinaison de fils longs et courts pour équilibrer la flexibilité, la surface et le retard du réseau de routage.

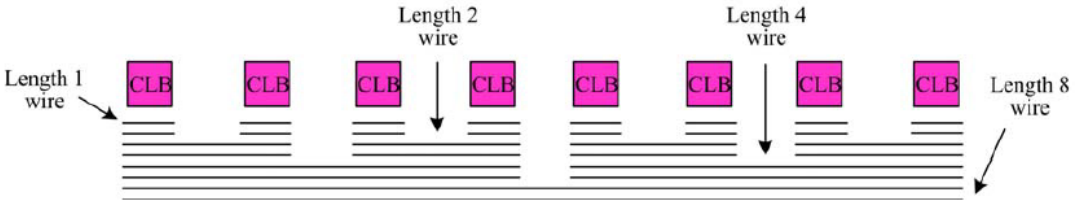



**Figure 1.11.** Distribution des segments d'un canal

Généralement, les broches de sortie d'un bloc peuvent se connecter à n'importe quelle piste de routage à travers des transistors de passage. Chaque transistor de passage forme une sortie à trois états qui peut être activée ou désactivée indépendamment. Cependant, une technique de câblage à conducteur unique peut également être utilisée pour connecter les broches de sortie d'un bloc aux pistes de routage adjacentes. Les architectures FPGA commerciales modernes ont évolué vers l'utilisation de pistes de routage directionnelles à un conducteur. Des travaux montrent que si le câblage directionnel à conducteur unique est utilisé à la place du câblage bidirectionnel, une amélioration de 25% de la surface, de 9% du retard et de 32% du retard à la surface peut être obtenue [26]. Tous ces avantages sont obtenus sans apporter de changements majeurs dans le flot de conception des FPGAs.

### 6.3. Flot de conception d'une architecture à base de plateforme FPGA

L'un des principaux aspects de recherche des FPGAs est le développement du flot de conception logiciel nécessaire pour cartographier les applications matérielles sur FPGA. L'efficacité et la qualité d'un FPGA dépendent largement de ce flot de conception. Il prend une description de conception d'application dans un langage de description matérielle (HDL) et le convertit en bitstream qui sera finalement programmé sur FPGA. La Figure 1.12 montre un flot complet pour programmer un circuit d'application sur un FPGA à mailles.

La synthèse logique permet de transformer une description HDL (VHDL ou Verilog) en un ensemble de portes booléennes et de bascules. Après la synthèse logique, des optimisations dépendant de la technologie sont effectuées. Ces optimisations transforment le réseau booléen indépendant de la technologie en un réseau de portes dans la bibliothèque technologique donnée. Le mappage technologique pour les FPGAs transforme le réseau booléen donné en un ensemble de blocs disponibles sur un FPGA. Pour une architecture FPGA traditionnelle, le réseau booléen est transformé en LUTs et bascules. Les algorithmes de cartographie technologique optimisent un réseau booléen donné pour un ensemble de fonctions objectives différentes, y compris la profondeur, la surface de silicium et la consommation énergitique.

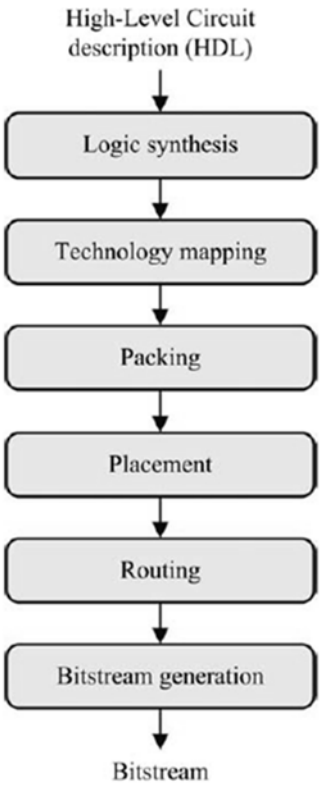



**Figure 1.12.** Flot de configuration d'un FPGA

La phase « Packing » regroupe les LUTs et les bascules en BLE, et regroupe les BLEs en groupe de BLEs. Ces groupes de BLEs sont ensuite mappés sur les CLBs d'un FPGA. L'objectif principal de cette phase est de regrouper les LUTs et les bascules et les BLEs de sorte que les communications entre les groupes soit minimisées, ce qui réduira aussi l'utilisation des ressources des FPGAs.

La phase de « Placement » est réalisé par l'algorithme de placement qui détermine la position des CLB et entrées/sorties dans un netlist compilée sur les blocs CLB et I/OB respectifs de l'architecture FPGA. L'objectif principal de l'algorithme de placement est de placer les blocs connectés les uns près des autres de sorte que des ressources de routage minimales soient requises pour acheminer leurs connexions. L'algorithme de placement peut également servir à répondre à d'autres exigences architecturales ou d'optimisation, telles que l'équilibrage de la densité de fils entre les FPGAs.

Une fois que les instances du netlist sont placées sur FPGA, les connexions entre différentes instances sont routées en utilisant les ressources de routage disponibles. Le problème de routage FPGA consiste à acheminer les signaux de telle sorte que pas plus d'un signal n'utilise la même ressource de routage.

Une fois que le netlist est placée et routée sur FPGA, le bitstream est généré pour le netlist. Ce bitstream est est programmé sur FPGA en utilisant un bitstream loader.

Le générateur bitstream lit les informations de mappage (Technology Mapping), et les informations du Packing et Placement pour programmer les SRAMs des LUTs. Les informations de routage d'un netlist sont utilisées pour programmer correctement les bits SRAM des CBs et des SBs.

## 7. Conclusion

Ce chapitre a initialement présenté une introduction des systèmes multiprocesseurs sur puce, les avantages que présentent de tels systèmes et les défis que relevent ces architectures pour motiver le besoin de sélection d'une plate-forme dans la conception des systèmes embarqués. Diverses approches pour la conception et la réalisation de circuits tout numériques (Microprocesseurs, GPP, ASIC, DSP...etc.) ont été présenté tout en relatant les avantages et inconvénients des unes et des autres en termes de facilité de conception et d'adaptabilité. Ces différentes technologies sont aussi comparées aux FPGAs pour avoir des élements de comparaisons et choisir la solution présentant le meilleur compris entre performance, consommation des ressources logiques et énergitique.

Nous avons aussi éxposé l'architecture typique d'un FPGA, et le flot logiciel pour programmer des conceptions matérielles sur FPGA. Toute ces solutions de réalisation d'architectures numériques nous interrogent à détailler un aspect primordiale de la conception des systèmes sur puce, celui de l'interconnexion sur puce. L'émergence du paradigme des NoCs sera introduit avant de faire un tour d'horizon des réseaux sur puce.

# Chapitre II

# Chapitre II

# Etat de l'art sur les réseaux sur puce

## 1. Introduction

À mesure que la densité de la conception VLSI augmente, la complexité de chaque composant d'un système augmente de manière significative. Pour s'adapter à la densité d'intégration croissante des transistors, aux fréquences de fonctionnement plus élevées et aux temps de mise sur le marché plus courts, les systèmes sur puce multiprocesseurs sont de plus en plus nécessaires dans l'industrie des semi-conducteurs d'aujourd'hui. Ces systèmes, connus aussi sous le nom de Multi-Processor System-on-Chip (MP-SoC), utilisent des structures de bus pour la communication sur puce et intègrent des éléments fonctionnels hétérogènes complexes sur une seule et unique puce. Cependant, les concepteurs de SoC font face de nos jours à un nouveau défi dans la conception des interconnexions sur puce, au-delà de l'évolution d'un nombre croissant d'éléments de traitement. Traditionnellement, les systèmes de communication utilisent une architecture de type bus partagé. Une telle architecture souffre d'un manque d'évolutivité et de prévisibilité, et ne peut désormais répondre aux exigences croissantes des futurs SoCs en termes de performances, de puissance, de synchronisation, et d'évolutivité, etc.

Pour relever les défis de la productivité de conception et d'intégrité du signal des conceptions de systèmes de nouvelle génération, une architecture d'interconnexion structurée et évolutive, « Network-on-Chip », a été récemment proposée pour atténuer le problème complexe de la communication sur puce.

## 2. Concept de design centré communication

Une application peut être représentée comme un ensemble d'unités de calcul qui nécessitent un ensemble de blocs de communication pour transmettre des informations entre ces unités. Pour distinguer l'impact sur la performance de ces deux composants principaux, le temps de calcul est dominé par le retard de porte tandis que le temps de communication est dominé par le retard de l'interconnexion. Lorsque la quantité d'unités de calcul d'une application est faible, les blocs de communication peuvent être effectués sur une base ad-hoc. Cependant, avec la réduction de la taille des transistors au cours des dernières années, le retard de la logique de calcul diminue continuellement comparé aux fils de connexion. Ainsi, nous ressentons le besoin d'une architecture de communication sur puce structurée et évolutive pour s'adapter aux applications de plus en plus complexes sur une seule puce. Cela se traduit par la conception d'une architecture de communications sur puce de plus en plus importante, et promeut le concept de design de la conception centrée calcul à la conception centrée communication.

### 2.1. Système multiprocesseurs sur puce

Le système sur puce est un concept architectural développé au cours des dernières décennies, dans lequel un processeur ou peu de processeurs, avec une mémoire et un ensemble de périphériques associés reliés par des bus sont tous implémentés sur une seule et même puce. Selon la loi de Moore, la tendance vers les puces de traitement à plusieurs cœurs est maintenant bien établie. Les processeurs à faible consommation d'énergie combinés à des accélérateurs matériels sont le choix préféré pour la plupart des concepteurs pour offrir le meilleur compromis entre performance et consommation d'énergie, étant donné que la puissance de calcul augmente de façon exponentielle en fonction du calcul de la dissipation de puissance dynamique [27]. Par conséquent, cette tendance impose de répartir les tâches d'application en plusieurs éléments de traitement où :

- Chaque élément de traitement peut être activé ou désactivé individuellement, économisant ainsi de l'énergie,
- Chaque élément de traitement peut fonctionner à sa propre tension et fréquence d'alimentation optimisées,
- Il sera plus facile d'équilibrer la charge entre les noyaux du processeur et de distribuer la chaleur à travers la matrice.

En général, les méthodes de sélection de quelques blocs de type ad-hoc peuvent fonctionner en s'appuyant sur l'expérience du concepteur. Cependant, une telle

approche ne peut fonctionner pour la conception des systèmes multiprocesseurs sur puce qui sont de plus en plus complexe. Par conséquent, la conception des systèmes sur puce de nos jours nécessite des techniques qui peuvent fournir une méthode efficace permettant à une puce de calculer des applications complexes et d'adapter judicieusement la superficie sur une seule puce en fonction des tendances technologiques actuelles.

## 2.2. Système de communication conventionnel sur puce

Un schéma de communication est composé d'une base d'interconnexion, des interfaces physiques et des protocoles en couches qui permettent la communication sur puce entre les composants d'un système MP-SoC. L'augmentation de la complexité de ce dernier rend cruciales les exigences de communication intra-puce. Les systèmes gourmands en données (Data-intensive Systems) tels que les dispositifs multimédias, les installations mobiles et les plates-formes multiprocesseurs ont besoin d'un système d'interconnexion flexible et évolutif pour gérer un grand flux de données sur puce. Habituellement, les connexions point-à-point sont adoptées comme des ensembles de connexions sur puce pour applications spécifiques qui connectent les modules de niveau supérieur. Cependant, à mesure que la densité et la longueur des connexions augmentent avec la complexité du système, l'architecture de communication basée sur les fils point-à-point n'est plus réalisable en raison de sa faible extensibilité et réutilisabilité. Plus précisément, les signaux sont acheminés par des connexions métalliques, à travers la puce, qui ne sont généralement pas mis à l'échelle avec la technologie.

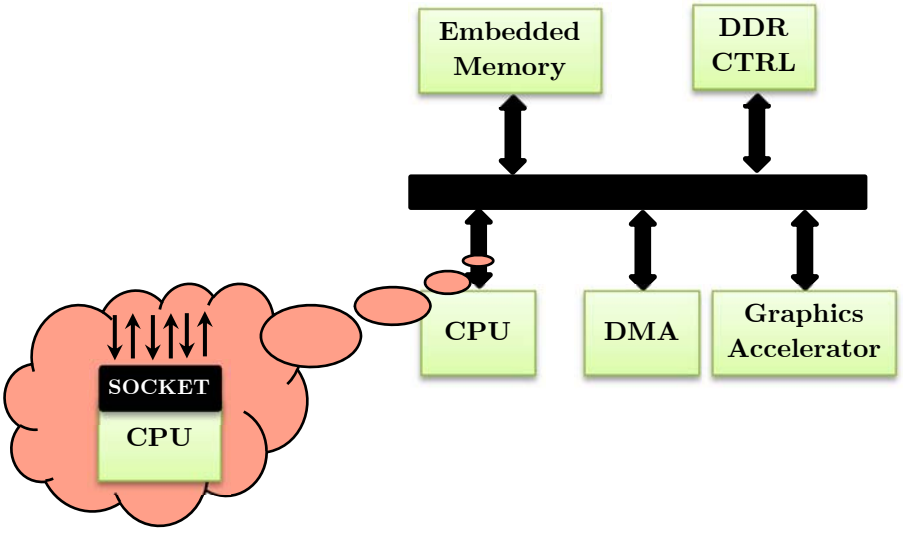



**Figure 2.1.** Exemple d'un Bus partagé pour système sur puce

Le délai de propagation, la dissipation de puissance et la fiabilité seront les problèmes sérieux des connexions métalliques dans la technologie submicronique profonde de VLSI. Des chercheurs se sont intéressés à l'évolution des technologies de silicium, démontrant qu'au-delà de 50nm les câbles globaux vont prendre 6-10 cycles pour se propager, ce qui dépassera de loin les retards et rendra difficile la synchronisation des longues connexions [28]. Le suivi de l'état de tous les éléments et la gestion centralisée de la communication globale entre les modules de niveau supérieur ne sont plus réalisables. Par conséquent, des modèles d'interconnexion sur puce par bus réutilisables tels que AMBA d'ARM [29] et CoreConnect d'IBM [30] sont couramment utilisés dans la conception des systèmes MP-SoCs actuels, de sorte que les modules peuvent partager le même groupe de fils d'interconnexion dans une communication par bus partagé.

Les bus partagés sur puce fournissent un ensemble de fils d'entrée et de sortie (appelés interfaces de connexion « Socket Interface ») et définissent un langage de communication inter-cœur commun appelé protocole de bus. Les sockets gèrent le protocole de bus et les exigences de trafic. Dans les bus sur puce, l'interface de socket est incluse dans le noyau, il n'y a pas de distinction entre le protocole de socket et le protocole de bus. Avoir un protocole de bus ou de socket normalisé commun permet une intégration facile « plug-and-play » sans avoir besoin d'un socket redessiné.

Un bus partagé est généralement basé sur un décodeur d'adresse centralisé qui pilote un multiplexeur unique qui sélectionne le maître, mais envoie des signaux à tous les dispositifs (esclaves) connectés en utilisant une logique de forte ventilation, en propageant les signaux de données. Ensuite, l'esclave sélectionné répond à la demande du maître en définissant des signaux de données de lecture (ou des accusés de réception) qui atteignent éventuellement l'IP maître. La synchronisation de tous les signaux de bus sur puce est assurée par une horloge unique partagée par tous les IPs maîtres et esclaves connectés au bus partagé. Outre les connexions et la mémoire tampon, l'arbitrage est un élément clé de la définition d'un bus. Ce processus répond aux requêtes principales en donnant accès au bus et à l'adresse de conduite et aux signaux de données de lecture/écriture à un esclave après avoir décodé la destination de la transaction. Si l'esclave est incapable de répondre immédiatement, le transfert est soit retardé, soit reporté ultérieurement.

Les bus partagés réduisent les coûts en partageant les connexions et en mettant en œuvre un protocole de communication bien défini qui permet d'ajouter ou d'échanger de nouvelles adresses IP ou de nouveaux cœurs de processeur entre les systèmes utilisant le même standard de bus.

Cependant, l'interconnexion sur puce par bus partagé ne permet qu'une communication à la fois en fonction de l'arbitrage, ainsi la bande passante moyenne de communication de chaque élément de traitement est inversement proportionnelle au nombre total d'IPs dans un système. Ce caractère rend ces architectures à base de bus partagé intrinsèquement non évolutives pour un système complexe dans la conception des systèmes MP-SoCs actuels. Implémenter plusieurs bus sur puce dans une architecture hiérarchique ou d'une manière séparée peut alléger cette contrainte d'évolutivité, mais elle nécessite un regroupement d'éléments de traitement spécifique à l'application et la conception de différents protocoles de communication pour répondre aux exigences de l'application. De plus, chaque fois qu'une nouvelle application doit être conçue, ou qu'un nouveau jeu de périphériques doit être ajouté, une puce conçue avec des bus simples manquera de moyens pour déterminer efficacement la faisabilité, sans parler de l'optimalité [31]. En outre, les tentatives visant à garantir la qualité de service (QoS) pour les performances du système consisteront en une tâche manuelle intensive. Par conséquent, la conception à base de bus partagé doit être remplacée avec une méthode flexible, évolutive et réutilisable.

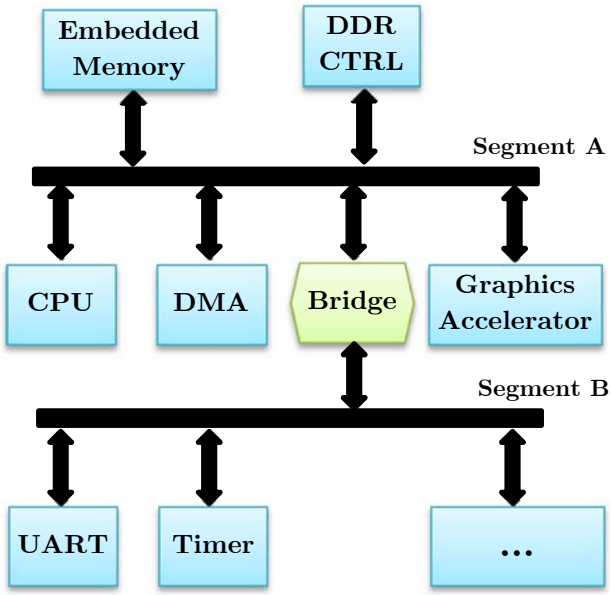



**Figure 2.2.** Exemple d'un Bus Hiérarchique partagé pour système sur puce

## 2.3. Émergence des réseaux sur puce

Depuis que les technologies de traitement actuelles ont permis de placer plus de processeurs et de cœurs sur une seule puce à travers des architectures MP-SoCs, exigeant ainsi un débit haut, une latence faible et des services de communication fiables, les infrastructures de communication sur puce à base de bus n'ont pu répondre à ces besoins. Essayer de réaliser de telles conceptions avec une architecture à base de bus

pourrait être problématique pour un certain nombre de raisons, y compris le temps de mise sur le marché, les problèmes de performance et l'évolutivité. Les variations des paramètres de l'appareil compliquent encore les problèmes de synchronisation et de fiabilité. Un changement de paradigme vers une conception centrée communication, plutôt qu'une conception centrée calcul, semble être l'approche la plus prometteuse pour résoudre ces crises de communication [32-37]. Par conséquent, au cours des dernières années, une nouvelle méthodologie appelée Network-on-Chip a été introduite comme solution pour résoudre ces problèmes en introduisant une architecture de communication structurée et évolutive.

## 3. Concept des réseaux sur puce

Network-on-Chip a été proposé ces dernières années comme une solution prometteuse de réseau de communication sur puce (On-Chip Communication Network) pour offrir une meilleure évolutivité, performance et modularité pour les architectures MP-SoCs actuelles [31, 33, 38, 39]. Les longues connexions croisées sur puce sont structurées et divisées en plus petits segments, ainsi leurs propriétés électriques peuvent être optimisées et bien contrôlées. Les signaux sont transmis en mode pipeline pour améliorer la fréquence de fonctionnement et faire face au problème d'intégrité du signal en communication. Les réseaux sur puce peuvent également promouvoir la productivité de conception en soutenant la modularité, qui s'avère pratique pour la vérification et la réutilisation à un niveau d'abstraction plus élevé. Les données transmises à travers une puce sont traitées par les unités intermédiaires de contrôle du réseau le long de son itinéraire de manière distribuée.

Le réseau sur puce est un réseau d'interconnexion sur puce à usage général qui offre de grandes promesses pour atténuer la complexité toujours croissante de la communication des conceptions MP-SoCs modernes. Un NoC préconise un style de conception centrée communication, où une épine dorsale de communication à usage général sera d'abord déployée ; les logiques spécifiques à l'application, telles que les processeurs, les sous-systèmes de mémoire, les contrôleurs de périphérique, etc., seront ensuite mappées sur des emplacements vides pré-alloués pour former un système complet. Ceci est analogue au processus moderne d'aménagement du territoire où les infrastructures routières et de communication sont posées avant la conception et la construction de bâtiments spécifiques. Selon cette direction, l'architecture NoC en mosaïque de style ville-bloc proposée dans [28, 30] a gagné en popularité grâce à sa simplicité et sa flexibilité. Mimant la disposition de bloc de ville moderne dans une telle architecture, la zone de la puce est divisée en carreaux rectangulaires où les adresses

IP logiques client sont placées. Les « rues » entre les tuiles sont réservées aux structures de routage de réseau sur puce à usage général prédéfinies. Chaque tuile du réseau sur puce à deux mailles comprend un routeur à 5 ports qui peut transférer des données via deux canaux unidirectionnels (ou bidirectionnels) vers la tuile locale, ainsi que vers les routeurs voisins nord, ouest, sud et est. En d'autres termes, le réseau générique sur puce est formé par une grille de mailles de routeurs. Chaque routeur est non seulement responsable des besoins de communication de sa maille associée, mais aussi achemine les trafics intermédiaires destinés ou en provenance d'autres mailles.

L'approche NoC offre les avantages suivants :

- **Evolutivité :** Comme la communication entre différents nœuds est réalisée par le routage de paquets, un grand nombre d'IPs peuvent être connectés sans avoir besoin d'utiliser de longues connexions. Au lieu de cela, les unités de traitement peuvent être connectées à l'infrastructure de communication via des liaisons courtes et des interfaces standards. De plus, la bande passante du réseau varie avec le nombre d'IPs dans la conception. Par conséquent, le paradigme NoC fournit une architecture de communication hautement évolutive.
- **Réutilisation de la conception :** La réutilisation est reconnue comme l'une des techniques les plus efficaces pour améliorer la productivité de conception. La modularité de l'approche NoC offre un grand potentiel pour la réutilisation des routeurs de réseau et d'autres IPs existants tels que les cœurs de processeurs et les codecs multimédia. Les routeurs, l'interconnexion et les protocoles de communication de niveau inférieur peuvent être conçus, optimisés et vérifiés une seule fois et réutilisés ultérieurement dans un grand nombre de produits. De même, de nombreux IPs existants qui ont été conçus avec certains protocoles en tête (CoreConnect, AMBA ...etc.) peuvent être réutilisés dans des conceptions NoC en utilisant des couches qui peuvent efficacement interfacer les IPs à base de bus existants et l'infrastructure de communication NoC.
- **Prévisibilité :** La nature structurée des fils dans les NoCs facilite les paramètres électriques bien contrôlés et optimisés. À leur tour, ces paramètres contrôlés permettent l'utilisation de circuits de signalisation agressifs qui peuvent réduire considérablement la dissipation de puissance et le délai de propagation. En plus de cela, faire face à des problèmes de conception physique tels que la diaphonie, et l'immunité au bruit devient plus facile en raison de cette régularité.

## 4. Généralités sur les réseaux sur puce :

Une architecture NoC typique consiste en plusieurs segments de fils et de routeurs. Dans un style régulier, les fils de connexion et les routeurs sont configurés comme des grilles de rues d'une ville, tandis que les unités de calcul (par exemple, les noyaux de processeur logique) sont placées sur des îlots séparés par des fils de connexion. Un module d'interface réseau transforme les paquets de données générés à partir des unités de calcul en des Flits de longueur fixe. Les paquets de données se composent d'un flit d'en-tête (ou de tête), d'un flit de queue et d'un certain nombre de flits de corps entre les deux. Cette chaine de flits sera routée vers la destination prévue en se déplaçant d'un routeur à son routeur voisin.

Chaque routeur possède cinq ports d'entrée et cinq ports de sortie correspondant aux directions nord, est, sud et ouest, ainsi que l'élément de traitement local PE. Chaque port se connectera à un autre port sur le routeur voisin via un ensemble de fils d'interconnexion physique (canaux). La fonction du routeur est d'acheminer les paquets entrant depuis chaque port d'entrée vers un port de sortie approprié, puis vers les destinations finales. Pour réaliser cette fonction, un routeur est équipé d'une mémoire tampon d'entrée (Buffer) pour chaque port d'entrée, d'un commutateur (cross-bar 5x5) pour rediriger le trafic vers le port de sortie souhaité et d'une logique de contrôle nécessaire pour assurer l'exactitude des résultats de routage.

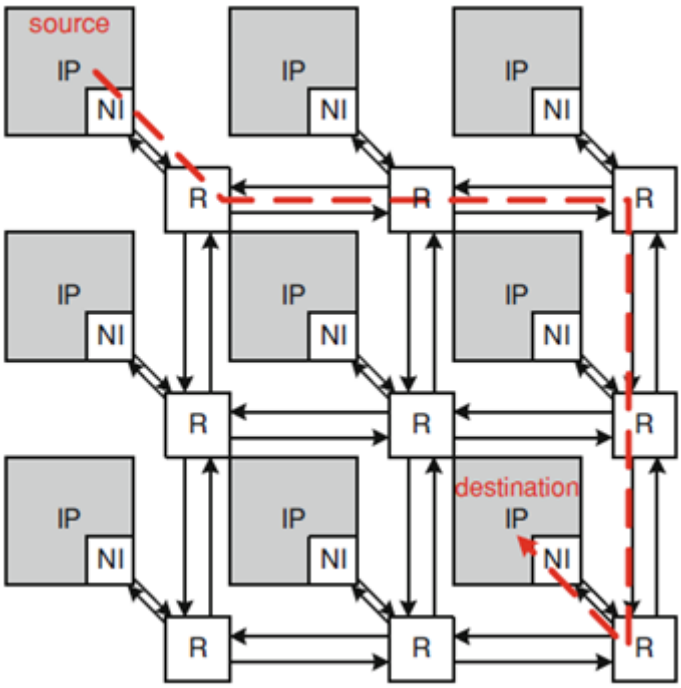



**Figure 2.3.** Architecture typique d'un NoC dans une topologie maillée (Mesh)

Habituellement, pour chaque paquet de données, le flit d'en-tête correspondant spécifie sa destination prévue. Après avoir examiné le flit de tête, la logique de commande du routeur déterminera quelle direction de sortie doit acheminer tous les flits (corps et queue) suivants associés à ce paquet de données en fonction de l'algorithme de routage appliqué.

### 4.1. Architecture conventionnelle d'un réseau sur puce

Un réseau sur puce typique se compose d'éléments de traitement informatique appelés PE, d'interfaces réseau (NI : Network Interface) et de routeurs. Les deux derniers comprennent l'architecture de communication. L'interface réseau est utilisée pour mettre en paquets les données avant d'utiliser les routeurs pour voyager le long du NoC. Chaque PE est attaché à un NI qui connecte le PE à un routeur local.

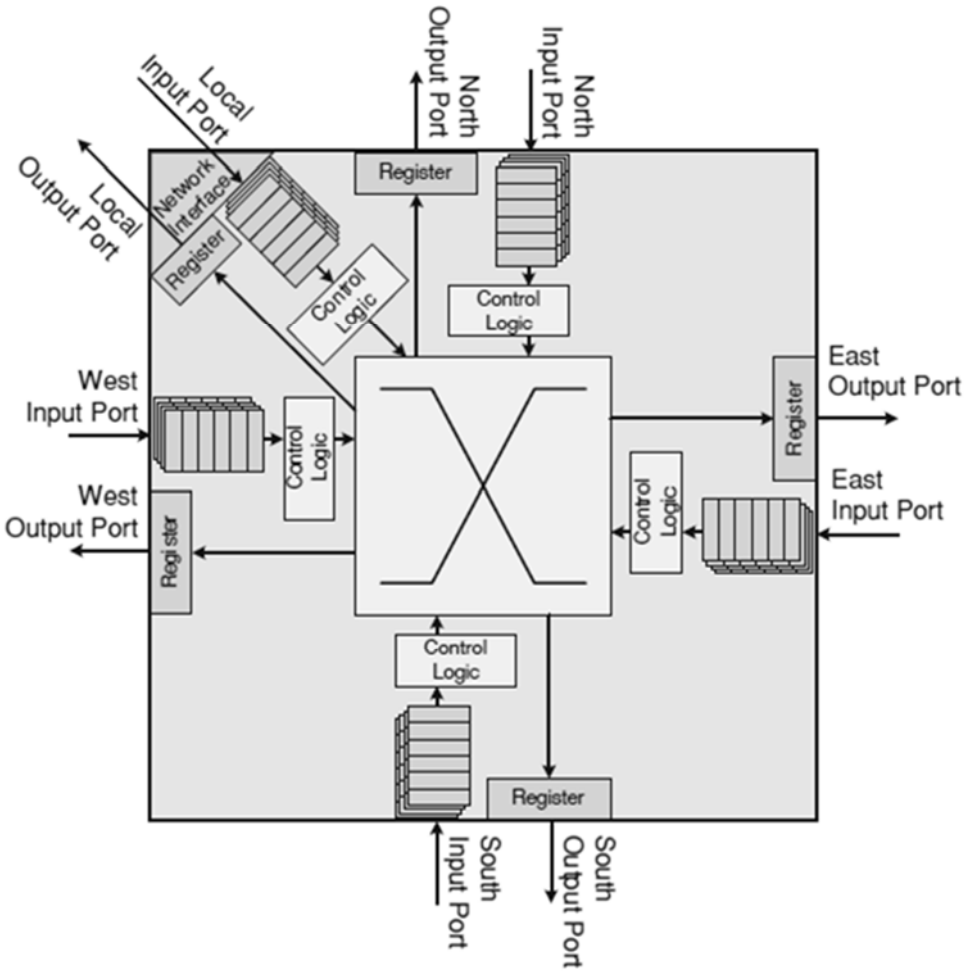



**Figure 2.4.** Architecture d'un routeur typique pour réseau sur puce

Lorsqu'un paquet a été envoyé à partir d'un PE source vers un PE de destination comme indiqué sur la Figure 2.3, le paquet est transmis d'un routeur à un routeur voisin sur le réseau via la décision prise par chaque routeur. Dans certaines architectures

NoC équipées de mécanismes de contrôle d'erreur, les NI sont également utilisées pour coder et décoder les données par le code de contrôle d'erreur appliqué.

Un routeur NoC est composé de commutateurs, de registres et d'une logique de commande qui effectuent collectivement l'acheminement et l'arbitrage de canal pour guider le flux de paquets dans le réseau, comme illustré à la Figure 2.4.

Pour chaque routeur, le paquet est d'abord reçu, avant d'être stocké dans une mémoire tampon d'entrée (Buffer). Ensuite, la logique de contrôle du le routeur est responsable de prendre la décision de routage et l'arbitrage de canal. Finalement, le paquet accordé traversera le crossbar qui l'aiguillera jusqu'à la sortie du routeur suivant, et le processus se répètera jusqu'à ce que le paquet arrive à sa destination.

### 4.2. Architecture d'un routeur conventionnel :

Les paquets livrés par les routeurs sont partitionnés en flit (FLow control unIT). Chaque flit entrant dans un routeur d'un paquet est stocké dans un buffer jusqu'à ce qu'il puisse passer au routeur suivant. Le premier flit dans le buffer sera traité par la logique de commande pour déterminer s'il est autorisé à être transmis et dans quelle direction de sortie il doit se diriger. La décision prise par l'unité de contrôle est basée sur le résultat du calcul du routage, de l'arbitrage et de l'espace dans le buffer du routeur suivant. Une fois la configuration de contrôle terminée, la commande passe à travers le commutateur du crossbar dans la direction de sortie souhaitée. Etant donné que toutes les décisions sont établies par la logique de contrôle du côté de l'entrée, les flits ne seront pas bloqués sur les ports de sortie.

L'approche des buffers en entrée dans une conception de routeur est généralement adoptée pour maintenir la disponibilité du canal jusqu'à ce que les emplacements des buffers soient épuisés. Des buffers plus grands apporteront un meilleur débit et une plus faible latence, mais entraîneront cependant une surcharge importante et une consommation d'énergie élevée. Habituellement, un buffer est mis en œuvre sous la forme d'une file d'attente de type FIFO (First In First Out), dans laquelle les données peuvent être traitées selon l'ordre chronologique de leur arrivée. D'autres mécanismes tels que le système d'allocation dynamique de buffers [40], qui écarte la complexité avec les performances, peut atteindre une meilleure utilisation de l'espace tampon en utilisant une structure de mémoire de liste-liée (linked-list memory structure).

Les deux principaux composants d'une logique de contrôle sont les modules de routage et d'arbitrage. Le module de routage traite les paquets pour générer les demandes de direction à chaque direction d'entrée, tandis que le module d'arbitrage au

niveau de chaque direction de sortie reçoit ces demandes et produit un signal d'accord pour indiquer la demande choisie, qui pourra traverser la direction de sortie correspondante.

L'algorithme de routage avec une logique simple est populaire dans la conception de routeur sur puce moderne. Notez que le nombre de demandes de direction est basé sur le nombre de directions de sortie existantes dans ce même routeur. Dans une topologie en maillage simple (Mesh 2D), cinq ports sont connectés à l'extérieur, y compris les quatre routeurs voisins et un élément de traitement local. La principale difficulté de la conception d'un arbitre réside dans sa propriété d'équité, ce qui signifie que les différents demandeurs devraient recevoir une quantité raisonnable de service de l'arbitre en fonction de leurs besoins individuels. La complexité logique d'un arbitre augmente avec le nombre de demandes d'entrée. La zone de circuit et le délai de calcul sont des préoccupations majeures dans la conception de l'arbitre pour un routeur sur puce. Le crossbar utilise un multiplexeur pour chaque sortie de routeur. L'entrée de commande d'un multiplexeur est générée par l'arbitre correspondant. Chaque port de sortie peut sélectionner au plus un élément à envoyer dans chaque cycle d'horloge en fonction du résultat d'arbitrage respectif. La consommation en surface et la latence d'un crossbar, qui sont affectées par le nombre total de ports de données et la largeur du bus, peuvent occuper une part importante de la conception dans un routeur.

Les transmissions de paquets dans le routage de type « Wormhole » sont segmentées en trois types de flits : en-tête, corps et queue. Les flits d'en-tête sont chargés d'initier et de réserver la bande passante du canal à chaque routeur pour les flits suivants de ce même paquet. De ce fait, les flits du corps et de queue seront guidés par les chemins que le flit d'en-tête a créés, et la bande passante du canal réservé est libérée par le flit de queue pour terminer le transit d'un paquet au routeur actuel.

### 4.3. Mécanisme de contrôle de flux des paquets

La performance de l'architecture de communication NoC est dictée par son mécanisme de contrôle de flux paquets. L'ajout de buffer au sein du réseau améliore considérablement l'efficacité d'un mécanisme de contrôle de flux de paquets, car un buffer peut augmenter l'allocation des canaux adjacents. Sans buffer, les deux canaux doivent être alloués à un paquet (ou à des flits) pendant plusieurs cycles consécutifs, autrement le paquet sera abandonné ou mal acheminé [32]. Plus spécifiquement, avec un contrôle de flux utilisant des buffers, tout paquet arrivant sur un routeur doit d'abord occuper certaines ressources, telles que la bande passante du canal et la capacité du buffer, en fonction de la méthodologie de contrôle de flux. Chaque routeur

doit jongler entre plusieurs flux de données d'entrée provenant de plusieurs ports d'entrée et les acheminer vers les ports de sortie appropriés avec la plus grande efficacité.

Les méthodes de contrôle de flux moyennant des buffers peuvent être classées en fonction de leur granularité d'allocation des buffers et d'allocation de bande passante de canal [32] : contrôle de flux paquet-buffer et contrôle de flux par flit-buffer. L'allocation des ressources par unité de flit peut atteindre plus d'efficacité d'utilisation du stockage que par unité de paquet. Deux types d'architectures de contrôle de flux de type flit-buffer sont couramment utilisés dans les NoC : le contrôle de flux « Wormhole » et le contrôle de flux de canal virtuel.

### 4.3.1. Contrôle de flux paquet-buffer

Le contrôle de flux paquet-buffer alloue les ressources du réseau paquet par paquet. Le contrôle de flux store-and-forward et le contrôle de flux virtual-cut-through sont des exemples de ce type de contrôle de flux.

Dans la méthode store-and-forward, chaque routeur doit s'assurer qu'il a déjà reçu et stocké un paquet entier avant de le transmettre au routeur suivant. Tandis que la méthode virtual-cut-through peut transmettre un paquet tant qu'il y a suffisamment d'espace dans le buffer pour recevoir un paquet dans le routeur suivant. Par conséquent, le contrôle de flux virtual-cut-through introduit un délai de communication inférieur à celui du contrôle de flux store-and-forward. Cependant, le contrôle de flux paquet-buffer nécessite plus d'emplacement buffer dans un routeur en raison de son utilisation inefficace du stockage de tampon. En outre, l'allocation de canaux en unités de paquets augmentera la latence de contention.

### 4.3.2. Routeur basé sur le contrôle de flux Wormhole

Le contrôle de flux Wormhole améliore les performances grâce à une granularité plus fine de l'allocation des messages au niveau flit au lieu du niveau paquet. Cette technique permet une utilisation plus efficace des buffers comparé aux mécanismes de contrôle de flux paquet-buffer, puisque la taille des buffers dans chaque routeur peut être réduite de manière significative [41, 42].

Une architecture typique d'un routeur pour réseau sur puce en pipeline à trois étages basée sur le contrôle de flux Wormhole est illustrée à la Figure 2.5. Chaque port d'entrée dispose d'un buffer en entrée basé sur une FIFO, qui peut être vu comme un seul canal virtuel utilisé pour contenir des flits bloqués. Pour faciliter le routage basé sur le contrôle de flux Wormhole [32], le module de calcul de routage (RC : Routing

computation) enverra un signal de requête de canal à l'allocateur de commutation (SA : Switch Allocator) pour les données dans chaque d'entrée. Si le tampon en aval du routeur voisin a un espace vacant, l'allocateur de commutation (SA) attribuera le canal et acheminera les flits de données à travers le commutateur crossbar vers le routeur suivant désigné à l'étape de traversée de commutateur (ST : Switch Traversal).

Cependant, la technique de commutation basée sur le contrôle de flux Wormhole économise la taille de la mémoire tampon au détriment du débit, puisque le canal appartient à un paquet, mais les tampons sont alloués sur une base flit-par-flit. En tant que tel, un paquet inactif peut continuer à bloquer un canal même lorsqu'un autre paquet est prêt à utiliser le même canal, ce qui conduit à une utilisation inefficace des ressources. C'est le bien connu problème de blocage de tête de ligne (HoL : Head of Line). Par conséquent, une architecture de routeur basée sur le contrôle de flux canal virtuel VC : Virtual Channel) a été proposée pour réduire l'effet de blocage et améliorer la latence du réseau.

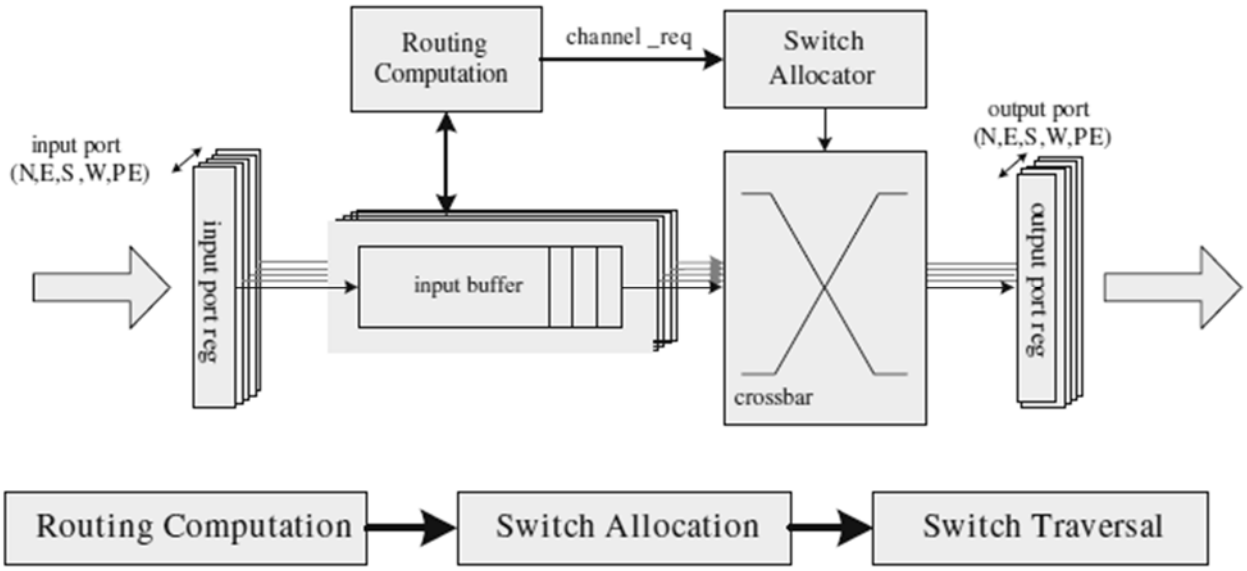



**Figure 2.5.** Conception d'un routeur type basé sur un flux de contrôle Wormhole

### 4.3.3. Routeur basé sur le contrôle de flux de canal virtuel

Le contrôle de flux de canal virtuel affecte plusieurs chemins virtuels, chacun avec sa propre file d'attente de buffer, au même canal physique ; il augmente ainsi le débit jusqu'à 40% par rapport au contrôle de flux de wormhole et aide à éviter les problèmes de congestion possibles [43-45].

Une architecture de routeur de contrôle de flux à canal virtuel, comme le montre la Figure 2.6, peut être considérée comme un remède aux limites du contrôle de flux wormhole. En multiplexant plusieurs canaux virtuels dans le même buffer d'entrée, un

paquet inactif ne bloquera plus d'autres paquets prêts à être acheminés en utilisant le canal physique partagé. Dans un routeur basé sur un contrôle de flux de canal virtuel, les flits sont acheminés via un pipeline à quatre étapes : calcul de routage (RC), allocation de canal virtuel (VCA), allocateur de commutateur (SA) et traversée de commutateur (ST).

Un flit entrant qui arrive à un routeur est d'abord mémorisé dans une file d'attente de canaux virtuels d'entrée appropriée et attend d'être traité. Lorsqu'un flit d'en-tête atteint le sommet de sa file d'attente du buffer de canal virtuel et entre dans l'étage RC, il est décodé par le module RC et génère une requête de direction associée. La requête de direction de cette commande est ensuite envoyée au module VCA pour atteindre le canal virtuel sur le routeur suivant. Il peut y avoir des conflits entre des paquets qui demandent le même canal virtuel sur le routeur. Les paquets qui ne sont pas choisi seront bloqués à l'étape VCA et le flit suivant à l'étape précédente sera également bloqué en raison de l'échec de contention. Notons que les processus de RC et VCA ne se déroulent réellement que sur le flit d'en-tête. Les flits de corps suivants et la queue d'un paquet accèdent simplement à la décision d'acheminement acquise par le flit d'en-tête et ne nécessitent aucun autre traitement aux étapes RC et VCA. Une fois qu'une décision sur la sélection du canal virtuel de sortie est prise à l'étape VCA, le module SA attribuera les canaux physiques aux flits intra-routeur. Les paquets accordés avec un canal physique traverseront le crossbar vers le buffer d'entrée du routeur suivant pendant l'étape ST, et le processus se répètera jusqu'à ce que le paquet arrive à sa destination.

Identique à un routeur basé sur le contrôle de flux Wormhole, un routeur basé sur un contrôle de flux de canal virtuel, utilisé pour une architecture NoC maillée (Mesh 2D), contient dix canaux de communication de données câblés, physiques et unidirectionnels. Deux canaux de données unidirectionnels sont connectés, chacun dans une direction opposée en tant que canaux d'entrée et de sortie, aux routeurs voisins. Le crossbar, qui est une matrice de commutation 5x5 utilisée pour connecter un canal d'entrée à un canal de sortie, dans ces deux architectures de routeur typiques peut prendre en charge jusqu'à dix canaux unidirectionnels pour la transmission de données. Cependant, l'utilisation de la bande passante n'est pas flexible dans ce routeur NoC à 5 entrées et 5 sorties câblées.

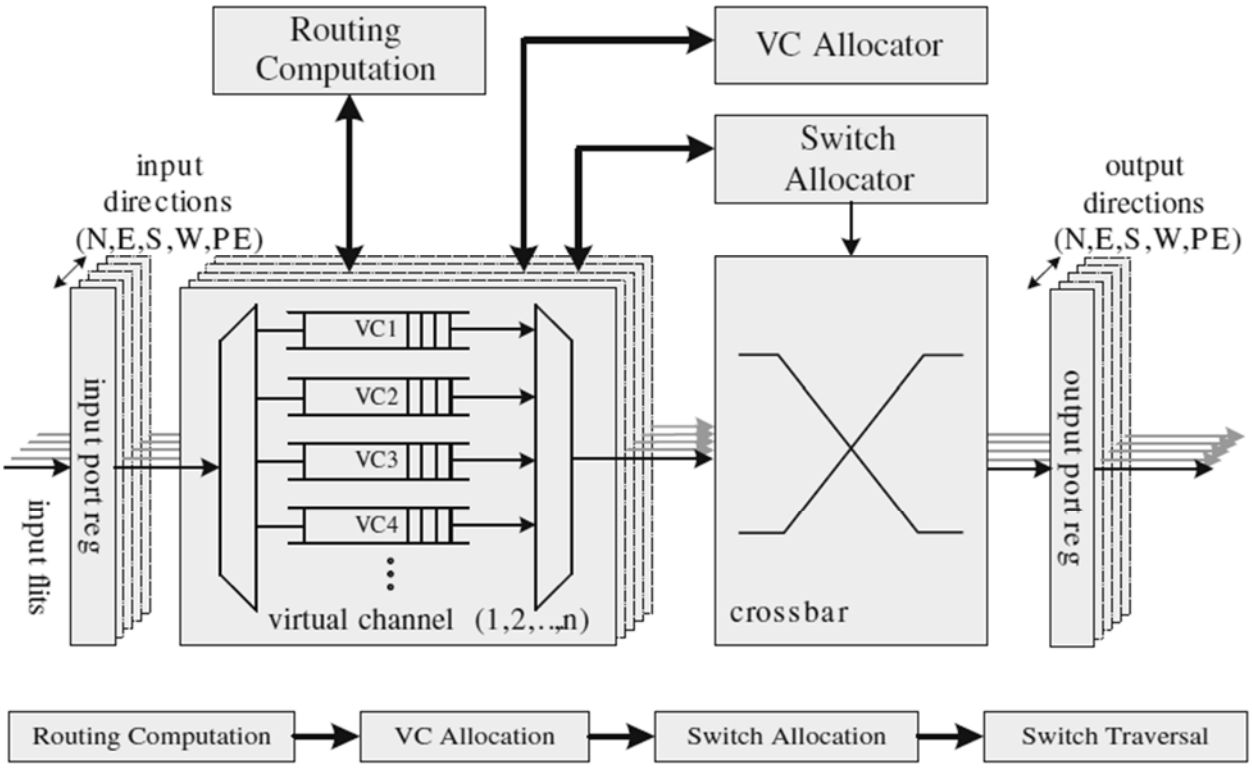



**Figure 2.6.** Architecture de routeur typique basée sur le contrôle de flux Virtual Channel

Prenons l'exemple d'un flit $f_A$ dans les canaux virtuels des buffers d'entrée « ouest » demande le canal de sortie « nord » qui est utilisé par un autre flit $f_B$ provenant des canaux virtuels des buffers d'entrée « ouest ». Le flit $f_A$ doit attendre le canal de sortie « nord » mais n'a aucune chance d'utiliser l'autre canal libre de la direction « nord » car il est câblé pour le canal d'entrée, qui ne peut que recevoir des données du routeur voisin. Cependant, si les dix canaux unidirectionnels sont remplacés par des canaux bidirectionnels, l'utilisation du canal sera plus efficace. De retour à l'exemple précédent, alors que le routeur voisin n'utilise pas l'autre canal dans la direction nord, le routeur local peut automatiquement reconfigurer ce canal comme un second canal de sortie et la contention dans la direction « nord » peut être soulagée.

### 4.4. Techniques de routage et d'arbitrage.

Un problème général concernant les algorithmes de routage et d'arbitrage peut être énoncé comme suit : soit un graphique d'application qui peut être représenté par un modèle de trafic unique, et une architecture de communication. Trouver une fonction de décision sur chaque routeur pour sélectionner un port de sortie.

Le problème ci-dessus a trois parties principales : un modèle de trafic, une architecture de communication NoC et un algorithme qui satisfait le mieux un ensemble d'objectifs définis par l'utilisateur. Tout d'abord, les modèles de trafic connus à l'avance peuvent être traités par un algorithme d'ordonnancement. D'un autre côté, les modèles de trafic dynamiques ou stochastiques reposent sur l'utilisation d'un algorithme de routage avec un degré variable d'adaptation pour acheminer les paquets. Nous nous concentrerons sur les modèles non connus à l'avance.

Deuxièmement, les architectures de communication NoC peuvent avoir des topologies différentes. Le plus courant est un maillage 2-D régulier, fréquemment utilisé pour afficher le comportement des algorithmes de routage adaptatif. D'autres travaux, tels que [46], traitent des régions irrégulières dans des mailles. Notre objectif est indépendant de la topologie.

La troisième partie traite des algorithmes eux-mêmes et des objectifs à atteindre. Deux algorithmes principaux utilisés pour déterminer où et quand un paquet va se déplacer sont : le routage et l'arbitrage. Un algorithme de routage décide dans quelle direction chaque paquet d'entrée doit se déplacer. L'arbitrage est le processus consistant à décider quelle demande de paquet d'entrée doit être accordée lorsqu'il y a plus d'une demande de paquet d'entrée pour le même port de sortie.

Un routeur typique dans un NoC est responsable du déplacement des paquets reçus des buffers d'entrée, avec ses algorithmes de routage et d'arbitrage, vers les ports de sortie. Les décisions prises par un routeur sont basées sur les informations collectées sur le réseau. Les décisions centralisées se réfèrent à la prise de décisions à partir des informations recueillies sur l'ensemble du réseau [47]. Les décisions distribuées font référence à la prise de décisions basée uniquement sur les informations générées par le routeur local ou les routeurs proches. Le routage distribué, permet aux NoCs de prendre de l'ampleur sans se soucier de l'ordre croissant de complexité au sein d'une unité de routage centralisée. Un exemple de routage centralisé est l'algorithme AntNet [48], qui dépend des informations globales pour prendre des décisions de routage, a donc besoin de buffer « ant », de tables de routage et de mécanismes d'arbitrage supplémentaires sur chaque nœud.

Il existe des algorithmes de routage distribués qui ne reposent que sur des informations locales. Ces algorithmes ont été proposés comme étant efficaces et conservant des frais généraux faibles et une évolutivité élevée. Les algorithmes de routage dans cette catégorie incluent les algorithmes déterministes et adaptatifs. Sous des configurations de trafic réalistes qui posent le problème des zones de congestion du

trafic « hotspot », le routage déterministe XY n'a pas réussi à éviter les points chauds et a entraîné des latences moyennes élevées [49]. Le routage adaptatif guide le routeur pour qu'il réagisse aux points chauds créés par différents modèles de trafic, en permettant à un paquet dans le tampon d'entrée de demander plusieurs ports ou directions de sortie [50]. Alors que des algorithmes de routage minimaux empêchent la création de « livelock », le routage adaptatif introduit la possibilité d'interblocage, ce qui peut être évité en appliquant des restrictions de modèle de virage impair-pair à la décision de routage [36].

Comme présenté dans [49], le routeur DyAD passe dynamiquement du routage déterministe au routage adaptatif lorsque la congestion est détectée, étant donné que le routage déterministe atteint une faible latence de paquets avec des taux d'injection de paquets faibles. Les nœuds voisins envoient une indication pour utiliser le routage adaptatif lorsque leurs buffers sont remplis au-dessus d'un seuil prédéfini. Dans ces conditions, le routeur impose que les paquets soient acheminés dans la direction ayant le plus d'emplacements disponibles dans ses buffers d'entrée. Cet algorithme adaptatif minimal, utilisé en présence de points chauds et de taux de congestion élevés, repousse le point de saturation du trafic dans le réseau. Une autre extension du routage adaptatif est l'algorithme Neigh-on-Path (NoP) [51], qui permet à chaque routeur de surveiller deux sauts dans le buffer d'entrée des routeurs afin de détecter les encombrements potentiels plus tôt. En détectant plus tôt le niveau de remplissage du tampon, les itinéraires peuvent mieux éviter la congestion. DyXY est un algorithme qui utilise un historique des niveaux de remplissage du buffer pour prendre des décisions [52]. Les algorithmes présentés dans [53, 54] utilisent des variantes du niveau de remplissage du buffer pour prendre des décisions.

En plus de prendre une décision de routage basée sur les informations des buffers des paquets en aval, l'autre partie de la prise de décision d'un routeur est l'arbitrage des paquets. Lorsque plusieurs paquets d'entrée sont désignés pour être transférés vers la même destination de saut suivant, des algorithmes d'arbitrage tels que round-robin ou premier arrivé-premier-servi (FCFS : First Come First Served) ont été proposés pour résoudre le conflit de port de sortie. Ces algorithmes d'arbitrage pourraient être conçus pour soulager les buffers en amont avec une congestion plus élevée. L'algorithme CAIS (Contention-Aware Input Selection) [41] est un algorithme d'arbitrage amélioré qui contribue à réduire la situation d'encombrement du routage en soulageant les points chauds du trafic en amont, déterminés par les demandes provenant du trafic en amont.

D'autres travaux ont été proposés pour traiter une certaine variance des algorithmes de routage ou d'arbitrage [56-58]. Parfois, nous classons les premiers comme des

méthodes d'évitement de la congestion ; en d'autres termes, ils évaluent les conditions du réseau en aval pour éviter d'envoyer des paquets vers les zones encombrées afin de ne pas aggraver les conditions de congestion. Nous classons ces derniers parmi les méthodes de soulagement de la congestion ; en d'autres termes, ils évaluent les conditions du réseau en amont pour déterminer quelle zone a le plus de congestion à envoyer en premier afin de diffuser rapidement la situation encombrée.

### 4.5. Contrôle de qualité de service (QoS : Quality of Service)

Il existe un large éventail de possibilités pour mettre en œuvre des services garantis sur un réseau. En se référant aux mécanismes de QoS de pointe pour les NoCs, ces derniers peuvent être catégorisés en deux types de schémas : en mode orienté connexion (commutation par circuit) et en mode sans-connexion (commutation par paquet).

#### 4.5.1. Mode orienté connexion (connection-oriented mode)

Dans les modes orientés connexion, les paquets de service garantis (GS : Guaranteed Service) traversent certains canaux ou buffer qui leur ont été réservés. Plus précisément, le chemin de connexion, entre la paire « source et destination » des paquets GS, est construit au moment où ils sont injectés sur le réseau [59-62]. Cependant, ce type de pré-allocation statique peut entraîner une latence de service élevée et ne tient pas compte des points de congestion (hotspots) créés par des changements temporels dans les exigences de données, conduisant ainsi à un NoC plutôt inextensible.

Le mécanisme de QoS orienté connexion est fiable pour répondre aux exigences QoS, car les connexions sont créées garantissant des limites strictes pour des flux spécifiques. Deux types de modèles de programmation pour la construction de la phase d'installation ont été présentés : la programmation centralisée et la programmation distribuée. D'une part la programmation centralisée qui configure les réservations par un gestionnaire de configuration qui prend en charge toutes les ressources du réseau. D'autre part, les modèles de programmes distribués permettent à toutes les réservations de ressources d'être gérées par chaque routeur local. La méthode centralisée est plus simple à réaliser, mais ne convient qu'aux systèmes de petite taille. Malgré la surcharge matérielle dans les routeurs, un modèle de programme distribué a acquis une popularité à grande échelle en raison de sa meilleure flexibilité.

Toutefois, le mécanisme de QoS orienté connexion est soumis à un surcoût matériel plus important en termes de contrôle et de stockage pour les réservations de ressources et d'évolutivité médiocre, car la complexité augmente avec l'ajout de chaque nœud. De plus, l'utilisation de la bande passante est inefficace et l'allocation des ressources doit

être considérée dans le pire des cas. De plus, la phase d'installation du trafic garanti présente une surcharge de temps qui peut entraîner une inefficacité pour les applications non déterministes.

### 4.5.2. Mode sans-connexion (connection-less mode)

Le mode sans-connexion est un moyen alternatif de prendre en charge différents niveaux de service dans les NoCs où les autorités de ressources sont hiérarchisées en fonction de l'exigence de qualité de service d'un flux de trafic [63]. C'est une technique distribuée qui permet de classer le trafic en différents niveaux de service. Ces niveaux de service peuvent souvent coïncider avec différents canaux virtuels à l'intérieur du commutateur. Étant donné que deux flux de trafic ayant des exigences de qualité de service différentes sont présentés simultanément sur le même canal, le flux dont la priorité est plus élevée peut interrompre celui dont la priorité est plus faible pour parcourir ce canal antécédemment [63, 64]. Il est plus adaptatif au trafic réseau et aux hotspots potentiels et peut mieux exploiter le réseau.

Différents des modes orientés connexion, les modes sans-connexion n'exécutent aucune réservation de ressources. En revanche, de multiples flux de trafic partagent la même priorité ou la même ressource, ce qui peut provoquer des conditions imprévisibles [51]. Le trafic avec un niveau de service plus élevé est garanti de manière relative dans un mode sans-connexion en donnant la priorité à chaque type de flux de trafic. Cependant, alors que le mode sans-connexion offre une prise en charge QoS plus grossière comparé au mode orienté connexion, il peut offrir une meilleure adaptation de la communication au trafic variable de réseau. De plus, une meilleure utilisation de la bande passante et moins de coût matériel peuvent être réalisés puisque le trafic est alloué aux ressources du réseau dynamiquement. En tenant compte des exigences de performance pour chaque niveau de service, un concepteur de réseau peut sélectionner une bande passante appropriée implémentée dans un NoC pour répondre aux contraintes de QoS et économiser le coût de câblage [61, 63].

Bien que la communication orientée connexion garantit des limites strictes pour plusieurs paramètres de trafic, une décision erronée de réservation de ressource peut entraîner une pénalité de performance inattendue. Alors que dans un réseau sans-connexion, une attribution de priorité non optimale a moins de dégradation du débit, bien qu'elle fournisse un support de QoS grossier. Comme indiqué dans [54], les services garantis nécessitent une réservation de ressources pour le pire des cas dans un mode orienté connexion, ce qui provoque beaucoup de gaspillage de ressources. En outre, une modélisation quantitative et une comparaison de ces deux schémas, fournies dans [55],

ont montré que dans une application à débit binaire variable, la technique sans-connexion offre de meilleures performances en termes de délai de transmission de bout en bout. Ces comparaisons peuvent aider à concevoir un NoC spécifique à l'application en utilisant un schéma de QoS approprié.

### 4.6. Fiabilité des réseaux sur puce

La tendance à la construction de grands systèmes de calcul incorporés à une architecture multi-cœurs a abouti à une relation bilatérale impliquant la fiabilité et la tolérance aux pannes. Bien que le rendement ait toujours été un problème critique dans la mise en œuvre des circuits haute performance, le document de la Carte routière internationale pour les semi-conducteurs (International Technology Roadmap for Semiconductor ITRS) [68] stipule que « l'assouplissement à 100% des dispositifs et des interconnexions réduit considérablement les coûts de fabrication, vérification et test ». Le principe général de la tolérance aux pannes pour tout système peut être divisé en deux catégories :

1- Emploi de la redondance matérielle pour masquer l'effet des fautes.
2- Auto-identification de la source de défaillance et compensation de l'effet par un mécanisme approprié.

Si nous pouvons faire fonctionner une telle stratégie, le système sera capable de se tester et de se reconfigurer, lui permettant de fonctionner de manière fiable tout au long de sa durée de vie.

Un certain nombre de méthodes tolérantes aux fautes ont été proposées dans [69, 70] pour les systèmes de communication à grande échelle. Malheureusement, ces algorithmes ne sont pas adaptés à un NoC, car ils induiront des surcharges significatives de surface et de ressources. Dumitras et al. ont proposé un algorithme de routage basé sur l'inondation pour NoC, appelé communication stochastique « stochastic communication », qui est dérivé du mécanisme de tolérance de panne utilisé dans le réseau informatique et des champs de base de données distribués. Un tel algorithme de communication stochastique sépare le calcul de la communication et fournit une tolérance aux fautes pour les défaillances sur puce [68, 71]. Cependant, afin d'éliminer la surcharge de communication élevée de l'algorithme de tolérance aux fautes basé sur les inondations, Pirretti et al. ont promu un algorithme de marche aléatoire redondant qui peut réduire considérablement les consommations tout en maintenant un niveau utile de tolérance aux pannes [72]. Cependant, l'idée de base, d'envoyer des informations redondantes par trajets multiples, pour atteindre la tolérance aux pannes peut provoquer une charge de trafic beaucoup plus élevée dans le réseau, et la

caractéristique de diffusion probabiliste peut également entraîner un comportement imprévisible supplémentaire sur le chargement du réseau.

Par conséquent, dans un routeur distribué pour NoC prenant en compte la mise en œuvre matérielle pratique, le schéma de contrôle d'erreur utilisé pour détecter / corriger les erreurs transitoires entre routeurs dans un NoC doit avoir une consommation en surface plus petite et une latence plus faible. Un code de contrôle d'erreur qui s'adapte à différents degrés de détection et de correction et qui a un faible temps d'attente facilitera son intégration dans un routeur. La méthode à tolérance aux fautes utilisant la détection d'erreur nécessite un buffer de retransmission supplémentaire spécialement conçu pour les NoCs lorsque les erreurs sont détectées.

## 4.7. Exploration des topologies des NoCs

Avec l'avènement des puces multiprocesseurs (CMP : Chip Multi-Processors) et des systèmes sur puce multi-cœur (MC-SoC : Multi-Core System-on-Chip), le paradigme du réseau sur puce a été proposé comme une solution viable au problème de la connexion du nombre croissant d'IPs intégrés. Le problème de communication devrait devenir beaucoup plus important avec la demande pour une plus grande puissance de traitement posée par la majorité des domaines d'application et la mise à l'échelle de la technologie.

Même si l'architecture de communication basée sur Noc introduit l'évolutivité architecturale et l'amélioration des performances, une conception minutieuse est absolument nécessaire pour atteindre ces gains. Pour cela nous nous intéressons à un certain nombre de problèmes architecturaux liés aux topologies NoC largement acceptées. De plus, nous présenterons les solutions académiques et commerciales disponibles pour la modélisation et la synthèse topologique, pour mettre en évidence les principales caractéristiques et limitations pour chacune d'entre elles. Afin de quantifier l'efficacité de chaque sélection architecturale, des repères appropriés sont également requis.

La topologie fait référence à la structure du réseau et à son organisation, car elle détermine les connexions entre les nœuds sur puce. Plus précisément, il s'agit du nombre de PEs (cœurs de traitement ou éléments de stockage), des routeurs, des liaisons de communication, ainsi que de leur interconnexion. De même, l'exploration topologique traite de l'évaluation de topologies alternatives afin de quantifier l'efficacité de chacune d'entre elles selon divers critères de conception. En fonction des résultats de l'exploration de la topologie, les concepteurs effectuent la sélection de la topologie. Lors de cette tâche, la topologie NoC la plus appropriée, qui satisfait les exigences de

communication de l'application et impose le coût minimum (en termes de consommation d'énergie / de puissance, de surface de silicium, etc.), est sélectionnée. Il est nécessaire de noter que lors de la sélection de la topologie, les contraintes de niveau physique sont également prises en compte.

Jusqu'à nos jours, les chercheurs proposent l'utilisation de diverses topologies de base telles qu'un bus, une étoile, un maillage, un point-à-point, ainsi que des topologies hiérarchiques, qui peuvent avoir des topologies identiques ou différentes localement et globalement.

### 4.7.1. Des topologies régulières ou irrégulières

Une première classification des topologies NoC concerne leur régularité. Plus précisément, en fonction de la structure de régularité de la disposition du réseau, le NoC est caractérisé comme régulier ou irrégulier. Ces deux topologies présentent des avantages qui devraient être exploités par le domaine d'application cible. Plus spécifiquement, une topologie de NoC classique suppose une distribution homogène des routeurs, ce qui conduit entre autres à réduire le temps et le coût de conception. Même s'il est possible d'incorporer une telle topologie à des conceptions de SoC à usage général, il convient surtout aux architectures à base de tores. De plus, les topologies régulières sont hautement réutilisables et imposent l'effort minimum de re-conception, au cas où elles seraient utilisées pour différentes applications ou architectures.

Malgré les avantages mentionnés précédemment, l'utilisation de topologies régulières n'est pas largement acceptée pour les produits commerciaux car elles imposent un certain nombre de défauts. Ces limitations se produisent principalement en raison de l'utilisation non optimale du réseau d'interconnexion, ce qui entraîne à son tour un retard et une consommation d'énergie accrus. Afin de pallier ces limitations, l'adoption de topologies irrégulières ou personnalisées est également proposée. Puisque ces topologies sont principalement orientées application, elles conviennent aux SoCs constitués de cœurs hétérogènes. De plus, la conception de topologies irrégulières nécessite des algorithmes de routage plus avancés prenant en compte la non-uniformité du réseau sous-jacent.

Par exemple, supposons une architecture NoC où chacun des routeurs doit être attaché à un nombre différent de nœuds. Dans le cas où une topologie régulière est utilisée, certains routeurs resteront inutilisés, ce qui entraînera des retards importants, des surconsommations de puissance et de surface de silicium. D'un autre côté, il est possible de concevoir une architecture NoC hétérogène, où chaque routeur a un nombre différent de ports. Une telle personnalisation des composants matériels entraîne des

performances plus élevées, par rapport au cas où une topologie régulière est utilisée. Cependant, l'hétérogénéité introduite impose un effort de conception supplémentaire à la fois pour les routeurs, ainsi que pour l'ensemble du NoC. La Figure 2.7 donne un exemple de ces deux topologies.

En conclusion, mentionnant qu'il n'est pas possible de concevoir une seule topologie de NoC adaptée à tous les domaines d'application, car chacune d'entre elles introduit des caractéristiques et contraintes inhérentes qui doivent être exploitées autant que possible afin de maximiser l'efficacité du réseau de communication. A cet effet, outre les deux solutions extrêmes mentionnées précédemment, les designers intègrent également à leurs conceptions des topologies hétérogènes, ou homogènes par morceaux. Ces topologies consistent en des blocs de construction de réseau hautement configurables (par exemple : des routeurs), qui peuvent être personnalisés pour un domaine d'application spécifique (par exemple multimédia, télécom, etc.).

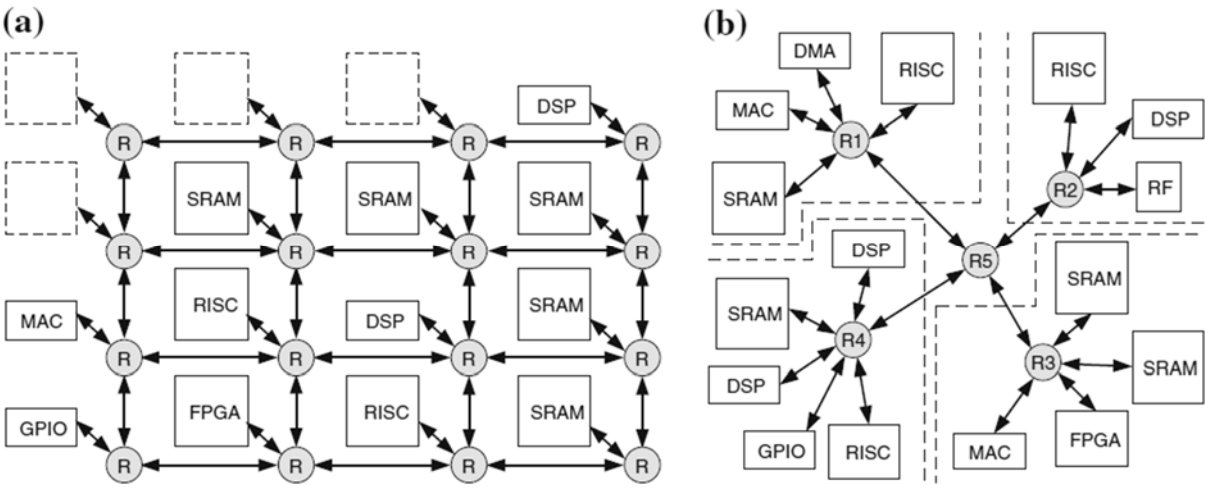



**Figure 2.7.** Topologies de NoC régulières (a) et irrégulières (b)

### 4.7.2. Topologies directes ou indirectes

De même, il est possible de classer la topologie du réseau soit directement, soit indirectement. Pour une topologie directe, chaque nœud a une liaison directe point-à-point avec un sous-ensemble d'autres nœuds du 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 des nœuds dans le système. Par conséquent, il existe un compromis fondamental entre la connectivité et le coût. Un certain nombre de produits NoC existants incorporent une topologie directe, parmi d'autres sont le Nostrum [73], SoCBus [74], Proteo [75], Octagon [76], ...etc.

La majorité des topologies de réseau dirigées ont une implantation orthogonale (disposition) et les nœuds incorporés sont habituellement disposés dans un espace orthogonal à « **n** » dimensions. Un tel arrangement architectural présente l'avantage concurrentiel de la simplicité. Par conséquent, il est possible d'utiliser des algorithmes de routage légers et rapides qui permettent d'obtenir un routage de paquets avec des performances comparables, mais avec une complexité nettement inférieure. Les instanciations typiques de cette topologie sont le maillage à « **n** » dimensions, le tore, le tore plié, l'hyper-cube et l'octogone.

### 4.7.3. Topologies de NoC 2D

Un maillage 2D, représenté schématiquement sur la Figure 2.8, est la topologie la plus simple et la plus populaire pour les NoCs. Il se compose d'un maillage MxN de commutateurs interconnectant des unités de calcul (PEs) placés avec les routeurs. Chaque routeur, sauf ceux sur les bords, est connecté à quatre routeurs voisins et un PE. Dans ce cas, le nombre de routeurs est égal au nombre de PEs. Afin de réaliser le chemin de communication physique entre les PEs et les routeurs, nous utilisons des canaux de communication, chacun constitué de deux liaisons unidirectionnelles entre un routeur et un PE, ou deux commutateurs. La topologie de maillage (Mesh 2D) suppose que toutes les liaisons ont la même longueur, et impose donc une régularité inhérente qui simplifie considérablement la conception physique. En outre, il est plus facile de prévoir les exigences de surface de silicium pour les topologies maillées, car elles augmentent presque linéairement avec l'augmentation du nombre de PEs. En dehors de ces avantages, l'utilisation de la topologie maillée présente également certains inconvénients. La pléthore de routeurs incorporés dans cette topologie conduit généralement à des régions congestionnées au-dessus de l'appareil. Pour ce faire, une conception minutieuse et une cartographie de l'application doivent être effectuées pour éviter l'accumulation de trafic, en particulier au centre du maillage. Notez que les topologies maillées 2D sont plus susceptibles de présenter des problèmes d'encombrement au centre de l'architecture, plutôt qu'à la périphérie, en raison des algorithmes de routage employés.

Tore (Torus) est une autre topologie directe, qui est formée par une grille à « **n** » dimensions avec « k » routeurs dans chaque dimension. Différentes instanciations de cette topologie ont été proposées. La Figure 2.9 représente 1-D tore, qui est essentiellement un réseau en anneau composé de « **k** » routeurs. Même si cette topologie est simplement suffisante pour être incorporée dans les conceptions ;

Cependant, cela entraîne une évolutivité et une performance limitées avec l'augmentation des nœuds.

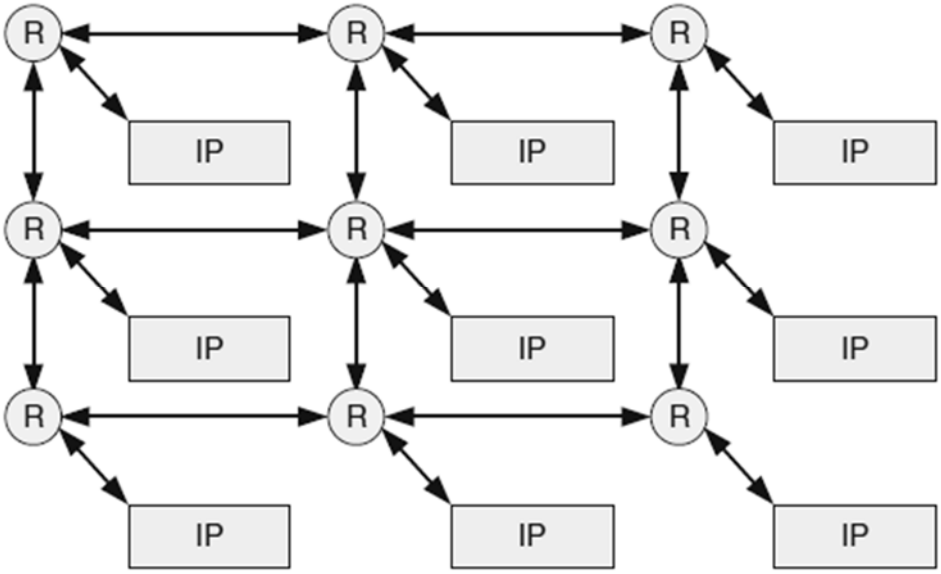



**Figure 2.8.** Topologie de maillage 2D (Mesh 3x3)

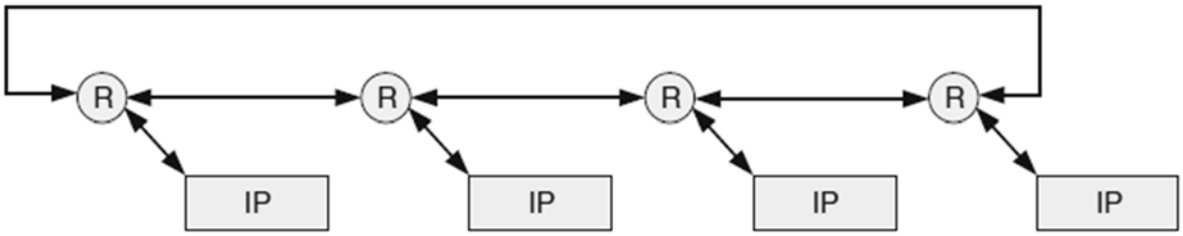



**Figure 2.9.** Topologie tore 1-D

De même, la topologie tore (2-D), représentée sur la Figure 2.10, montre la disposition d'un maillage régulier sauf que les PEs sur les bords sont connectés aux routeurs du bord opposé. Chaque routeur a cinq ports, l'un connecté à la ressource locale (local IP) et les autres connectés aux routeurs voisins les plus proches. La limitation de cette topologie affecte les connexions à long terme, car leur retard peut être considérablement augmenté, en fonction de la taille de l'architecture. Par conséquent, afin d'éviter tout problème de violation de synchronisation, la longueur, le retard et la consommation d'énergie de ces liaisons doivent être soigneusement pris en compte lors de la phase de conception.

Comme les NoCs sont généralement utilisés pour des conceptions spécifiques aux applications, les designers préfèrent généralement modifier (ou étendre) les topologies mesh ou tore en ajoutant des liens de dérivation afin d'obtenir des performances encore

plus élevées au prix d'une plus grande surface de silicium. Nous devons mentionner aussi que les topologies de maille et de tore peuvent être implémentées en tant que réseaux directs ou indirects.

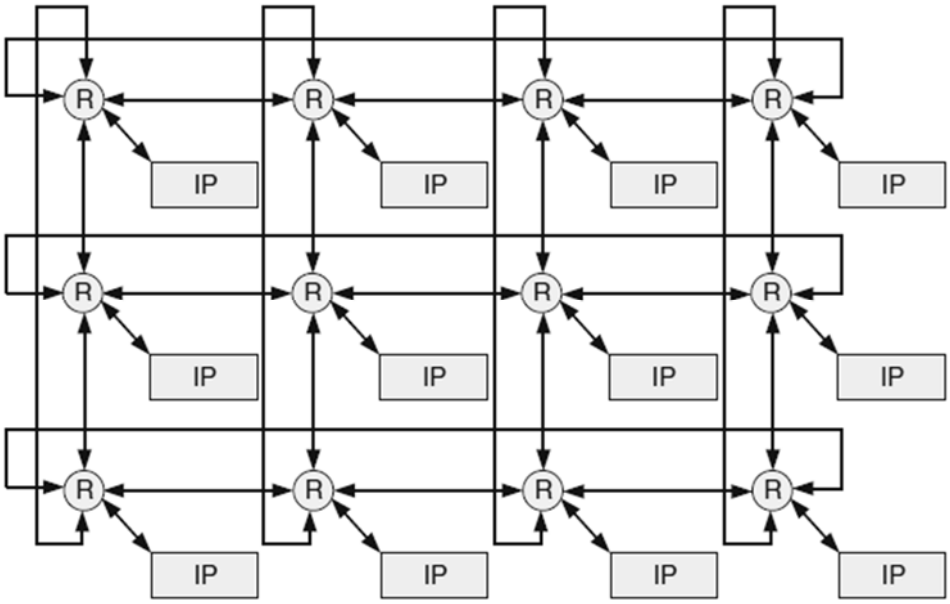



**Figure 2.10.** Topologie tore 2-D

Octagon est une autre topologie directe bien établie trouvée dans les NoCs. Sa configuration la plus simple, telle qu'elle est illustrée à la Fig. 2.11, consiste en un anneau de huit routeurs reliés par 12 liaisons bidirectionnelles.

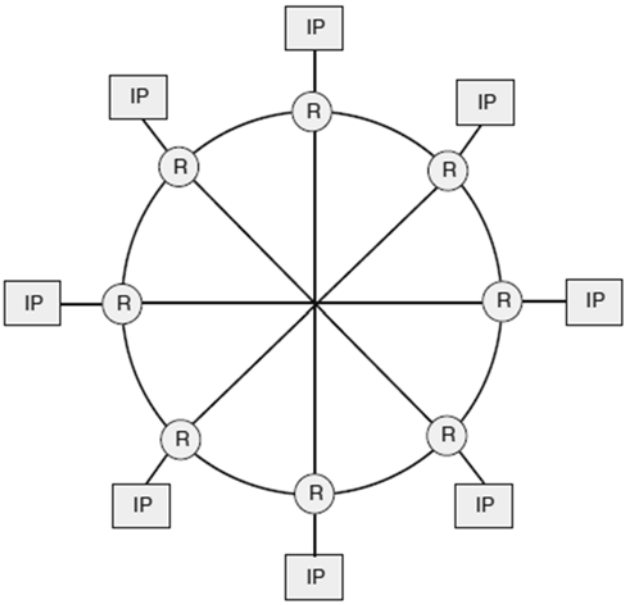



**Figure 2.11.** Topologie octogone

Ces liens fournissent une communication à deux sauts entre n'importe quelle paire de routeurs dans l'anneau. Cela permet l'utilisation d'algorithmes simples pour effectuer

un routage rapide et efficace du plus court chemin. Cette topologie suppose que chaque IP est associé à un IP et à un routeur, alors que pour accomplir la communication nécessaire entre n'importe quelle paire d'IP, deux transitions au maximum sont nécessaires dans l'unité octogonale de base. Dans ce cas, une plate-forme se compose de plus de huit IPs, l'octogone est étendu à l'espace multidimensionnel. Plus précisément, lors de cette approche, l'un des IPs par octogone est utilisé comme pont entre différents octogones. Bien entendu, un tel type de mécanisme d'interconnexion impose généralement une augmentation significative de la complexité du câblage.

Outre les topologies directes mentionnées précédemment, un certain nombre de topologies indirectes sont également largement utilisées pour la conception d'architectures NoC. Les Figures 2.12 et 2.13 donnent deux topologies nommées fat-tree et butterfly fat-tree, respectivement. Dans les deux topologies, les IPs sont connectés au routeur externe de l'architecture, alors que les routeurs ont des liaisons point-à-point vers d'autres commutateurs.

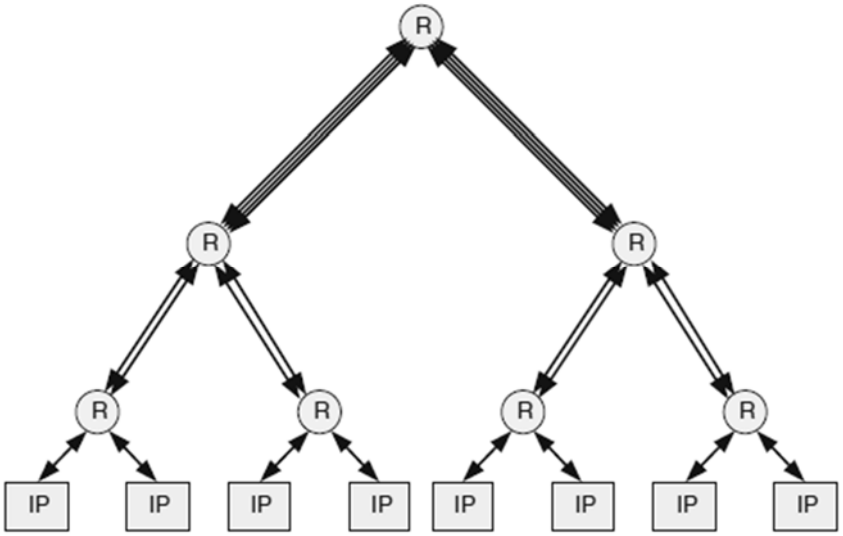



**Figure 2.12.** Topologie arbre élargi (Fat-Tree)

Plus spécifiquement, lorsqu'une architecture intègre ce type de réseau d'interconnexion, les unités de traitement et les modules de mémoire sont affectés aux feuilles des arbres, les routeurs sont placés aux sommets, alors que le chemin de communication consiste à monter et descendre une partie de l'arbre.

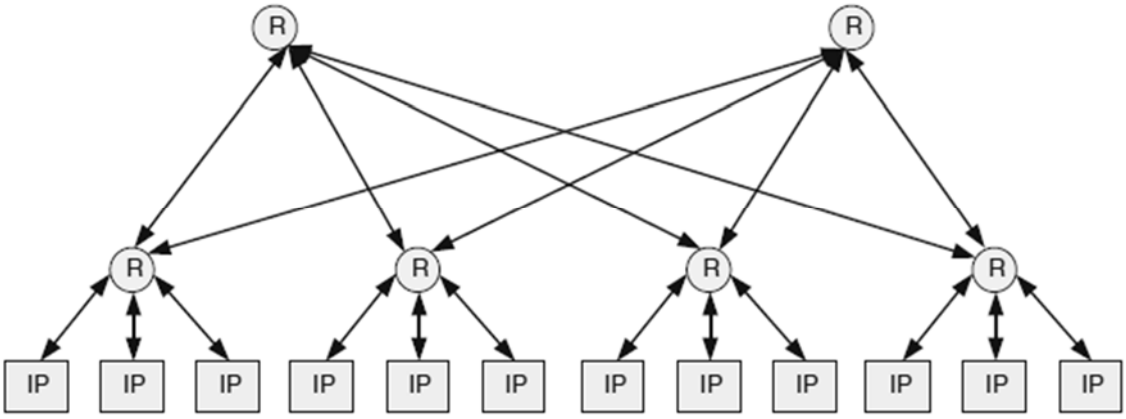



**Figure 2.13.** Topologie papillon arbre élargi (Butterfly-Fat-Tree)

Un réseau de papillons, représenté sur la Figure 2.13, peut être uni-, ou bidirectionnel. Par exemple, un simple réseau papillon unidirectionnel se compose de huit ports d'entrée, de huit ports de sortie et de trois niveaux de routeur, chacun d'eux contenant quatre routeurs. Dans un réseau papillon bidirectionnel, toutes les entrées et sorties sont du même côté du réseau. Lors de cette approche, tous les paquets arrivant aux entrées sont d'abord acheminés de l'autre côté du réseau, puis retournés et renvoyés à la sortie correcte [32].

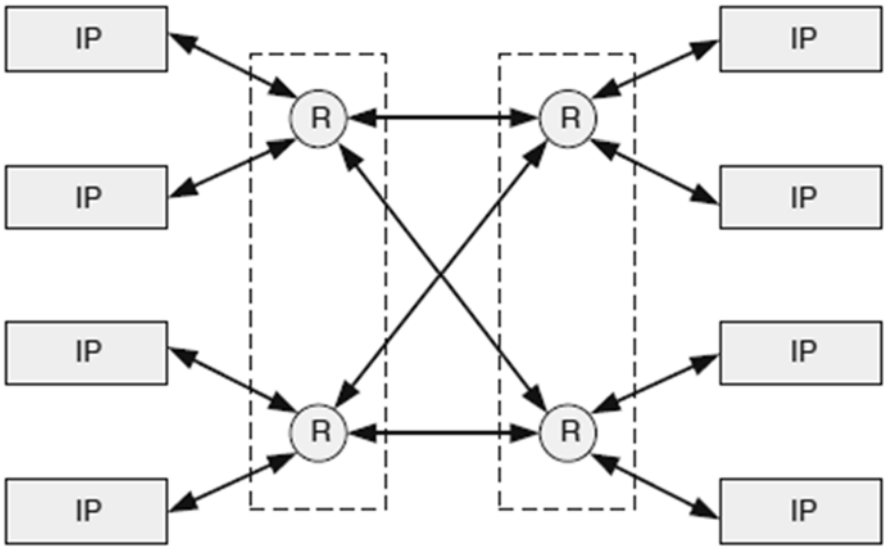



**Figure 2.14.** Topologie papillon avec quatre entrées, quatre sorties et deux étages de routeurs contenant chacun deux routeurs

Un réseau d'étoiles se compose d'un routeur central au milieu de l'étoile et de ressources de calcul, ou sous-réseaux, dans les pointes de l'étoile. Les besoins en capacité du routeur central sont assez importants, car tout le trafic entre pointes passe par le

routeur central. Cela provoque une possibilité importante de congestion au milieu de l'étoile. La Figure 2.15 montre une instance typique d'une topologie en étoile.

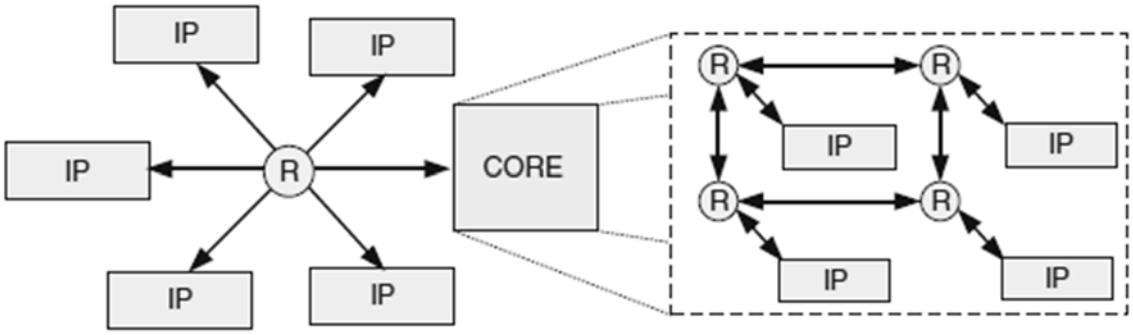



**Figure 2.15.** Topologie en étoile

Finalement, il existe des conceptions dans lesquelles l'utilisation de NoC est entièrement personnalisée, ou des topologies ad-hoc sont appropriées. La Figure 2.16 montre un exemple typique de deux topologies personnalisées de NoC. Ces topologies sont composées d'un mélange de topologies de bus partagées, directes et indirectes. L'avantage compétitif de ces topologies spécifiques à l'application (personnalisées) est l'amélioration significative de la performance globale du réseau. D'autre part, l'inconvénient de cette sélection concerne la pénalité dans le câblage structuré, qui est néanmoins l'un des principaux avantages offerts par les architectures régulières des réseaux sur puce. Les designers doivent s'efforcer d'offrir un compromis acceptable entre ces deux approches en insérant des liens longs spécifiques à l'application dans des architectures régulières.

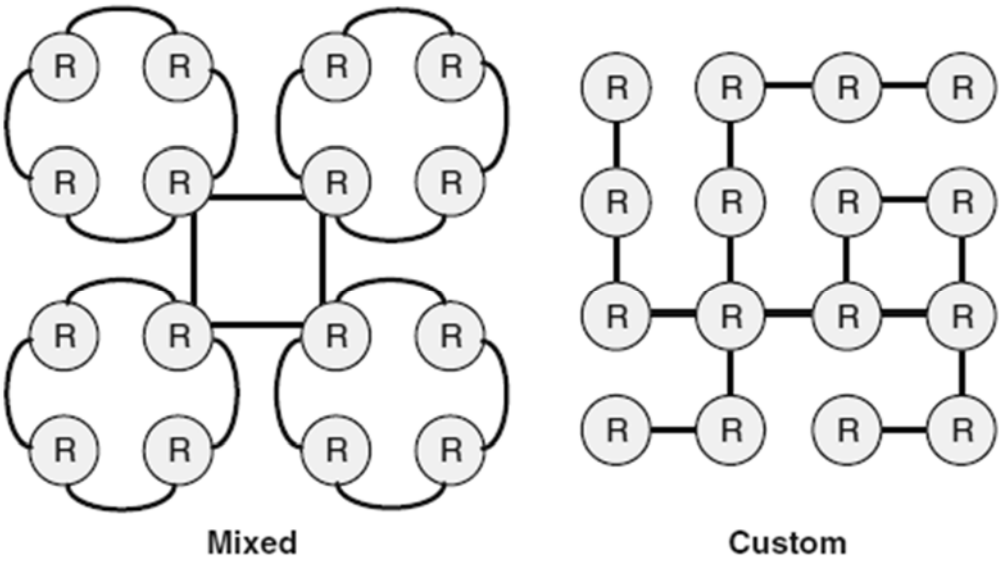



**Figure 2.16.** Topologies personnalisées de NoC

Même si une telle sélection booste les performances du réseau, sans impact notable sur la régularité du réseau et ses exigences de consommation en surface de silicium, les

chemins longue distance des topologies régulières imposent généralement des connexions avec de longueurs et de métriques variables (ce qui n'est pas souhaitable).

### 4.7.4. Topologies de NoC 3D

L'empilage de puces tridimensionnelles (3D) est présenté comme la technologie miracle qui peut maintenir l'élan de Moore et alimenter la prochaine vague de produits électroniques grand public. Outre la flexibilité imposée par ce nouveau paradigme de conception, l'un des principaux défis auxquels les concepteurs sont confrontés aujourd'hui en matière d'intégration 3D est de réaliser l'interconnexion entre les composants au sein d'une couche et entre les couches de manière évolutive et efficace. Une solution viable à ce problème est l'utilisation de NoCs.

Les topologies mentionnées précédemment sont appliquées pour fournir une communication de trafic pour les architectures planes (2D). Cependant, au cours des dernières années, les universités, les centres de recherche et l'industrie se sont efforcés de concevoir des architectures 3D où un certain nombre de PEs sont affectés à différentes couches, tandis que la connectivité entre couches adjacentes est assurée par TSVs (Through Silicon-Via). En combinant les avantages apportés par l'intégration 3D avec l'augmentation de la capacité de calcul des réseaux de communication, ce nouveau paradigme de conception semble prometteur pour fournir l'infrastructure de communication pour la prochaine génération de systèmes complexes. Plus spécifiquement, la localité le long de l'axe z conduit entre autres à un retard d'interconnexion significativement réduit, une structure d'interconnexion canonique, une flexibilité accrue, ainsi qu'une intégration de systèmes et de technologies dissemblables. Cependant, pour que ces architectures soient largement acceptées, des méthodologies et des outils de conception novateurs, qui tiennent compte des caractéristiques inhérentes à cette nouvelle technologie de traitement, sont absolument essentiels [77].

Ce nouveau concept permet aussi d'appliquer presque toutes les topologies 2-D mentionnées précédemment au domaine 3D, si elles sont étendues de manière appropriée. En supposant que nous ayons une topologie maillée, il est alors possible de faire la transition des architectures 2D vers les architectures 3D NoC en répartissant également les tuiles sur les couches de l'architecture 3D. Cependant, comme nous le montrons à la Figure 2.17, la topologie de maillage 3D impose l'utilisation de commutateurs à 7 ports plutôt que le port à 5 ports de l'architecture 2D correspondante [78].

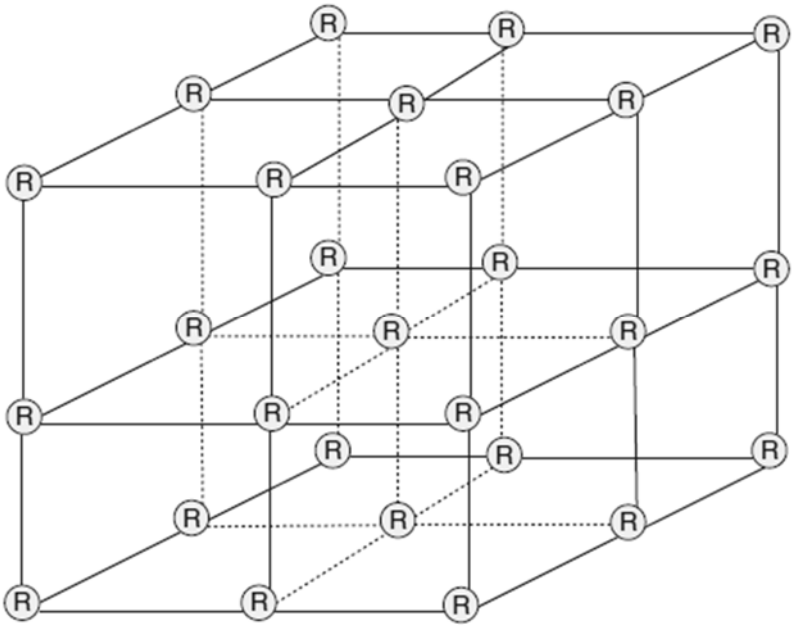



**Figure 2.17.** Topologie Mesh 3-D

## 5. Les métriques pour les réseaux sur puce

Pour comparer l'impact des configurations des réseaux sur puce sur les performances, nous analysons certaines métriques spécifiques au NoC afin d'expliquer comment elles influencent les performances du système tels que la scalabilité, la complexité ou la robustesse.

### 5.1. Degré (Degree) :

Le degré de topologie fait référence au nombre de liens à chaque nœud. Une topologie en anneau a un degré de 2 car il y a deux liens à chaque nœud, tandis qu'une topologie en tore a un degré de 4, chaque nœud ayant 4 liens le reliant à 4 nœuds voisins. Un degré plus élevé nécessite plus de ports sur les routeurs, ce qui augmente la complexité de la mise en œuvre.

### 5.2. Nombre de sauts (Hop Count) :

Le nombre de liens que traverse un paquet de la source à la destination, définit le nombre de sauts. Il s'agit d'une métrique très simple et utile pour la latence du réseau, car chaque nœud et chaque liaison subit un certain délai de propagation, même s'il n'y a pas de conflit. Le nombre maximum de sauts est donné par le diamètre du réseau ; A titre d'exemple, le nombre maximum de sauts d'une topologie en octogone est quatre, celui d'une topologie en maillage 3x3 est de quatre, tandis qu'une topologie en tore 3x3 optimise le nombre de sauts à deux.

### 5.3. Charge maximale du canal (Maximum Channel Load) :

Cette mesure est utile pour estimer la bande passante maximale que le réseau peut prendre en charge, ou le nombre maximal de bits par seconde (bps) pouvant être injectés par chaque nœud (routeur) dans le réseau avant sa saturation. Elle est définie comme étant relative à la largeur de bande d'injection. Ainsi, lorsque la charge sur un canal est de 2, cela signifie que le canal est chargé avec deux fois la largeur de bande d'injection.

### 5.4. La latence (Latency) :

Pour qu'un flit atteigne les cores de destination (par exemple, des éléments de traitement), il doit parcourir un chemin constitué d'un ensemble de liens et de routeurs (switches). La latence est définie comme le temps qui s'écoule entre le début de l'injection des volumes dans le réseau au niveau d'un IP source et son arrivée à l'IP destination. Cette latence $Ls_i$ dans chaque routeur peut être calculée comme suit :

$$L_{si} = \frac{b_i}{R_i} + T_i \tag{2.1}$$

avec $R_i$ est la bande passante du service, et $T_i$ est la latence maximale du service pour un routeur $s_i$ .

### 5.5. Charge du réseau (Network Load) :

La charge de communication est une valeur relative du taux de paquets entrant par rapport au taux de paquets sortant sur toutes les liaisons. Considérons que $D_r(t)$ est le nombre maximum de paquets pouvant, dans des circonstances idéales, être transmis sur toutes les liaisons à l'instant $t$, et $A_r(t)$ est le nombre réel de paquets arrivés sur toutes les liaisons à l'instant $t$. La charge de communication $L(t)$ peut être définie comme le rapport entre le taux de départ $D_r(t)$ et le taux d'arrivée $A_r(t)$ comme suit :

$$L(t) = \frac{A_r(t)}{D_r(t)} = \frac{\sum_{i=1}^{N_l} \overline{\alpha}_{l_i}}{N_l R t} \tag{2.2}$$

où $l_i$ est le nombre de paquets arrivés dans la liaison, $R$ est la bande passante de chaque lien $l_i$, et $N_l$ est le nombre de liens unidirectionnels transportant des paquets, en considérant que tous les liens ont la même bande passante « $R$ ».

### 5.6. La bande passante :

La bande passante pour chaque core $c_i$ représente le nombre de bits qui arrivent à ce core par seconde (bps). La bande passante globale $T$(t) est la somme des bandes

passantes de chaque core de destination $c_i$ durant l'intervalle [0, t]. Elle peut être calculée comme suit :

$$T(t) = \sum_{i=1}^{N_d} \bar{\alpha}_{c_i}(t) \quad (2.3)$$

où $N_d$ est le nombre de cores sélectionnés en tant que destinations, et $\alpha_{ci}(t)$ est la courbe d'arrivée qui représente le nombre cumulé de bits arrivés (c'est-à-dire accumulés) dans le core de destination jusqu'à l'instant t.

### 5.7. Coûts en ressources logiques :

Dans la conception des NoCs, trois sources de consommation de surface en FPGA peuvent être identifiées, à savoir les routeurs, les cores et les liens. Les routeurs ont deux composants principaux : les buffers pour stocker temporairement les flits et la logique pour implémenter l'algorithme de routage. La consommation en surface des liaisons dépend de leur longueur à l'intérieur de la puce. La consommation en surface totale peut alors être calculée comme suit :

$$A = \sum_{i=1}^{Ns} A_s(i) + \sum_{j=1}^{Nc} A_c(j) + \sum_{k=1}^{N_l} A_l(k) \quad (2.4)$$

où $Ns$ est le nombre de routeurs, $Nc$ est le nombre de cores IP, $N_l$ est le nombre de liaisons bidirectionnelles, $A_s(i)$ *et* $A_c(j)$, *et* $A_l(k)$ est la surface en FPGA requise pour le routeur $i$, le core IP $j$ et la liaison $k$ respectivement.

### 5.8. Consommation énergétique :

L'énergie totale peut être décomposée en énergie consommée sur les routeurs et en énergie consommée par des fils ou des liens entre les noyaux et les commutateurs. L'énergie totale $\mathcal{E}(t)$ peut être calculée comme suit :

$$\mathcal{E}(t) = \sum_{i=1}^{N_l} \bar{\alpha}_{l_i}(t)\, E_{l_i} + \sum_{j=1}^{N_s} \bar{\alpha}_{s_j}(t)\, E_{s_j} \quad (2.5)$$

où $\bar{\alpha}_{l_i}(t)$ et $\bar{\alpha}_{s_j}(t)$ sont le nombre de bits arrivés jusqu'à l'instant t des liaisons $l_i$ et $s_j$ respectivement. $N_l$ et $N_s$ sont le nombre de liaisons et de routeurs impliqués dans le transfert du flux de paquets l'application. $E_{li}$ est l'énergie consommée lors du transport d'un bit sur une liaison $l_i$ et $E_{sj}$ est l'énergie consommée lors des opérations d'écriture sur le buffer et de routage d'un bit à l'intérieur de chaque routeur $s_j$.

### 5.9. Le point de saturation :

Pour un système de communication sur puce basé sur un NoC, la latence reste prédictible et est raisonnable tant que le taux d'injection des données sur le réseau est en deçà d'un certain niveau appelé : le point de saturation. Au-delà de ce niveau, la

latence du réseau sera plus élevée et moins prédictible, ainsi le système est qualifié de « saturé ». Par conséquent, le point de saturation sépare les deux niveaux de fonctionnement du système de communication.

## 6. Conclusion

En conclusion, nous pouvons mentionner qu'il n'est pas possible de trouver une topologie de NoC optimale, car son efficacité dépend fortement des contraintes inhérentes au domaine d'application cible. Nous avons déjà souligné que la topologie maillée est la plus facile à implémenter car elle introduit une complexité de conception minimale. Plus spécifiquement, la topologie maillée présente certaines propriétés souhaitables, telles qu'un schéma d'adressage très simple et de multiples routes source-destination, qui confèrent une robustesse aux perturbations du réseau. Les topologies de faible dimension sont favorisées dans le cas où il y a une demande accrue de bande passante entre les commutateurs et le retard causé par les routeurs est comparable au délai inter-routeurs. D'un autre côté, si l'application cible présente un degré de localité élevé au niveau du schéma de communication, il existe alors une demande pour des topologies de dimension d'ordre supérieur.

# Chapitre III

# CHAPITRE III

# Conception d'une plateforme pour la génération des NoCs de type Wishbone

## 1. Introduction

L'architecture d'interconnexion « Wishbone » destinée aux Système sur puce pour les cores IPs est une méthodologie de conception flexible à utiliser avec les IPs à semi-conducteurs. Le but de cette architecture est de favoriser la réutilisation des conceptions (des IPs déjà développés) en atténuant les problèmes d'intégration du système sur puce. Ceci est accompli en créant une interface commune entre les IPs. Cela améliore la portabilité et la fiabilité du système et accélère le délai de mise sur le marché pour l'utilisateur final.

Auparavant, les IPs utilisaient des schémas d'interconnexion non standard qui les rendaient difficiles à intégrer. Cela a nécessité la création d'une logique de collage personnalisée pour connecter chacun des cores ensemble. En adoptant un schéma d'interconnexion standard, les cores peuvent être intégrés plus rapidement et plus facilement par l'utilisateur final. Cette spécification peut être utilisée pour des composants logiciels, ou matériels.

Cette spécification ne nécessite pas l'utilisation d'outils de développement ou de matériel cible spécifiques. En outre, il est entièrement compatible avec pratiquement tous les outils de synthèse logique. Cependant, les exemples présentés dans la spécification utilisent le langage de description matérielle VHDL. Ceux-ci sont présentés uniquement à titre de commodité pour le lecteur et devraient être facilement compris

par les utilisateurs d'autres langages de description du matériel (tels que Verilog). Des outils basés sur des schémas peuvent également être utilisés.

L'interconnexion « Wishbone » est conçue comme une interface à usage général. En tant que tel, elle définit l'échange de données standard entre les modules (IPs cores). Elle ne tente pas de réguler les fonctions spécifiques à l'application du core IP. Les architectes de Wishbone ont été fortement influencés par trois facteurs :

- Premièrement : le besoin d'une bonne solution d'intégration System-on-Chip fiable.
- Deuxièmement : une spécification d'interface commune était nécessaire pour faciliter les méthodologies de conception structurées pour les équipes travaillant sur de grands projets.
- Troisièmement, ils ont été inspirés par les solutions d'intégration de systèmes traditionnelles offertes par les bus micro-ordinateurs tels que le bus PCI (Peripheral Component Interconnect) le bus VME (VERSAModule Eurocard).

En fait, l'architecture Wishbone est analogue à un bus micro-ordinateur dans la mesure où :

- Ils offrent tous les deux une solution d'intégration flexible qui peut être facilement adaptée à une application spécifique ;
- Ils offrent tous les deux une variété de cycles de bus et de largeurs de chemin de données pour résoudre divers problèmes système ;
- Ils permettent tous les deux la conception de produits par divers concepteurs (ce qui fait baisser les coûts de la conception tout en améliorant les performances et la qualité).

Cependant, les bus traditionnels des micro-ordinateurs sont fondamentalement handicapés pour être utilisés comme une interconnexion pour système sur puce. En effet, ils sont conçus pour piloter de longues connexions de signaux et de systèmes de connecteurs hautement inductifs et capacitifs. À cet égard, les SoCs sont beaucoup plus simples et plus rapides. De plus, les solutions à base de SoCs ont un riche ensemble de ressources d'interconnexion, ceux qui n'existent pas dans les bus micro-ordinateurs car ils sont limités par les boitiers des circuits intégrés et des connecteurs mécaniques.

En somme, les architectes « Wishbone » ont tenté de créer une spécification suffisamment robuste pour assurer une compatibilité complète entre les IPs cores. Cependant, il n'a pas été trop spécifié afin de limiter indûment la créativité des principaux développeurs ou de l'utilisateur final.

## 2. Les caractéristiques du « Wishbone »

L'interconnexion Wishbone facilite le concept de la réutilisation dans la conception des SoCs en créant un protocole standard d'échange de données. Les caractéristiques de ce protocole sont :

- Des interfaces matérielles pour des IPs simples, compactes et logiques qui nécessitent très peu de logique.
- Prise en charge des méthodologies de conception structurées utilisées par les équipes des grands projets.
- Un ensemble complet de protocoles de bus de transfert des données comprenant :
  - Cycle Lecture/Ecriture,
  - Cycle de transfert en bloc,
  - Cycle Lecture/Modification/Ecriture (RMW)
- Largeurs de bus de données modulaires et tailles d'opérandes jusqu'à 64 bits.
- Prend en charge différentes méthodes d'interconnexion inter-IPs : bus partagé, connexion point-à-point, Crossbar, interconnexion pour flux de données.
- Le protocole de « handshaking » permet à chaque IP de contrôler (accélérer/réduire) sa vitesse de transfert de données.
- Prend en charge les transferts de données d'horloge unique.
- Prend en charge les étiquettes définis par l'utilisateur. Cette particularité peut être appliquée sur des informations à : un bus d'adresse, un bus de données ou un cycle de bus.
- Une conception synchrone qui assure la portabilité, la simplicité et la facilité d'utilisation.
- Spécification de synchronisation très simple et variable.
- Documentation standard qui simplifie l'utilisation.
- Indépendant de la technologie matérielle cible (FPGA, ASIC, etc.).
- Indépendant de la méthode de livraison (Soft-design, Firm-design ou Hard-design).
- Indépendant de l'outil de synthèse, du routeur et de la technologie des outils de mise en page.

## 3. Les objectifs du « Wishbone »

Les principaux objectifs de la spécification Wishbone sont :

- Créer un moyen d'interconnexion flexible à utiliser avec les IPs, destiné aux SoCs.

- Renforcer la compatibilité entre les IPs, et améliorer la conception basée sur la réutilisation.
- Créer une norme robuste, sans contraindre la créativité des designers.
- Rendre facile la compréhension du designer et l'utilisateur final.
- Rendre structurée la conception en équipe. Chaque membre de l'équipe peut s'interfacer avec la Spécification « Wishbone ». Une fois tous les sous-ensembles ont été complétés, le système complet peut être intégré.
- L'utilisation d'interface commune permet d'éliminer la large documentation d'interface habituelle.
- Permettre aux utilisateurs de créer des composants SoCs sans enfreindre les droits de brevet des autres.
- Identifier les technologies d'interconnexion critiques du système sur puce et les placer dans le domaine public le plus tôt possible.
- Soutenir un modèle d'entreprise dans lequel les fournisseurs IPs peuvent coopérer, mais peut également rivaliser sur le marché commercial. Cela améliore la qualité globale et la valeur des produits. Ce modèle permet également d'offrir des IPs open-source.
- Créer une architecture qui a un chemin de transition en douceur pour soutenir de nouvelles les technologies. Cela augmente la longévité de la spécification, et l'adapte aux exigences.
- Créer une architecture qui prend entièrement en charge la génération automatique des SoCs. Cela permet de générer des composants avec des générateurs paramétriques.

## 4. Terminologies de la spécification Wishbone

Pour éviter toute confusion et pour clarifier les exigences de conformité, cette spécification utilise cinq mots-clés :

- Règle,
- Recommandation,
- Suggestion,
- Autorisation,
- Observation.

### 4.1. Règle

Les règles forment le cadre de base de la spécification. Elles sont parfois exprimées sous forme de texte et parfois sous forme de figures, de tableaux ou de dessins. Toutes les règles doivent être suivies pour assurer la compatibilité entre les interfaces. Les règles sont caractérisées par un style impératif.

### 4.2. Recommandation

Chaque fois qu'une recommandation apparaît, les concepteurs seraient avisés de suivre les conseils donnés. Faire autrement pourrait entraîner des problèmes gênants ou de mauvaises performances. Bien que cette spécification ait été conçue pour prendre en charge des systèmes haute performance, il est possible de créer une interconnexion conforme à toutes les règles, mais dont les performances sont très médiocres. Dans de nombreux cas, un concepteur a besoin d'un certain niveau d'expérience avec l'architecture du système pour concevoir des interfaces offrant des performances optimales. Les recommandations sont justement basées sur ce type d'expérience et sont fournies à titre indicatif à l'utilisateur pour optimiser les conceptions.

### 4.3. Suggestion

Les suggestions contiennent des conseils qui sont utiles mais pas vitaux. Le développeur est encouragé à considérer le conseil avant de s'en passer. Certaines décisions de conception sont difficiles jusqu'à ce que l'expérience ait été acquise. Les suggestions aident les designers n'ayant pas encore acquis cette expérience. Certaines suggestions portent sur la conception d'interconnexions compatibles, d'autres sur la facilité d'intégration des systèmes.

### 4.4. Permission

Dans certains cas, une règle n'interdit pas spécifiquement une certaine approche de conception, mais l'utilisateur peut se demander si cette approche pourrait violer l'esprit de la règle, ou si elle peut conduire à un problème subtil. Les permissions rassurent les concepteurs qu'une certaine approche est acceptable et ne posera pas de problèmes.

### 4.5. Observation

Les observations n'offrent aucun conseil spécifique. Elles apportent simplement des clarifications. Elles énoncent les implications de certaines règles et attirent l'attention sur des choses qui pourraient autrement être négligées. Elles donnent également la logique derrière certaines règles, de sorte à comprendre pourquoi la règle doit être suivie.

## 5. Interconnexion « Wishbone » dans les systèmes sur puce

L'interconnexion Wishbone est une architecture pour connecter des circuits numériques ensemble pour former un de système sur puce. Elle résout un problème très fondamental dans la conception de circuits intégrés, « comment connecter des fonctions réalisées sur circuits intégrés ensemble d'une manière simple, flexible et portable ? ». Ces fonctions sont généralement fournies sous la forme de «IP cores» (cœurs de propriété intellectuelle), que les designers de systèmes peuvent acheter ou concevoir eux-mêmes.

Sous cette topologie, les IPs sont les blocs fonctionnels de base du système SoC. Ils sont disponibles dans une grande variété de fonctions telles que les microprocesseurs, les éléments de calcul (PEs), les interfaces de disque, les contrôleurs de mémoire, les contrôleurs de réseau...etc.

Généralement, les IPs sont développés indépendamment les uns des autres et sont liés ensemble et testés par un intégrateur système tiers. Wishbone aide l'intégrateur système en standardisant les interfaces IP Cores. Cela rend beaucoup plus facile la connexion des IPs, et donc beaucoup plus facile de créer un système sur puce personnalisé.

Wishbone utilise une architecture Maître/Esclave (MASTER/SLAVE). Cela signifie que les modules fonctionnels avec des interfaces MASTER initient des transactions de données vers les interfaces SLAVE participantes. Comme le montre la Figure 3.1, les blocs MASTER et SLAVE communiquent via une interface d'interconnexion appelée INTERCON. L'INTERCON est considéré comme un « Nuage » (Cloud) qui contient des circuits. Ces circuits permettent aux IPs MASTER de communiquer avec les IPs SLAVE. Le terme « cloud ou nuage » est emprunté à la communauté des télécommunications. Souvent, les systèmes téléphoniques sont modélisés comme des nuages qui représentent un système de commutateurs téléphoniques et des dispositifs de transmission. Les combinés téléphoniques sont connectés au nuage et sont utilisés pour passer des appels téléphoniques. Le nuage lui-même représente un réseau qui achemine un appel téléphonique d'un endroit à un autre.

L'analogie du « Cloud » est utilisée car Wishbone peut être modélisé d'une manière similaire. Les interfaces MASTER/SLAVE (qui sont analogues aux téléphones) communiquent sur une interconnexion (qui est analogue au «nuage» du réseau téléphonique). Le réseau d'interconnexion Wishbone peut être modifié par les développeurs du système pour répondre aux besoins. Dans la terminologie de Wishbone, cela s'appelle « une interconnexion variable ». Cette interconnexion variable permet de modifier la manière dont les interfaces (MASTER / SLAVE) communiquent entre elles. Par exemple, une paire d'interfaces MASTER et SLAVE peut communiquer avec des topologies point-à-point, de flux de données, de bus partagé ou par un commutateur crossbar.

Le schéma d'interconnexion variable est très différent de celui utilisé dans les bus traditionnels tels que les bus PCI, VMEbus et ISA. Ces systèmes utilisent des circuits imprimés avec des connecteurs câblés. Par conséquent les interfaces de ces bus ne peuvent pas être modifiées, ce qui limite considérablement la communication entre les éléments du système.

Wishbone surmonte cette limitation en permettant de changer l'interconnexion du système. Ceci est possible car les puces ont des interconnexions flexibles et peuvent être ajustés. Celles-ci peuvent être «programmées» sur puce en utilisant les langages de description de matériel tel que VHDL ou Verilog. Les bibliothèques d'interconnexion peuvent également être formées et partagées.

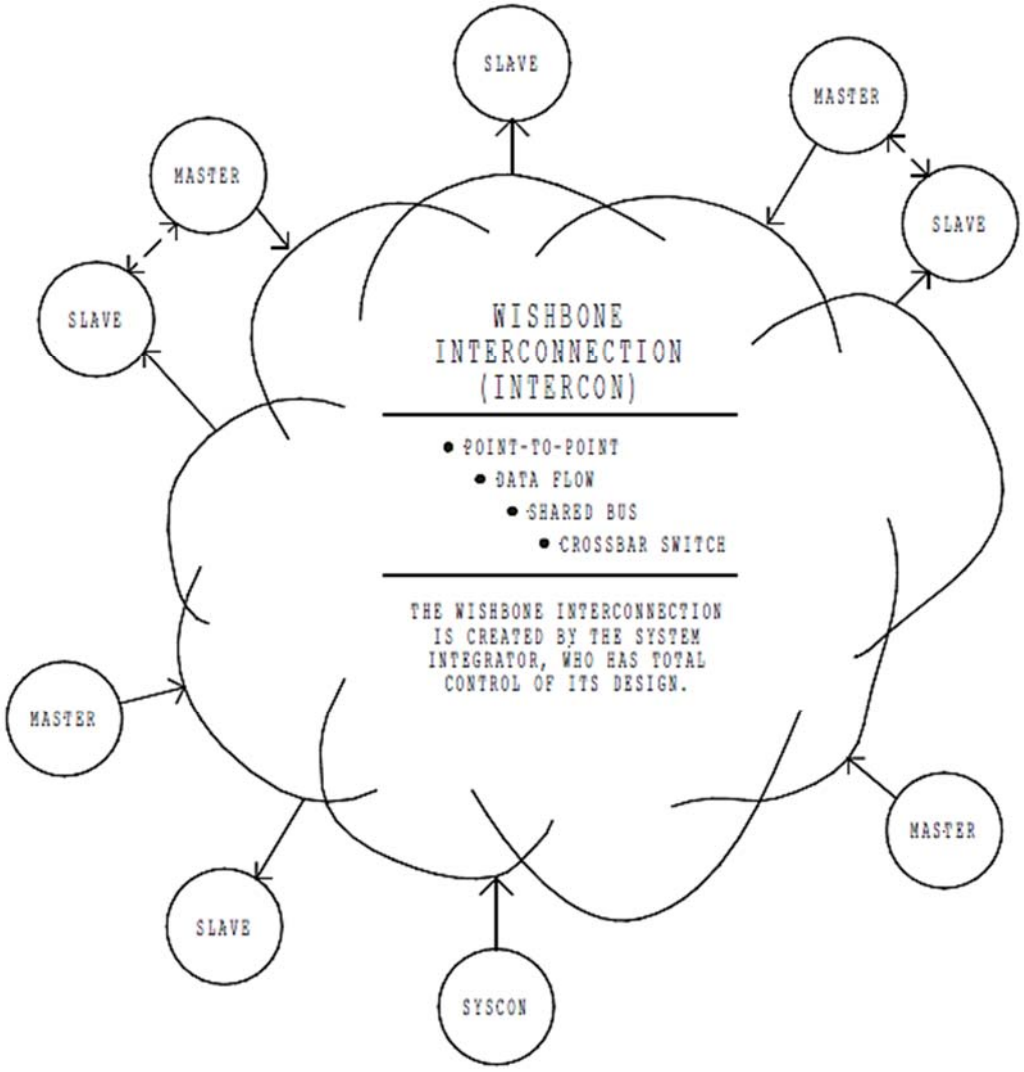



**Figure 3.1.** L'interconnexion Wishbone

L'interconnexion « Wishbone » elle-même n'est rien de plus qu'un grand circuit synchrone. Il doit être conçu pour fonctionner logiquement sur une gamme de fréquence presque infinie. Cependant, chaque circuit intégré a des caractéristiques physiques qui limitent la fréquence maximale du circuit. Dans la terminologie Wishbone, cela s'appelle une spécification de synchronisation variable. Cela signifie qu'un circuit compatible Wishbone fonctionnera théoriquement normalement à n'importe quelle vitesse, mais que sa vitesse maximale sera toujours limitée par la technologie de processus du circuit intégré.

Dans la suite de notre travail, nous définissons nos interconnexions Wishbone en utilisant le langage de description du matériel VHDL. Cela nous permet de définir pleinement nos interconnexions dans une manière qui correspond le mieux à l'application. Cependant, cela nous permet également de partager nos interconnexions avec les autres, qui peuvent les ajuster pour répondre à leurs propres besoins. De plus, l'interconnexion Wishbone permet de changer que la manière avec laquelle les IPs sont connectés.

Il y a quatre types définis d'interconnexion « Wishbone » :

- Point-à-point
- Bus partagé
- Interrupteur de barre transversale
- Commutateur Crossbar

L'interconnexion-hors-puce est le cinquième type possible. Cependant, les implémentations hors puce correspondent généralement à l'un des quatre autres types de base. Par exemple, les interfaces Wishbone sur deux circuits intégrés différents peuvent être connectées à l'aide d'une interconnexion point à point. La spécification Wishbone ne dicte pas comment ceux-ci sont implémentés, car l'interconnexion elle-même est une interface IP appelée « INTERCON ». Il est possible d'utiliser ou modifier un INTERCON déjà disponible ou en créer un.

### 5.1. Interconnexion point à point

L'interconnexion point à point est la manière la plus simple de connecter deux IPs Wishbone ensemble. Comme le montre la Figure 3.2, l'interconnexion point à point permet à une seule interface MASTER de se connecter à une interface SLAVE unique. Par exemple, l'interface MASTER peut être sur IP de microprocesseur et l'interface SLAVE sur une de mémoire.

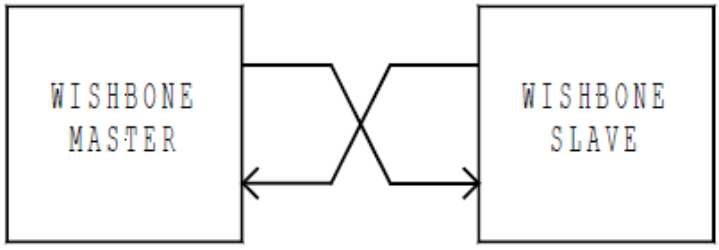



**Figure 3.2.** Interconnexion Wishbone Point-à-Point

### 5.2. Interconnexion en flux de données

L'interconnexion en flux de données est utilisée lorsque les données sont traitées de manière séquentielle. Comme illustré par la Figure 3.3, chaque IP dans cette architecture possède à la fois une interface MASTER et une interface SLAVE. Les données circulent de noyau à noyau. Ce processus est appelé interconnexion en pipeline. L'architecture en flux de données exploite le parallélisme, accélérant ainsi le temps d'exécution. Cela suppose, bien entendu, que les IPs ont des temps d'exécution similaires. Dans la pratique, un tel cas peut être plus complexe à adopter, mais cela illustre bien comment l'architecture en flux de données peut fournir un haut degré de parallélisme lors de la résolution de problèmes.

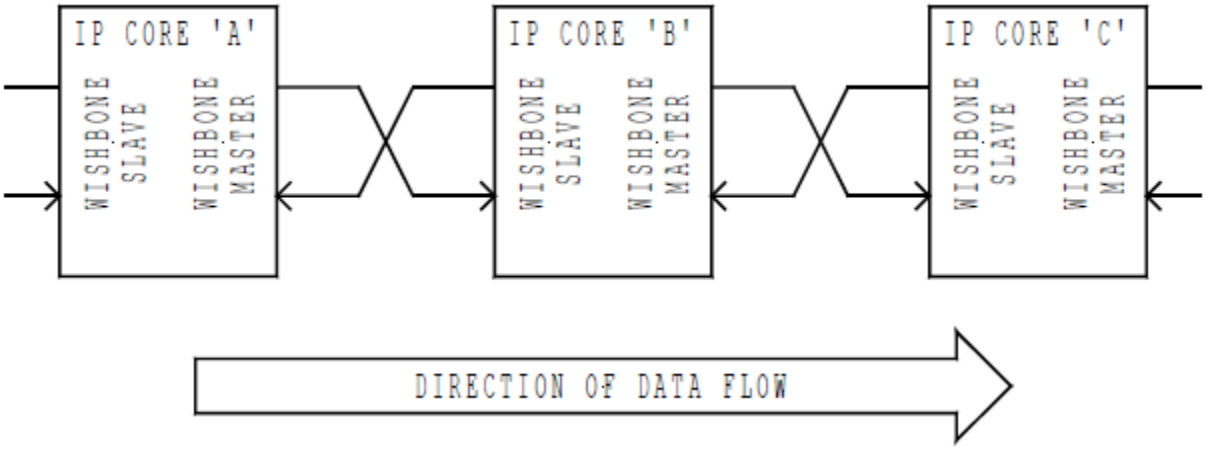



**Figure 3.3.** Interconnexion Wishbone en Flux de Données

### 5.3. Interconnexion en bus partagé

L'interconnexion en bus partagé est utilisée pour connecter deux IPs MASTER ou plus avec un ou plusieurs IPs SLAVE (Figure 3.4). Dans cette topologie, un MASTER initie un cycle de bus vers un SLAVE cible. L'IP SLAVE participe alors à un ou plusieurs cycles de communication avec le MASTER à travers la Bus. Un arbitre (non représenté sur la Figure 3.4) détermine quand un IP MASTER peut accéder au bus partagé. L'arbitre agit comme un «policier de la circulation» pour déterminer quand et comment chaque IP MASTER accède à la ressource partagée. Généralement, l'arbitre de type « Round Robin » est le plus usuel car il accorde l'accès sur une base prioritaire ou égale, respectivement. Le principal avantage de cette technique est que les systèmes d'interconnexion partagés sont relativement compacts. De plus, une telle interconnexion nécessite moins de ressources logiques et de routage que d'autres configurations, en particulier le commutateur crossbar. Son principal inconvénient est que les MASTER doivent attendre avant d'avoir accès à l'interconnexion. Ceci dégrade la vitesse globale à laquelle un MASTER peut transférer des données.

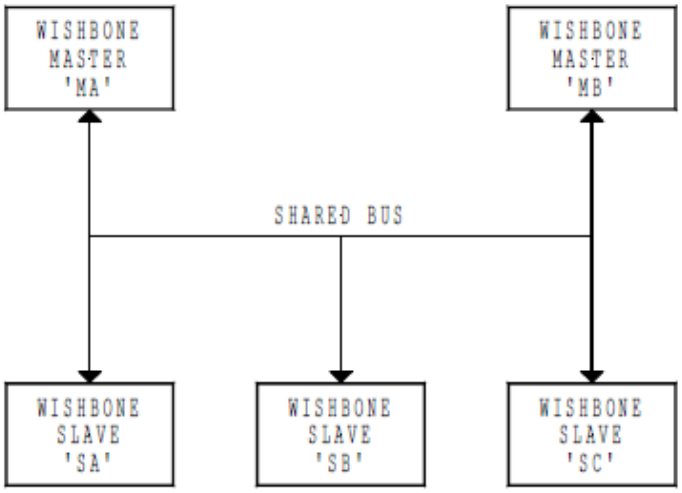



**Figure 3.4.** Interconnexion Wishbone en Bus Partagé

### 5.4. Interconnexion par commutateur Crossbar

L'interconnexion du commutateur crossbar est utilisée lors de la connexion de deux ou plusieurs MASTER ensemble afin que chacun puisse accéder à deux ou plusieurs SLAVE. Un diagramme est illustré à la Figure 3.5. Dans l'interconnexion par commutateur crossbar, un MASTER initie un cycle de communication adressable vers un SLAVE cible. Un arbitre (non représenté sur la Figure 3.5.) détermine quand chaque MASTER peut avoir accès au SLAVE indiqué. Contrairement à l'interconnexion par bus partagé, le commutateur crossbar permet à plus d'un MASTER d'utiliser l'interconnexion (tant que deux MASTER n'accèdent pas au même SLAVE simultanément).

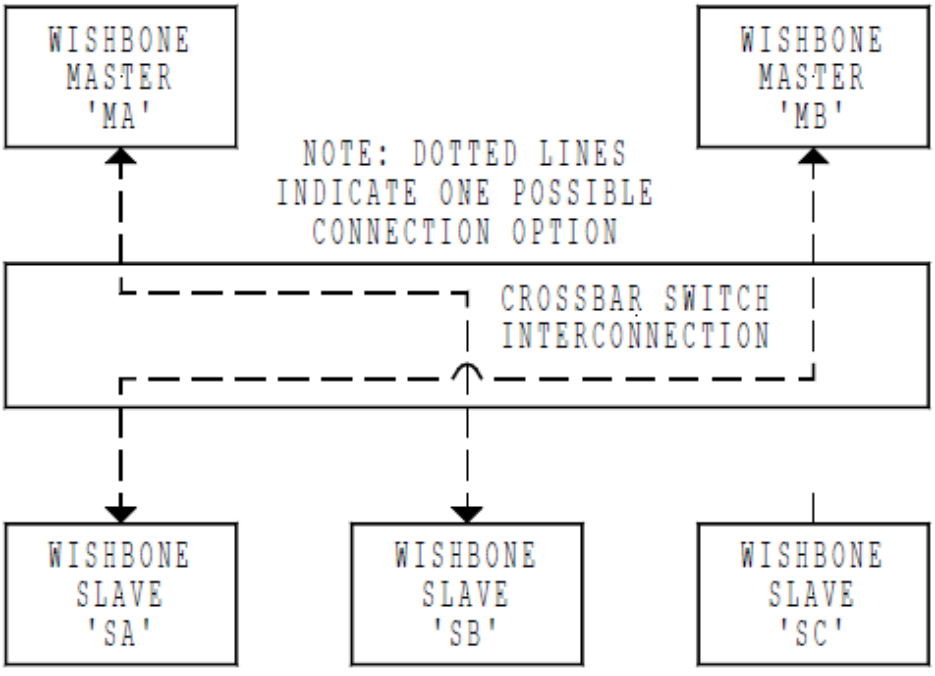



**Figure 3.5.** Interconnexion Wishbone par Commutateur Crossbar

Selon ce modèle, chaque MASTER dispose d'un «canal» sur le commutateur, où les données sont transférées entre le MASTER et le SLAVE via une liaison de communication privée. La figure montre deux canaux possibles qui peuvent apparaître sur le commutateur.

Le taux global de transfert de données du commutateur crossbar est supérieur à celui du bus partagé. Seulement, il nécessite plus de logique d'interconnexion et de ressources de routage que le bus partagé. En règle générale, un commutateur crossbar avec deux MASTER et deux SLAVE prend deux fois plus de logique d'interconnexion qu'un système de bus partagé similaire (avec deux MASTER et deux SLAVE).

## 6. Les signaux de l'interface Wishbone

Les interfaces Wishbone MASTER et SLAVE peuvent être connectées ensemble de plusieurs façons. Cela nécessite que les signaux d'interface Wishbone et les cycles de bus soient conçus de manière très flexible et réutilisable. Les signaux ont été définis avec les exigences suivantes :

Les signaux permettent aux interfaces MASTER et SLAVE de prendre en charge les interconnexions point-à-point, flux de données, bus partagé et commutateur crossbar.

Les signaux permettent trois types de base de cycle de bus. Ceux-ci incluent :

- Lecture / écriture simple (SRW : Single Read / Write),
- Lecture / écriture bloquée (BRW : Block Read / Write),
- Lecture / modification / écriture (RMW : Read / Modify / Write).

Le fonctionnement de ces cycles de bus sont décrit dans toutes les documentations Wishbone.

- Un mécanisme de « handshaking » est utilisé pour que l'interface MASTER ou l'interface SLAVE puisse ajuster le débit de transfert de données pendant un cycle de bus. Ceci permet d'ajuster la vitesse de chaque cycle de bus (ou phase) soit par l'interface MASTER soit par l'interface SLAVE. Cela signifie que tous les cycles de bus Wishbone fonctionnent à la vitesse de interface la plus lente.
- Le mécanisme de « handshaking » permet à un SLAVE d'accepter un transfert de données, de rejeter un transfert de données avec une erreur ou de demander au MASTER de réessayer un cycle de bus. Ceci est réalisé en générant les signaux [ACK_O], [ERR_O] ou [RTY_O] respectivement. Chaque interface doit prendre en charge le signal [ACK_O], par contre l'acquittement des deux autres signaux est facultatif.

- Comme le présente la Figure 3.6, tous les signaux sont disposés de sorte que les interfaces MASTER et SLAVE puissent être connectées directement pour former une interface point-à-point simple. Cela permet de construire des interfaces Wishbone très compactes et efficaces.
- Tous les signaux sur les interfaces MASTER et SLAVE sont des entrées ou des sorties, unidirectionnels. Cependant, il est permis (et parfois avantageux) d'utiliser des signaux bidirectionnels dans la logique d'interconnexion si le dispositif cible le supporte.
- Les largeurs du bus d'adresse et de données peuvent être modifiées pour s'adapter à l'application. 8, 16, 32 et 64 bits pour le bus de données de bits et de 0 à 64 bits pour le bus d'adresses.

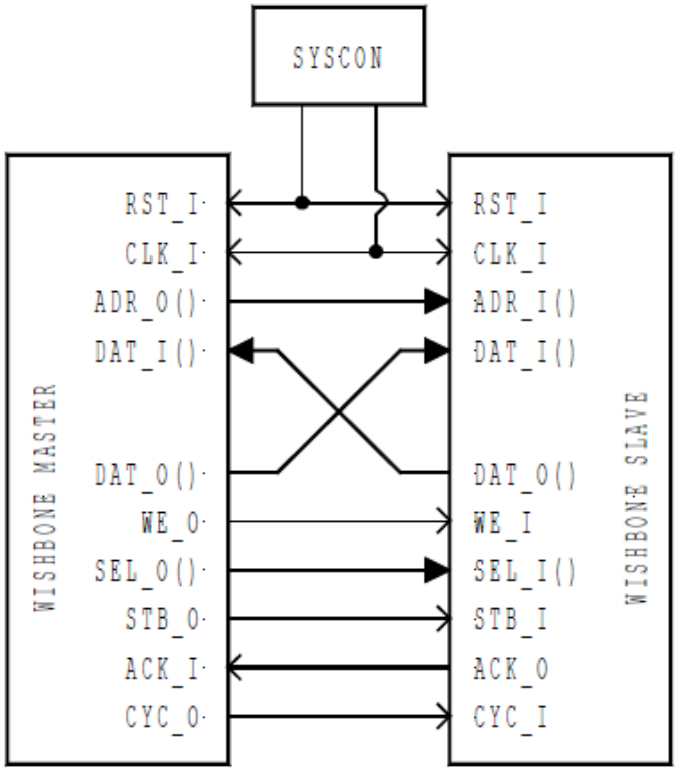



**Figure 3.6.** Signaux pour interconnexion Wishbone Point-à-Point

## 7. Plateforme de génération des réseaux sur puce de type Wishbone

Généralement, une architecture d'un SoC basé sur une interconnexion de type NoC consiste en un ensemble de routeurs entourés d'éléments de processeur (PEs). Ces PEs peuvent être des IPs, des mémoires ou des CPU, connectés au réseau via les interfaces réseau (NI). Ces interfaces ont pour rôle d'adapter les IPs au réseau. Un réseau sur puce doit répondre aux exigences de l'application tout en gérant les coûts, la surface des puces et les contraintes d'alimentation. Nous devons choisir la bonne topologie de réseau, l'algorithme de routage adéquat, la meilleure stratégie de stockage, la taille

optimale des buffers, etc. Ainsi, plusieurs études ont été réalisées sur la conception [33, 79], l'exploration topologique [31,80-82], les algorithmes de routage [83], la microarchitecture du routeur [84-88] garantie de service (QoS) [89], tests et vérifications [82]. Néanmoins, un plus grand débit et une plus faible latence ne sont pas les seuls paramètres à prendre en compte. Une utilisation réduite de la surface FPGA ou une plus faible consommation d'énergie peuvent représenter des objectifs cibles de la conception d'un réseau sur puce. Certains des travaux récents sur les réseaux sur puce se sont concentrés sur la complexité de la conception [89].

Un tour d'horizon des NoCs nous a permis d'identifier les contraintes de la conception d'une architecture de réseau sur puce. Le travail réalisé dans cette section consiste à proposer une plateforme pour la génération des réseaux sur puce au niveau VHDL sans pour autant avoir une grande connaissance de ce langage de description, tout en offrant un large choix de possibilités.

## 7.1. Architecture du routeur générique de base

Afin de développer cette plate-forme, nous commençons par définir le routeur générique de base. Ce routeur générique prend en charge plusieurs fonctionnalités à la fois, ce qui augmente la complexité de la conception. Pour répondre à toutes ces contraintes, nous proposons une architecture de routeur supportant plusieurs standards en termes de topologie, d'algorithmes de routage, de stratégie de stockage, d'arbitrage, de mécanismes de commutation.

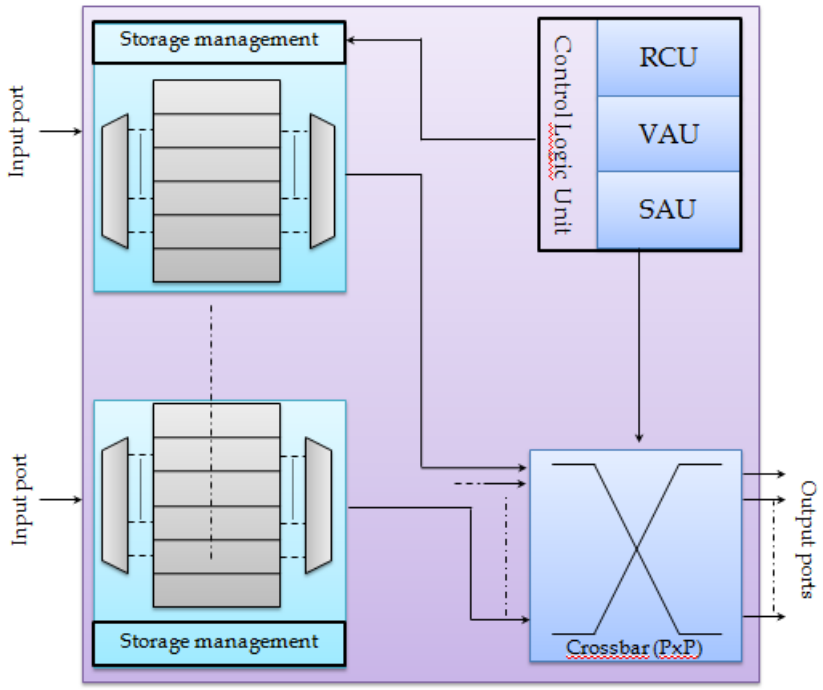



**Figure 3.7.** Architecture du routeur générique utilisé dans la plateforme

L'architecture du routeur générique est illustrée par la Figure 3.7. L'idée principale derrière cette architecture est de construire un routeur de base avec des composants génériques pour offrir la possibilité de passer facilement d'une configuration à une autre sans conflit. Cette architecture est grandement inspirée de celle des routeurs à canaux virtuels (ViChaR).

Les paquets entrants sont stockés dans des buffers (FiFo) situés à chaque port d'entrée du routeur. La stratégie des files d'attente à l'entrée est présente une flexibilité qui sera utile pour la génération de différents types de NoC.

L'unité de contrôle logique (Control Logic Unit) au sein de cette architecture permet de gérer toute la logique nécessaire au bon fonctionnement du routeur afin d'aiguiller les paquets vers les ports de sortie correspondant à la destination optimale. Elle est composé de :

L'Unité d'Allocation de Canaux Virtuels (VAU) : c'est la logique de gestion des emplacements dans les buffers. Elle stocke les paquets de manière à éviter qu'un paquet reste bloqué dans le buffer en attendant que le port de sortie attribué soit libre. Ce module associe les demandes de ressources provenant des canaux virtuels d'entrée aux canaux virtuels disponibles sur les routeurs en aval. Comme l'allocateur de canaux virtuels établit une correspondance entre les demandes de ressources provenant des canaux virtuels d'entrée et les canaux virtuels de sortie disponibles, sa surcharge matérielle augmente avec le nombre de canaux virtuels utilisés dans chaque direction. La complexité et la latence d'une VAU dépendent de la plage de sélection du routage. Selon notre conception du routeur générique, le circuit de routage renvoie les canaux virtuels de sortie candidats dans un seul canal physique pour être sélectionnés par un canal virtuel d'entrée. La taille de la requête de l'arbitrage d'entrée est égale au nombre de canaux virtuels, **V**, partageant un canal physique. Avec le souci de la consommation en FPGA du circuit et de la latence, le choix de l'allocateur pour effectuer une attribution de canal virtuel est très utile. De plus, elle permet aussi à l'utilisateur de définir entre autre le mécanisme de commutation à utiliser parmi la commutation par circuit (pour un NoC à service garantie) ou la commutation par paquet (pour un NoC de type meilleur effort).

L'unité de calcul de routage (Routing Computation Unit) : Elle pilote toute la partie du routage des paquets (et / ou des flits). Plusieurs types de routages sont possibles et sont effectués en fonction du port d'entrée, des coordonnées de la destination et du trafic au sein du réseau.

Unité d'allocation de commutation (SAU) : cette unité fait le lien entre le routage calculé et les ports de sortie via le crossbar (5X5). La SAU alloue un créneau temporel du routeur au crossbar pour déplacer les paquets à partir d'un canal virtuel d'entrée vers un canal physique de sortie. Dans un NoC conventionnel, la SAU est réalisée avec un arbitrage en deux étapes, où l'arbitre a un nombre égal de lignes d'entrée et de lignes de sortie, et sa sortie est le signal d'attribution.

## 7.2. Flot de conception pour la génération des réseaux sur puce de type Wishbone

Afin de simplifier et automatiser la génération de Network on Chip de type Wishbone au niveau VHDL, nous avons développé cette plate-forme qui à partir d'un certain nombre de paramètres d'entrée permet de générer le code VHDL du NoC en question. Le flot de conception est décrit en plusieurs étapes :

- Première étape : Dans cette phase l'utilisateur choisit la topologie et la dimension (nombre de routeur à utiliser dans le réseau). Ce premier choix permet de définir une catégorie du routeur parmi plusieurs fournis dans une bibliothèque d'IPs que nous avons conçu (Figure 3.8).

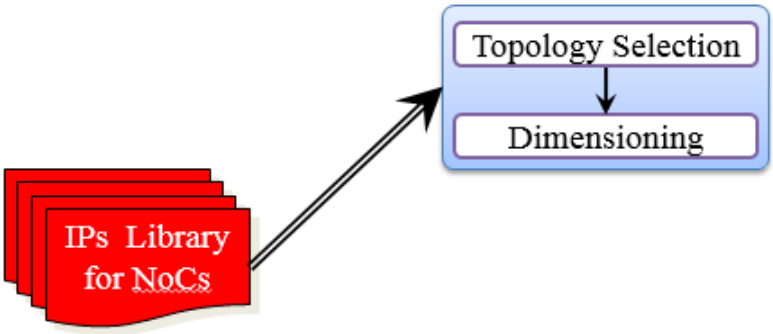



**Figure 3.8.** 1ère étape dans le flot de conception d'un NoC

- Deuxième étape : Comme illustré dans la Figure 3.9, d'autres paramètres sont saisis par l'utilisateur :
  - Stratégie des files d'attentes, Choix du contrôle de flux, Paramétrisation des buffers (choix de la largeur et de la profondeur des FiFo), Largeur des paquets de données, Algorithme de routage, Choix du type d'arbitrage,

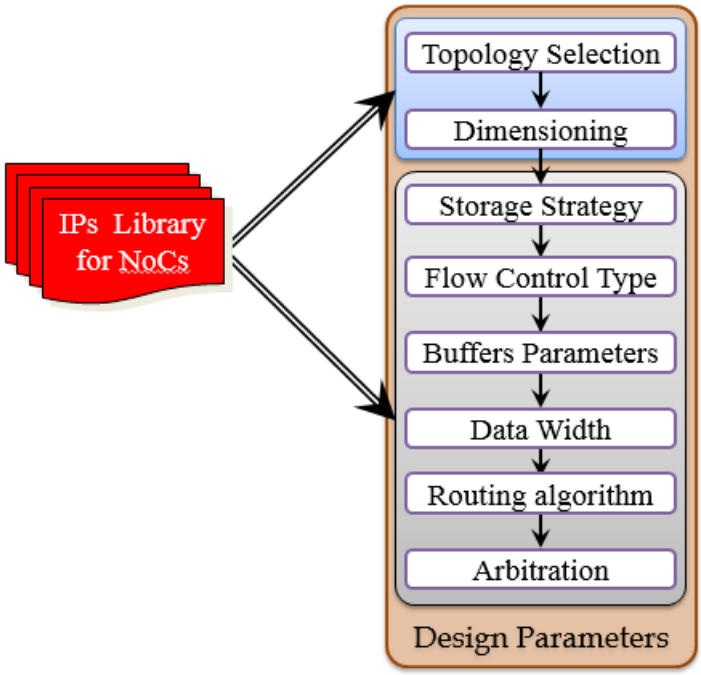



**Figure 3.9.** 2ème étape dans le flot de conception d'un NoC

- Troisième étape : La Figure 3.10 présente l'étape de génération du code VHDL de la plateforme. Une fois le code VHDL généré, le rapport est synthétisé par l'outil de développement ISE Design Suite de Xilinx. Ce rapport nous permet d'avoir une première idée sur le coût en ressources logiques du NoC ainsi que la fréquence de fonctionnement maximale.

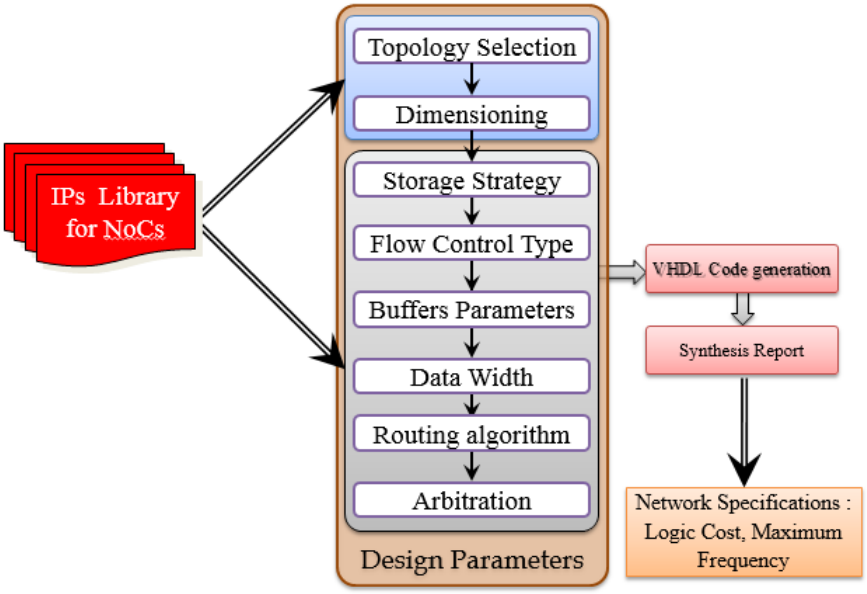



**Figure 3.10.** 3ème étape dans le flot de conception d'un NoC

- Quatrième étape : illustrée par la Figure 3.11, c'est l'étape de test et de validation de l'architecture NoC générée. Pour cela, nous connectons différents IPs au réseau afin de quantifier la bande passante et la latence du NoC dans différentes conditions.

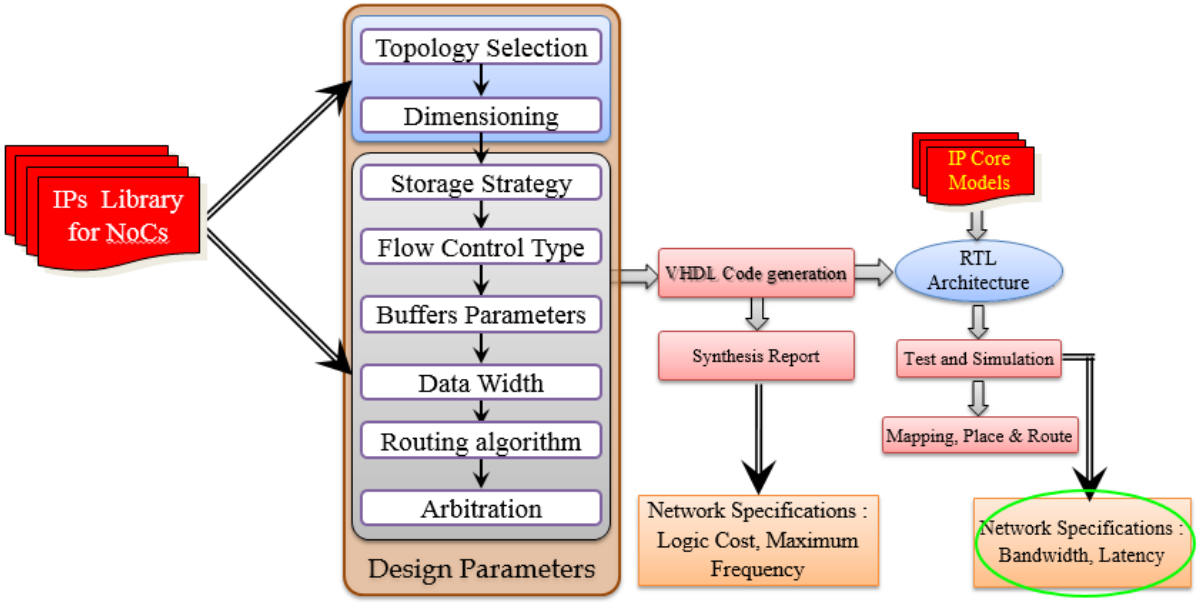



**Figure 3.11.** 4ème étape dans le flot de conception d'un NoC

## 8. Adaptation et connexion des éléments Wishbone avec le NoC

Afin d'adapter notre réseau sur puce aux cœurs IP Wishbone, une interface réseau est utilisée. Cela permet aux utilisateurs de simplement brancher des cœurs IP Wishbone « Plug and Play » [90]. Une telle facilitation permettra aux IPs de type Wishbone d'être reconnus rapidement et automatiquement par le réseau dès le branchement, et sans avoir recours à une adaptation du protocole. De plus, une telle approche permet l'installation de ces composants Wishbone en requérant un minimum d'intervention de la part de l'utilisateur et par conséquent en minimisant les erreurs de manipulation et de paramétrage.

Etant donné que « Wishbone » est une architecture d'interconnexion maître et esclave, nous avons conçus deux interfaces réseau différentes Wb_MNI (Wishbone Master Network Interface) pour les IPs maîtres Wishbone et Wb_SNI (Wishbone Slave Network Interface) pour les IPs esclaves [90]. L'exemple de la connexion d'un composant Wishbone Maître à travers l'interface réseau est illustré par la Figure 3.12.

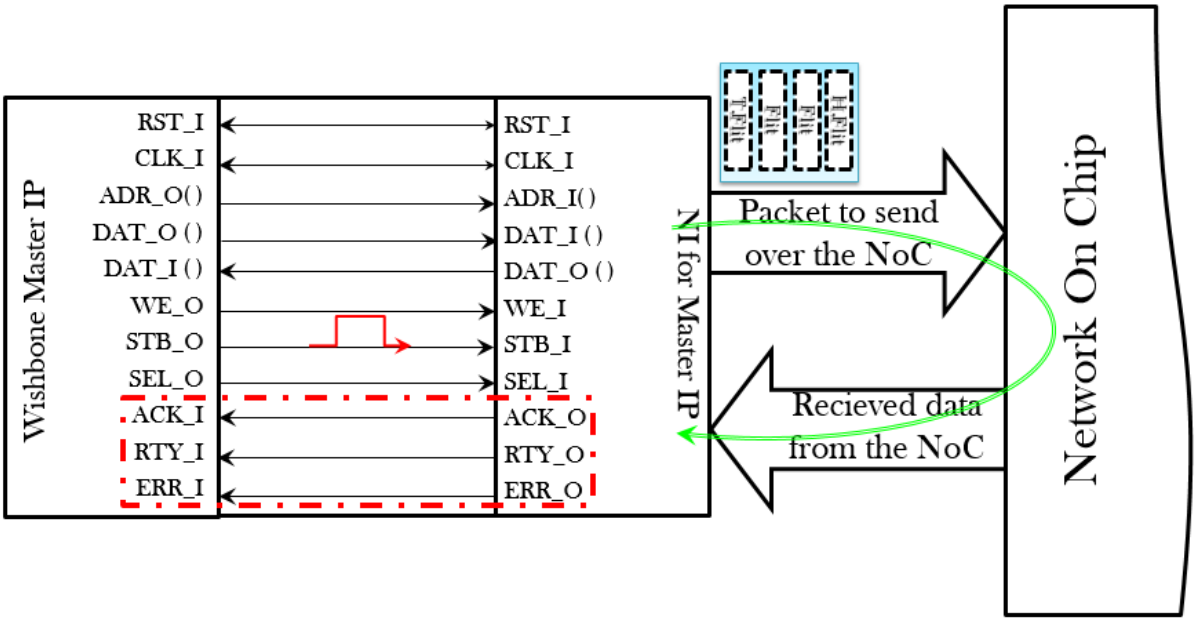



**Figure 3.12.** Adaptation et connexion des composants Wishbone au réseau sur puce

## 8.1. Fonctionnement de l'interface réseau pour les composants Wishbone Maîtres (Wb-MNI)

Le composant Wishbone reste en attente jusqu'à ce que des signaux « Strobe » ou « Cycle » soient confirmés. Ensuite, le contrôleur sauvegarde le paquet avec ses signaux Wishbone puis calcule l'adresse de destination. Le contrôleur vérifie si la requête est une demande d'écriture, puis commence l'envoi et reste dans un état d'attente jusqu'à ce que le signal d'acquittement soit reçu de la part de l'esclave. Une fois ce signal reçu, le contrôleur revient à l'état d'attente pour une autre transaction Wishbone. Si la requête est une demande de lecture, le contrôleur commence l'envoi et reste en état d'attente jusqu'à ce que les données et le signal d'acquittement soient reçus, le contrôleur confirme son signal d'acquittement pour indiquer que les données ont été reçues et l'opération terminée avec succès. Le contrôleur revient son état d'attente pour une autre transaction Wishbone. La Figure 3.13 résume l'organigramme de l'interface du réseau pour les composants Wishbone Maîtres.

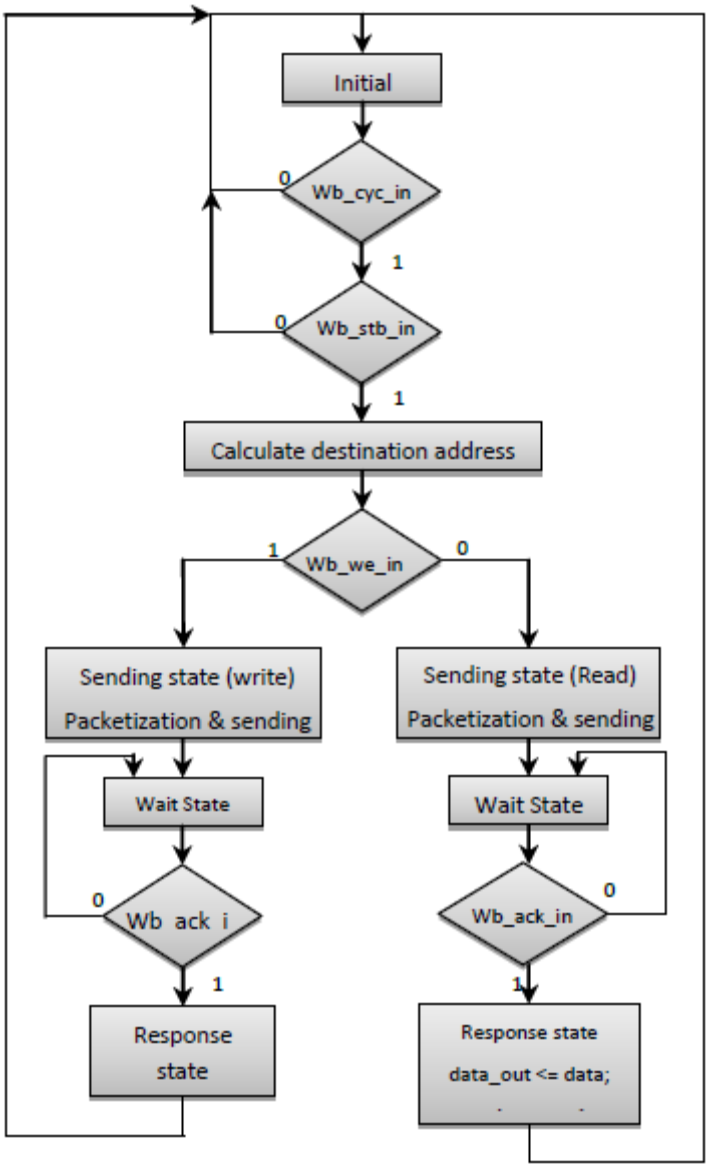



**Figure 3.13.** Organigramme de l'interface réseau pour les composants Wishbone Maîtres

## 8.2. Fonctionnement de l'interface réseau pour les composants Wishbone Esclaves (Wb_SNI)

L'interface du réseau des composants esclaves Wishbone est similaire à celle des composants maîtres. Le contrôleur commence par un état d'attente jusqu'à ce qu'un signal « Strobe » soit activé. Le contrôleur calcule l'adresse de destination de retour puis attend le signal d'acquittement du composant Esclave. Une fois ce signal d'acquittement reçu, le contrôleur lance l'état de mise en paquets et envoie le signal de réponse avant de revenir à l'état d'attente initial et d'attendre une autre transaction de Wishbone. La Figure 3.14 montre l'organigramme de l'interface du réseau esclave du Wishbone.

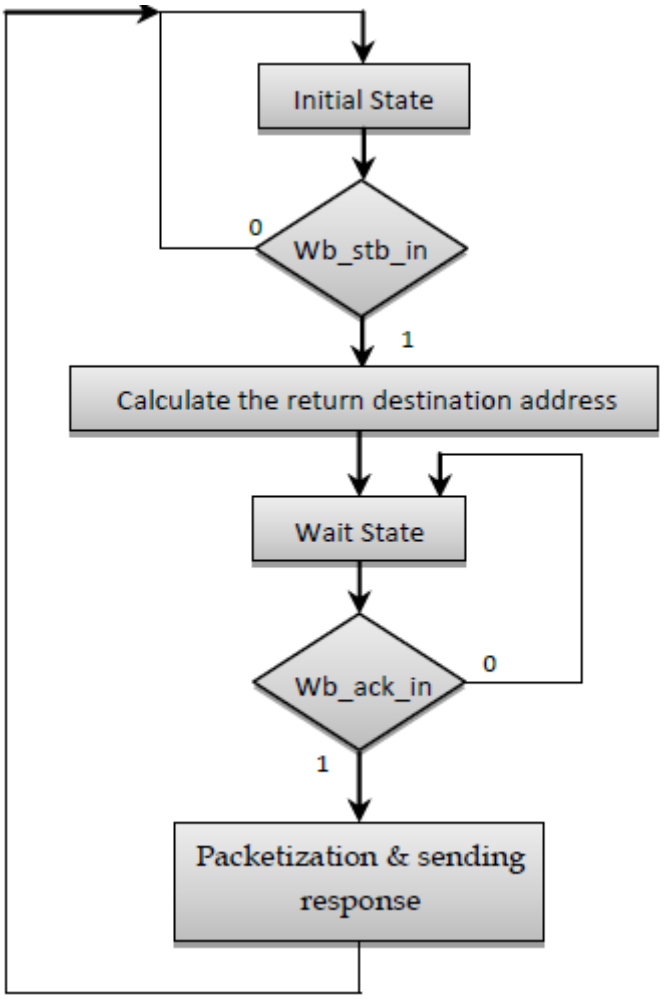



**Figure 3.14.** Organigramme de l'interface réseau pour les composants Wishbone Esclaves

## 9. Test de la plateforme pour la génération des réseaux sur puce de type Wishbone

Des approches rapides et précises pour analyser les mesures critiques telles que les performances, la consommation d'énergie ou la tolérance aux pannes du système sont importantes pour guider le processus de conception. Cependant, pour pouvoir être utilisées dans une boucle d'optimisation ou faire des choix de conception précoces, les techniques d'analyse doivent pouvoir être traitées et fournir un retour d'information significatif aux concepteurs. Durant les phases de conception ultérieures, nous pourrons obtenir des estimations plus précises grâce à la simulation.

La latence de communication et la bande passante du réseau sont des indicateurs de performance communs intéressants. Bien qu'il soit relativement plus facile de trouver la latence de communication pour un trafic de service garanti, l'analyse de la latence moyenne pour un trafic de type « Best Effort » reste une tâche difficile. Par conséquent, le nombre moyen de sauts ou le délai de paquet libre sont généralement utilisés pour approcher la latence moyenne des paquets.

L'analyse des performances dépend en grande partie de diverses hypothèses simplificatrices sur les caractéristiques du réseau ou du trafic (trafic uniforme ou trafic en rafales, par exemple) et suppose généralement un routage déterministe en raison de la difficulté à gérer le problème le plus général. Les approches basées sur la simulation sont toujours populaires pour l'exploration architecturale des réseaux sur puce en raison de leur précision, de leur flexibilité et de leur capacité à exécuter de véritables charges de travail.

Le principal problème avec les approches basées sur la simulation est le compromis entre le niveau de détail de la mise en œuvre et le temps de simulation. Les modèles détaillés peuvent fournir des résultats très précis, mais le temps de simulation peut être prohibitif. Une simulation réaliste des traces synthétiques ou une accélération matérielle peuvent être utilisées pour améliorer la vitesse de simulation. Il s'agit donc de directions ouvertes pour les chercheurs.

Pour valider fonctionnellement l'approche de la plateforme pour la génération des réseaux sur puce de type Wishbone, nous exposons dans cette section les résultats de simulation de plusieurs NoCs connectés à des modules de communication. Pour cette partie de simulation, nous avons utilisé l'outil logiciel ISE 14.1 Design Suite de Xilinx pour : synthétiser les architectures, effectuer l'analyse temporelle et visualiser les schémas RTL sur la plateforme FPGA Virtex-7 XC7V2000T. L'environnement de simulation multi-langage HDL ModelSim 9.2 a été utilisé pour la simulation et le débogage.

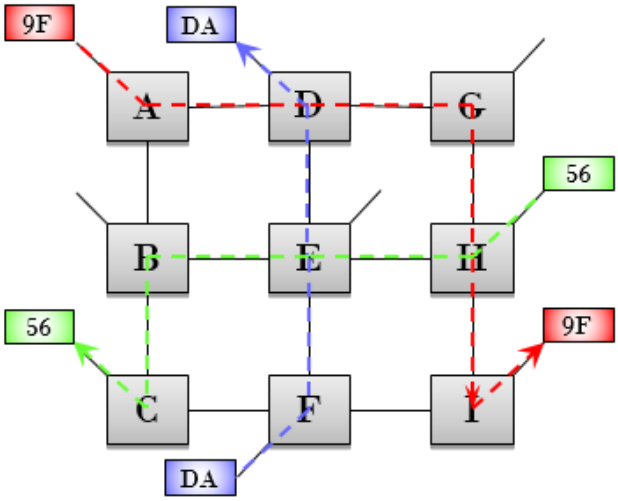
### 9.1. Simulation d'un NoC 3x3 Mesh généré par la plateforme

Pour cette première simulation nous testerons la plateforme pour la génération d'un réseau sur puce de topologie 3x3 Mesh. Les routeurs sont configurés pour utiliser des paquets de données composées de 8 bits.

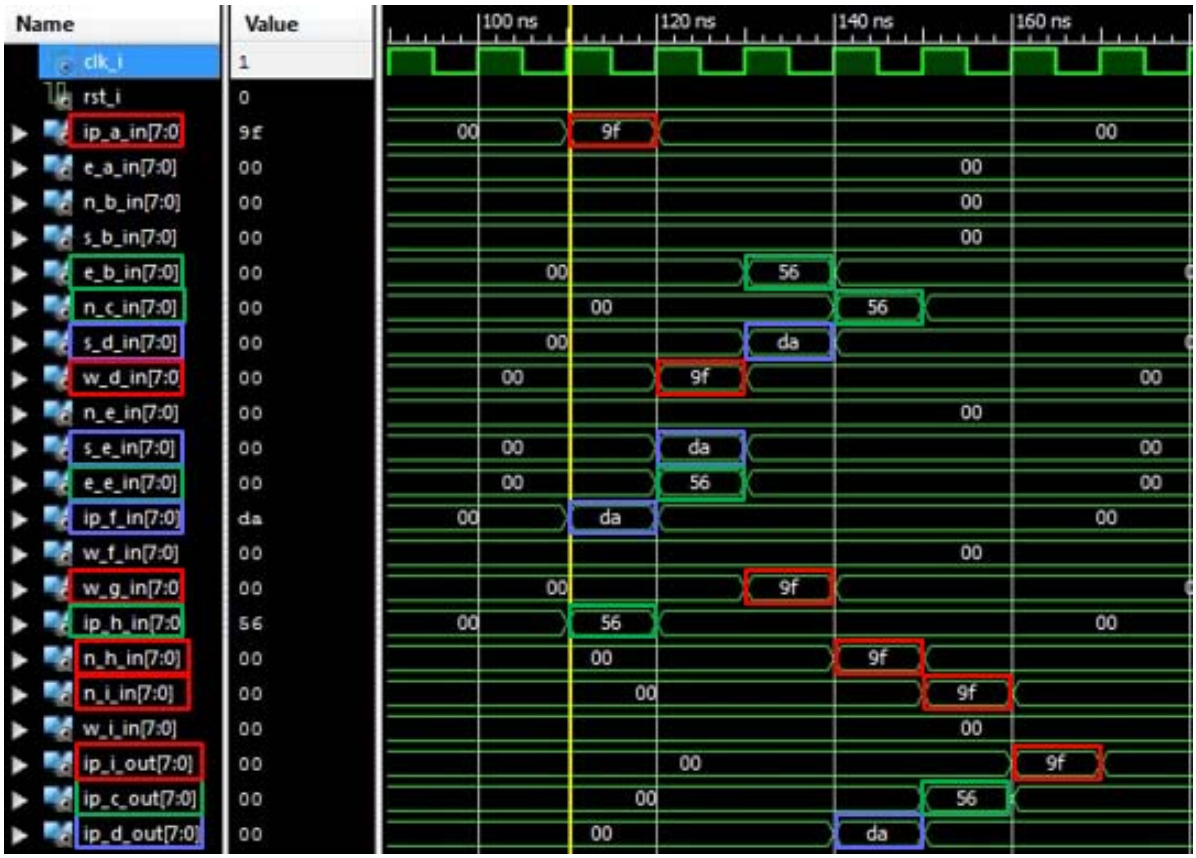
- NoC 3x3 Mesh utilisant un algorithme de routage XY conventionnel,
- Commutation par paquet S&F, arbitrage par Round Robin,
- Envoi de 3 paquets de 8 bits vers des destinations différentes.

Une fois le NoC généré par la plateforme, nous procédons au test en configurant le réseau comme suit :

- Des paquets sont générés par des émetteurs de types Wishbone, et envoyés vers des récepteurs du même type. Ces paquets sont de taille identique et générés de façon aléatoire dans le but de tester le NoC généré.




(a)




(b)

**Figure 3.15.** Résultats de simulation d'un NoC 3x3 Mesh utilisant un algorithme de routage XY

La figure 3.15 est une capture d'écran des résultats de simulation. On peut voir que les trois paquets sont injectés dans le NoC au même instant $t_0$. De plus, on peut aussi constater que le transfert de ces paquets de routeur vers le routeur voisin, jusqu'à atteindre la destination, ne nécessite qu'un cycle d'horloge du fait de la faible charge en données du réseau. Chaque routeur permet simultanément la réception et l'envoie

des données. Pour une topologie maillée régulière, l'algorithme XY conventionnel offre une performance décente.

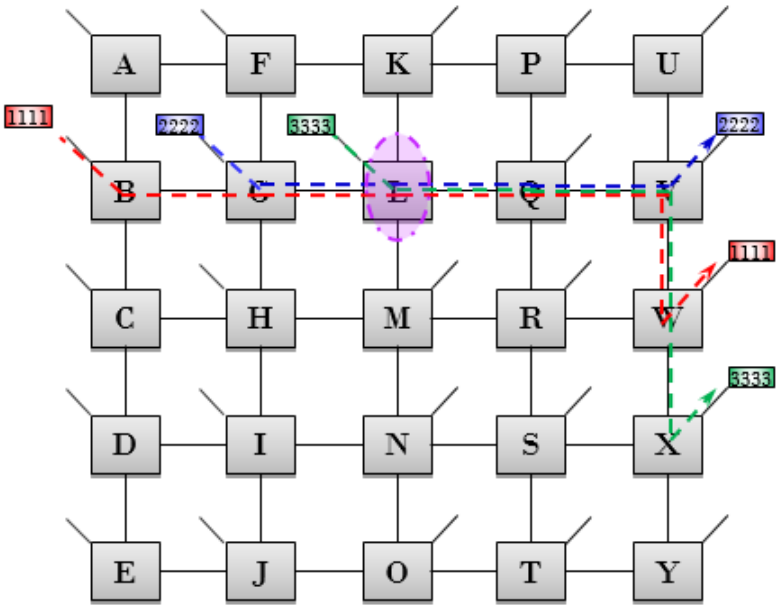
### 9.2. Simulation d'un NoC 5x5 Mesh et comparaison de deux algorithmes routage différents

Dans cette deuxième simulation, nous utilisons la plateforme pour la génération d'un NoC de type Mesh 5x5, avec un arbitrage en round robin et une commutation par paquet Store-&-Forward.

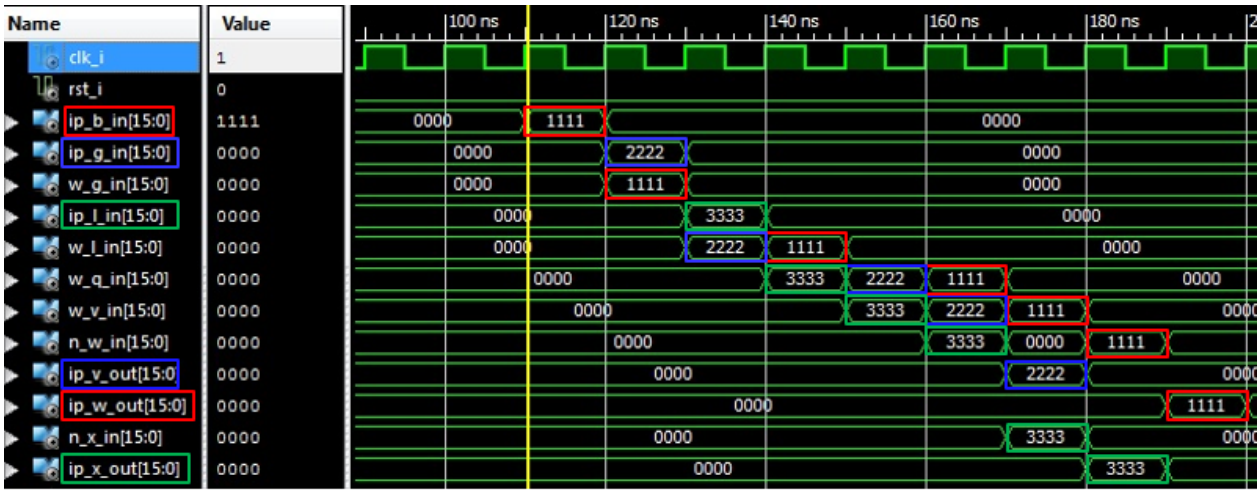
Afin de tester les limites de ce réseau, et comparer l'efficacité des algorithmes de routage XY (conventionnel et adaptatif), nous effectuons une simulation avec l'envoi de 3 paquets de 16 bits en provoquant des situations de congestion.

**1ère simulation :**

- Algorithme de routage XY conventionnel,
- Commutation par paquet S&F,
- Envoi de 3 paquets de 16 bits avec gestion de situations de congestions,
- Arbitrage par Round Robin.




(a)



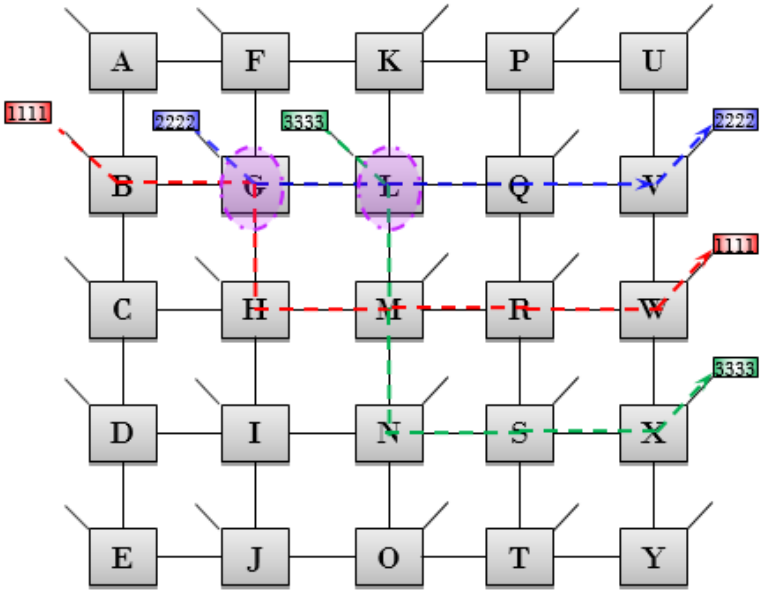

(b)

**Figure 3.16.** Résultats de simulation d'un NoC 5x5 Mesh avec algorithme de routage XY conventionnel

Les mêmes paquets déjà utilisés dans la simulation ci-dessus sont envoyés sur un réseau similaire utilisant un algorithme adaptatif. Les résultats de la simulation ci-dessus seront discutés dans la section suivante afin de comparer les résultats des deux simulations.

**2ème simulation :**

- Algorithme de routage XY adaptatif,
- Commutation par paquet S&F,
- Envoi de 3 paquets de 16 bits avec gestion de situations de congestions,
- Arbitrage par Round Robin.




(a)


(b)

**Figure 3.17.** Résultats de simulation d'un NoC 5x5 Mesh utilisant un algorithme de routage XY adaptatif

Les figures 3.16 et 3.17 illustrent deux simulations semblables, la première mettant en avant un routage XY conventionnel et la seconde un routage XY adaptatif. Dans les deux simulations, les trois paquets sont injectés à des intervalles réguliers d'un cycle d'horloge afin de provoquer une première situation de congestion au port de sortie « East » du routeur « G » et une autre situation de congestion au port de sortie « East » du routeur « L ».

En observant de plus près la figure 3.16, nous pouvons remarquer que pour un routage XY conventionnel, si deux paquets demandent un même port de sortie, l'arbitre « Round Robin » attribuera une priorité à l'un des deux paquets, selon le sens défini, afin d'éviter le conflit (dans le cas présent P= « 2222 » est envoyé en premier via le port « East » du routeur « G »). Par conséquent, le paquet n'ayant pas la priorité devra attendre un cycle d'horloge supplémentaire avant d'être envoyé vers le routeur suivant (dans le cas présent P= « 1111 » est envoyé en seconde position via le port « East » du routeur « G »).

Une situation de congestion similaire se produit au niveau du routeur « L » où plusieurs paquets requièrent le même port de sortie « East ». De façon analogue, l'arbitrage attribuera les priorités, et les paquets « 3333 », « 2222 », « 1111 » sont

envoyés respectivement dans l'ordre. Un tel algorithme de routage permet facilement de surmonter les situations de conflits entre les paquets. Cependant, les paquets sont parfois contraints de passer des cycles d'attente dans le routeur avant d'être envoyés via le port de sortie choisi. Les paquets « 1111 » et « 2222 » doivent patienter deux cycles et un cycle d'horloge respectivement pour arriver à leurs destinations, contrairement au paquet « 3333 », qui a été favorisé, et n'aura aucun cycle d'attente durant son parcours.

L'algorithme de routage adaptatif XY permet de supprimer ces délais d'attente lors des situations de congestion. La figure 3.17 démontre l'efficacité du routage adaptatif XY face à aux mêmes situations de congestion. La première situation de conflit se présente au niveau du routeur « G », et contrairement au routage conventionnel, les deux paquets « 1111 » et « 2222 » n'auront aucun cycle d'attente et seront aiguillés simultanément vers les ports « South » et « East » respectivement. Le paquet « 1111 » pourra atteindre sa destination via un chemin différent. Une seconde situation similaire se reproduira au niveau du routeur « L » avec les paquets « 2222 » et « 3333 ». Pour ce cas-là, le paquet « 2222 » sera routé vers le port de sortie « East » alors que le paquet « 3333 » sera routé vers le port de sortie « South ». Un tel aiguillage entrainera une troisième situation au niveau du routeur « M » avec les paquets « 1111 » et « 3333 ». Dans ce dernier cas, le routage adaptatif XY permettra encore une fois d'éviter la congestion et les paquets « 1111 » et « 3333 » seront orientés vers les ports de sortie « East » et « South » respectivement.

D'autre part, en comparant les : figure 1.16 et 1.17, nous observons que le paquet « 3333 » nécessite le même nombre de cycles d'horloge pour atteindre sa destination quel que soit le type de routage XY utilisé (conventionnel ou adaptatif). Cependant, avec un routage XY adaptatif, les latences des paquets « 1111 » et « 2222 » sont réduites par rapport à un routage XY conventionnel. La latence du paquet « 2222 » est réduite de 5 cycles d'horloge à 4, et celle du paquet « 1111 » passe de 8 cycles d'horloge à 6 cycles. Une telle optimisation de la latence peut s'expliquer par le caractère dynamique du routage adaptatif qui permet d'aiguiller les paquets vers les ports de sortie les moins sollicités, en parcourant parfois des distances supérieures sans pour autant augmenter la latence des paquets au sein du réseau. Une telle approche s'apparente à un mécanisme de déflection qui permet d'éviter que les paquets restent bloqués dans les buffers en attendant qu'ils aient l'accès aux ports de sortie.

### 9.3. Comparaison des performances pour un NoC 5x5 Mesh : Routage XY conventionnel et Routage XY adaptatif

Pour avoir des éléments de comparaison sur la complexité de la mise en œuvre des différentes solutions et les performances des réseaux selon le routage utilisé, nous exposons à travers les figures 3.18 et 3.19 quelques résultats de synthèse de la simulation des deux approches pour la topologie Mesh.

Pour deux NoCs aux caractéristiques quasi-identiques, excepté l'algorithme de routage, on peut facilement voir que le routage adaptatif est plus coûteux en ressources logiques qu'un routage conventionnel. Cette différence en ressources logiques, bien présente quel que soit la taille des paquets utilisée sur le réseau, s'explique par l'augmentation de la complexité pour un routage adaptatif par rapport au routage conventionnel. Néanmoins, un tel coût élevé se traduit aussi par une fréquence de fonctionnement maximale plus élevée. Pour les utilisateurs, il s'agira surtout de trouver le meilleur compromis pour la solution d'interconnexion. De ce fait, choisir un réseau de type « Guaranteed Service » revient à favoriser une latence plus faible et une fréquence maximale plus élevée sera au détriment d'un coût logique plus élevé. En outre, un NoC de type « Best Effort » correspond à un routage XY conventionnel qui offre une performance moyenne pour un coût inférieur en ressources logiques.

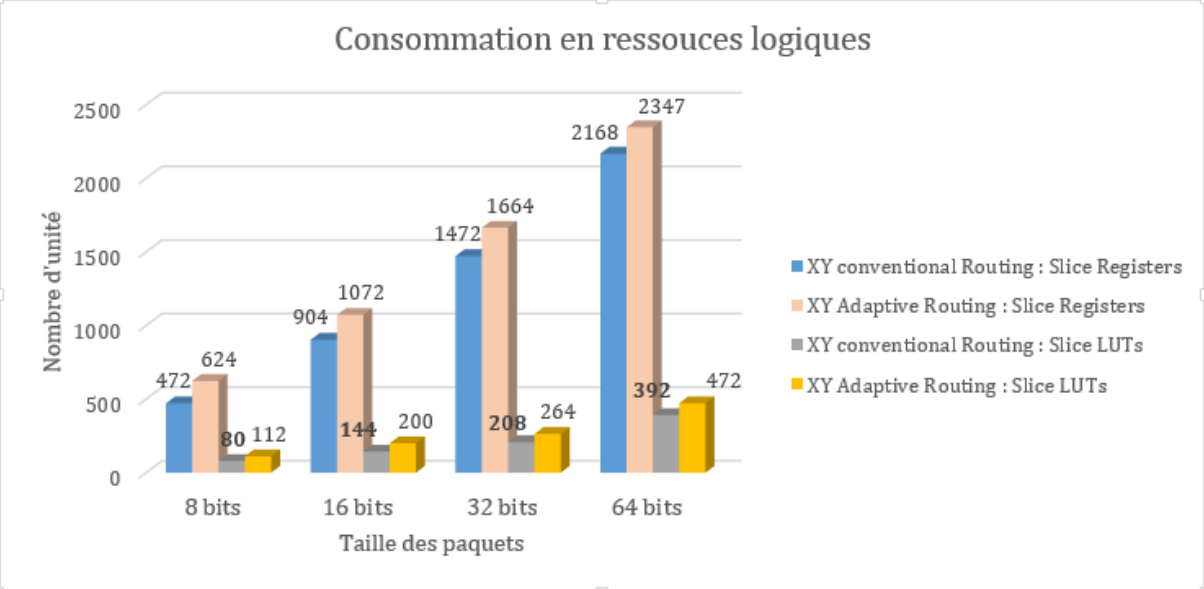



**Figure 3.18.** Consommation en Silce Registers, Slice LUTs pour différentes largeurs de paquets : NoC 5x5 Mesh (routage XY Conventionnel et routage XY adaptatif)

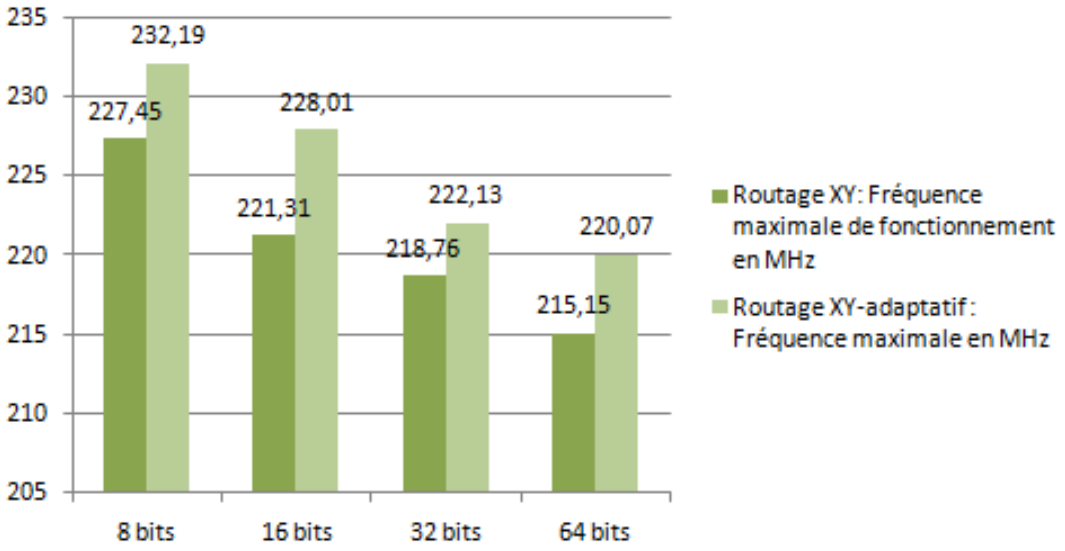



**Figure 3.19.** Fréquence de fonctionnement maximale pour différentes largeurs de paquets : NoC 5x5 Mesh (routage XY Conventionnel et routage XY adaptatif)

Une technique appelée « réseau orienté service » (Service Oriented Network) a émergée, où l'en-tête du paquet contient un type PE de destination plutôt qu'une adresse de routeur de destination. Le type d'un PE est attribué en fonction du service offert par le PE. Le routeur transmet ensuite automatiquement un paquet au meilleur PE de ce type. Ainsi, même si un routeur de destination actuel devient soudainement indisponible, un paquet sera dynamiquement redirigé vers le meilleur PE suivant sans aucune interruption. En adoptant cette approche orientée service, nous pouvons établir un transfert de paquets au niveau du matériel pour protéger contre les défaillances des routeurs.

### 9.4. Génération et simulation d'un NoC en Arbre à 4 étages

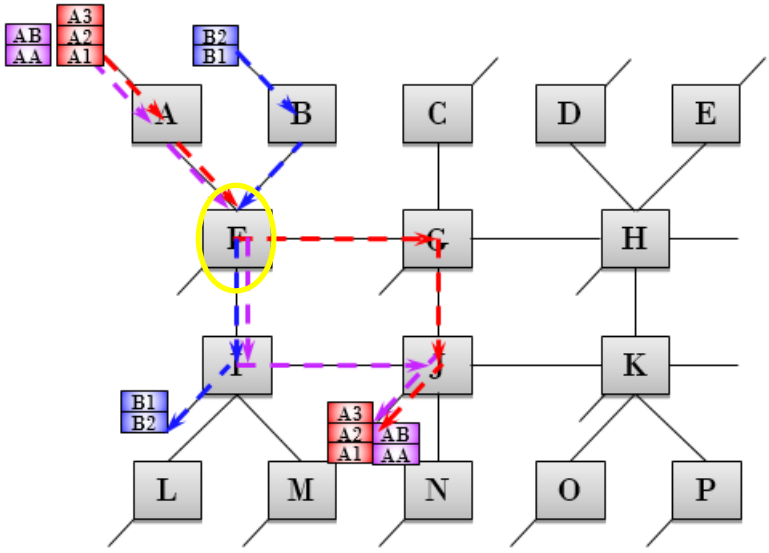
Durant cette partie de simulation, nous nous intéressons à un autre type de topologie différent des topologies maillées. Les topologies en arbre sont parmi les topologies les plus fréquentes dans les architectures des réseaux sur puce. Dans une première phase, nous configurant la plateforme pour un NoC en arbre en quatre étages, contenant 16 routeurs ayant la configuration suivante :

- Commutation par paquet (Wormhole),

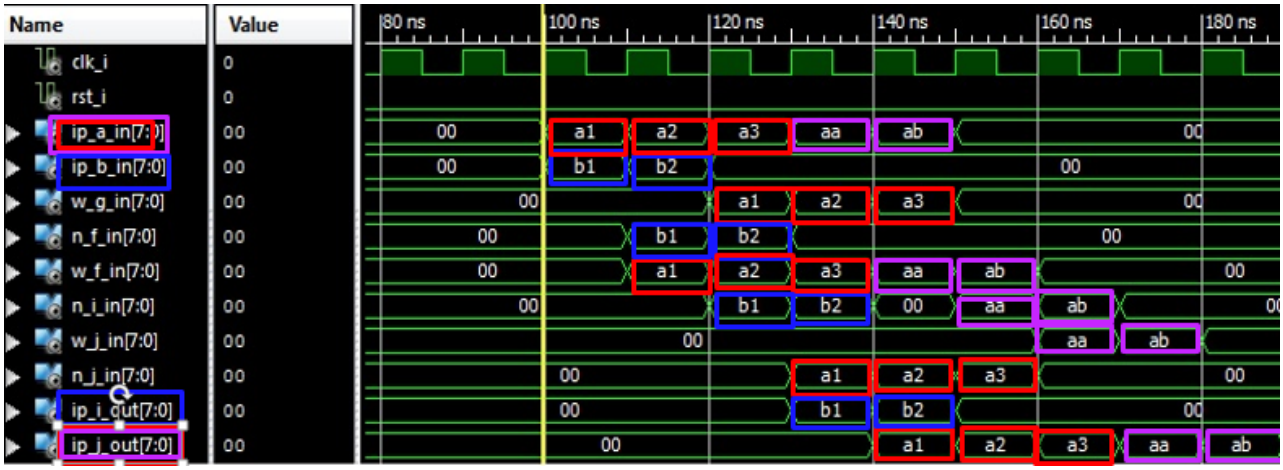
- Plusieurs paquets contenant des Flits de 8 bits de taille avec situation de congestion,

- Algorithme XY adaptatif, et un arbitrage par Round Robin (priorité à droite).

La figure ci-dessous nous résume la simulation du NoC durant le transfert des paquets.




(a)




(b)

**Figure 3.20.** Résultats de simulation d'un NoC Arbre 4 étages (16 Routeurs) utilisant un algorithme de routage XY adaptatif, commutation par paquet « Wormhole »

La figure 3.20 met en évidence les résultats de simulation du NoC en arbre à quatre étages contenant seize routeurs. L'algorithme de routage utilisé étant adaptatif, nous pouvons facilement distinguer la différence entre les chemins empruntés par les paquets « A1, A2, A3 » et « AA, AB ». Il est a noté aussi que l'ensemble des flits de chaque paquet se suivent dans leurs trajectoires respectives. L'explication réside dans le caractère dynamique du routage qui peut, pour une même source et destination, attribuer des chemins différents en fonction du trafic au sein du réseau. De plus, l'injection simultanée des paquets « A1, A2, A3 » et « B1, B2 » dans le réseau fait qu'une première situation de congestion apparait dans le routeur « F », où les paquets requièrent la même sortie. Afin de réduire la latence, les flits « A1, A2, A3 » sont déviés vers un autre itinéraire qui aura, dans le cas présent, une latence identique à l'itinéraire préférentiel. Une fois les flits « B1, B2 » sont transmis, le paquet suivant (c.-à-d. « AA,

AB ») peut être aiguillé vers le chemin préférentiel qui est désormais disponible. Ces exemples, qui sont certes assez basiques, nous renseignent sur les scénarii possibles lors des situations de congestion. Cependant, ils restent loin des cas de saturation du réseau, où le NoC peut être beaucoup plus chargé en données, ce qui provoquera des situations de congestion encore plus complexes. Dans ces cas-là, éviter les délais d'attente dans les buffers devient quasi-impossible, et l'objectif sera plutôt de réduire la latence moyenne des paquets et optimiser au maximum la bande passante moyenne du NoC.

### 9.5. Comparaison des performances : NoC Mesh 4x4 et NoC Arbre (4 étages – 16 Routeurs)

Pour une évaluation réelle de l'architecture en arbre précédente, nous comparons ses performances et son coût en ressources logiques avec la topologie 4x4 maillée semblable comportant le même nombre de routeur (c.à.d. 16 routeurs). Les deux réseaux ayant des configurations similaires en termes de : nombre de routeur, taille de paquets, largeur de Flits, arbitrage, mécanisme de commutation et algorithme de routage adaptatif, dès lors la consommation en ressources logique constitue un premier élément de comparaison pour l'évaluation des deux réseaux. Par ailleurs, la fréquence maximale de fonctionnement est aussi un critère de comparaison pertinent qui permet de jauger de manière objective les performances des différents réseaux d'interconnexion.

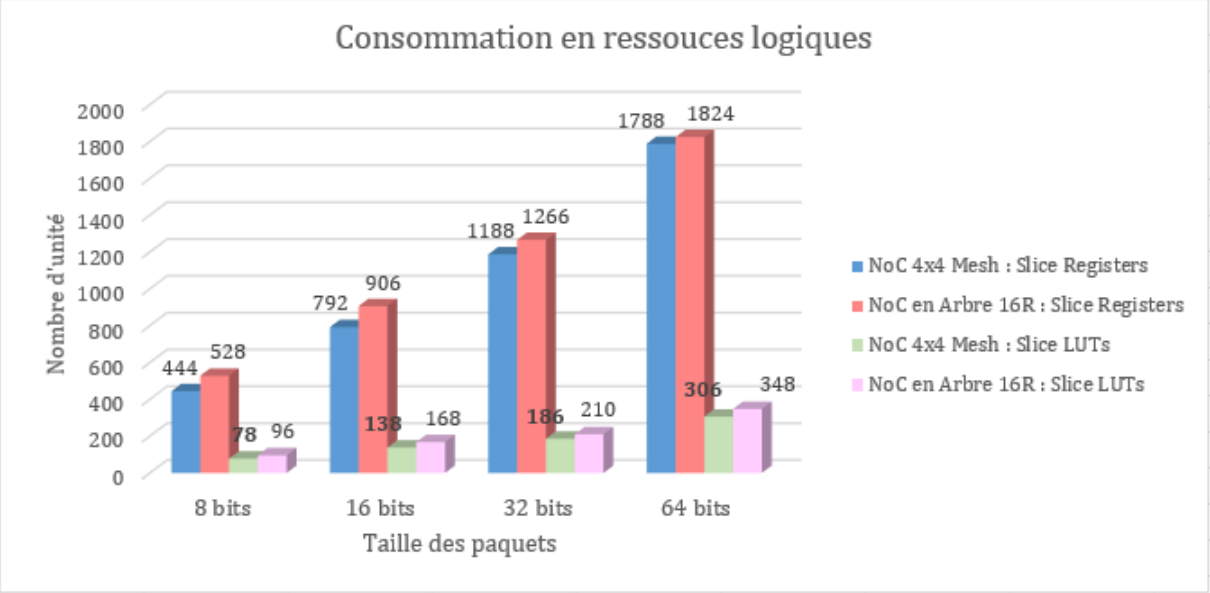



**Figure 3.21.** Consommation en Silce Registers, Slice LUTs pour différentes largeurs de paquets : NoC 4x4 Mesh, NoC en Arbre à 4 étages-16 Routeurs

Pour des données variant de 8 à 64 bits, il est transparent d'après la figure 3.21 qu'une topologie en Mesh est moins coûteuse que la topologie en arbre à quatre étages (16R) en « Slice Registers » de l'ordre de 8.5% à 32% (au meilleur des cas). Cette faible

consommation est tout autant visible en consommation en « Slice LUTs » qui oscille entre des pourcentages de 9% à 25%. Ces résultats sont tout à fait conformes aux caractéristiques de chacune des topologies en termes de complexité. L'architecture en Mesh (maillée) est réputée pour sa faible complexité et sa facilité de mise en œuvre, ce qui en fait une solution moins coûteuse en surface FPGA et une fréquence maximale légèrement plus élevée. Ce léger avantage en fréquence reste limité aux alentours des 3%, ce qui rend la performance de la topologie en arbre tout à fait respectable face aux topologies régulières maillées. Ces performances à l'avantage de la topologie maillée, confortent sa position en tant que solution la plus adoptée pour les réseaux sur puce.

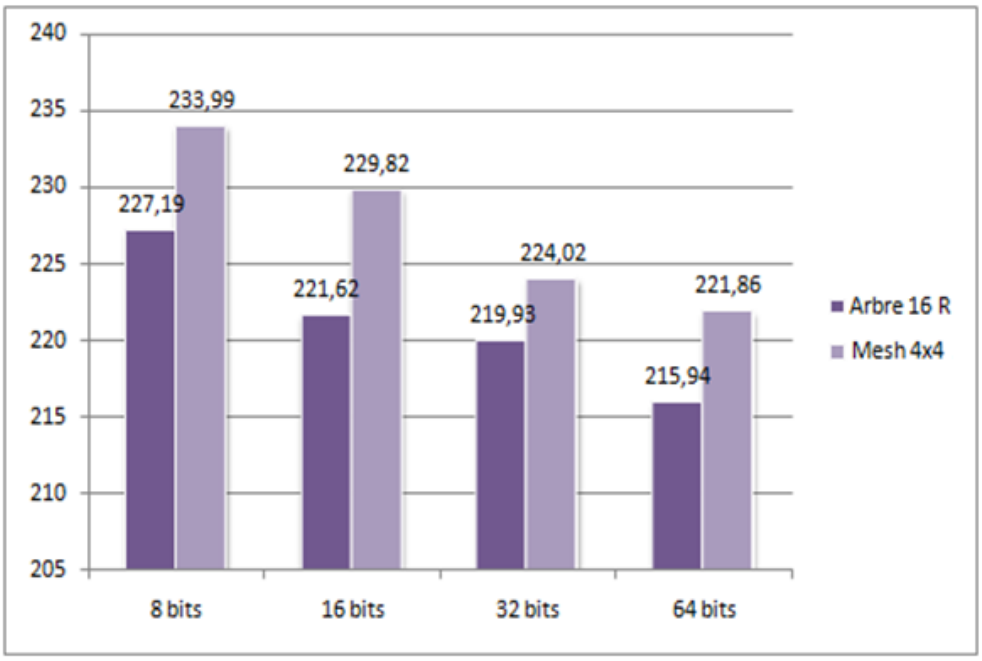



**Figure 3.22.** Fréquence de fonctionnement maximale pour différentes largeurs de paquets : NoC 4x4 Mesh, NoC en Arbre à 4 étages-16 Routeurs

## 9.6. Comparaison des performances de différentes architectures NoCs

- Taux d'injection de Flits : Le taux d'injection de paquets (**PIR** : Packet Injection Rate) correspond au nombre de paquets de données pouvant être envoyés en un cycle d'horloge. Par exemple, si un IP a un PIR de 0,5, cela signifie qu'il peut envoyer 50 paquets de données en 100 cycles d'horloge. En revanche, le taux d'injection de Flits (**FIR** : Flit Injection Rate) est la valeur du PIR multipliée par le nombre de flits dans chaque paquet de données. Nous avons estimé le $FIR_{max}$ en simulant différents NoCs: Mesh 4 × 4, Arbre à quatre étages 16 routeurs, Torus 4x4. Ces NoCs sont générés par la plateforme avec des algorithmes de routage adaptatif, une même taille des

paquets et des flits, un arbitrage en round robin et un même mécanisme de commutation (Store and Forward). Chaque NoC est connecté au nombre maximal de modules de communication de type Wishbone (c.à.d. 16 IPs) à travers les interfaces de type Wishbone présentés dans la section 8.

En outre, dans notre cas d'étude, le $FIR_{max}$ est obtenu lorsque le réseau fonctionne sans composant indisponible et que les modules Wishbone envoient et reçoivent des paquets de données uniquement au routeur voisin situé du côté opposé du réseau. Le $FIR_{max}$ varie en fonction du type du NoC. Pour un Mesh 4x4 il est estimé à 0,35 pour toute taille du réseau et toute largeur de données, il sera de 0,3 pour un NoC de type arbre et 0,4 pour un NoC de type Torus.

- La latence : La latence minimale ($latency_{min}$) pour traverser le réseau de la source à la destination est définie par l'équation (5.1). $N_{Rout}$ et $latency_{Rmin}$ sont respectivement le nombre de routeurs traversés et la latence minimale pour traverser un routeur. L'équation suivante prend en compte les cycles d'horloge supplémentaires du contrôle de flux de données utilisé pour nos routeurs. Cette technique de contrôle de flux de données nécessite deux cycles de latence par transmission de routeur :

$$\text{Latency}_{\text{min}} = N_{Rout} * \text{latency}_{\text{Rmin}} + N_{Rout} * 2 \qquad (5.1)$$

Dans les applications en temps réel, la latence des paquets de données dépend du trafic de réseau. Nous avons évalué la latence moyenne pour différents NoCs mais avec le même nombre de routeur dans chacun des réseaux. Chaque module envoie et reçoit des paquets de données. Les destinations des paquets de données sont générées de manière aléatoire. L'émission de paquets de données est effectuée au PIR maximum. La figure 3.23 donne les latences moyennes minimale et maximale pour chaque type de NoC en termes de cycles d'horloge en tenant compte de la fréquence de fonctionnement maximale obtenue avec la plateforme FPGA Virtex-7 XC7V2000T de Xilinx. Il en résulte que, pour un NoC 4x4 Mesh les latences moyennes minimale et maximale sont respectivement de 96,88 et 124,76 cycles d'horloge. Celles du NoC en arbre 16R sont logiquement supérieures et valent respectivement 131,48 et 169,79. La topologie en Torus étant la plus performante des trois, offre de meilleures latences et ont les valeurs respectives 59,02 et 70,84.

- La bande passante : la bande passante maximale d'un routeur dépend de la largeur du bus de données de n bits, de la fréquence de fonctionnement f et

de la FIR. La bande passante maximum est donnée par (2) en considérant le nombre maximum d'IP de type Wishbone connectées de 16 :

$$\text{Throughput}_{\max} = n * \text{FIR}_{\max} * f \tag{5.2}$$

La Figure 3.24 indique la bande passante maximale d'un routeur pour les différents NoCs simulés avec des largeurs de bus de données différentes. La bande passante maximale théorique d'un NoC est défini par la bande passante maximum d'un routeur multipliée par le nombre de routeurs au sein du NoC. Si nous considérons une taille de bus de données de 64 bits dans lequel un IP connecté par routeur et la fréquence de fonctionnement maximale du routeur dans la technologie FPGA Virtex-7 XC7V2000T de Xilinx, le débit maximal est de 30,58 Gb/s. La comparaison réalisée est en adéquation avec la littérature dans le sens où la topologie en tore est la plus performante des trois. Néanmoins une performance aussi élevée est obtenue au prix d'une consommation plus en ressources logiques plus importante. L'analyse des deux figures nous permet facilement de constater que la topologie maillée mesh est un bon compromis entre performance et coût en FPGA. De plus, celle en arbre 16R, présente une haute latence, une faible bande passante, le tout combiné à une consommation plus élevée en ressources logiques.

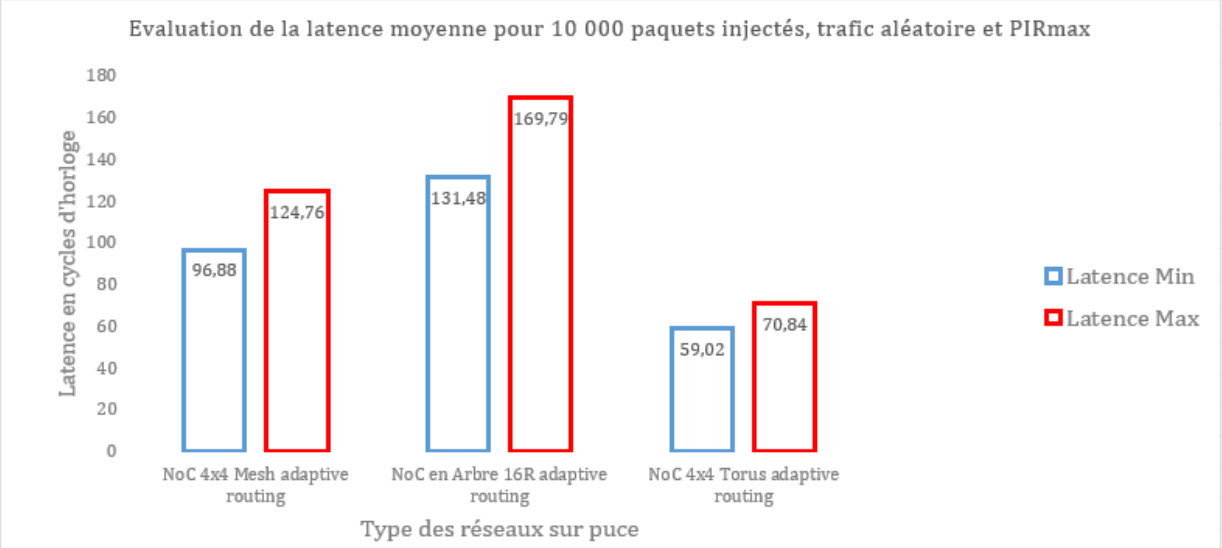



**Figure 3.23.** Evaluation des Latences minimales et maximales pour un total de 10.000 paquets injectés, trafic aléatoire et $\text{PIR}_{\max}$

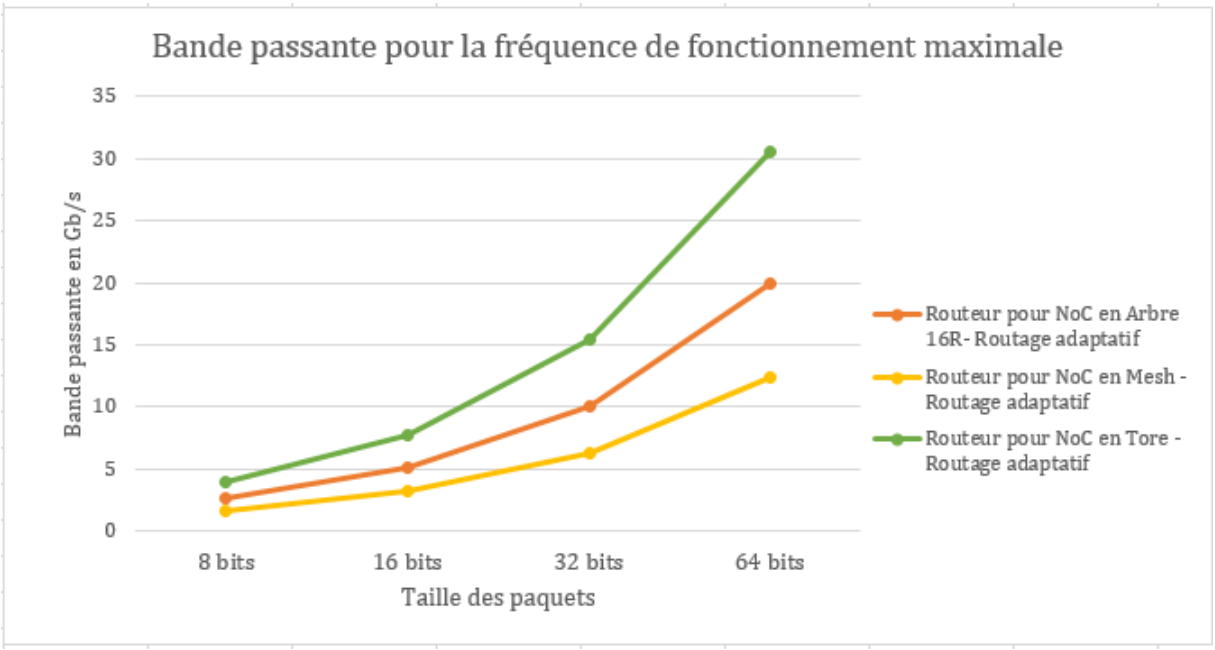



**Figure 3.24.** Bande passante d'un routeur pour les différents NoC configurés pour différentes largeurs de données

## 10. Conclusion

Dans ce chapitre nous avons proposé une plateforme pour la génération des réseaux sur puce pour les composants de type Wishbone. Cette plateforme prend en considération différentes configuration de NoC en termes de topologies, algorithmes de routage, mécanismes de commutation. Ce large choix nous offre différentes possibilités en matière de performance et / ou coût.

Dans une première partie, le protocole Wishbone est détaillé afin de poser les bases de notre travail. Le choix du protocole Wishbone est motivé par la possibilité de disposer de composants Wishbone et de pouvoir les connecter facilement (Plug-and-Play) gagnant ainsi un temps de développement considérable. L'adaptation du protocole Wishbone aux NoCs se fait à l'aide des interfaces Wishbone développés. Ces interfaces sont classées en deux catégories : interface pour composant Wishbone maître, et interface pour composant Wishbone esclave. Les deux assurent une parfaite adaptation des IPs de type Wishbone.

Dans la deuxième partie de ce chapitre nous nous sommes intéressés au flux de conception de réseaux sur puce à travers la plateforme pour la génération des réseaux sur puce. Cette plateforme offre différentes possibilités de configuration des réseaux sur puce. La plateforme est testée et les performances des NoCs générés sont évaluées. Une comparaison est faite entre certaines architectures générées par cette même plateforme

et les résultats sont discutés. Un tel outil permet de générer de le code VHDL de réseaux sur puce avec une grande flexibilité tant au niveau de la performance qu'au niveau de la souplesse. Les résultats de synthèse FPGA des NoCs générés par la plateforme présentent des performances attrayantes. Ils démontrent l'efficacité de cet outil pour l'interconnexion des systèmes sur puce à base de composants (IPs) de type Wishbone.

# Chapitre IV

# CHAPITRE IV

# Architecture d'un réseau sur puce sans buffer (Buffer-less NoC) pour les applications d'encodage vidéo : étude de cas H.264/AVC

## 1. Introduction

De jours en jours, le numérique est passée de l'utilisation professionnelle à l'informatique personnelle pour atteindre une omniprésence au quotidien au cours des dernières décennies. Cette omniprésence est presque imperceptible et pourtant partout autour de nous, elle se manifeste par la prolifération des systèmes embarqués dans notre quotidien. Un système embarqué est un système informatique matériel-logiciel, conçu pour exécuter des tâches spécifiques (contrairement à un système polyvalent) et qui est généralement intégré dans un système ou un dispositif plus important. Des exemples courants de systèmes embarqués comprennent les montres numériques, les contrôleurs de la circulation, les téléphones mobiles, les lecteurs de musique / vidéo, les tablettes, les moniteurs de santé et les voitures modernes. L'évolution des systèmes embarqués est d'une rapidité importante et leur marché connaît une croissance fulgurante, comme en témoigne les rapports publiés par l'International Data Corporation en 2011 [91] et 2014 [92], qui stipulent que 5,4 milliards de systèmes embarqués ont été livrés en 2010, 10 et 15 milliards sont attendus respectivement en 2015 et 2018. En outre, 7,5 milliards de processeurs embarqués ont été utilisés en 2010, 15 et plus de 20 milliards seront probablement nécessaires pour 2015 et 2018 respectivement.

Le marché des systèmes embarqués comprend un ensemble varié d'industries couvrant l'automobile, la communication, le multimédia, l'énergie, la santé, l'industrie et le transport etc. Les industries de la communication et de la consommation représentaient 48% du chiffre d'affaires du marché des systèmes embarqués en 2010 [91], traduisant les demandes et les attentes des utilisateurs sur les appareils grand public tels que téléphones portables, assistants numériques personnels, appareils photo numériques, consoles de jeux. Le domaine des multimédias est une combinaison de diverses formes de contenu telles que le texte, l'audio, la vidéo, l'image et l'animation pour fournir des informations ou du divertissement aux utilisateurs. Habituellement, les données vidéo et audio doivent être compressées avant le stockage ou la transmission car le volume de données d'origine est trop important et les données compressées doivent être décodées avant l'affichage ou pour un traitement ultérieur. La compression est également appelée codage ou encodage, et la décompression en tant que décodage. Par conséquent, le logiciel ou dispositif pour compresser et décompresser des données vidéo/audio est appelé codeur (encodeur) vidéo/audio et décodeur respectivement. L'encodeur et le décodeur sont abrégés en « **codec** » pour plus de commodité. Bien que de nombreux algorithmes de codage vidéo et audio aient été développés, ce sont les normes de codage vidéo et audio qui garantissent l'interopérabilité entre logiciels et matériels fournis par plusieurs fournisseurs, ce qui rend les communications multimédias pratiques. L'avènement de l'encodeur H.264/AVC en 2003 a permis un codage plus efficace et robuste d'une part et a accru les exigences en termes de performance, coût en ressources logiques, consommation énergétique. Son implémentation matérielle nécessite un travail minutieux de par sa complexité [93]. De plus, l'exigence croissante en fréquence de fonctionnement oblige les designers à optimiser l'interconnexion entre les différents blocs du système afin d'éviter qu'elle ne soit un goulot d'étranglement des performances [94].

L'émergence des réseaux sur puce comme solution d'interconnexion pour les applications des systèmes sur puce a apporté une évolutivité, flexibilité et réutilisabilité pour les systèmes sur puce. Inspirés des réseaux informatiques, ils offrent un bon compromis entre le coût et la performance. Dans ce chapitre nous présentons une solution d'interconnexion à base de NoC pour l'encodeur H.264/AVC afin d'optimiser l'exécution de l'application, réduire le coût logique et la consommation énergétique.

## 2. Vue d'ensemble de l'encodeur H.264/AVC

Un codeur vidéo prend en entrée une séquence vidéo, effectue une compression, et produit ensuite en sortie des données de train de bits qui peuvent être décodées vers

une séquence vidéo par un décodeur vidéo conforme à la norme. Un signal vidéo est une séquence d'images. Il a une fréquence d'images définie comme le nombre d'images par seconde (fps). Pour les applications typiques du consommateur, une fréquence de 30 fps est suffisante. Cependant, cette fréquence pourrait atteindre 60 voire 120 pour des applications très haut de gamme ou aussi peu que 10 ou 15 pour la vidéoconférence sur une liaison de communication à faible bande passante.

Une trame est une matrice à deux dimensions de pixels de couleur. Sa taille est appelée résolution d'image. Une trame de définition standard (SD) a 720 x 480 pixels par trame alors qu'une trame haute définition complète (Full-HD) en a 1920 x 1088. Il existe un grand nombre de variations de taille de trame développées par diverses applications telles que les écrans d'ordinateur et les écrans des téléviseurs.

Un pixel couleur est composé de trois composantes élémentaires : R, G et B. Chaque composant est numérisé en données 8 bits pour les applications grand public, en 12 ou 16 bits pour les applications haut de gamme. Le débit de données pour un signal vidéo brut est énorme. Par exemple, une séquence Full-HD à 30 fps aura un débit de données de 30 x 1920 x 1088 x 3 x 8 = 1,5Gbps, ce qui impraticable pour les systèmes de communication d'aujourd'hui. Heureusement, en tirant parti des caractéristiques du système visuel humain et de la redondance dans le signal vidéo, nous pouvons comprimer les données de deux ordres de grandeur sans scarifier la qualité de la vidéo décompressée.

Pour fournir une meilleure compression des images vidéo, la norme H.264/AVC est développée conjointement par le groupe d'experts des images animées (MPEG) et le groupe d'experts du codage vidéo de l'UIT-T (VCEG) et publiée en 2003. Comme l'ensemble des systèmes de codage traditionnels tel que MPEG-1 et MPEG-2, H.264/AVC est basé sur la compensation de mouvement et le codage inversé, comme le montre la Figure 4.1. Cependant, en H.264/AVC, plusieurs technologies de codage avancées ont été introduites, à savoir l'estimation de mouvement multiple, l'estimation inter-trame et l'estimation multi-trame [93].

Une séquence vidéo en H.264 est construite par plusieurs groupes d'images, dont chacune comprend plusieurs trames. Chaque trame comprend une ou plusieurs tranches qui sont construites par plusieurs macroblocs. Un macrobloc est une unité en forme de bloc (par exemple 16x16 = 256 pixels) qui nécessite 256 x 3 = 768 octets pour la représenter au format RGB et 256x(1+1/4+1/4) = 384 octets pour une représentation au format 4 :2 :0. Si l'on peut trouver lors du décodage un macrobloc M' similaire à M, il suffit d'obtenir la différence entre M et M' par le codage. Si M et M' sont très

similaires, la différence devient très faible, de même que la quantité de données à transmettre/stocker. Une autre façon d'interpréter la similarité est la redondance (spatiale ou temporelle). La redondance spatiale résulte de la similarité entre un pixel (région) et ses pixels voisins dans une trame. La redondance temporelle résulte d'une modification lente du contenu vidéo d'une image à la suivante. Les informations de redondance peuvent être identifiées et supprimées à l'aide d'outils de prédiction.

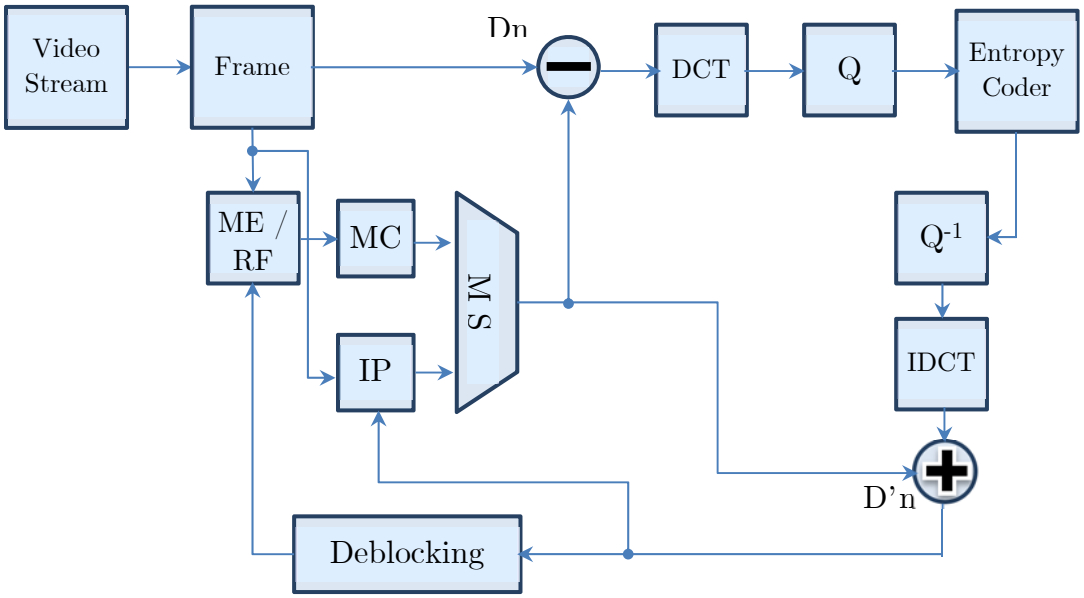



**Figure 4.1.** Schéma fonctionnel du codeur/décodeur H.264

Dans une région d'image avec un changement régulier, un macrobloc est susceptible d'être similaire à ses macroblocs voisins en couleur ou en texture. Par exemple, si tous ses voisins sont en rouge, nous pouvons prédire qu'un macrobloc est également rouge. Généralement, nous pouvons définir plusieurs fonctions de prédiction ; chacune prend les valeurs de pixel des macroblocs voisins comme son entrée et produit un macrobloc prédit comme sa sortie. Pour effectuer une prédiction intra-trame, chaque fonction est évaluée et celle qui résulte de la plus petite erreur est choisie. Seul le type de fonction et l'erreur doivent être codés et stockés/transmis. Cet outil est également appelé intra prédiction et une fonction de prédiction est également appelée mode de prédiction [94].

L'ensemble du processus de codage est illustré par la Figure 4.2. La prédiction inter-frame, également appelée inter-prédiction, identifie la redondance temporelle entre les trames voisines. Nous appelons la trame en cours de traitement la trame-actuelle (current-frame) et la voisine la trame de référence (reference-frame). Nous essayons de trouver dans la trame de référence un macrobloc de référence très similaire au

macrobloc actuel de l'image en cours. Le processus est appelé estimation du mouvement. L'estimation de mouvement vise à exploiter la similarité entre deux trames consécutives afin de réduire la redondance temporelle.

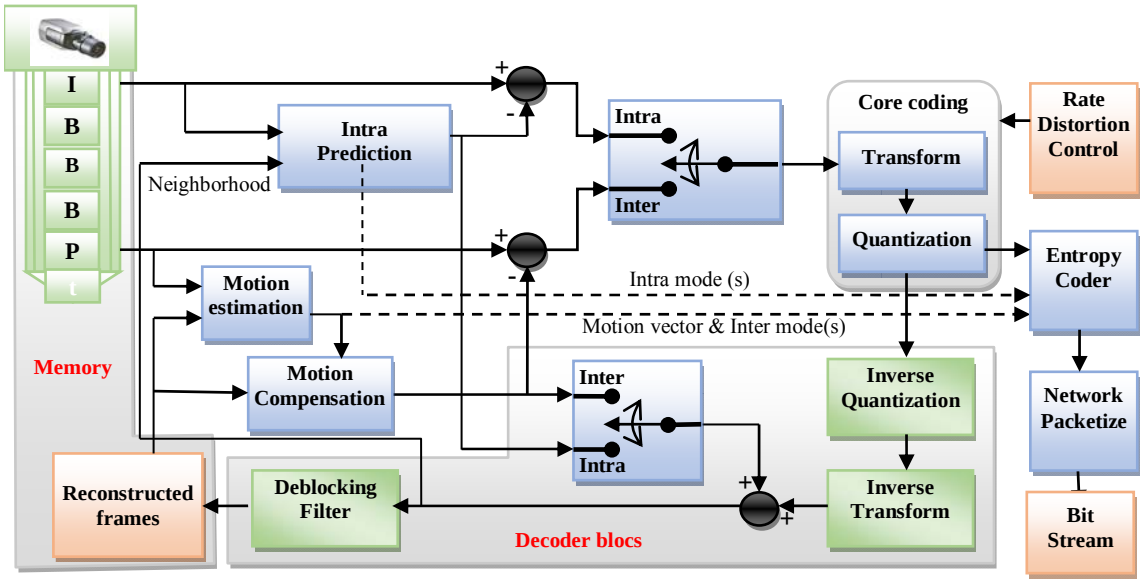



**Figure 4.2.** Architecture de base pour le codeur H.264/AVC [95]

Les normes de codage vidéo précédentes étaient basées sur l'estimation de mouvement à taille de bloc fixe (FBSME), qui avait l'inconvénient de dépenser le même effort pour estimer le mouvement d'objets statiques ou dynamiques. De plus, une telle approche donne de piètres performances dans le cas de deux objets se déplaçant dans deux directions différentes dans un bloc. D'autre part, le codeur H.264 / AVC est basé sur VBSME (Variable Block-Size Motion Estimation), ce qui permet d'obtenir une plus grande efficacité en utilisant une taille de bloc plus grande pour les objets statiques et une taille plus petite pour les mobiles d'entre eux. L'estimation de mouvement de taille de bloc variable de l'encodeur H.264/AVC est effectuée par : Estimation de mouvement d'entier (IME) et estimation de mouvement de trame (FME). Un estimateur de mouvement compare le macrobloc actuel avec des macroblocs candidats dans une fenêtre de recherche dans la trame de référence. Après avoir trouvé le macrobloc candidat le mieux adapté, seul le déplacement et l'erreur doivent être codés et stockés/transmis. Le déplacement de l'emplacement du macrobloc actuel vers celui du meilleur bloc candidat est appelé vecteur de mouvement (MV : Motion Vector). En d'autres termes, l'estimation de mouvement détermine le MV qui entraîne la plus petite erreur d'inter-prédiction. Une fenêtre de recherche plus grande donnera une meilleure prédiction au détriment d'un temps d'estimation plus long.

Un vecteur de mouvement, obtenu à partir de l'estimation de mouvement, est adéquat pour récupérer un bloc de la trame de référence. Cependant, nous n'avons pas

besoin de l'encoder/transmettre en totalité car il existe une similarité (ou redondance) entre les MVs des blocs voisins. Au lieu de cela, nous pouvons avoir une prédiction de vecteur de mouvement (MVP : Motion Vector Prediction) en fonction des MVs des blocs voisins et juste traiter la différence, appelée différence de vecteur de mouvement (MVD : Motion Vector Difference), entre le MV et son MVP. Dans la plupart des cas, le MVD est beaucoup plus petit que son MV associé.

Nous appelons la différence entre le macrobloc actuel et le macrobloc estimé une erreur de prédiction. On l'appelle aussi erreur résiduelle. L'erreur résiduelle est propre au domaine spatial et peut être représentée dans le domaine fréquentiel en appliquant une transformation en cosinus discrète (DCT). La DCT peut être considérée comme : représenter un bloc d'image avec une somme pondérée de motifs élémentaires. Les poids sont appelés coefficients. Un macrobloc d'erreurs résiduelles est généralement divisé en blocs 4 x 4 ou 8 x 8 plus petits avant d'appliquer la DCT sur ces blocs un par un.

Les coefficients générés par la DCT contiennent des composantes d'image de diverses fréquences. Etant donné que le système visuel humain est plus sensible aux composantes basses fréquences et moins sensible aux hautes fréquences, nous pouvons les traiter avec une résolution différente au moyen de la quantification. La quantification rejette effectivement certains bits les moins significatifs (LSBs) d'un coefficient. En donnant de plus petites étapes de quantification aux composants basses fréquences et des étapes de quantification plus importantes aux hautes fréquences, nous pouvons réduire la quantité de données sans altérer la qualité visuelle.

Codage et de décodage doivent reconstruire la trame vidéo à la fin. Dans l'encodage, la trame reconstruite au lieu de l'originale doit être utilisée comme référence car aucune trame d'origine n'est disponible dans l'extrémité de décodage. Pour reconstruire, nous effectuons une quantification inverse et une DCT inverse pour obtenir des résidus reconstruits. Notez que le résidu reconstruit n'est pas identique au résidu d'origine car la quantification est irréversible, par conséquent, la distorsion est introduite ici. Nous ajoutons ensuite des données de prédiction au résidu reconstruit pour obtenir une image reconstruite. Pour un macrobloc intra-estimé, nous effectuons une fonction de prédiction sur ses macroblocs reconstruits voisins alors que pour un macrobloc inter-estimé, nous effectuons une compensation de mouvement. Les deux méthodes donnent une version reconstruite du macrobloc actuel.

Soit un vecteur de mouvement donné, la compensation de mouvement récupère un macrobloc reconstruit, de la trame de référence, pointé par la partie entière du MV. Si le MV a une partie fractionnaire, il effectue une interpolation sur l'image récupérée

pour obtenir l'image reconstruite finale. Habituellement, l'interpolation est effectuée deux fois, l'une pour la précision d'un demi-pixel et l'autre pour la précision d'un quart de pixel.

Après que chaque macrobloc d'une trame ait été reconstruit un par un, nous obtenons une trame reconstruite. Une fois que le processus de codage/décodage est effectué sur le macrobloc, il existe des artefacts de blocage entre les limites des macroblocs. Le filtre de déblocage (Deblocking Filter) est utilisé pour éliminer ce type de bords artificiels.

## 3. Etat de l'art sur les implémentations matérielles du codeur H.264/AVC

Le processus de conception des systèmes numériques a considérablement évolué avec l'émergence de nouvelles technologies telles que FPGA (Field Programmable Logic Array) pour prendre une place prépondérante dans la conception des systèmes embarqués. Ces technologies offrent des architectures matérielles aussi polyvalentes que les logiciels, ce qui constitue un bon compromis entre flexibilité d'utilisation et puissance de traitement. Un tel compromis est encore plus adapté aux systèmes embarqués destinés à différentes applications dans différents domaines. Dans ce contexte, la technologie FPGA permettra de concevoir et de mettre à niveau des systèmes informatiques en temps réel, tout en optimisant les ressources matérielles. Parallèlement, le progrès technologique continue d'augmenter la densité d'intégration des transistors dans les circuits intégrés (IC) en raison de la complexité croissante des applications et des fonctionnalités. Cela nécessite le développement de systèmes embarqués haute performance capables de répondre aux besoins informatiques récents [96]. De plus, le désir de réduire le temps de mise sur le marché met plus de pression sur les concepteurs pour obtenir de nouveaux produits compétitifs. Manifestement, les architectures à bus partagé ont été considérées pendant longtemps comme la solution la plus attrayante en raison de leur faible coût logique et de leur simplicité de mise en œuvre [97]. Cependant, cette solution n'a pas réussi à fournir des performances élevées avec un grand nombre d'IPs connectées. La bande passante d'un bus est partagée par tous les IPs connectés ce qui le rend insuffisant. De plus, pour connecter un grand nombre de cœurs IP, la longueur du bus devrait être importante ce qui implique une consommation électrique importante. En outre, pour une telle architecture, il est possible de faire une seule opération de lecture (ou d'écriture) à la fois. Toutes ces limitations ont conduit les concepteurs à proposer des alternatives. De ce fait, Network-on-Chip est apparu comme une solution prometteuse pour surmonter certaines

limitations d'architectures basées sur le bus [31] [33]. Ce concept résout les problèmes de complexité d'intégration en fournissant une solution de communication sur puce à la fois évolutive, réutilisable, robuste et à faible consommation avec une bande passante plus importante [98]. De même, le réseau sur puce est composé de liens réseau et de routeurs (R). Ces derniers sont connectés à des éléments de traitement (PE) via une interface réseau (NI : Network Interface). Chaque routeur a cinq ports pour les quatre directions (est, ouest, sud et nord) et le PE locale. Dans les NoCs traditionnels, chaque routeur est composé de Crossbar, d'un arbitre, d'une logique de calcul et de Buffers, ce qui conduit à des surconsommations de puissance et de surface de silicium et une latence significative.

Il parait évident que le besoin de systèmes de communication sur puce performants réside dans la complexité des applications de traitement du signal telles que les standards de compression vidéo : MPEG 2, MPEG 4, H.264 et bien d'autres [99] [100]. Ces normes doivent répondre aux exigences des systèmes en temps réel et en très haute définition, ce qui les rend de plus en plus sophistiqués. H.264/Advanced Video Coding est une compression vidéo orientée bloc basée sur la compensation de mouvement utilisée pour les contenus vidéo à : l'enregistrement, la compression et la distribution. Jusqu'à présent, plusieurs solutions de mise en œuvre matérielle H.264/AVC ont été proposées dans le but d'en optimiser le fonctionnement [101]. La Figure 4.3 illustre l'architecture de la chaîne de codage intra proposée par [101]. Certains travaux [101-107] se sont focalisés sur différents objectifs tels que : la consommation d'énergie, la fréquence maximale de fonctionnement, le pipelining de données, le parallélisme, ... etc. en proposant un intra-codage et blocs inter-codes. Ces implémentations utilisent souvent une interconnexion en cascade, ce qui est un réel inconvénient pour le parallélisme des données, car certains modules doivent être dupliqués. D'une part, cette duplication permet d'exécuter l'encodeur en mode pipeline ce qui a pour résultat une meilleure fréquence de fonctionnement. D'autre part, elle constitue un point faible dans la mesure où ces blocs dupliqués sont redondant. Par conséquent, elle engendre un coût supplémentaire en ressources logiques et énergétiques. Certains auteurs ont exploité des points de similarité entre des blocs redondants pour réduire le coût du FPGA en connectant les IPs de l'encodeur H.264/AVC via un réseau sur puce [108]. Plus précisément, un réseau sur puce permet d'enlever ces blocs redondants dans les deux chaînes et se contente d'utiliser un seul bloc en faisant un aiguillage intelligent du flux de données pour à la fois utiliser les ressources de manière efficace et réduire considérablement les ressources logiques nécessaires à l'exécution de l'application.

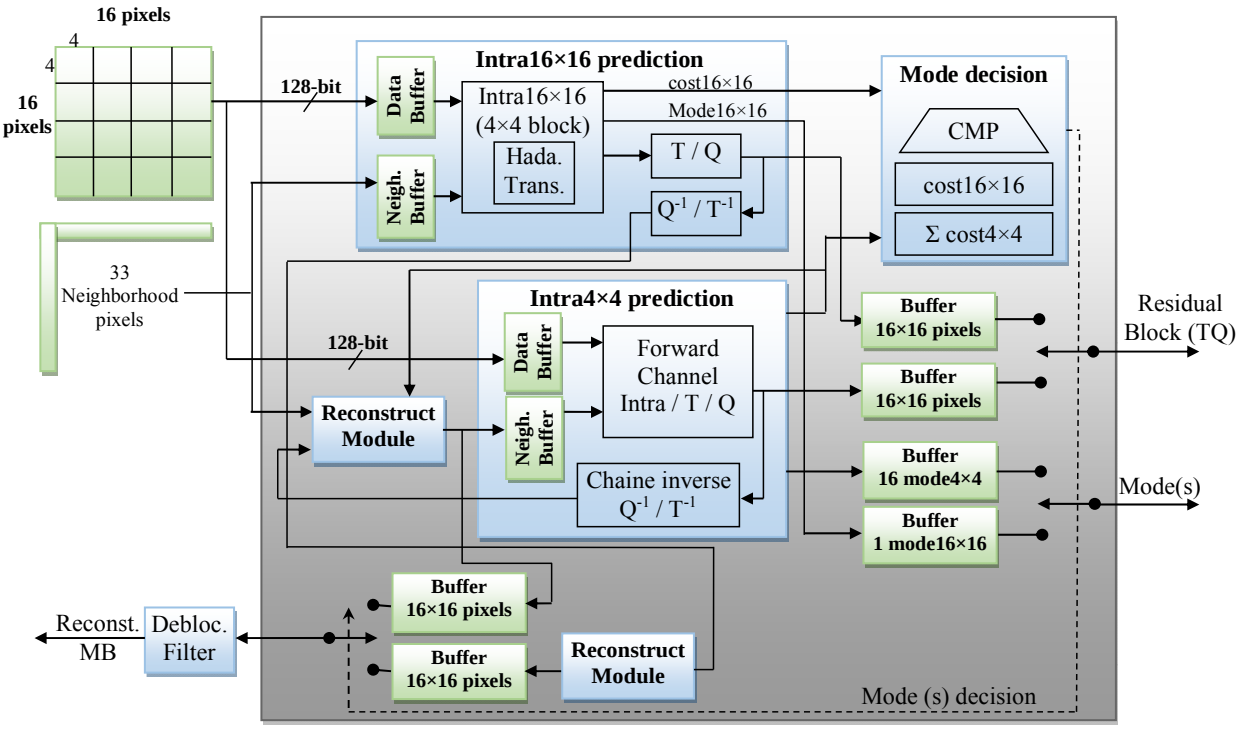



**Figure 4.3.** Architecture matérielle proposée par [101] pour le module de codage Intra. [101]

Bien qu'une telle solution se soit révélée plus rapide avec un coût équivalent, il reste une marge d'optimisation en termes de ressources logiques. De plus, l'utilisation d'un réseau sur puce pour l'interconnexion des modules H.264/AVC n'est pas aussi efficace qu'elle peut sembler l'être, car les systèmes de codage vidéo nécessitent une interconnexion sur puce qui garantit la qualité de service (QoS) et assure un système de communication bien adapté pour les applications vidéo.

En pratique, l'évaluation des performances du système basé sur NoC est souvent connue après la mise en œuvre du matériel car elle montre à quel point la communication peut être prévisible en fournissant des garanties de service. Par exemple, les applications vidéo fonctionnant en temps réel ont besoin de plus de garanties de performances plutôt que d'une prédiction sur le comportement de l'interconnexion sur puce. Par conséquent, le type de réseaux utilisés dans de telles applications est de type service garanti (GS NoCs). Pour d'autres applications, n'ayant pas de contraintes de sur la latence et la bande passante et que seul l'accomplissement de la communication est nécessaire, un réseau de type meilleur effort (BE NoC) peut satisfaire les exigences de l'interconnexion du système [32].

Dans ce chapitre, nous proposons une nouvelle solution basée sur l'architecture NoC pour les applications vidéo offrant des garanties de bande passante aussi élevées que la

connexion point-à-point, avec la flexibilité et l'évolutivité d'un réseau sur puce. La principale originalité de notre travail est d'adapter un réseau sur puce à travers un mécanisme de commutation de circuit afin de réduire la latence. En outre, cette approche vise à assurer une fréquence de fonctionnement élevée, à optimiser l'utilisation des FPGAs et à réduire la consommation d'énergie en supprimant les buffers du routeur.

## 4. Proposition d'architecture d'un routeur sans buffer pour l'encodeur H.264/AVC

L'évolution des applications vidéo a donné naissance à de nouveaux systèmes temps réel (RTS : Real Time Systems), plus complexes mais aussi avec plus d'exigences qui ont poussé les chercheurs à proposer des solutions basées sur des implémentations matérielles partielles ou totales du codec H.264 / AVC. Au cours des dernières années, plusieurs implémentations matérielles (IP matérielles) ont été proposées pour les modules H.264/AVC [101] [109-112]. Différentes stratégies de communication ont été utilisées pour implémenter ces IPs sur des dispositifs FPGA afin d'assurer un traitement en temps réel. Généralement, la solution point à point a longtemps été l'approche d'interconnexion la plus adoptée [101] [113]. Les bus partagés sont également utilisés pour connecter des IPs [114] avec un processeur embarqué. Le processeur est généralement utilisé pour la gestion des tâches.

Dans le codeur H.264, nous notons que les IPs élémentaires sont organisées en groupes distincts. Les IPs dans le même groupe communiquent souvent entre eux mais ne communiquent pas avec les IPs des autres groupes, à l'exception d'un bloc IP unique. En ce sens, un NoC peut être utilisé pour les IPs de chaîne intra-codage et un autre NoC pour les IPs de filtrage. Dans certaines études récentes, plusieurs auteurs ont démontré l'utilité de l'utilisation de la structure NoC dans de nombreuses applications [31] [115] [116]. Traditionnellement, les réseaux sur puce ont souvent été conçus en utilisant des architectures de routeurs contenant des buffers afin de stocker les données circulant sur le NoC, en particulier lorsque ces architectures ont été améliorées par l'avènement du concept de canal virtuel. Ce concept a permis une meilleure gestion du goulot d'étranglement tout en surmontant les problèmes de congestion. En fait, l'efficacité de la bande passante du NoC est améliorée par ces buffers car ils minimisent le taux de paquets perdus ou écrasés dans le réseau.

Dans [101], les auteurs ont proposé une architecture matérielle pour le module d'intra-codage utilisé dans l'encodeur H.264. En utilisant un bus de données (taille de bus 32 bits), des architectures matérielles basées sur le parallélisme de données sont

proposées pour l'intra-prédiction, la transformée entière, la quantification, la transformée inverse, la quantification inverse et les modules du mode de décision. Dans [101], les auteurs ont reproduit ces architectures en utilisant une taille de bus de 128 bits afin d'assurer une bonne gestion de la mémoire. Comme mentionné dans la Figure 4.3 [101], le module intra-prédiction est décomposé en deux modules (l'Intra4 × 4 et l'Intra16 × 16) en utilisant à la fois la même chaîne de modules (Integer Transform, quantization, Inverse quantization). Dans [101], les auteurs utilisent une interconnexion point à point avec des multiplexeurs / démultiplexeurs dans des cas multi-entrées/sorties. Certaines adresses IP (T / Q / Q-1 et T-1) sont utilisées plusieurs fois pour assurer un traitement rapide et éviter l'utilisation de multiplexeurs / démultiplexeurs. La duplication des modules IP H264 / AVC augmente considérablement les ressources FPGA utilisées. Pour surmonter les limitations du bus partagé et bloquer la duplication, une autre approche exploitant les réseaux sur puce a été mise en place [117]. L'idée derrière cette solution est de tirer le meilleur parti du principe de réutilisation des IPs. Cette approche permet d'exploiter les similarités entre la chaîne intra et la chaîne inter en supprimant les blocs redondants afin de réduire les ressources FPGA.

Ainsi, cette solution n'est pas très efficace et l'optimisation des ressources n'est pas optimale car le NoC utilisé dans ce travail est basé sur des buffers pour le stockage du flux de données. Plus précisément, l'utilisation de buffers dans le routeur augmente le coût logique et diminue la fréquence de fonctionnement, ce qui constitue l'inconvénient majeur de cette approche.

Conceptuellement, la suppression des buffers peut sembler un choix audacieux car nous prévoyons une latence plus élevée avec une bande passante plus faible. Cependant, ces buffers présentent également des inconvénients importants en termes de coût des ressources FPGA, de consommation d'énergie statique et dynamique et de complexité de conception. En conséquence, certains auteurs ont tenté de réduire leur impact en travaillant sur le concept de routeur à bas coût [96] [118-119]. D'autres travaux exploraient comment supprimer totalement les buffers dans les architectures de routeur [120-122]. De plus, il a été démontré qu'une réduction de 60% des buffers peut réduire de près de 40% la consommation d'énergie dans les meilleurs cas sans altération significative des performances du système et l'élimination des buffers entraîne une réduction considérable de la latence [120]. Pour ces raisons, nous avons choisi d'adapter cette approche à l'encodeur vidéo (étude de cas : H.264 / AVC) afin de fournir un système d'interconnexion sur puce plus efficace. La Figure 4.4 montre l'architecture du réseau sur puce proposée basée sur un routeur sans tampon (Buffer-Less Router).

Comme présenté dans la Figure 4.4, nous avons fait le choix d'une architecture 2D Mesh pour la solution d'interconnexion basée sur NoC. Une telle architecture se caractérise par sa régularité et sa facilité de mise en œuvre. Le routeur sans buffer (BLR) proposé constitue l'élément de base du réseau (Figure 4.4). Il est simplement composé d'un Crossbar et d'une table de commutation et d'une logique de déviation. Le rôle de la table de commutation est d'attribuer le port de sortie de chaque paquet en fonction de son étiquette. La particularité de l'algorithme de routage par déflexion pour routeur sans buffer proposé est l'utilisation d'une technique de commutation au lieu d'utiliser un algorithme de routage XY conventionnel. Il est basé sur la déviation et sa politique de déviation basée sur les priorités à plusieurs niveau afin d'éviter les blocages et les congestions.

Plus précisément, le mécanisme de commutation repose sur des étiquettes ajoutées dans le paquet d'en-tête. Cette étiquette est utilisée par chaque routeur sans buffer (BLR) pour envoyer les paquets à leur adresse de destination.

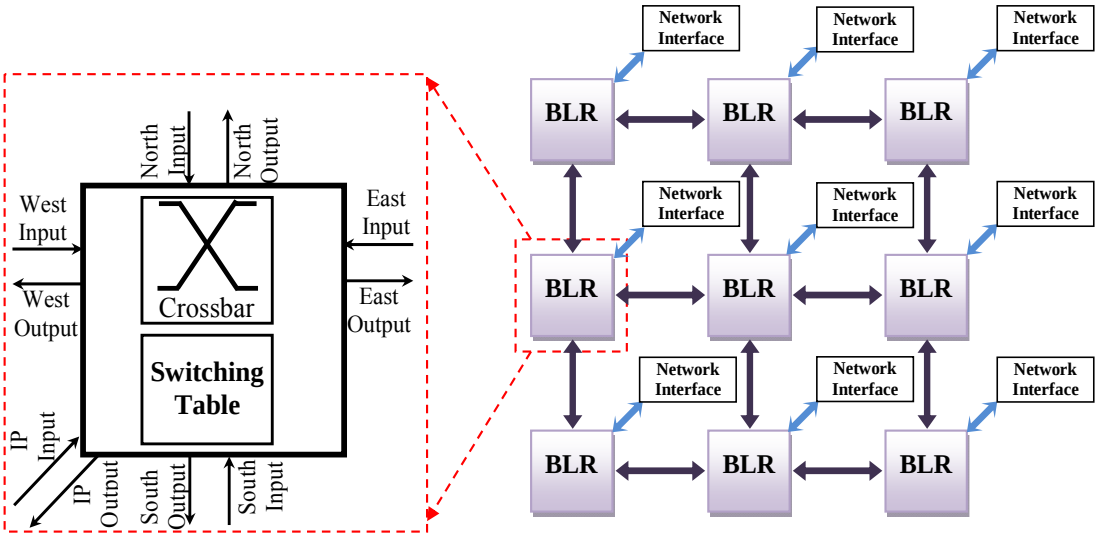



**Figure 4.4.** Architecture propose du réseau sur puce sans buffer

## 4.1. Algorithme de routage utilisé

Les routeurs dans les réseaux sur puce sans buffers (Buffer-less NoCs) ne possèdent pas de mémoire (buffer) pour stocker les paquets. Ces buffers de mémorisation sont généralement implémentés en tant que files d'attente de type FiFo, utilisées pour stocker les paquets en attente de transfert vers le port de sortie. Le routage par déflexion a attiré une attention significative en tant que méthode viable pour résoudre les conflits dans les réseaux sans buffer. Les déviations multiples peuvent améliorer de manière significative l'utilisation dans les réseaux comportant plusieurs chemins possibles.

Un conflit se produit lorsque, selon une table de routage, plusieurs flux de trafic arrivant sur un routeur doivent être acheminés via le même port de sortie. Dans ce cas, un seul flux est acheminé via le lien optimal défini par la table de routage. En l'absence d'un schéma de résolution de conflits, les flux restants sont ignorés car le nœud ne possède pas de buffers. Au lieu de mettre des buffers de sauvegarde ou de supprimer les paquets, le routage par déviation permet de les détourner temporairement du chemin prescrit par la table de routage.

La figure 4.5 résume le flux de la méthode de routage par déflexion pour un routeur à cinq ports. L'algorithme utilisé vérifie d'abord l'en-tête du paquet entrant pour obtenir les coordonnées de destination. Pour respecter la règle de base qui fournit la destination de sortie des paquets, le sous-ensemble des connexions possibles est utilisé pour construire un routage valide. Les informations de priorité sont également incluses dans le paquet en entrée. La latence accumulée peut être augmentée, car les priorités des paquets sont efficacement attribuées. Par ailleurs, les coordonnées des paquets sont vérifiées pour effectuer le routage pour une transmission efficace au sein du NoC.

Après avoir vérifié les informations d'en-tête des paquets entrants, une table de commutation doit être générée pour répertorier les options de routage de la direction de préférence, de la direction de déflexion et de la priorité. Dans la table de commutation, la direction de déflexion est fixe et correspond à la direction d'entrée. Il est important de noter que la table de commutation détermine les ports de sortie pour tous les paquets entrants.

Le modèle d'E/S, qui prend la décision de routage de paquets, est mis en place selon la table de commutation. Cinq directions différentes pour le paquet entrant au routeur actuel et sortant vers le routeur suivant. Afin d'atteindre les objectifs de faible puissance et de réduction de consommation logique, l'arithmétique simple sans multiplication et division est utilisée. Cependant, le routeur conserve des performances élevées de communication.

L'en-tête de chaque paquet comporte quatre champs : des bits valides, des bits de priorité, des coordonnées actuelles et des coordonnées de destination. La méthode proposée utilise ces champs pour construire le chemin de données. Le bit valide est utilisé pour classer le paquet entrant est valide ou défectueux. Lorsque le paquet provient de l'autre routeur valide, le bit valide est mis à « 1 », sinon il est mis à « 0 ». Cependant, lorsque les paquets arrivent au routeur cible, le bit valide est également mis à « 0 ». Lorsque les paquets arrivent dans le routeur, tous les bits valides sont

vérifiés si les paquets sont valides ou non. Tous les paquets du module sont affectés avec des priorités différentes en fonction de la latence de la transmission. Selon le format standard de l'en-tête, le paquet avec le nombre de sauts le plus élevé doit être routé en premier. Par conséquent, la méthode proposée compare tous les sauts pour déterminer la priorité de chaque paquet entrant. Pour un routeur disposant de cinq ports, la priorité la plus faible est le numéro « 1 » et la priorité la plus élevée est le numéro « 5 ».

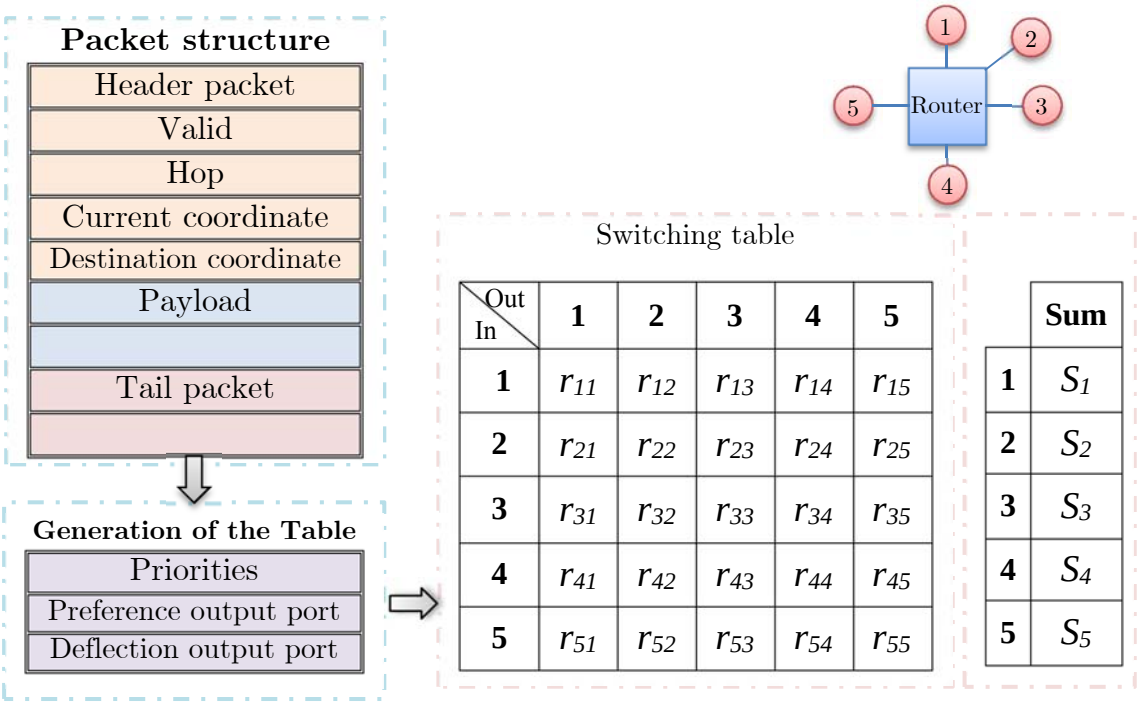



**Figure 4.5.** Algorithme de routage par déflexion pour Bufferless NoC

## 4.2. Table de commutation

Les priorités sont attribuées aux paquets à partir de chaque direction d'entrée. L'affectation est basée sur la comparaison du champ comportant le nombre de sauts dans l'en-tête du paquet. Afin de déterminer si le paquet sera acheminé vers la direction de préférence ou vers la direction de déflexion, la distance entre la coordonnée courante et la coordonnée de destination doit être analysée. Soit ($x_a$, $y_a$) les coordonnées actuelles et ($x_d$, $y_d$) la coordonnée de destination. La direction « Dir » peut être calculé par la fonction (4.1).

$$Dir \begin{cases} x_{Dir} = x_d - x_a \\ y_{Dir} = y_d - y_a \end{cases} \quad (4.1)$$

Pour chaque paquet entrant, une fonction de signe est utilisée pour mesurer la position relative. La fonction de signe de $x_{DisC}$ et $y_{DisC}$ peut être définie comme suit :

$$sign(x_{Dir}) = \begin{cases} -1, & if\ x_{Dir} < 0 \\ 0, & if\ x_{Dir} = 0 \\ 1, & if\ x_{Dir} > 0 \end{cases} \quad (4.2)$$

$$sign(y_{Dir}) = \begin{cases} -1, & if\ y_{Dir} < 0 \\ 0, & if\ y_{Dir} = 0 \\ 1, & if\ y_{Dir} > 0 \end{cases} \quad (4.3)$$

Les coordonnées *Dir* sont directement considérées comme l'entrée de la fonction « signe ».

Soit *i* la direction d'entrée et *j* la direction de sortie. Les paquets entrants de chaque sens d'entrée sont affectés de quatre choix de sortie, et des valeurs de différents coûts sont attribuées à chaque choix. Une fois que la matrice de configuration E/S a été remplie avec les valeurs du coût, la somme peut être facilement calculée en fonction de la direction de sortie. La fonction de sommation peut alors être exprimée comme suit :

$$S_j = \sum_{k=1}^{5} r_{ik} \quad (4.4)$$

où *k* représente différentes valeurs des directions de sortie *j*.

La fonction « coût de routage » *c* peut être définie comme suit :

$$c_{ij} = \begin{cases} 0, & si\ \ i \neq j\ \&\ d_{ij} = vraie, \\ 1, & si\ \ i \neq j\ \&\ d_{ij} = faux, \\ 2, & si\ \ i = j. \end{cases} \quad (4.5)$$

avec *dij* une indication sur la distance la plus proche de la localisation actuelle à la destination. En d'autres termes, lorsque la distance restante jusqu'à la destination via le port de sortie *j* est plus courte que celle vers le port d'entrée *i*, *dij* prend la valeur « vrai » ; sinon, *dij* prend la valeur « faux ». Il convient de noter aussi que plus la valeur de « *cij* » est grande, plus la pénalité est élevée, c'est-à-dire que la décision de routage est pire. A partir de la fonction « coût de routage », les entrées de la matrice de routage peuvent être calculées en utilisant la valeur pondérée :

$$r_{ij} = w_i . c_{ij} \quad (4.6)$$

où $w_i$ est la valeur pondérée du port d'entrée *i*. La valeur pondérée représente les priorités du paquet. Pour une architecture en Mesh, la priorité la plus basse du paquet $w_i = 1$, la plus haute $w_i = 4$ et $w_i = 0$ lorsqu'il n'existe aucun paquet valide.

La valeur de chaque cas est calculée par trois sommes. Les coûts de routage permettent de prendre la décision en fonction du résultat minimum de sommation.

Après avoir calculé la somme de la configuration d'E / S, le port de sortie du paquet est choisi afin qu'il puisse se rapprocher de la destination. Il effectue les étapes suivantes :

1. La sélection du port de sortie peut être choisie en fonction de la priorité du paquet. Lorsque tous les ports de sortie préférés sont occupés, l'itinéraire de déviation sera le candidat.
2. Lorsque tous les chemins de routage de déviation sont occupés, le chemin de routage par réflexion peut ainsi être choisi comme direction de sortie.
3. Lorsqu'il existe des paquets restants qui ne sont pas routés, les paquets doivent attendre le cycle suivant et partir de la première étape jusqu'à ce que tous les paquets soient routés.

Le port de sortie du paquet entrant est déterminé par sa priorité. Lorsque le paquet entrant a la priorité la plus élevée, il peut d'abord choisir le port de sortie. Après la sélection de certains ports de sortie, la valeur correspondante de la sommation doit être marquée pour éviter la sélection en double.

### 4.3. Connexion des modules H.264/AVC via le buffer-less-NoC

Étant donné que l'interconnexion sur puce peut constituer un goulot d'étranglement pour les performances du système, notre travail s'est concentré sur la solution d'interconnexion sur puce pour application d'encodage vidéo, codeur H.264/AVC dans notre cas. L'originalité de notre proposition est de tirer parti des atouts de Networks-on-Chip tout en supprimant l'inconvénient posé par les buffers. Ainsi, nous avons considéré H.264/AVC comme un système sur lequel nous n'avons pas agi, et nous avons simplement proposé une amélioration de l'interconnexion entre ses différents blocs.

Notre architecture proposée permet d'interconnecter les modules H.264 / AVC via une interface réseau comme le montre la Figure 4.6. Prenons comme exemple le cas de l'ensemble de la chaîne de codage intra. Premièrement, un microprocesseur effectue la configuration initiale du réseau en chargeant les tables de commutation de chaque routeur selon les différents scénarii. Deuxièmement, les paquets entrent dans le réseau destiné au bloc intra, à travers l'interface réseau (NI) qui attribue une première étiquette. Une fois le traitement effectué, les données traitées sont renvoyées avec une nouvelle étiquette, ce qui permettra l'accès à la DCT et au bloc de quantification. Ce changement d'étiquette est géré par l'interface réseau et le routeur connecté vérifie sa table de commutation et sélectionne le port de sortie correspondant.

Comme le montre la Figure 4.6, l'utilisation d'un NoC facilitera la connexion des IPs avec les avantages suivants :

- Utilisation des modules de traitement communs pour les deux chaînes de codage (inter et intra),
- Passage d'un niveau H.264/AVC à un autre sans changer la partie matérielle. Il permet même le passage d'un encodeur à un autre (transcodeur vidéo),
- Réduire la consommation d'énergie d'abord par la suppression des buffers puis par la réduction des ressources logiques utilisées ;
- Reconfigurabilité du matériel pour le traitement des données en temps réel,

D'un point de vue des perspectives, l'utilisation d'une solution (Buffer-les-NoC) pourrait s'avérer intéressante dans le cas de la reconfiguration dynamique au vue de l'évolutivité qu'offrent les réseaux sur puce.

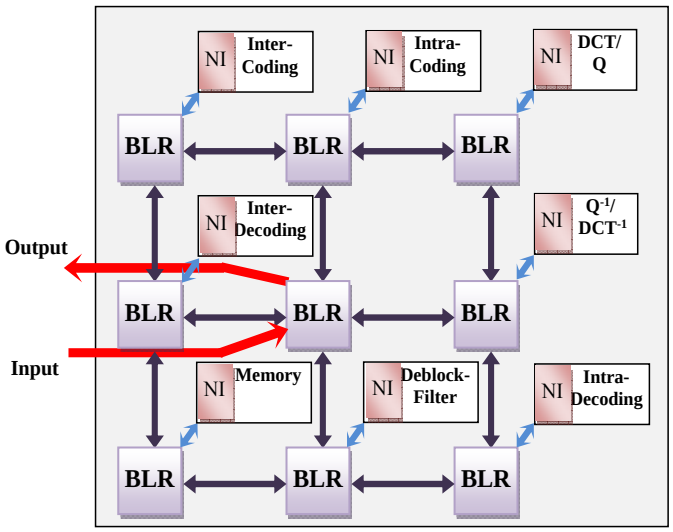



**Figure 4.6.** Mappage de la chaîne de codage intra-H264/AVC via le buffer-less NoC proposé

## 5. Résultats de la simulation et discussion

La description matérielle des composants H264/AVC et du réseau sans puce dédié aux applications vidéo ont été réalisés dans le langage de description du matériel (VHDL). Nous utilisons l'outil logiciel ISE 14.1 Design Suite de Xilinx pour : synthétiser les architectures, effectuer l'analyse temporelle et visualiser les schémas RTL. L'environnement de simulation multi-langage HDL ModelSim 9.2 a été utilisé pour la simulation et le débogage. La conception de l'ensemble de l'architecture des éléments H264/AVC interconnectés par le buffer-less NoC proposé a été réalisée avec l'outil de développement intégré EDK Design tool. En somme, l'ensemble du système a été simulé sur la plateforme FPGA Virtex-7 XC7V2000T via le logiciel ISE Design Suite 14.1 de Xilinx.

L'évaluation de notre architecture proposée, à base de buffer-less NoC, mise en œuvre pour les applications vidéo est réalisée après la génération du rapport de synthèse pour la topologie 3x3 2D-mesh, la commutation par paquet et un algorithme de routage conventionnel. L'analyse des performances en termes de Slices LUTs, de Slices registres et de fréquence de fonctionnement maximale est résumée dans le Tableau 4.1 pour différentes dimensions de réseau.

**Tableau 4.1.** Résultats de synthèse de la sur la plateforme de prototypage FPGA Virtex-7 XC7V2000T et comparaison entre l'architecture proposée Réseau sur puce sans buffer (Buffer-less NoC) et les travaux de [117]

| Performance metrics / Dimension the of mesh NoC | Proposed Buffer-less NoC | | | Conventional NoC [31] | | |
|---|---|---|---|---|---|---|
| | Number of slice registers | Number of slice LUTs | Maximum Frequency | Number of slice registers | Number of slice LUTs | Maximum Frequency |
| **NoC2×2** | 78/2,443,200 | 285/1,221,600 | 429.70MHz | 67/2,443,200 | 972/1,221,600 | 174.932MHz |
| **NoC3×3** | 191/2,443,200 | 603/1,221,600 | 427.56MHz | 127/2,443,200 | 2,154/1,221,600 | 172.828MHz |
| **NoC4×4** | 296/2,443,200 | 1,127/1,221,600 | 422.08MHz | 240/2,443,200 | 3,645/1,221,600 | 171.725MHz |

En mettant l'accent sur ces résultats, il semble clair que la solution proposée est facile à mettre en œuvre par la topologie 2-D Mesh et le coût logique relativement faible. De plus, il ressort de ces résultats que cette solution est assez efficace notamment pour des applications hautes performances telles que le codage vidéo (ou décodage) à travers sa fréquence de fonctionnement assez élevée. Ainsi, on peut facilement constater qu'un bon équilibre entre le faible coût logique et la fréquence de fonctionnement est atteint. En outre, il est également transparent à partir de la comparaison réalisée avec les résultats de synthèse de [117] (Tab.1) que notre proposition a réduit la consommation des LUTs d'environ 70% pour les trois dimensions de NoC mises en œuvre. De même, notre solution Buffer-less NoC fonctionne à 429,70 MHz, 427,56 MHz et 422,08 MHz pour les dimensions NoC 2x2, 3x3 et 4x4 mesh, ce qui est une fréquence maximale beaucoup plus élevée par rapport aux fréquences atteintes par les NoCs avec buffer [117].

En effet, notre approche semble tout à fait supérieure puisqu'elle fonctionne à une fréquence d'horloge plus rapide, ce qui est principalement dû à la suppression des buffers. Ces derniers ont un coût logique important et ralentissent toute la fréquence du système de communication pendant l'écriture (ou la lecture ou la suppression) des données.

Afin d'évaluer plus objectivement notre Buffer-less NoC proposé, nous effectuons une comparaison avec des travaux antérieurs qui ont obtenu une architecture similaire

mais avec un réseau sur puce conventionnel (NoC avec buffer) [117]. Pour ce faire, nous avons veillé à simuler la même chaîne de codage intra en utilisant les mêmes IPs utilisées par [117], puis nous nous sommes alignés sur les mêmes métriques à savoir : fréquence de fonctionnement maximale, coût logique et même taille de paquet pour avoir des repères communs. Comme dans le travail de [117], nous avons décomposé le module intra-prédiction en modules intra-prédictifs 4x4 et 16x16 et dans chaque sous-module inclus : modules DCT, et quantification, modules de quantification inverse et DCT inverse, comme détaillé dans la figure 4.6. Par la suite, nous avons comparé les résultats de synthèse de notre chaîne mise en œuvre connectée en utilisant le Buffer-less NoC avec ceux des auteurs [117] ; Cette comparaison est résumée dans le Tableau 4.2.

L'examen de ces résultats de mise en œuvre de la chaîne intra montre une légère optimisation des ressources FPGA jusqu'à 8,50% avec une amélioration significative de la vitesse de fonctionnement de 165,78 MHz à 197,61 MHz (19,20%), ce qui démontre l'efficacité de notre proposition.

**Tableau 4.2.** Comparaison des résultats de synthèse des IPs de H.264/AVC connectés par : un NoC 3x3 2-D Mesh avec buffers sur la plateforme FPGA XUPV5 [117] et un NoC 3x3 2-D Mesh sans buffers sur la plateforme FPGA XC7V2000T

| | | Number of slice registers | Number of slice LUTs | fully used LUT Slices % | Maximum Operating Frequency |
|---|---|---|---|---|---|
| Synthesis result of H.264/AVC IPs connected by 3×3 mesh conventional NoC [117] | **3×3 mesh conventional NoC** | 127 | 2,154 | 5.89% | 172.83MHz |
| | **Intra4×4 prediction** | 1,477 | 1,703 | 38.02% | 278.43MHz |
| | **Intra16×16 prediction** | 2,650 | 2,180 | 49.33% | 282.03MHz |
| | **DCT & Quantization** | 1,821 | 1,603 | 59.49% | 594.11MHz |
| | **Inverse Quantization & Inverse DCT** | 1,865 | 1,712 | 38.70% | 394.06MHz |
| | **Deblocking filter** | 6.702 | 8.890 | 29.20% | 188.64MHz |
| | **Synthesis results of the IPs of the chain connected by a conventional 3x3 mesh NoC** | **14,642** | **18,242** | **37.10%** | **165.78MHz** |
| Synthesis result of H.264/AVC IPs connected by the proposed 3×3 mesh NoC | **3×3 mesh buffer-less NoC** | 191 | 603 | 57.53% | 427.56MHz |
| | **Intra4×4 prediction** | 1,477 | 1,703 | 44.51% | 295.84MHz |
| | **Intra16×16 prediction** | 2,650 | 2,180 | 54.69% | 305.36MHz |
| | **DCT & Quantization** | 1,821 | 1,603 | 63.21% | 620.46MHz |
| | **Inverse Quantization & Inverse DCT** | 1,865 | 1,712 | 42.18% | 418.67MHz |
| | **Deblocking filter** | 6.702 | 8.890 | 31.98% | 207.94MHz |
| | **Synthesis results of the IPs of the chain connected by a buffer-less 3×3 mesh NoC** | **14,706** | **16,691** | **49.01%** | **197.61MHz** |

Le dernier tableau de résultats (Tableau 4.3.) présente les résultats de synthèse du système entier basé sur Microblaze, mis en œuvre en utilisant les trois approches différentes :

- Connexion des IPs du H.264/AVC en cascade [117],

- Connexion des IPs du H.264/AVC par un NoC avec buffer [117],

- L'approche proposée : Connexion des IPs du H.264/AVC par un NoC sans buffer.

Il ressort de ces résultats de synthèse (Tableau 4.3) que notre approche est aussi efficace ou meilleure que les deux autres approches [117] en termes de coût logique. Cependant, il est compréhensible, à travers la fréquence de fonctionnement, que l'interconnexion sur puce sans buffer soit plus rapide d'environ 27,49% que l'approche en cascade et d'environ 19,20% par rapport à la solution NoC avec buffer [117]. En outre, avec une telle réduction des ressources FPGA et l'abolition des buffers, nous prévoyons une diminution significative de la consommation d'énergie.

**Tableau 4.3.** Résultats de synthèse du système à base du Microblaze implémenté pour les IPs de l'encodeur H.264/AVC IPs pour les trois approches

| | **Number of slice registers used** | **Number of slice LUTs used** | **Maximum Frequency** |
|---|---|---|---|
| **SoC for H.264/AVC IPs connected in cascade [117]** | 23,589 | 25,092 | 155MHz |
| **SoC for H.264/AVC IPs using 3x3 Mesh conventional NoC [117]** | 21,768 | 23,112 | 165.78MHz |
| **SoC for H.264/AVC IPs using the proposed 3x3 Mesh buffer-less NoC** | **21,835** | **22,559** | **197.61MHz** |

## 6. Conclusion

A travers ce dernier chapitre, nous avons proposé une architecture à base de réseau sur puce sans buffer adapté aux applications vidéo. La principale originalité de notre solution est de tirer parti de ce nouveau paradigme de communication sur puce, Networks-on-Chip, tout en l'adaptant au domaine des applications vidéo. Les applications de codage vidéo (décodage) font partie des applications hautes performances, ce qui rend la solution NoC évidente, d'autant plus qu'elle a fait ses preuves et qu'elle a émergé par ses performances. L'idée était d'éliminer les buffers de l'architecture du routeur pour gagner en ressources logiques des FPGAs, mais aussi en fréquence de fonctionnement maximale. Une telle solution a nécessité un algorithme de routage adapté basé sur la déflexion afin de pallier aux problèmes de congestion et de blocage au sein du réseau.

Afin de prouver l'efficacité de la solution proposée, nous avons réalisé une simulation du NoC 3x3 mesh sans buffer ainsi que les modules élémentaires H.264/AVC sur la plateforme FPGA Virtex7-Xilinx dans le langage de description matérielle VHDL. Par

la suite, l'ensemble a été connecté à la plate-forme matérielle MicroBlaze. Une première comparaison entre le NoC sans buffer proposé et d'autres travaux similaires a été réalisée et elle a donné des résultats encourageants. En effet, les résultats de synthèse ont démontré l'amélioration, par rapport à la solution NoC avec buffer, en réduisant le coût logique de plus de 65%, tout en augmentant également la vitesse de fonctionnement pour opérer avec une fréquence de 175 MHz à 430 MHz (dans le meilleur des cas).

Ensuite, nous avons simulé l'ensemble du SoC afin de percevoir le comportement de l'ensemble du système sur puce et de le comparer aux travaux précédents, en l'occurrence l'approche basée sur le bus et l'approche basé sur Mesh 2-D 3x3 avec buffer. Plus précisément, notre approche NoC sans buffer proposée a réduit la consommation des LUTs utilisées de 6,43% par rapport à l'approche NoC buffer [117] et de 10,09% par rapport à l'approche en cascade. Une telle différence peut être expliquée par le gain du FPGA en raison de l'abolition des buffers par rapport au NoC conventionnel et de l'absence de blocs redondants utilisant l'architecture en cascade [117]. En outre, l'approche proposée sans buffer présente également de meilleures performances en améliorant la fréquence de fonctionnement d'environ 19,20% par rapport à l'approche NoC avec buffer et de 27,49% par rapport à l'approche en cascade [117]. Enfin, notre approche a démontré son adaptation aux applications vidéo, ce qui la rend plus facilement utilisable pour d'autres standards vidéo en raison de la polyvalence et l'évolutivité des réseaux sur puce.

# Conclusion et perspectives

# Conclusion et perspectives

## La réponse à la problématique

Depuis le début du nouveau millénaire, le monde de la conception de systèmes numériques a connu un phénomène sans précédent : une réduction rapide et persistante de la taille des entités dans le domaine nanométrique. Les progrès de la technologie des dispositifs et des techniques de fabrication ont permis aux concepteurs de pénétrer dans des territoires jusque-là inexplorés ; l'intégration de milliards de transistors sur puce est devenue maintenant une réalité. Cette ultra haute densité d'intégration (ULSI) a augmenté la complexité des applications ce qui a engendré une multiplication du nombre d'IPs dans les SoCs.

Cette poussée multi-core (multi-IP) a introduit un déplacement progressif du modèle de conception centré calcul vers une approche centrée communication. Les grands modules monolithiques sophistiqués cèdent la place à plusieurs éléments de traitement plus petits et plus simples fonctionnant en tandem. Cette tendance a conduit à une montée en puissance de la popularité des systèmes multi-core, qui se manifestent généralement sous deux formes distinctes : multiprocesseurs sur puces (CMP) et systèmes multiprocesseurs sur puce (MP-SoC).

La philosophie SoC s'articule autour de la technique Platform-Based Design (PBD), qui préconise la réutilisation des cores de propriété intellectuelle (IP) dans des modèles de conception flexibles qui peuvent être personnalisés pour répondre aux exigences des implémentations particulières. L'attrait d'une telle approche modulaire réside dans la période d'incubation « Time-to-Market » considérablement réduite, qui résulte directement de la complexité du circuit et de la réduction de l'effort de conception. L'ensemble du système peut maintenant être considéré comme une collection diversifiée de composants IPs préexistants intégrés sur une seule puce. L'architecture Wishbone basé sur le partage des IPs, se distingue des autres protocoles en étant un protocole à accès libre. Une communauté de développeurs a adapté à l'interface Wishbone à un très grand nombre de conception « open-source » pour des mémoires, processeurs, PEs, des périphériques auxiliaires, voire même des systèmes complets. Ces conceptions disponibles chez OpenCores, une fondation qui essaie de rendre disponible le matériel open-source, permettent d'exploiter encore plus la notion de partage et réutilisation des IPs.

Toutes ces architectures MP-SoC émergentes, exigent un haut débit, une faible latence et des services de communication globaux et fiables. Essayer de réaliser de telles

conceptions avec une structure de bus partagé pourrait être problématique pour un certain nombre de raisons, y compris les problèmes de performance, de coût et d'évolutivité. Un changement de paradigme semble être l'approche la plus prometteuse pour résoudre ces crises de communication. Par conséquent, au cours des dernières années, une nouvelle méthodologie appelée Network-on-Chip a été introduite comme un moyen pour résoudre ces problèmes en introduisant une architecture de communication structurée et évolutive.

Les travaux réalisés lors de cette thèse s'inscrivent également dans ce double contexte :

- Réalisation d'une solution d'interconnexion sur puce basé sur les réseaux sur puce (NoCs),
- Adaptation du protocole Wishbone pour les réseaux sur puce afin de tirer profit des IPs open-source disponibles et optimiser le temps nécessaire à la conception des systèmes sur puce.

Pour atteindre l'objectif principal ainsi que les objectifs secondaires, nous avons entamé nos travaux par une étude bibliographique détaillée au cours de laquelle nous nous sommes intéressés au concept des systèmes multiprocesseurs sur puce (MP-SoCs), les difficultés de conception, et la gestion de la communication au sein de ces systèmes. Nous avons ensuite présenté les différentes technologies utilisées pour l'implémentation d'applications numériques (CMP, GPP, ASIC, FPGA...etc.) en essayant d'avoir des éléments de comparaisons entre-elles. Cette comparaison permet de choisir la technologie offrant le meilleur équilibre performance/flexibilité. En outre, le paradigme des NoCs est présenté puis détaillé afin de mettre en avant la diversification et la richesse qu'offre les NoCs en termes de performance et d'architectures.

En matière de conception, nous avons réalisé une plateforme pour la conception des réseaux sur puce de type Wishbone. Pour cela, nous proposé un flot de conception qui permet à partir des paramètres d'entrée du réseau de générer le code VHDL qui sera par la suite testé et simulé. L'adaptation du Wishbone aux NoCs a été réalisée grâce au développement d'interface réseau (NI : Network Interface). Ces interfaces (Maître / Esclave) permettent une parfaite adaptation de tous les composants Wishbone au réseau généré par la plateforme. La plateforme est testée par la génération des codes VHDL de différents NoCs, avec différentes architectures, et différents niveaux de performance. Ces conceptions sont validées par simulation de l'implémentation matérielle sur plateforme de prototypage FPGA.

Parallèlement à cette plateforme réalisée pour la génération des NoCs de type Wishbone, une autre solution à base de réseau sur puce a été proposée. Une architecture

de réseau sur puce sans buffer (Buffer-less-NoC) adaptée pour les applications d'encodage vidéo. Cette architecture est simulée, ses performances sont comparées à d'autres travaux similaires et les résultats sont discutés.

Toutes les architectures ont été testées et simulées sur technologie FPGA pour une comparaison équitable, et la solution proposée a démontré sa supériorité en termes de performance et coût. La performance a été illustrée par une fréquence de fonctionnement supérieure, et le coût par une consommation inférieure en ressources logiques. L'idée de supprimer les buffers à confirmer son intérêt grâce particulièrement à l'algorithme de routage par déflexion qui a permis de surmonter les risques de congestion et de blocage.

### Perspectives et valorisation des résultats

Le travail réalisé à travers cette thèse a été concluant et prometteur et montre l'intérêt des approches proposées. Notre volonté était axée sur l'atteinte des objectifs et la réalisation d'une plateforme qui répond aux exigences des systèmes multiprocesseurs (MP-SoCs) modernes. Cependant des améliorations et optimisations sont toujours envisageables et notre travail servira de ligne directrice pour des travaux futurs.

Concernant la plateforme de génération des NoCs de type Wishbone, une modélisation haut niveau peut s'avérer efficace pour une meilleure exploitation, et une simplification de l'utilisation de la plateforme. L'exploration des topologies 3D peut constituer une perspective attirante car la plateforme réalisée dans ce travail ne permet de générer que des topologies 2D.

Un outil de modélisation des performances des NoCs peut aussi être une perspective intéressante, car fournir une solution d'interconnexion sur puce sur mesure reste envisageable si on arrive à prédire la performance du NoC avant même sa conception.

Du côté de l'architecture Buffer-less NoC, bien que l'encodeur H.264/AVC ait fait ses preuves, l'avènement de la norme H.265/HEVC, qui constitue la nouvelle référence dans le domaine, nous mène à vouloir tester cette approche sur ce standard, plus récent, plus performant et surtout plus exigeant en termes de communication sur puce.

# Références bibliographiques

# Références Bibliographiques

[1] Gordon E Moore, "Cramming more components onto integrated circuits," *Electronics*, Vol. 38, No. 8. (19 April 1965), pp. 114-117.

[2] International Technology Roadmap for Semiconductors, report 2012.

[3] R. Saleh, S. Mirabbasi, A. Hu, M. Greenstreet, G. Lemieux, P. P. Pande, C. Grecu, and A. Ivanov, "System-on-Chip: Reuse and Integration," Proceedings of the IEEE, vol. 94, no. 6, Jun. 2006, pp. 1050-1069.

[4] H. Nikolov, T. Stefanov, E. Deprettere, "Systematic and Automated Multiprocessor System Design, Programming, and Implementation", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 27, n° 3, 2008, pp. 542-555.

[5] W. Wolf, A. A. Jerraya, G. Martin, "Multiprocessor System-on-Chip (MPSoC) Technology", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol.27, n° 10, 2008, pp. 1701-1713.

[6] B. Poudel, N. K. Giri, A. Munir, "Design and comparative evaluation of GPGPU- and FPGA-based MPSoC ECU architectures for secure, dependable, and real-time automotive CPS", 2017 IEEE 28th International Conference on Application-specific Systems, Architectures and Processors (ASAP), July 10-12, 2017, Seattle, WA, USA.

[7] A. Kunimatsu, N. Ide, T. Sato, Y. Endo, H. Murakami, T. Kamei, M. Hirano, F. Ishihara, H. Tago, M. Oka, A. Ohba, T. Yutaka, T. Okada, M. Suzuoki, "Vector unit architecture for emotion synthesis", IEEE Micro, vol. 20, n° 2, 2000, pp. 40–47.

[8] W. Wolf, "The future of MultiProcessor Systems-on-Chip", The 41st Design Automation Conference, 2004, July 7-11, 2004, San Diego, CA, USA.

[9] A. K. Coskun, T. S. Rosing, K. Whisnant, "Temperature Aware Task Scheduling in MPSoCs", 2007 Design, Automation & Test in Europe Conference & Exhibition, April 16-20, 2007, Nice, France.

[10] J. Tan, D. Hua, "Emulation platform for dedicated to adaptive MPSoC-based NoC architecture for the art authentication application", 2016 8th IEEE International Conference on Communication Software and Networks (ICCSN), June 4-6, 2016, Beijing, China.

[11] D.E. Culler, J.P. Singh, A. Gupta, "Parallel Computer Architecture: A Hardware/Software Approach", Morgan Kaufman, San Francisco, 1999.

[12] E. Waingold, M. Taylor, D. Srikrishna, V. Sarkar, W. Lee, V. Lee, J. Kim, M. Frank, P. Finch, R. Barua, J. Babb, S. Amarasinghe, A. Agarwal, "Baring it all to software: raw machines", IEEE Computer, vol. 30, n° 9, 1997, pp. 86–93.

[13] S. Wenger, "H.264/AVC over IP", IEEE Transactions on Circuits and Systems for Video Technology, vol. 13, n° 7, 2003, pp. 645-656.

[14] S. Esposito, M. Violante, "System-level architecture for mixed criticality applications on MPSoC: A space application", 2017 IEEE International Workshop on Metrology for AeroSpace (MetroAeroSpace), June 21-23, 2017, Padua, Italy.

[15] E. Monmasson, M. N. Cirstea, "FPGA Design Methodology for Industrial Control Systems-A Review", IEEE Transactions on Industrial Electronics, vol. 54, n° 4, 2007, pp. 1824-1842.

[16] B. Ahmad, A. T. Erdogan, S. Khawam, "Architecture of a Dynamically Reconfigurable NoC for Adaptive Reconfigurable MPSoC", June 15-18, 2006, Istanbul, Turkey.

[17] T. von Sydow, B. Neumann, H. Blume, T. G. Noll, "Quantitative analysis of embedded fpgaarchitectures for arithmetic", in Proceedings of the International Conference on Application-Specific Systems, Architectures and Processors, September 2006, (ASAP '06), pp. 125–131.

[18] K. Karuri, R. Leupers, "Application analysis tools for ASIP design: application profiling and instruction-set customization", Springer, New York, 2011.

[19] M. Shafique, "Architectures for adaptive low-power embedded multimedia systems". Ph.D. thesis, Karlsruhe Institute of Technology, Germany, 2011.

[20] D. Talla, L. John, V. Lapinskii, B. Evans, "Evaluating signal processing and multimedia applications on simd, vliw and superscalar architectures", in Proceedings of the 2000 International Conference on Computer Design, 2000, pp. 163–172.

[21] I. Kuon, J. Rose, "Measuring the Gap Between FPGAs and ASICs", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 26, n° 2, 2007, pp. 203-215.

[22] A. DeHon, "Balancing interconnect and computation in a reconfigurable computing array (or, why you don't really want 100% LUT utilization", Proceedings of the 1999 ACM/SIGDA seventh international symposium on Field programmable gate arrays, February 21-23, 1999, Monterey, California, USA, pp. 69-78.

[23] V. Betz, A. Marquardt, J. Rose, "Architecture and CAD for Deep-Submicron FPGAs". Kluwer Academic Publishers, 1999.

[24] Z. Marrakchi, "Exploration and optimization of tree-based FPGA architectures", PhD Thesis, Paris, France, 2008.

[25] E.Ahmed, J.Rose, "The Effect of LUT and Cluster Size on Deep-submicron FPGA Performance and Density", Proceedings of the International Symposium on Field Programmable Gate Arrays, February 10-11, 2000, Monterey, California, USA pp. 3–12.

[26] G.Lemieux, E.Lee, M.Tom, and A.Yu, "Directional and Single-Driver Wires in FPGA Interconnect", IEEE International Conference on Field-Programmable Technology (ICFPT), December 6-8, 2004, Brisbane, NSW, Australia, pp. 41–48.

[27] F. N. Najm, ''A Survey of Power Estimation Techniques in VLSI Circuits,'' IEEE Transactions on Very Large Scale Integrations Systems, December 1994, vol. 2, n° 4, pp. 446–455.

[28] R. Ho, K. W. Mai, and M. A. Horowitz, ''the Future of Wires,'' Proceedings of the IEEE, April 2001, vol. 89, n° 4, pp. 490–504.

[29] ARM, AMBA Specification Rev 2.0, ARM Limited, 1999.

[30] IBM, 32-bit Processor Local Bus Architecture Specification Version 2.9, IBM Corporation.

[31] L. Benini and G. DeMicheli, ‘‘Networks on Chips: a New SoC Paradigm,’’ IEEE Transactions on Computers, January 2002, vol. 35, no. 4, pp. 70–78.

[32] W. J. Dally and B. Towles, “Principles and Practices of Interconnection Networks”, Morgan Kaufmann, 2004.

[33] W. J. Dally and B. Towles, ‘‘Route Packets, Not Wires: On-Chip Interconnection Networks,’’ in Proceedings of the Design Automation Conference, June 2001, Las Vegas, NV, USA, pp. 684–689.

[34] M. Kistler, M. Perrone, and F. Petrini, ‘‘Cell Multiprocessor Communication Network: Built for Speed,’’ IEEE Micro, May 2006, vol. 26, no. 3, pp. 10–23.

[35] L. Seiler, D. Carmean, E. Sprangle, T. Forsyth, P. Dubey, S. Junkins, A. Lake, R. Cavin, R. Espasa, E. Grochowski, T. Juan, M. Abrash, J. Sugerman, P. Hanrahan, ‘‘Larrabee: A Many-Core x86 Architecture for Visual Computing,’’ IEEE Micro, January 2009, vol. 29, n° 1, pp. 10–21.

[36] D. Wentzlaff, P. Griffin, H. Hoffmann, L. Bao, B. Edwards, C. Ramey, M. Mattina, C. C. Miao, J. F. Brown, and A. Agarwal, ‘‘On-Chip Interconnection Architecture of the Tile Processor,’’ IEEE Micro, September 2007, vol. 27, n° 5, pp. 15–31.

[37] J. Howard, S. Dighe, Y. Hoskote, S. Vangal, D. Finan, G. Ruhl, D. Jenkins, H. Wilson, N. Borkar, G. Schrom, F. Pailet, S. Jain, T. Jacob, S. Yada, S. Marella, P. Salihundam, V. Erraguntla, M. Konow, M. Riepen, G. Droege, J. Lindemann, M. Gries, T. Apel, K. Henriss, T. L. Larsen, S. Steibl, S. Borkar, V. De, R. Van Der Wijngaart, and T. Mattson, ‘‘A 48-Core IA-32 Message-Passing Processor with DVFS in 45nm CMOS”, February 2010, in Proceedings of the IEEE International Solid-State Circuits Conference, San Francisco, CA, USA, pp. 108–109.

[38] A. Hemani, A. Jantsch, S. Kumar, A. Postula, J. Oberg, M. Millberg, and D. Lindvist, ‘‘Network-on-Chip: an Architecture for Billion Transistor Era,’’ in Proceedings of the IEEE NorChip Conference, pp. 1–8, July 2000

[39] S. Kumar, A. Jantsch, J. P. Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja, and A. Hemani, ‘‘A Network-on-Chip Architecture and Design Methodology,’’ in Proceedings of the International Symposium on Very Large Scale Integration, April 2000, Pittsburgh, PA, USA, pp. 105–112.

[40] Y. Tamir and G. L. Frazier, ‘‘Dynamically-Allocated Multi-Queue Buffers for VLSI Communication Switches,’’ IEEE Transactions on Computers, June 1992, vol. 41, n° 6, pp. 725–737.

[41] W. J. Dally and C. L. Seitz, ‘‘The Torus Routing Chip,’’ Journal of Distributed Computing, January 1986, vol. 1, n° 4, pp. 187–196.

[42] P. Kermani and L. Kleinrock, ‘‘Virtual Cut-Through: A New Computer Communication Switching Technique,’’ Computer Networks, September 1979, vol. 3, n° 4, pp. 267–286.

[43] A. El-Naggar, A. Medhat, B. Al-Abassy, E. Massoud, H. Ibrahim, M. Khamis, A. Shalaby, “Performance evaluation of virtual channel flow control in centralized and distributed networks

for system on chip", The 29th International Conference on Microelectronics (ICM). December 10-13, 2017, Beirut, Lebanon.

[44] L. S. Peh and W. J. Dally, ''A Delay Model for Router Microarchitectures,'' IEEE Micro, January 2001, vol. 21, no. 1, pp.26–34.

[45] S. Moriam, G. Fettweis, "Fault Tolerant Deadlock-Free Adaptive Routing Algorithms for Hexagonal Networks-on-Chip", 2016 Euromicro Conference on Digital System Design (DSD). August 31 – September 2, 2016, Limassol, Cyprus.

[46] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, ''Routing Table Minimization for Irregular Mesh NoCs,'' in Proceedings of the Design Automation and Test in Europe Conference, April 16-20, 2007, Nice, France, pp. 1–6.

[47] M. A. Yazdi, M. Modarressi, and H. S. Azad, ''A Load-Balanced Routing Scheme for NoCBased System-on-Chip,'' in Proceedings of the Workshop on Hardware and Software Implementation and Control of Distributed MEMS, June 28-29, 2010, Besançon, France, pp. 72–77,

[48] M. Daneshtalab, A. A. Kusha, A. Sobhani, Z. Navabi, M. D. Mottaghi, and O. Fatemi, ''Ant Colony Based Routing Architecture for Minimizing Hot Spots in NOCs,'' in Proceedings of the Annual Symposium on Integrated Circuits and System Design, September 11-13, 2006, Steamboat Springs, CO, USA, pp. 56–61,

[49] J. Hu and R. Marculescu, ''DyAD–Smart Routing for Networks-on-Chip,'' in Proceedings of the Design Automation Conference, June 7-11, 2004, San Diego, CA, USA, pp. 260–263,

[50] J. Alshraiedeh, A. Kodi, "An adaptive routing algorithm to improve lifetime reliability in NoCs architecture", 2016 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT). September 19-20, 2016, Storrs, CT, USA.

[51] G. Ascia, V. Catania, M. Palesi, and D. Patti, ''Neighbors On-Path: A New Selection Strategy for On-Chip Networks,'' in Proceedings of the IEEE Workshop on Embedded Systems for Real Time Multimedia, October 26-27, 2006, Seoul, South Korea, pp. 79–84.

[52] M. Li, Q.A. Zeng, and W. B. Jone, ''DyXY - a Proximity Congestion-Aware Deadlock-Free Dynamic Routing Method for Network on Chip,'' in Proceedings of the Design Automation Conference, July 24-28, 2006, San Francisco, CA, USA, pp.849–852.

[53] E. Nilsson, M. Millberg, J. Oberg, and A. Jantsch, ''Load Distribution with the Proximity Congestion Awareness in a Network-on-Chip,'' in Proceedings of the Design Automation and Test in Europe Conference, December 2003, Munich, Germany, pp.1126–1127,

[54] J. Kim, D. Park, T. Theocharides, N. Vijaykrishnan, and C. R. Das, ''A Low Latency Router Supporting Adaptivity for on-Chip Interconnects,'' in Proceedings of the Design Automation Conference, June 13-17, 2005, Anaheim, CA, USA, pp. 559–564.

[55] D. Wu, B. M. Al-Hashimi, and M. T. Schmitz, ''Improving Routing Efficiency for Networkon-Chip through Contention-Aware Input Selection,'' in Proceedings of the Asia and South Pacific Design Automation Conference, January 24-27, 2006, Yokohama, Japan, pp. 36–41.

[56] N. Wang, P. Valencia, "Traffic Allocation: An efficient adaptive network-on-chip routing algorithm design" The 2nd IEEE International Conference on Computer and Communications (ICCC), October 14-17, 2016, Chengdu, China.

[57] P. Valencia, E. Muller, N. Wang, "ZigZag: An efficient deterministic Network-on-chip routing algorithm design", The 8th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), October 3-5, 2017, Vancouver, BC, Canada.

[58] S. Umapathy, M. Shah, N. Wang, "Encircle routing: An efficient deterministic network on chip routing algorithm", IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC), January 8-10, 2018, Las Vegas, NV, USA.

[59] K. Goossens, J. Dielissen, and A. Radulescu, ''The Æthereal Network on Chip: Concepts, Architectures, and Implementations,'' IEEE Design & Test of Computers, vol. 22, no. 5, pp. 414–421, October 2005

[60] P. Vellanki, N. Banerjee, and K. S. Chatha, ''Quality-of-Service and Error Control Techniques for Mesh-Based Network-on-Chip Architectures,'' ACM Very Large Scale Integration Journal, vol. 38, no. 3, pp. 353–382, January 2005

[61] Y. Chen, E. Matus, S. Moriam, G. Fettweis, "High Performance Dynamic Resource Allocation for Guaranteed Service in Network-on-Chips", IEEE Transactions on Emerging Topics in Computing, 2017.

[62] F. Samman, "Network-on-chip with guaranteed-bandwidth data communication service", International Conference on Computational Intelligence and Cybernetics. November 22-24, 2016, Makassar, Indonesia.

[63] E. Bolotin, I, Cidon, R. Ginosar, and A. Kolodny, ''QNoC: QoS Architecture and Design Process for Network-on-Chip,'' Elsevier Journal of System Architecture, vol. 50, no.2–3, pp. 105–128, February 2004

[64] M. D. Harmanci, N. P. Escudero, Y. Leblebici, and P. Ienne, ''Providing QoS to Connection- Less Packet-Switched NoC by Implementing DiffServ Functionalities,'' in Proceedings of the International Symposium on System-on-Chip, pp. 37–40, November 2004

[65] Z. Guz, E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, ''Efficient Link Capacity and QoS Design for Network-on-Chip,'' in Proceedings of the Design Automation and Test in Europe Conference, March 6-10, 2006, Munich, Germany, pp. 1–6.

[66] E. Rijpkema, K. G. W. Goossens, A. Radulescu, J. Dielissen, J. V. Meerbergen, P. Wielage, and E. Waterlander, ''Trade-offs in the Design of a Router with Both Guaranteed and Best-Effort Services for Networks-on-Chip,'' in Proceedings of the Design Automation and Test in Europe Conference, pp. 350–355, March 2003

[67] M. D. Harmanci, N. P. Escudero, Y. Leblebici, and P. Ienne, ''Quantitative Modeling and Comparison of Communication Schemes to Guarantee Quality-of-Service in Networks-on-Chip'', in Proceedings of the International Symposium on Circuits and Systems, May 2005, Kobe, Japan, pp. 1782– 1785,

[68] P. Bogdan, T. Dumitras, and R. Marculescu, ''Stochastic Communication: A New Paradigm for Fault Tolerant Networks on Chip,'' VLSI Design, vol. 2007, Article ID 95348, pp. 1–17, 2007

[69] Y. Hatanaka, M. Nakamura, Y. Kakuda, and T. Kikuno, ‘‘A Synthesis Method for Fault-Tolerant and Flexible Multipath Routing Protocols,’’ in Proceedings of the International Conference on Engineering of Complex Computer Systems, September 1997, Como, Italy, pp. 96–105,

[70] W. Stallings, Data and Computer Communications, Prentice Hall, 2007

[71] T. Dumitras, S. Kerner, and R. Marculescu, ‘‘Towards On-Chip Fault-Tolerant Communication,’’ in Proceedings of the Asia and South Pacific Design Automation Conference, pp.225–232, January 2003, Kitakyushu, Japan.

[72] M. Pirretti, G. M. Link, R. R. Brooks, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin, ‘‘Fault Tolerant Algorithms for Network-on-Chip Interconnect,’’ in Proceedings of the IEEE Computer Society Annual Symposium on VLSI, pp. 46–51, February 2004, Lafayette, LA, USA.

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

[74] D. Wiklund, L. Dake Liu, SoCBUS: “switched network on chip for hard real time embedded systems”, in Proceedings of International Parallel and Distributed Processing Symposium, April 22-26, 2003, Nice, France.

[75] I. Saastamoinen, M. Alho, J. Nurmi, “Buffer implementation for Proteo network-on-chip”. International Symposium on Circuits and Systems, May 25-28, 2003, Bangkok, Thailand.

[76] F. Karim A. Nguyen, S. Dey, “An interconnect architecture for networking systems on chips”. IEEE J. Micro High Perform. Interconnect, 2002, vol. 22, n° 5, pp. 36–45.

[77] A. Weldezion, M. Grange, D. Pamunuwa, L. Zhonghai, A. Jantsch, R. Weerasekera, H. Tenhunen, “Scalability of network-on-chip communication architecture for 3-D meshes”, in International Symposium on Networks-on-Chip, May 10-13, 2009, San Diego, CA, USA, pp. 114–123.

[78] K. Puttaswamy, G.H. Loh, “Thermal analysis of a 3D die-stacked high-performance microprocessor”, in ACM Great Lakes Symposium on VLSI (GLSVLSI), pp. 19–24, May 2006, Philadelphia, PA, USA

[79] K. Tatas, K. S. D. Soudris, A. Jantsch, “Designing 2D and 3D Network-on-Chip Architectures".

[80] J. Balfour and W. J. Dally. “Design tradeoffs for tiled CMP on-chip networks”. In ICS ’06: Proceedings of the 20th annual international conference on Supercomputing, pages 187–198, June 2006, Cairns, Queensland, Australia.

[81] R. Das, S. Eachempati, A. K. Mishra, N. Vijaykrishnan, and C. R. Das. “Design and evaluation of a hierarchical on-chip interconnect for next-generation CMPs”. In International Symposium on High-Performance Computer Architecture (HPCA), pages 175–186, Raleigh, North Carolina, 2009.

[82] B. Grot, J. Hestness, S. W. Keckler, and O. Mutlu. Express cube topologies for on-chip interconnects. In International Symposium on High-Performance Computer Architecture (HPCA), pages 163–174, Raleigh, North Carolina, 2009.

[83] J.-S. Kim et al., “On-chip network based embedded core testing,” in Proc. IEEE International SoC Conference, September 2004, Santa Clara, CA, USA, pp. 223–226.

[84] D. Seo, A. Ali, W.-T. Lim, N. Rafique, and M. Thottethodi. “Near optimal worst-case throughput routing for two-dimensional mesh networks”. In Proc. of the International Symposium on Computer Architecture (ISCA), vol. 33, n° 2, June 2005, Madison, WI, USA, pages 432–443.

[85] A. Kumar, L.-S. Peh, P. Kundu, and N. K. Jhay. Express virtual channels: Towards the ideal interconnection fabric. In Proc. of the International Symposium on Computer Architecture (ISCA), San Diego, CA, June 2007.

[86] P. Abad, V. Puente, J. A. Gregorio, and P. Prieto. Rotary router: An efficient architecture for cmp interconnection networks. In Proc. of the International Symposium on Computer Architecture (ISCA), San Diego, CA, June 2007.

[87] A. Kumar, P. Kundu, A. Singh, L.-S. Peh, and N. Jha. A 4.6tbits/s 3.6ghz single-cycle noc router with a novel switch allocator in 65nm cmos. In International Conference on Computer Design (ICCD), October 2007.

[88] H. Matsutani, M. Koibuchi, H. Amano, and T. Yoshinaga. Prediction router: Yet another low latency on-chip router architecture. In International Symposium on High-Performance Computer Architecture (HPCA), pages 367–378, Raleigh, North Carolina, 2009.

[89] J. Dielissen, et al., “Concepts and implementation of the Philips network-on-chip,” in IP-Based SOC Design, Nov. 2003.

[90] J. N. Lassen, “FPGA prototyping of asynchronous Networks-on-Chip,” M.Sc. thesis, Technical University of Denmark, DTU, 2008.

[91] M. Morales, S.Rau,M.J. Palma,M.Venkatesan, F. Pulskamp,A.Dugar, “Worldwide intelligent systems 2011–2015 forecast: the next big opportunity”. Technical report, International Data Corporation, September 2011

[92] D. Lund, C. MacGillivray, V. Turner, M. Morales, “Worldwide and Regional Internet of Things (IoT) 2014–2020 Forecast: A Virtuous Circle of Proven Value and Demand”, Market analysis, Technical report, International Data Corporation, May 2014.

[93] Advanced Video Coding for Generic Audiovisual Services, Recommendation H.264 and ISO/IEC 14 496-10 (MPEG-4) AVC, International Telecommunication Union-Telecommun. (ITU-T) and International Standards Organization/International Electrotechnical Commission (ISO/IEC) JTC 1 (2003)

[94] Marpe, D., Schwarz, H., Wiegand, T.: Context-Based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression Standard. IEEE Transactions on Circuits and Systems for Video Technology, vol 13, n° 7, 620–636, 2003.

[95] K. Messaoudi, “Traitement des signaux et images en temps réel : "implantation de H.264 sur MPSoC", Thèse, Université de Bourgogne, Dijon, France/ Université Badji-Mokhtar, Annaba, Algérie, 2012.

[96] C. A. Nicopoulos, D. Park, J. Kim, N. Vijaykrishnan, M. S. Yousif, C. R. Das : ViChaR: A Dynamic Virtual Channel Regulator for Network-on-Chip Routers. The 39th Annual

IEEE/ACM International Symposium on Microarchitecture 2006 (MICRO'06), December 9-13, 2006, Orlando, Florida, USA.

[97] T. C. Xu, P. Liljeberg, H. Tenhunen : "A Study of Through Silicon Via Impact to 3D Network-on- Chip Design", International Conference on Electronics and Information Engineering (ICEIE 2010), 1-3 August 2010, Kyoto, JAPAN.

[98] P. P. Pande, C. Grecu, M. Jones, A. Ivanov, and R. Saleh : "Performance evaluation and design trade-offs for network-on-chip interconnect architectures". IEEE Transaction on Computers, vol.54, n°8, 2005, pp. 1025–1040.

[99] T. Sikora : "The MPEG-4 video standard verification model". IEEE Transactions on Circuits and Systems for Video Technology, Vol. 7, n° 1, 1997, pp. 19-31.

[100] I. Richardson : "Real-time implementation of H.264 Video Coding". IEEE SOC Conference, September 17-20, 2008, Newport Beach, CA, USA.

[101] K. Messaoudi, E. B. Bourennane, S. Toumi, M. Touiza "Data-Parallelism Based Hardware Architecture for the Intra-Coding Module in the H.264 Encoder", Springer, The Arabian Journal for Science and Engineering, vol.39, n°5, 2014, pp. 3781-3797.

[102] S. Saponaraa, M. Martinab, M. Casulaa, L. Fanucci, G. Masera "Motion estimation and CABAC VLSI co-processors for real-time high-quality H.264/AVC video coding", Microprocessors and Microsystems, vol. 34, n° 7–8, 2010, pp. 316–328.

[103] Y. W. Huang, B. Y. Hsieh, T. C. Chen, L. G. Chen "Analysis, Fast Algorithm, and VLSI Architecture Design for H.264/AVC Intra Frame Coder", IEEE Transactions on Circuits and Systems for Video Tech, vol. 15, n° 3, 2005, pp. 378-401.

[104] B. A. Ringnyu, A. Tangel. E. Karabulut "Implementation of different architectures of forward 4×4 integer DCT for H.264/AVC encoder", The 10th International Conference on Electrical and Electronics Engineering (ELECO), November 30 – December 2, 2017, Bursa, Turkey.

[105] C. W. Ku, C. C. Cheng, G. S. Yu, M. C. Tsai, T. S. Chang "A High-Definition H.264/AVC Intra-Frame Codec IP for Digital Video and Still Camera Applications", IEEE Transactions on Circuits and Systems for Video Tech., vol. 16, n° 8, 2006, pp. 917-928.

[106] B. G. Kim, J. H. Kim, C. S. Cho "A Fast Intra Skip Detection Algorithm for H.264/AVC Video Encoding", ETRI Journal vol. 28, n° 6, 2006, pp. 721-731.

[107] R. Mukherjee, A. Banerjee, I. Chakrabarti, P. K. Dutta, A. K. Ray "An efficient FPGA based implementation of forward integer transform and quantization algorithm of H.264". IEEE Distributed Computing, VLSI, Electrical Circuits and Robotics (DISCOVER), August 13-14, 2016, Mangalore, India.

[108] I. J. Barge, C. Ababei, "H.264 video decoder implemented on FPGAs using 3×3 and 2×2 networks-on-chip". The International Conference on ReConFigurable Computing and FPGAs, December 6-7, 2017, Cancun, Mexico.

[109] T. C. Chen, C. Lian, and L. Chen : Hardware Architecture Design of an H.264/AVC Video Codec, Asia and South Pacific Conference on Design Automation, 24-27 January 2006, Yokohama, Japan.

[110] K. Babionitakis, G. Doumenis, G. Georgakarakos, G. Lentaris, K. Nakos, D. Reisis, I. Sifnaios, N. Vlassopoulos : A real-time H.264/AVC VLSI encoder architecture, Springer – Journal of Real-Time Image Processing, vol. 3, n° 1-2, 2008, pp. 43–59.

[111] K. Messaoudi, E. Bourennane, S. Toumi and G. Ochoa : Performance Comparison of Two Hardware Implementations of the Deblocking Filter Used in H.264 by Changing the Utilized Data Width, IEEE, The 7th International Workshop on Systems, Signal Processing and their Applications, 9-11 May 2011, Tipaza-Algeria.

[112] K. Messaoudi, E. Bourennane, S. Toumi, M. Touiza, A. Yahi : A Highly Parallel Hardware Implementation of the Deblocking Filter Used in H.264/AVC CODECs, In proceedings of the International Conference on Software Engineering and New Technologies, ICSENT'12, pp. 26-38, December 15-17, 2012, Hammamet, Tunisia.

[113] K. Messaoudi, E. B. Bourennane, S. Toumi : Reconfigurable Hardware Intelligent Memory Controller for H.264/AVC Encoders, International Journal of Computer Science and Information Security (IJCSIS), vol.8 n° 9, 2010, pp. 8-16.

[114] K. Messaoudi, M. Touiza, E.B. Bourennane, S. Toumi, Hardware/Software Co-Design with MicroBlaze Soft-Core Processor for the Integer Transform Algorithm Used in the H.264 Encoder, International Review on Computers and Software (I.Re.Co.S), vol. 5 n°. 3, 2010, pp. 348-354.

[115] G. Schelle and D. Grunwald : On chip interconnect exploration for multicore processors utilizing FPGAs, The 2nd Workshop on Architecture Research using FPGA Platforms (WARFP-2006), February 12, 2006, Austin, Texas, USA.

[116] K. Motamedi, N. Ioannides, M. H. Rümmeli, I. Schagaev : Reconfigurable Network on Chip Architecture for Aerospace Applications, 30th IFAC Workshop on Real-Time Programming and 4th International Workshop on Real-Time Software, PP. 131-136, October 12–14, 2009, Mrągowo, Poland.

[117] K.Messaoudi, H.Mayache, A.Benhaoues, E.Bourennane, S.Toumi : Connection of H.264/AVC hardware IPs using a specific Networks-on-Chip. Microprocessors and Microsystems, vol. 39, n° 8, 2015, pp. 609-620.

[118] J. Kim : Low-cost router microarchitecture for on-chip networks. The 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-42), December 12-16, 2009, New York, NY, USA.pp. 255-266.

[119] A. Kodi, A. Sarathy, and A. Louri : iDEAL. Inter-router dual-function energy and area-efficient links for network-on-chip (NoC) architectures, The 35th International Symposium on Computer Architecture (ISCA-35), June 21-25, 2008, Beijing, China.

[120] T. Moscibroda and O. Mutlu : A case for bufferless routing in on-chip networks. In Proceedings of the 36th Annual International Symposium on Computer Architecture (ISCA-36), June 20-24, 2009, Austin, USA. pp. 196-207.

[121] M. Hayenga, N. Jerger, and M. Lipasti : SCARAB: A single cycle adaptive routing and bufferless network. In proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-42), December 12-16, 2009, New York, NY, USA. pp.244-254.

[122] C. Gómez, M. E. Gómez, P. López, J. Duato : Reducing packet dropping in a bufferless NoC. In proceedings of the 14th International European Conference on Parallel Processing (Euro-Par'08), 26-29 August, 2008, Las Palmas de Gran Canaria, Spain.

# Liste des publications

# Liste des publications

Revues scientifiques :

[1] H. Mayache, S. Toumi, E. Bourennane, A. Benhaoues: « Buffer-less Network-on-Chip for video encoding applications: case study H.264/AVC ». Revue des Sciences et de la Technologie (Synthèse), de l'Université Badji- Mokhtar Annaba ; ISSN-1114-4924, 2018.

Communications internationales :

[1] H. Mayache, A. Benhaoues, E. Bourennane, K. Messaoudi, S. Toumi, « Designing Mesh 2x2 network on chip adapted for Wishbone protocol », ICESTI'12, International Conference on Embedded Systems in Telecommunications and Instrumentation, November 5–7, 2012, Annaba, Algeria.

[2] H. Mayache, E. Bourennane, A. Benhaoues, S. Toumi, K. Messaouidi, « A scalable Router and Effective XYZ Routing Algorithm », The 3rd International Conference on Information, Processing and Electrical Engineering (IEEE - ICIPEE'14), Tebessa-Algeria.

[3] H. Mayache, A. Benhaoues, E. Bourennane, K Messaoudi, S. Toumi, « Network on Chip Platform Generator », International Conférence on Embedded Systems in Télécommunications and Instrumentation (ICESTI'14), Annaba, 27-29 octobre 2014, Algérie.

Communications nationales :

[1] H.Mayache, « Mise en œuvre d'une architecture multiprocesseur autour d'un réseau sur puce (NoC) de type Wishbone. ». 1éres Journées Doctorales sur l'Automatique, les Télécommunications, l'Instrumentation et les Multimédia « JIDATIM'12 » 16-17 Janvier 2012, Annaba.

Autres publications et communications :

[1] K. Messaoudi, E. Bourennane, S. Toumi, H. Mayache, N. Messaoudi, O. Labbani : « Utilisation of the Array-OL specification language for self-generation of a memory controller especially for the H.264/AVC ». IJES 7(2):133-147 (2015)

[2] A. Benhaoues, S. Toumi, C. Tanougast, E. Bourennane, K. Messaoudi, H. Mayache : « Versatile digital architecture for mobile terminal ». Microprocessors and Microsystems - Embedded Hardware Design 39(6): 405-417 (2015)

[3] K. Messaoudi, H. Mayache, A. Benhaoues, E. Bourennane, S. Toumi : « Connection of H.264/AVC hardware IPs using a specific Networks-on-Chip ». Microprocessors and Microsystems - Embedded Hardware Design 39(8): 609-620 (2015)

[4] A. Benhaoues, H. Mayache, S. Toumi. « High Bit Rate Optical Transmission Using Midspan Spectral Inversion » revue des sciences et de la technologie (Synthèse), de l'Université Badji- Mokhtar Annaba ; ISSN-1114-4924, vol. 21, 2010.

[5] A. Benhaoues, H. Mayache, C. Tanougast, A. Dandache, S. Toumi, « Digital synthesis architecture for modulation and demodulation », The International Conference on Microelectronics, IEEE advancing Technology for Humanity, dec. 2013. Beirut, LIBAN.

[6] A. Benhaoues, H. Mayache, C. Tanougast, E. Bourennane, S. Toumi, « Universal Architecture for Demodulation and Modulation », International Conference on Embedded Systems in Telecommunications and Instrumentation, Annaba, Algeria, "ICESTI'12" November, 05-07, 2012.

[7] A. Benhaoues, S. Toumi, C.Tanougast, H.Mayache, « Architecture numérique générique et universel », 80ème Congrès de l'ACFAS, Association Francophone pour le savoir »Montréal, Canada.

A. Benhaoues, Z.Asradj, C.Tanougast, H.Mayache, S. Toumi, « Digital synthesis architecture for modulation and demodulation », International Congress on Telecommunication and Application 2014, ICTA'14. Bejaia – Algeria.

A. Benhaoues, E. Bourennane, C.Tanougast, H. Mayache, S. Toumi, K. Messaoudi, « Implementation of Universal Digital Architecture using 3D-NoC for Mobile Terminal ", The International Conference on Control, Decision and Information and Technologies, IEEE France, 2014.