République algérienne démocratique et populaire
Ministère de l'enseignement supérieur et de la recherche scientifique
Université El Haj Lakhdar BATNA
Faculté des Sciences
Département Informatique

Mémoire

pour obtenir le diplôme de

Magistère en Informatique

Option: Ingénierie des Systèmes Informatiques (ISI)

Une architecture d'un réseau sur puce (NoC) dédiée au routage de paquets

Présenté et soutenu publiquement par

Noureddine DJABALLAH

JURY

Dr. Abdelmajid ZIDANI	MCA	Université de Batna	Président
Pr. Azeddine BILAMI	Prof.	Université de Batna	Rapporteur
Dr. Brahim BELATTAR	MCA	Université de Batna	Examinateur
Dr. Mohamed Chaouki BABAHENINI	MCA	Université de Biskra	Examinateur

Année universitaire : 2010 - 2011

إِلَيكِ يَا إِبنَتي تَسنيم

A ma fille Tasnime

ملخـــص

في الشبكة على الرقاقة، يلعب التوجيه دورا هاما حيث ان خوارزمية التوجيه هي التي تقرر الطريق الذي يتوجب اتخاذه لارسال الرسالة من مصدرها الى وجهتها. في هذه المذكرة نقوم بعرض و تقييم خوارزمية جديدة تسمي OEC ($OddEvenCongestion$) مخصصة لطبولوجيات ذات شبكات بسيطة ثنائية الابعاد و التي توفر توجيها مكيفا خال من الجمود مع الاخذ بعين الاعتبار ظروف ازدحام الشبكة في المنطقة المجاورة. تكمن اهمية التوجيه المقترح في ديناميكيته التي تعتمد على مؤشر الازدحام الذي يسمح بتجاوز نقاط الاحتقان باستخدام مسارات بديلة. و قد تم تطوير بنية لتقييم اداء خوارزمية التوجيه المقترح. و لقد لوحظ ان نتائج المحاكاة للتوجيه OEC فيما يتعلق بالكمون المتوسط هي الافضل مقارنة بالتي تحصلنا عليها باستعمال خوارزمات التوجيه $OddEven$ و XY .

الكلمات الرئيسية : شبكة على الرقاقة، نظام على الرقاقة، موجه، ازدحام.

Résumé

Dans un NoC ou réseau sur puce, le routage joue un rôle très important. C'est l'algorithme de routage qui décide du chemin à emprunter pour envoyer le message de sa source à sa destination. Dans ce mémoire, nous présentons et évaluons un nouvel algorithme de routage appelé *OEC* (Odd Even Congestion) dédié aux topologies de maillage simple à deux dimensions, qui fournit un routage adaptatif et sans interblocages en tenant compte des conditions de la congestion du réseau dans la proximité. L'intérêt de notre stratégie de routage réside dans son aspect dynamique basé sur l'*Indice de Congestion* qui permet de contourner les points de congestion en utilisant des chemins alternatifs. Une architecture a été développée afin d'évaluer les performances de notre algorithme. Il a été observé que les résultats de simulation de *OEC* en terme de latence moyenne sont meilleurs que ceux obtenus par les algorithmes *OddEven* et *XY*.

Mots clés : Réseau sur puce, système sur puce, routeur, congestion.

Abstract

In a NoC or Network on Chip, routing plays an important role. A routing algorithm defines a route which message traverses to get to destination. In this paper, we present and evaluate a novel deadlock-free routing algorithm called *OEC* (Odd Even Congestion) for a two dimensional mesh topologies without virtual channels, which provides an adaptive routing based on the network's congestion conditions. The point of our dynamic routing strategy based on the turn model approach is its simplicity and inherent freedom from deadlock. It use an alternate route to bypass the congestion points. An architecture was developed to evaluate the performance of our algorithm. Experimental and simulation results indicate that the proposed OEC routing scheme provides better average latencies compared to *OddEven* and deterministic *XY* routing algorithms.

Keywords : Networks on Chip, System on Chip, router design, congestion.

Remerciements

Je remercie tout d'abord monsieur Azzeddine BILAMI, de m'avoir fait l'honneur d'encadrer mon magistère et pour ses conseils avisés durant ce travail.

Je souhaite remercier l'ensemble des membres de mon jury d'avoir accepter de juger et d'évaluer ce travail de magistère, mais aussi pour avoir passé du temps dans la lecture de ce mémoire.

J'exprime aussi mes remerciements a monsieur Samir ZIDAT et monsieur Nacereddine BENABBAS pour leur aide et pour l'intérêt qu'ils ont manifesté à l'égard de ce travail.

Je tiens à remercier tout spécialement mon cher ami Mohiedine pour le temps consacré à la lecture de ce mémoire et lui témoigner toute ma reconnaissance pour ses encouragements.

Je tiens également à remercier toute ma famille, et tout particulièrement mes parents qui m'ont toujours appuyé et soutenu dans mes études depuis mon plus jeune âge.

Enfin, une pensée très tendre pour ma femme, pour son soutien et sa patience et une immense marque d'affection pour notre petite fille Tasnime.

Table des matières

Table des figures

Glossaire

ALG : Asynchronous Latency Guarantees.

ASPIN : Asynchronous Scalable Predictable Interconnect Network.

BE : Best-Effort.

CB : Credit Based.

CHAIN : CHip Area INterconnect.

CS : Circuit Switching.

DSM : Deep Sub-Micron.

DSPIN : Distributed Scalable Predictable Interconnect Network.

FAUST : Flexible Architecture of Unified System for Telecom.

FIFO : First In First Out.

FLIT : FLow control unITes.

FSM : Finite State Machine.

GALS : Globally Asynchronous Locally Synchronous.

GS : Guaranteed Services.

IP : Intellectual Property.

LUT : LookUp Table.

MANGO : Message-passing Asynchronous Network-on-chip providing Guaranteed services over OCP interfaces.

NoC : Network on Chip.

OE : Odd Even.

OEC : Odd Even Congestion.

OCP : Open Core Protocol.

OSI : Open Systems Interconnection.

PHIT : PHysical flow control unITs.

PS : Packet Switching.

QoS : Quality of Service.

SPIN : Scalable Programmable Integrated Network.

SoC : System-on-Chip. Système sur puce ou système mono-puce en français, est un dispositif complet embarqué sur une puce, pouvant comprendre de la mémoire, un ou plusieurs microprocesseurs, des périphériques d'interface, ou tout autre composant indispensable à la réalisation de la fonction attendue.

TDM : Time Division Multiplexing.

TDMA : Time Division Multiple Access.

Time-to-market : Temps de mise sur le marché. Le temps de mise sur le marché est l'un des plus importants facteurs qui détermine le profit d'un système, qui se répercute directement sur le coût de la conception. Ce temps représente le début de la conception du produit jusqu'au moment de son lancement sur le marché.

VC : Virtual Channel.

Introduction

Avec l'évolution rapide et de plus en plus complexe de la taille des SoCs (System On Chip) et la multiplication des éléments présents sur une même puce, les besoins de communication et d'interconnexion des modules IPs (Intellectuel Property) à l'intérieur d'une puce deviennent de plus en plus importants et constituent une partie fondamentale lors de la conception de tels systèmes.

Plusieurs facteurs comme la réduction des temps de mise sur le marché (Time to market), les besoins de flexibilité et de la bande passante ont nécessité l'adoption d'une méthodologie de conception adéquate.

Le concept émergent de Noc (Network on Chip) ou réseau sur puce à été adopté par les chercheurs pour répondre à la complexité croissante et aux exigences de communication des futurs systèmes sur puce. Ces réseaux sont conçus à l'image des réseaux informatiques classiques, avec néanmoins une certaine spécificité, qui découle de l'environnement de ces réseaux. Il est clair que le paradigme NoC permet de construire des architectures flexibles et extensibles de systèmes sur puce et offre, par la suite une solution performante et économique.

Dans les NoCs, les algorithmes de routage sont utilisés pour déterminer le chemin que doit suivre un paquet pour atteindre sa destination. Les algorithmes de routage peuvent être généralement classés comme déterministes et adaptatifs. Même si les routages déterministes sont plus simple à implémenter, ils restent un choix de routage non approprié au sein des réseaux dans lesquels le trafic est irrégulier avec des risques de congestion. A l'inverse, les routages adaptatifs présentent une solution bien adaptée aux phénomènes de congestion dans les NoCs.

C'est dans cet esprit que le travail réalisé dans ce mémoire propose un algorithme de routage appelé OEC (Odd Even Congestion), qui fournit un routage adaptatif et sans interblocages en tenant compte de l'encombrement local du routeur dans une topologie de maillage simple à deux dimensions.

La finalité de cette étude est d'apporter une solution dynamique et flexible aux phénomènes de congestion dans les NoCs en utilisant des chemins alternatifs.

Le travail présenté dans ce mémoire de magistère a été divisé en quatre chapitres.

Le premier chapitre présente un état de l'art des interconnexions. Il s'intéresse en particulier aux éléments constitutifs de l'architecture d'un NoC ainsi qu'aux protocoles de communication nécessaires à la transmission de l'information au sein des NoCs.

Le deuxième chapitre présente les différents algorithmes de routages dans les NoCs, les techniques de gestion des flux et quelques exemples de NoCs existants.

Le troisième chapitre présente la description détaillée de notre approche.

Le quatrième, et dernier chapitre décrit la partie implémentation et présente les différents résultats de simulation obtenus.

Enfin, une conclusion récapitule l'intérêt de ce mémoire et présente les futurs améliorations possibles de notre travail.

1 Network on Chip : état de l'art

1.1 Les systemes monopuce (SoC)

Durant les deux dernières décennies, on est passé de la conception de circuits composés de quelques milliers de portes a des systèmes mono puce structurées (SoC [1]) intégrant un nombre croissant d'unités de traitement, matérielles et logicielles, des interfaces d'entrée et de sortie, des mémoires pour répondre aux exigences de nouvelles applications émergentes.

Cette évolution entraîne une augmentation des besoins de communication à l'intérieur de la puce (intra-chip) pour les échanges de données et le contrôle des traitements entre les différentes unités ou cœurs (IP [2]) qui composent le système mono puce.
Par conséquent, un des problèmes principaux que rencontrent les concepteurs de SoC, est la réalisation des structures d'interconnections permettant aux IPs de communiquer entre eux.

Ces nouvelles structures de communication devront répondre aux exigences de :

- ***Flexibilité :*** en offrant la possibilité d'interconnecter de nombreux IPs hétérogènes grâce à un interfaçage adéquat (interface based design [RS97]).
- ***Extensibilité :*** consiste à pouvoir augmenter le nombre d'IPs interconnectées tout en conservant le même niveau de performance dans les communications.
- ***Implantation distribué :*** permettant de faire cohabiter plusieurs domaines d'horloge des différents modules. On s'oriente vers une implantation globalement asynchrone et localement synchrones (GALS [MHK99]).
- ***Robustesse :*** en évitant lors de la conception les erreurs de transmissions générés par les bruits sur les interconnections liés aux effets submicronique profonds (DSM [3]) dans les technologies en dessous de 90 nm.

D'autre part, elles doivent aussi offrir une bande passante assez conséquente.

1.2 Les différentes structures d'interconnexions

Dans cette section, nous allons exposer les divers standards d'interconnexions qui sont utilisés pour assurer les communications entre les différents blocs fonctionnels d'un SoC. Les deux principales techniques employées la plupart du temps par les concepteurs sont les connexions directes (point à point) et les connexions bus.

1.2.1 La connexion point à point

La connexion point à point (figure 1) est considérée comme la connexion la plus simple qui soit pour connecter deux IPs. L'avantage de cette connexion est qu'elle n'a pas besoin d'avoir un arbitrage pour l'accès au médium de communication. Cependant, elle impose beaucoup de contraintes, notamment concernant les interfaces d'entrées/sorties pour l'intégration de nouveaux blocs fonctionnels (problème de flexibilité). De plus, cette technique ne fournit aucune tolérance aux fautes et pannes pouvant survenir sur un lien ou un bloc fonctionnel et qui rendent le SoC inutilisable[].

1. System on Chip
2. Intellectual Property
3. Deep Sub-Micron

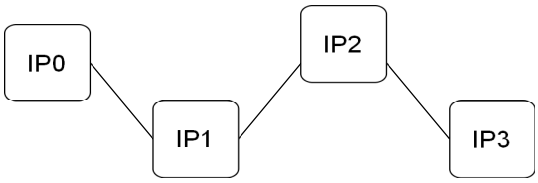

FIG. 1 – La connexion point à point

1.2.2 La connexion complète

Quand toutes le IPs sont reliées les uns aux autres par des connexions point à point (figure 2), la connexion est dite complète. Cette structure tolère mieux les pannes, mais limite énormément l'extensibilité surtout lorsque le nombre d'unités fonctionnelles à interconnecter est important. De plus le taux d'utilisation moyen des liens est globalement assez faible.

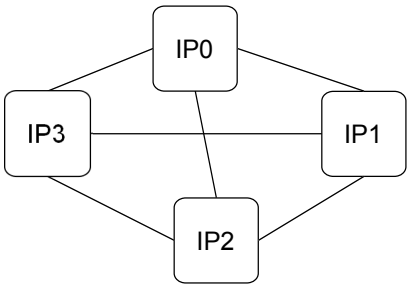

FIG. 2 – La connexion complète

1.2.3 Le bus standard

Un bus standard ou partagé est un ensemble de connexions autour duquel communiquent plusieurs cœurs ou IPs [Rab00]. Il est aujourd'hui le moyen de communication le plus répandu dans l'industrie. Comme l'illustre la figure 3, un bus standard ou partagé est composé de lignes transportant les signaux (données et contrôles) entre une source et une cible et d'un arbitre pour designer l'utilisateur des services du bus à un instant donné.
Le bus est beaucoup plus flexible et mieux réutilisable, comparé aux liaisons point à point. En effet, un cœur peut être ajouté, déplacé ou supprimé facilement.
Cependant, il ne permet qu'une seule communication à la fois entre les IPs raccordés. La bande passante allouée à chaque IP diminue donc avec le nombre d'IPs communicants.
De plus, le mécanisme d'arbitrage dans le bus implique un délai non négligeable pour traiter les demandes d'accès concurrentes.
Enfin, du fait de la longueur des fils et de nombre de cœurs connectés, la consommation d'énergie devient plus importante et les délais de transmission peuvent alors dépasser la période d'horloge du système [Lem06].

1.2.4 Le bus hiérarchique

Afin de surmonter les inconvénients du bus standard énoncés précédemment, la structure de bus hiérarchique a été introduite. Le concept de cette interconnexion consiste à connecter plusieurs bus standards à l'aide de ponts (figure 4).

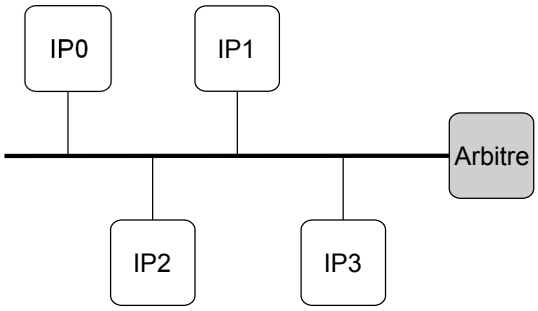

FIG. 3 – La connexion du bus standard

Les différents segments du bus hiérarchique possèdent des liaisons courtes avec peu d'IPs connectées, offrant une faible capacité sur chacun de ces segments et donc une faible consommation.
Cependant, l'accès d'un bus à un autre au travers d'un pont, implique un coût en latence d'où l'importance de bien répartir les différentes IPs communicantes pour limiter l'utilisation du pont et ainsi favoriser le fonctionnement parallèle des bus.

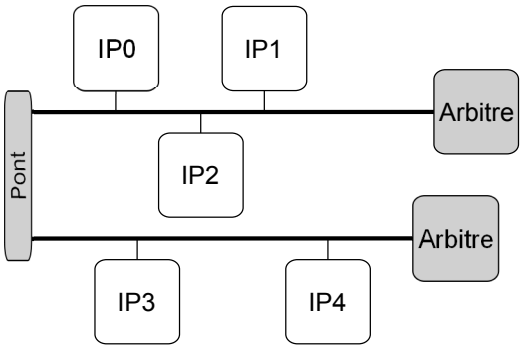

FIG. 4 – La connexion du bus hiérarchique

1.2.5 Le crossbar

Dans cette interconnexion, aussi appelée commutateur de croisement, les IPs d'entrée sont connectées directement aux IPs de sortie sans étage intermédiaire (figure 5).
Les crossbars consomment généralement moins d'énergie et offrent des débits et un parallélisme supérieurs à ceux des bus traditionnels. Par contre, la complexité de câblage d'une telle architecture est très grande. Elle augmente avec le carré du nombre d'IPs communicantes.

1.2.6 Le réseau

Les limites intrinsèques (débit maximal, scalabilité) des bus ont conduit, depuis quelques années, à étudier de nouvelles architectures de réseaux intégrés, donnant ainsi naissance vers le début des années 2000 au concept de réseau sur puce ou NoC[4].
Ce nouveau type d'interconnexion a été développé pour pouvoir connecter efficacement les différents IPs d'un système monopuce (SoC). I'idée consiste à adapter les solutions des réseaux locaux et des réseaux d'interconnexions de machines parallèles au contexte des SoCs, tout en respectant, lors de la

4. Network on Chip

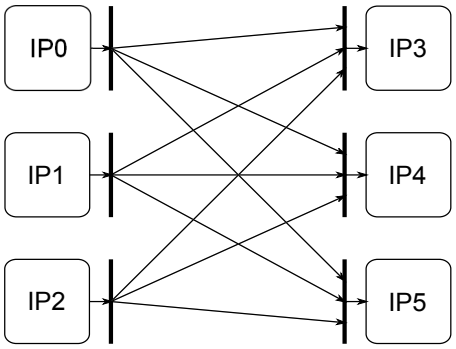

FIG. 5 – La connexion crossbar

phase de conception, les nombreuses contraintes spécifiques aux circuits intégrés comme la latence, la surface du silicium et la consommation d'énergie.

1.3 Les caractéristiques des NoCs

Dans cette section, nous allons décrire les principales caractéristiques d'un NoC, à savoir les composants constitutifs de ce type d'interconnexion, les architectures ou topologies utilisées qui définissent comment sont interconnectés les différents IPs et les protocoles de transfert des données. Les problèmes d'interblocage dans les NoCs sont aussi présentés.

1.3.1 Les composants de base d'un NoC

Comme l'illustre la figure 6, un réseau sur puce est constitué de trois types de composants. Les interfaces réseau sont les éléments qui permettent d'accéder au réseau sur puce. Les nœuds de routage sont chargés de diriger les données qu'ils reçoivent vers leurs destinataires. Les liens de communication assurent le transfert des données entre deux composants du réseau sur puce (interfaces réseau ou nœuds de routage).

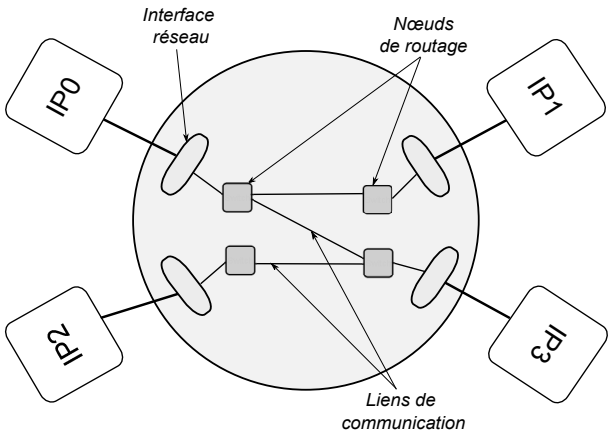

FIG. 6 – Structure réseau sur puce

1.3.1.1 Les interfaces réseau Les interfaces réseau sont les points d'entrée-sortie d'un réseau sur puce. Une interface réseau fournit un ensemble de services de communication de haut niveau (e.g. paquétisation, contrôle de flux, ordonnancement des communications, . . .) aux composants qui lui sont

connectés. Elle se sert des primitives de communication de base du réseau sur puce pour répondre aux appels à ses services de communication.
Comme l'illustre la figure 7, une interface réseau est divisée en deux parties : une partie frontale (front end) qui est reliée à un composant fonctionnel du circuit, une partie arrière (back end) qui est reliée au réseau sur puce.

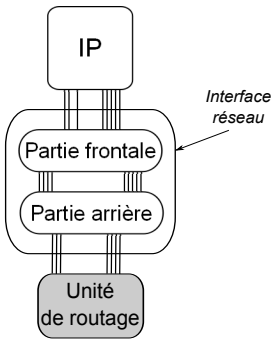

FIG. 7 – Architecture d'une interface réseau

Pour faciliter la réutilisation des composants fonctionnels, la partie frontale d'une interface réseau implémente généralement un protocole de communication point à point standardisé. Les principaux standards actuels sont le VCI (Virtual Component Interface), l'OCP (Open Core Protocol), l'AXI (Advanced eXtensible Interface) et le DTL (Device Transaction Level). Les services offerts par l'interface peuvent etre classés en trois catégories: les services d'adaptation, les services réseaux et les services fonctionnels.

Services d'adaptation L'adaptation n'est pas uniquement la mise en forme des données issues de l'unité de traitement dans un format compatible avec les modes de transfert du réseau (et inversement)[Lem06]. Il s'agit aussi d'adapter les débits et les latences entre les flux de l'unité de calcul et ceux du réseau. Enfin, dans un système contenant plusieurs horloges, l'adaptateur permet d'émettre et de recevoir des paquets provenant de nœuds de calculs ayant des fréquences d'horloge différentes.

Services réseaux Dans la partie interface réseau [Lem06], l'adaptateur gère la construction et l'ordonnancement des envois des paquets, la fiabilisation des transactions à travers les contrôles de flux et les détections/corrections d'erreurs, l'adressage des paquets et enfin l'établissement ou la fin d'une connexion pour des circuits commutés.

Services fonctionnels Outre les services d'interfaçage entre le réseau et les éléments de traitement, l'adaptateur peut fournir des services fonctionnels liés au contrôle et à la configuration des unités de calcul[Lem06].

1.3.1.2 Les nœuds de routage

Le rôle d'un nœud de routage est d'aiguiller les données qu'il reçoit vers leurs destinataires[DSM10]. La figure 8 illustre l'architecture générique d'un nœud de routage qui est constituée des éléments suivants :

Les files d'attente qui sont utilisées pour stocker les données en transit. Elles permettent aussi de lisser le trafic sans bloquer les émetteurs (tant que les files ne sont pas pleines).

Le commutateur ou matrice de commutation qui connecte les files d'attente d'entrée aux files d'attente de sortie. Un crossbar ou plusieurs multiplexeurs en cascade sont utilisés afin de multiplexer les ports d'entrées vers les ports de sortie.

L'unité de routage et d'arbitrage qui définit comment doit être configuré le commutateur pour que les données contenues dans les tampons d'entrée soient correctement aiguillées. De plus, il doit gérer les conflits d'accès lorsque plusieurs files d'attente d'entrée doivent être connectées à la même file de sortie.

1.3.1.3 Les liens Le rôle principal d'un lien de communication est de transférer des données entre deux composants d'un réseau sur puce. Avec l'avancement des progrès technologiques, cette tâche est de plus en plus difficile à accomplir sans erreur. En effet, les fils utilisés pour réaliser les liens de communication d'un réseau sur puce sont de plus en plus fins alors que les distances qu'ils doivent parcourir ne diminuent pas. Ces fils sont donc de plus en plus sensibles aux phénomènes de bruits et il est donc difficile de garantir l'intégrité des valeurs qu'ils communiquent.
Une solution permettant de résoudre ce problème est d'utiliser des répéteurs, le plus souvent des buffers, pour segmenter les fils ayant de longues distances à parcourir. Une solution complémentaire est d'utiliser des codes correcteurs et/ou détecteurs d'erreurs.

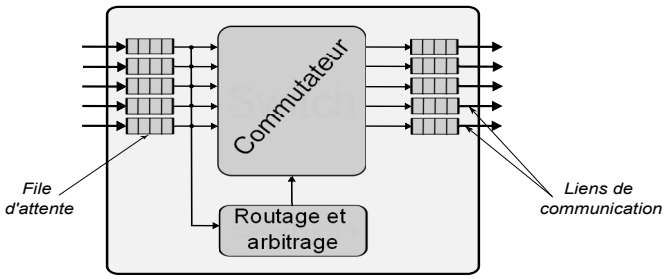

FIG. 8 – Architecture générique d'un nœud de routage

1.3.2 Les couches réseau dans les NoCs

Le modèle OSI[5] décrit le protocole de communication d'un réseau sous forme de sept parties (physique, liaison de données, réseau, transport, session, présentation et application) appelées couches OSI. Or, cette représentation OSI n'est pas adaptée à la description d'un NoC car celui-ci n'intègre pas toutes ces couches, seules les quatre premières couches sont implantées.

Couche physique (Physical layer) Défini la structure physique et les protocoles nécessaires (nature d'interconnexion et largeur en nombre de bits des mots de données) pour établir la communication au niveau des liens entre les différents routeurs.

5. Open Systems Interconnection

Couche de liaison ou commutation (Link or Switching layer) Utilise la couche physique pour implémenter le mécanisme de transmission de données à travers le réseau (échange de données entre les différents routeurs avec détection et correction d'erreur de transmission).

Couche réseau ou routage (Network or Routing layer) Elle détermine la technique de routage et d'arbitrage utilisée pour faire propager les messages dans le réseau.

Couche transport (Transport layer) Cette couche est chargée de la transformation des messages en paquets, et inversement. Elle assure le contrôle de flux de bout en bout afin d'éviter que des données ne soient envoyées vers une destination qui n'a plus de place disponible pour les recevoir. De plus elle gère l'adaptation de protocole entre le NoC et les IPs via l'adaptateur réseau.

Dans les NoCs, les couches ISO concernant l'application peuvent être regroupées en une seule couche prise en charge par les IPs communicantes dans le réseau.

1.3.3 Les topologies

La topologie d'un réseau sur puce définit comment sont interconnectés ses différents composants. Elle est souvent modélisée sous la forme d'un graphe dans lequel Les nœuds de routage et les liens de communication sont respectivement associés aux sommets et aux arrêtes de ce graphe. Cette modélisation permet de comparer les topologies entre elles en utilisant les propriétés suivantes des graphes :

- ***Le diamètre :*** il est égal à la distance (en nombre de liens de communication) maximale qui sépare deux nœuds de routage du réseau. Les performances du réseau, en termes de débit et de latence, sont liées aux longueurs des plus courts chemins entre les sommets du graphe. C'est pourquoi le diamètre est un indicateur très regardé.
- ***Le degré d'un nœud de routage :*** il est égal aux nombre de liens de communication qui lui sont connectés.
- ***La largeur de bissection :*** il est égale au nombre minimal de liens de communication nécessaires pour séparer le réseau en deux ensembles disjoints de même taille (en nombre de nœuds de routage). Cela permet d'évaluer le coût de transfert des données d'un ensemble à l'autre.
- ***La régularité :*** elle est respectée lorsque tous les nœuds de routage ont le même degré. En effet, la topologie est dite régulière, lorsque tous les sommets du graphe du réseau ont exactement le même nombre d'arrêtes les reliant aux sommets adjacents. Dans le cas contraire, la topologie est irrégulière[DT01].

Dans la littérature, les topologies sont généralement réparties en trois classes :

- ***Directes :*** chaque routeur est connecté à au moins un élément communicant(IP).
- ***Indirects :*** les routeurs peuvent être connectés seulement à d'autres routeurs.
- ***Hybrides :*** l'interconnexion des routeurs est une combinaison des réseaux directes et indirects.

Plusieurs topologies existent dans la littérature afin d'offrir le meilleur compromis possible entre les performances requises par l'application et le coût matériel engendré par le réseau lui-même. Nous avons choisi de présenter les plus couramment utilisées.

1.3.3.1 La topologie de maillage à deux dimensions La topologie la plus souvent employée est celle de structure 2D régulière représentée sur le figure 9. Elle est simple de mise en œuvre et facilement implantable sur une technologie silicium car tous les liens ont la même longueur[RAR09]. Dans cette topologie, chaque sommet (un nœuds de routage) est connecté au moins à deux autres sommets. L'adresse d'un sommet dans un tel réseau est définie par l'ensemble des coordonnées dans

chaque dimension. Cependant, ce type de réseau possède une distance moyenne relativement grande et une bande passante de bissection limitée, ce qui réduit les performances lorsque le réseau est très chargé. Parmi les réseaux développés qui utilisent cette topologie, on trouve : HERMES [MCM04], NOSTRUM [NO06] et FAUST[6][Gro07].

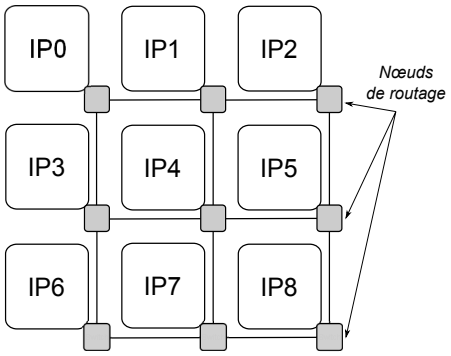

FIG. 9 – La topologie de maillage à deux dimensions (2D mesh)

1.3.3.2 La topologie Tore C'est une topologie dérivée de la structure 2D possédant la particularité d'un repliement des bords extérieurs sur eux-mêmes (Figure 10). Elle offre ainsi une bande passante légèrement supérieure à celle de la 2D maillée mais elle est plus complexe à implémenter et très difficile à tester.

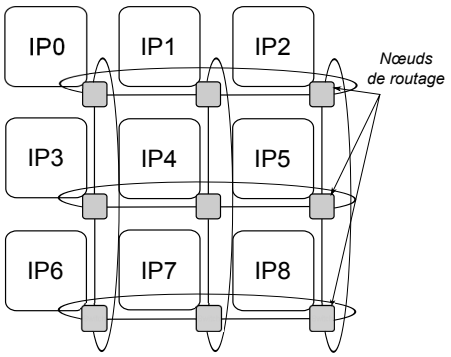

FIG. 10 – La connexion tore (torus)

1.3.3.3 La topologie anneau Une topologie en anneau ressemble à une topologie en bus, sauf qu'elle n'a pas de fin ni de début, elle forme une boucle. Cette architecture peut être très efficace mais dans certain cas d'utilisation, elle offre des performances inférieures à celles d'un Mesh [NW06]. La topologie en anneau est une structure facilement intégrable sur silicium (Figure 11). Cependant, elle n'est pas extensible car ses performances se dégradent (la bande passante devient limitée) au fur et à mesure que le nombre d'IPs connectées augmente.

6. Flexible Architecture of Unified System for Telecom

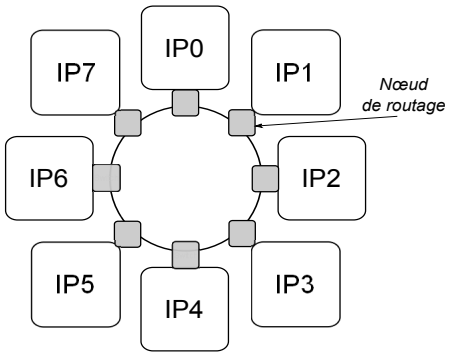

FIG. 11 – La topologie anneau (ring)

1.3.3.4 La topologie anneau à cordes La topologie anneau à cordes (Figure 12) a été introduite pour remédier aux problèmes de performance de la connexion anneau. Elle permet de réduire la latence en offrant la garantie de ne traverser au maximum que deux routeurs pour n'importe quelle communication au sein du réseau.

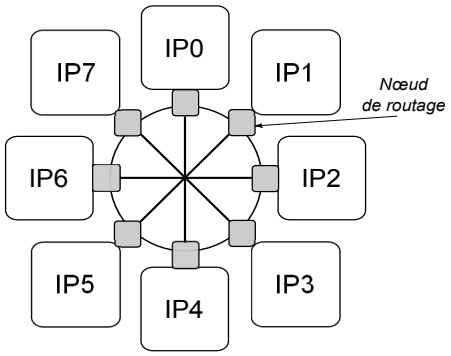

FIG. 12 – La topologie anneau à cordes (chordal ring)

1.3.3.5 La topologie papillon La figure 13 montre un réseau de papillon 8x8 sommets dans lequel les données transitent, à partir des IPs d'entrée sur la gauche, à travers trois étages de nœuds de routage pour atteindre les IPs de sortie sur la droite [HH10].

1.3.3.6 La topologie Benes Dans l'architecture Benes, toutes les IPs sont classés en deux types en fonction de leurs rôles et positions [ZG09]. Tous les éléments entre les IPs d'entrée et ceux de sortie dans le réseau sont appelés nœuds de routage ou de commutation. Chaque nœud de routage est adressé par une suite d'entiers (*position*, *étage*), où *position* représente l'endroit dans l'étage et *étage* désigne le numéro de l'étage dans le réseau comme le montre la figure 14.

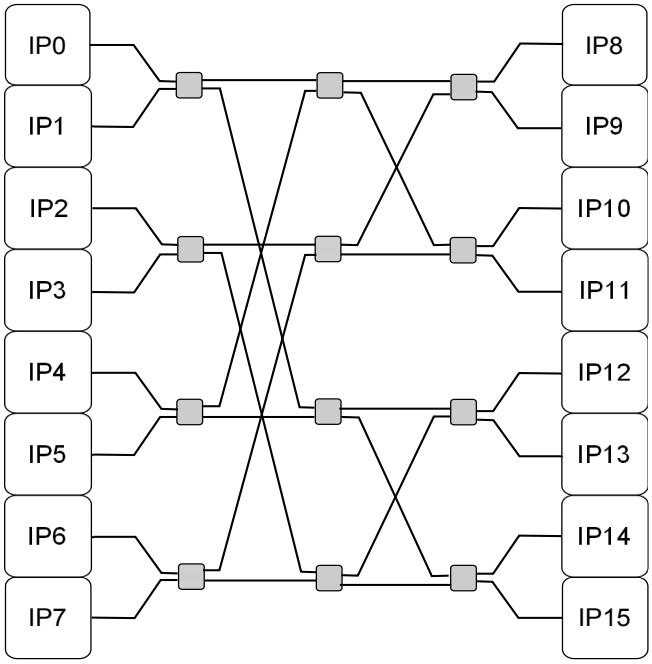

FIG. 13 – La topologie papillon (butterfly)

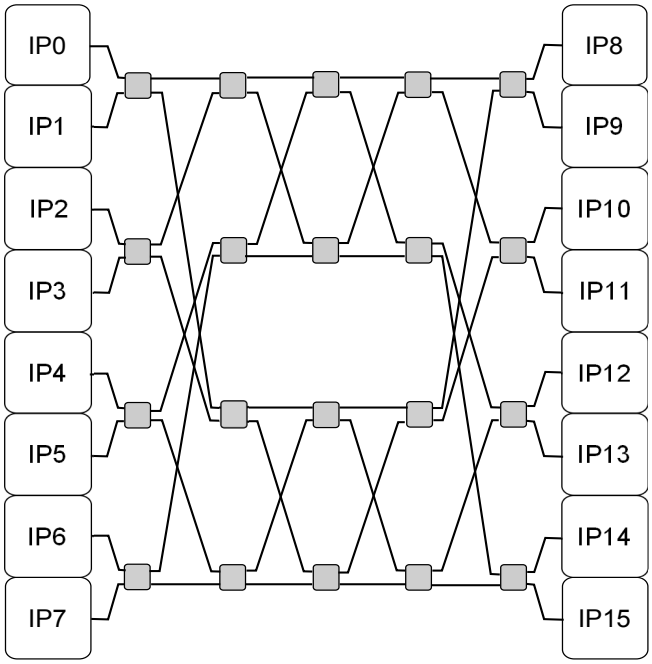

FIG. 14 – La topologie benes

1.3.4 La notion de paquet

Les communication entres les différents IPs au sein d'un réseau sont gérées par la propagation des messages. Afin d'être échangés efficacement, les messages sont souvent divisés en un ou plusieurs paquets (figure 15) avant d'être transmis. Un paquet est la plus petite unité de communication qui contient :

- ***l'en-tête "header":*** qui véhicule l'information nécessaire au routage et au séquençage du paquet.
- ***le corps "payload":*** qui indique la charge utile d'un paquet (contient les données transportées).
- ***la queue "tail":*** qui désigne la fin du paquet.

Le paquet est composé de flits[7]. Le flit représente la plus petite quantité d'information utilisée par la plupart des mécanismes de contrôle de flux. La position du flit dans un paquet détermine s'il s'agit d'un flit en-tête (head flit), d'un flit de corps de paquet (body flit ou payload flit) ou un flit de queue de paquet (tail flit). En fonction de la taille de flit et de la largeur des liens, plusieurs cycles peuvent être nécessaires pour transmettre un flit (typiquement le nombre de bits d'un flit correspond au nombre de fils dans un lien). A l'instar du paquet, le flit est formés d'un ou de plusieurs phits[8]. Un phit est l'unité d'information qui est transférée à travers les liens durant un cycle d'horloge.

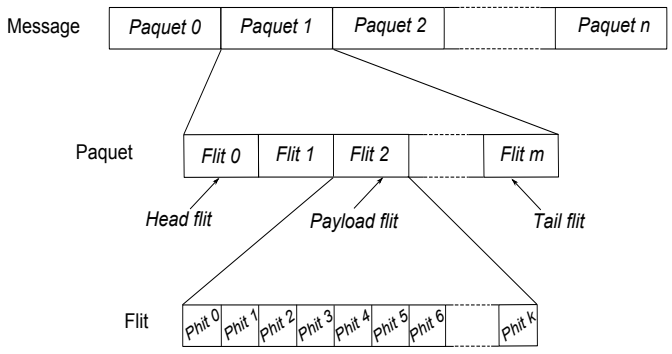

FIG. 15 – La structure du paquet

1.3.5 Les techniques de commutation

Les techniques de commutation définissent comment sont utilisées les différentes ressources d'un réseau sur puce pour acheminer des données. Les deux techniques de commutation suivantes sont principalement utilisées par les réseaux sur puce.

1.3.5.1 La commutation de circuit [9]

Cette technique de commutation consiste, dans un premier temps, à réserver les ressources nécessaires du réseau sur puce pour établir une connexion logique entre l'émetteur et le récepteur pour toute la durée de la communication. Cela signifie que des éléments du réseau (liens, nœuds) sont alloués pour le transfert de données et ils ne peuvent pas être utilisés pour des communications entre d'autres couples d'émetteur/récepteur tant que la connexion est active. Puis dans un deuxième temps à utiliser cette connexion logique pour transférer les données depuis l'émetteur vers le récepteur. Une fois le transfert terminé, la connexion réservée est alors désactivée.

7. FLow control unITes
8. PHysical flow control unITs
9. en anglais, Circuit Switching

Le principal avantage de ce mécanisme de réservation est d'assurer que deux flots de communication ne rentreront jamais en conflit une fois leurs connexions logiques établies. IL est bien adapté lorsque l'on veut effectuer des transferts de données importants, ce qui rentabilise le temps passé à négocier la connexion. Ce type de commutation permet donc à un réseau sur puce de facilement offrir des garanties de service sur les latences et les débits. L'inconvénient de cette méthode de commutation est d'avoir une latence importante à cause du temps nécessaire à l'établissement de la connexion.

1.3.5.2 La commutation de paquet [10]

Le principe de cette technique de commutation est de découper un message en un ensemble de paquets indépendants qui voyagent séparément dans le réseau. Pour que chaque paquet puisse atteindre sa destination, des informations, dites de routage, leurs sont ajoutées. Ces informations de routage sont utilisées par les nœuds de routage pour aiguiller les paquets qu'ils reçoivent vers leurs destinations. Contrairement à la commutation de circuits, les ressources nécessaires à l'acheminement d'un paquet ne sont pas réservées avec la commutation de paquets, donc il n'y a pas besoin d'établir une connexion au préalable. Les paquets ainsi émis peuvent suivre des chemins différents et sont ré-assemblés à l'arrivée.

Il peut donc survenir que plusieurs paquets transitant par le même nœud de routage veuillent être aiguillés vers le même lien de communication. Ce type de conflit d'accès à une ressource partagée est appelé contention. Pour résoudre cette situation, les nœuds de routage emploient des composants appelés arbitres qui déterminent quel paquet peut emprunter le lien de communication. Il peut aussi arriver que des paquets se perdent, auquel cas ils devront être retransmis.

Par conséquent, les délais de communications sont non-déterministes d'où la nécessité d'avoir des mécanismes de contrôle de flux et de congestion pour cette technique de commutation.

1.3.6 Les modes de commutation de paquets

Il existe différents modes de commutation de paquets qui définissent comment sont transférés les paquets entre deux routeurs via un lien de communication. La commutation de paquets présente principalement trois principaux modes de commutation (ou politiques de mémorisation des paquets au sein du réseau) [RGR03] :

différé "Store-And-Forward" (SAF), passage à travers "Cut-Through" (CT) et trou de ver "Worm-Hole" (WH).

1.3.6.1 Store and forward Dans ce mode de commutation, les nœuds de routage vont stocker totalement les paquets avant de les renvoyer, comme le montre la figure 16. Cela implique que chaque routeur doit être en mesure de stocker la totalité d'un paquet de données[Mur09].
Comme les ressources de mémorisation sont très coûteuses en termes de surface et de consommation, la taille de paquet est donc limitée. De plus, le délai des paquets augmente à chaque routeur car tous les flits du paquet doivent être reçus par le routeur actuel avant d'être envoyé au suivant. La latence totale est donc la multiplication du temps de transfert d'un paquet entre deux routeurs par le nombre de routeurs du réseau traversés par le paquet de sa source à sa destination.

1.3.6.2 Virtual Cut Through Ce mode de commutation [Mur09] a été proposé afin de réduire la latence des paquets à chaque étape de routage. Dans ce mode de commutation, un paquet peut commencer à être transféré au routeur suivant avant la réception complète de ce paquet par le routeur

10. en anglais, Packet Switching

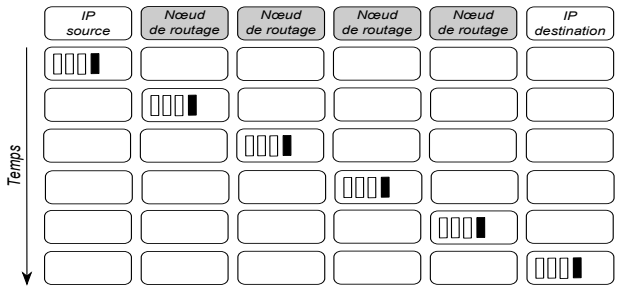

FIG. 16 – Commutation Store And Forward (SAF)

actuel, comme illustré dans la figure 17.
Au lieu d'attendre que tout le paquet arrive dans le routeur avant d'examiner l'en-tête pour déterminer les paramètres de routage, celui-ci est examiné dès son arrivée dans le routeur. Du coup il ne faudrait même pas sauvegarder le paquet qui peut être directement transmis au routeur suivant s'il garantit le stockage du paquet dans sa totalité. Le routeur doit pouvoir garder le paquet (dans sa totalité) si le routeur suivant n'est pas disponible. La capacité de mémorisation du routeur est donc la même que pour le mode SAF mais la latence est diminuée, en absence de contention, puisqu'il n'est plus nécessaire d'attendre la réception complète du paquet pour qu'il passe d'un routeur à l'autre.

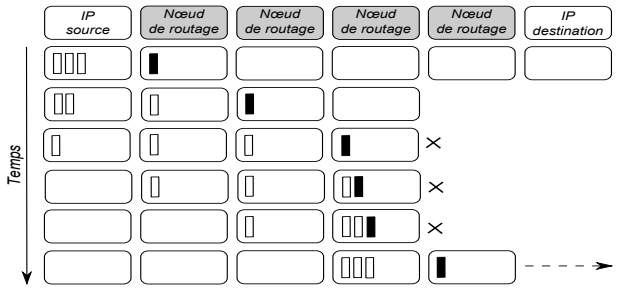

FIG. 17 – Commutation Virtual Cut Through (VCT)

1.3.6.3 Wormhole Le mode de commutation wormhole [Fle02] ou trou de ver est basée sur le principe que les flits d'un paquet peuvent être transmis d'un routeur à l'autre dès qu'il y a de la place pour un flit et plus nécessairement pour un paquet complet (figure 18).
Plus précisément, dans ce mode de commutation, on considère qu'un paquet est composé de flits et que seul un petit nombre de flits peut être stocké dans chaque routeur.
l'en-tête du paquet (i.e., le flit de tête) contient la destination. Les paquets progressent flit par flit dans le réseau. Dès que l'en-tête d'un paquet arrive dans un routeur, ce dernier le traite et détermine le canal de sortie que doit emprunter le paquet. Si ce canal est disponible, le flit d'en-tête est retransmis directement sur ce dernier, le reste des flits le suivent à la queue.

Les principaux avantages du mode de commutation de trou de ver sont d'avoir une faible latence en moyenne et surtout de permettre d'utiliser des files d'attentes de faible taille. Cette dernière propriété est la principale raison pour laquelle ce mode de commutation de paquets est utilisé. En effet, la surface occupée par les points de mémorisation est une des contraintes les plus fortes des réseaux sur puce.
Cependant, il peut arriver qu'un paquet occupe plusieurs routeurs en même temps, et dans ce cas, il est possible qu'il bloque la transmission d'autres paquets, aboutissant à une contention en cascade,

voire même à une situation de blocage cyclique appelée interblocage (cf. Les différents algorithmes de routages dans les NoCs).

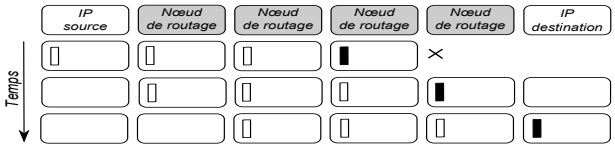

FIG. 18 – Commutation WormHole (WH)

2 Les différents algorithmes de routages dans les NoCs

Le mécanisme de routage au sein du NoC est responsable de la transmission correcte et efficace des paquets à travers le réseau. L'algorithme de routage s'occupe de résoudre le problème du choix de routage à prendre au niveau de chaque routeur. Son rôle est donc de définir le chemin (i.e. la suite des liens de communication à utiliser) que doit suivre un paquet pour atteindre sa destination. En fonction des caractéristiques d'un NoC et de ses besoins, plusieurs types d'algorithmes de routage peuvent être utilisés.

2.1 Classification des algorithmes de routage

Alors que les techniques de commutation spécifient comment sont stockées et transportées les données entre les routeurs, les techniques de routage déterminent les chemins que les données vont emprunter pour atteindre leurs destinations.

De nombreuses techniques de routage ont été proposées dans la littérature. Elles sont classées selon différents critères comme le montre la figure 19.

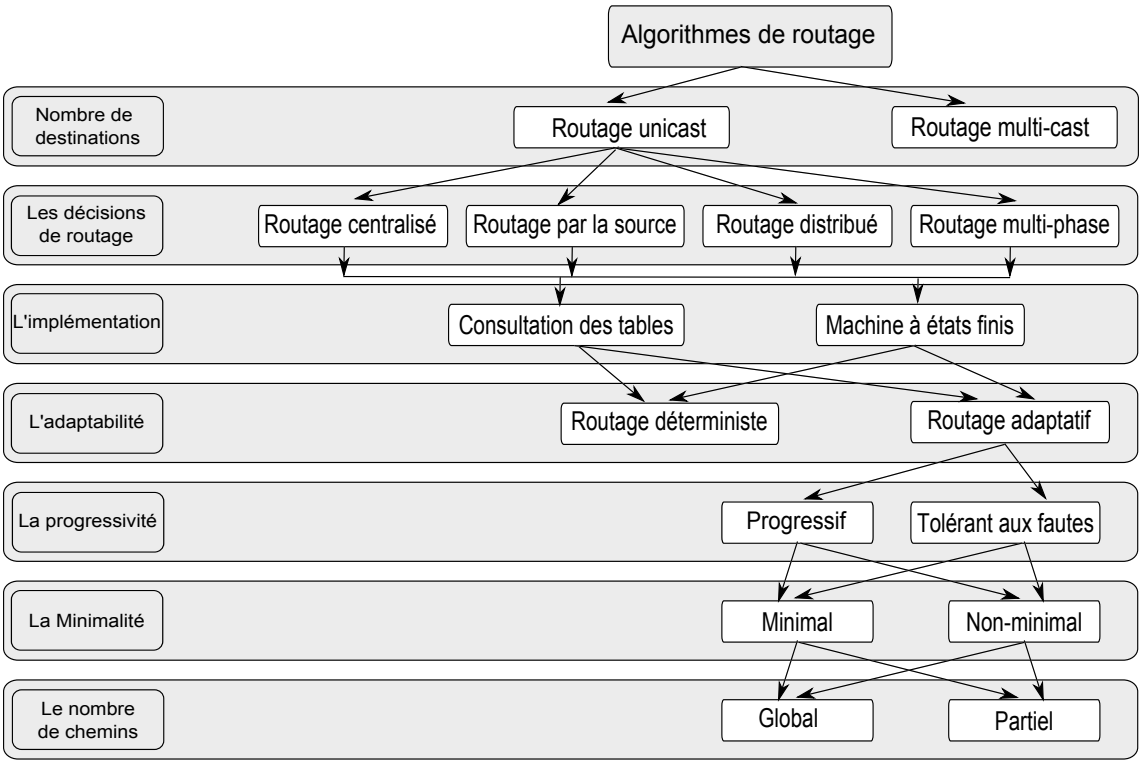

FIG. 19 – Les algorithmes de routage

2.1.1 Le nombre de destinations

Selon ce critère, les algorithmes de routage peuvent être divisés en deux classes : le routage unicast et le routage multicast. Dans le routage unicast, les paquets d'un même message ont une destination unique, tandis que dans le cas du routage multicast les paquets ont des destinations multiples.

Pour Les communications intra-puce, la technique de routage unicast semble être une bonne approche pratique en raison de la présence des interconnexions point à point entre les différentes composantes d'une puce [AIS09].

2.1.2 Les décisions de routage

Basé sur les décisions du routage, les algorithmes unicast peuvent être classés selon l'endroit où la décision est prise en quatre catégories : le routage centralisé, le routage par la source, le routage distribué et le routage multi-phase.

Dans un routage centralisé, c'est un nœud qui possède toutes les informations sur l'état du réseau. Ce nœud est donc en mesure de calculer à chaque instant le chemin optimal entre deux nœuds. Ainsi tout nœud source désirant établir une connexion doit s'adresser au nœud "principal", ce qui augmente le temps pour calculer une route. De plus, il existe un problème de fiabilité important. En effet, si ce nœud de routage venait à être hors service, ou si un des liens le reliant au reste du réseau était coupé, il y aurait alors un impact sur le bon fonctionnement du réseau.

Avec un routage par la source, la suite des liens de communication à utiliser est indiquée dans les informations de routage de chaque paquet. Le rôle d'un nœud de routage se limite donc à extraire et à appliquer les informations de routage contenues dans un paquet.

Au contraire, avec un routage distribué, un paquet ne contient que l'adresse de sa destination. Un nœud de routage doit donc calculer pour chaque paquet vers quel lien de communication il doit l'aiguiller.

La combinaison des algorithmes de routage par la source et la destination nous donne un nouveau routage hybride appelé routage multi-phase.

2.1.3 L'implémentation

Les algorithmes de routage peuvent également être classés en fonction de leur mise en œuvre ou implémentation : la consultation des tables (LUT [11]) et Machine à états finis ou FSM [12] (cf. Annexe A). Les algorithmes de LUT sont plus populaires dans la mise en œuvre. Leurs implémentation logicielle consiste à stocker une table de consultation dans chaque nœud. Nous pouvons ainsi changer l'algorithme de routage par le remplacement des entrées de la table de consultation. Les algorithmes de routage basés sur une FSM peuvent être implémentés soit en logiciel ou en matériel [AIS09].

2.1.4 L'adaptabilité

Dans une stratégie de routage déterministe, encore appelée statique, le chemin des paquets est déterminé à partir des adresses de sa source et de sa destination [DA93]. L'avantage des algorithmes statiques réside dans leur simplicité d'implémentation. Cependant le manque de flexibilité dans l'attribution des routes les rend tributaires d'une sensibilité accrue aux phénomènes de congestion. Les approches déterministes fournissent toujours le même chemin entre une paire source / destination et elles constituent le meilleur choix pour des profils de trafic uniformes ou réguliers.

A l'inverse, dans une stratégie de routage adaptatif, le chemin de routage est décidé dans les routeurs avant chaque saut des paquets et implique des mécanismes dynamiques d'arbitrage (par exemple basé sur l'encombrement local de lien) [GY93]. Ainsi, les routages adaptatifs sont utilisés afin de modifier, à tout moment, le chemin pris par le paquet en fonction de l'état du réseau. Ces algorithmes peuvent être minimaux s'ils fournissent un chemin qui est de longueur minimale. Ils peuvent avoir différents buts comme limiter au maximum la latence d'un paquet, équilibrer dynamiquement le trafic en

11. en anglais, LookUp Table
12. en anglais, Finite State Machine

tenant compte des zones de congestion ou encore permettre un routage même avec des liens défectueux.

Cependant, les routages adaptatifs sont plus complexes dans leur mise en place et peuvent facilement créer des interblocages (cf. Les problèmes d'interblocages) entre les paquets, étant donné que ces derniers peuvent se déplacer librement dans le réseau [BD02].

2.1.5 La progressivité

Dans le cas des algorithmes progressifs, chaque opération de routage alloue un nouveau canal avant que le flit ne soit transmis. La longueur du chemin augmente mais pas forcément en se rapprochant de la destination. Le routage glouton est toujours progressif.

Les algorithmes de routage tolérant aux pannes sont principalement utilisés lorsque aucune progression n'est possible. Le but de la tolérance aux pannes est d'éviter la faille totale du système malgré la présence de fautes dans un sous ensemble de ses composants élémentaires. La tolérance de panne est d'autant meilleure que le nombre de composants en panne est grand (avec la garantie du bon fonctionnement du système).
Dans ce genre d'algorithme, l'en-tête est autorisé à revenir en arrière, en libérant les canaux précédemment réservés et l'historique du routage est pris en compte afin de s'assurer que les mêmes chemins ne sont pas essayés à plusieurs reprises. Le routage tolérant aux pannes ne peut pas être déterministe.

2.1.6 La Minimalité

Un algorithme de routage est de plus court chemin ou minimal si chaque décision de routage est faite pour atteindre la cible en passant par les routes les plus courtes. Ainsi, avant chaque saut d'un paquet, une fonction de routage va calculer, à partir des identifiants du routeur local et de la destination, quelle(s) est (ou sont) la (ou les) sortie(s) du routeur permettant au paquet d'arriver à sa destination en suivant le (ou les) chemin(s) le (ou les) plus court(s). Ces algorithmes ont l'avantage d'éviter les problèmes d'interblocages en offrant une latence réduite. Mais ils sont moins flexibles lorsqu'un routeur est congestionné.

Dans les algorithmes de routage non-minimal, le paquet peut suivre n'importe quel chemin entre le nœud source et celui de destination. Les routages non-minimaux offrent une grande flexibilité en termes de chemins possibles, mais peuvent conduire à des situations d'interblocage dynamique et augmenter le temps de livraison du paquet.

2.1.7 Le nombre de chemins

Dans le routage adaptatif, le choix peut être partiel, c'est à dire effectué parmi un nombre restreint de chemins. Il peut également être global, le paquet pouvant alors être routé dans l'ensemble du réseau avant d'arriver à destination.

2.2 Les stratégies de stockage

Afin de ne pas être obligé de détruire des paquets en conflits, ces derniers peuvent être stockés dans des files d'attentes. Or, dans la majorité des NoCs, la surface consommée par un routeur provient de la place prise par ces files. Dès lors, un compromis doit être trouvé lors de la conception du routeur pour limiter au maximum la surface utilisée par les files d'attente sans pour autant dégrader les performances requises.

Les files d'attentes sont principalement caractérisées par leur taille et leur position au sein du routeur.

La taille des files d'attente

La taille comprend la largeur des files en nombre de bits et leur profondeur définissant le nombre de mots qui peuvent être stockés. Le premier paramètre qui influe directement sur la taille des files d'attente est le mode de commutation choisi [Ler07]. En effet, les files pourront contenir un ou plusieurs paquets. Concernant la largeur de la file, elle ne pourra pas être plus petite que celle d'une unité de contrôle de flux (flit). De plus, la taille doit être rigoureusement déterminée car elle peut affecter le coût en termes de surface et de consommation d'énergie des routeurs. Enfin, une taille inappropriée peut induire des congestions au sein du réseau avec une réduction de la bande passante utile.

La position des files d'attente

Les files d'attente peuvent être placées à différentes positions dans le routeur : en entrée, en sortie ou en sortie virtuelle [RGR03].

Files d'attente en entrée : Dans ce premier cas, chaque port d'entrée du routeur possède une file d'attente. Bien que cette technique soit la moins coûteuse en surface, elle peut induire une saturation à cause du blocage de tête de ligne (head-of-line) [KHM87]. Cela arrive lorsqu'une donnée en tête de file ne peut accéder au port de sortie associé, bloquant ainsi les autres paquets de la file, même si leur sortie est libre.

Files d'attente en sortie : Lorsque les files d'attentes sont positionnées en sortie du routeur, chaque port de sortie a autant de files d'attente que de ports d'entrée. Cette technique offre de meilleures performances que la précédente mais la surface est forcément plus importante.

Files d'attente en sortie virtuelle ou canaux virtuels : L'idée des files d'attente en sortie virtuelle est de combiner les avantages des deux techniques précédentes. Ainsi, chaque port d'entrée possède plusieurs files d'attente servant à répartir les paquets entrant en fonction de leur destination ou bien de leur priorité. On parle alors de canaux virtuels car pour un unique canal physique, il y aura autant de canaux virtuels que de files d'attente.
Bien qu'ils impliquent une augmentation de la surface et de la latence, à cause de la mémorisation et du contrôle nécessaires, les canaux virtuels permettent d'éviter les interblocages (les paquets bloqués sont stockés dans différents canaux virtuels pour laisser passer les autres paquets), d'optimiser l'utilisation des liens (liens rarement laissés dans un état oisif) et d'augmenter les performances du réseau (amélioration de la latence et de la bande passante) [Dal90].

2.3 Les problèmes d'interblocages

Le fait que les paquets de données sont transmis entre les nœuds de routage sans contrôle de routage global au sein du NoC, il est possible d'obtenir des blocages mutuels entre paquets. Ces blocages, aussi appelés interblocages, sont le plus souvent dus à l'algorithme de routage utilisé. Il en va de même pour les techniques servant à les empêcher [Lem06].

Il existe deux types d'interblocages : les interblocages statiques (deadlock) et les interblocages dynamiques (livelock).

2.3.1 Les interblocages statiques

Lorsqu'une dépendance existe entre plusieurs paquets comme dans l'example de la figure 20, ils peuvent se bloquer mutuellement et produire un interblocage statique. En effet, cela se produit lorsqu'un nœud de routage est en attente d'envoyer un paquet vers un autre nœud de routage qui est lui aussi en attente et ainsi de suite, jusqu'au nœud de routage initial. De toutes les commutations de

paquets, c'est la technique wormhole qui est la plus à générer ce genre d'interblocages car les messages sont autorisés à détenir un grand nombre de ressources tout en voulant en requérir d'autres [Fle02][NM93].

L'approche classique pour éviter les interblocages est bien évidemment de restreindre la fonction de routage de telle sorte que l'apparition de cycles dans le graphe de dépendance des ressources (ici les canaux) soit impossible.

Dans une topologie maillée, les interblocages peuvent être évités en choisissant un algorithme de routage approprié, comme par exemple le fait d'interdire certains virages [GN92], ou encore grâce aux canaux virtuels. En effet, si un paquet se trouve être bloqué dans un canal virtuel, il pourra être doublé par un autre paquet utilisant le même canal physique mais stocké dans un autre canal virtuel.

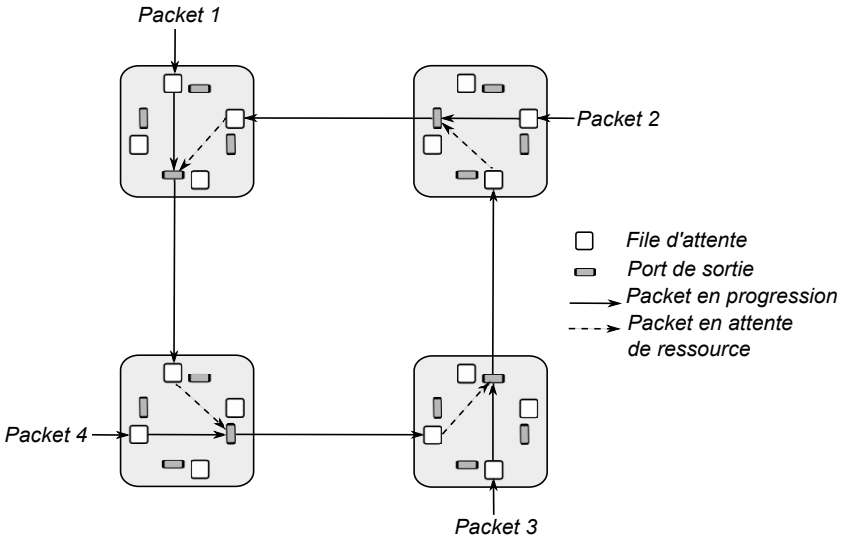

FIG. 20 – Un exemple de deadlock impliquant quatre paquets.

2.3.2 Les interblocages dynamiques

Les interblocages dynamiques se produisent lorsqu'un paquet reste indéfiniment dans le réseau sans jamais atteindre sa destination [Lem06]. La cause peut provenir d'une congestion du réseau qui ferait dérouter systématiquement le paquet, ou bien encore d'une mauvaise gestion des priorités lorsque le lien de sortie est toujours occupé par des paquets considérés comme plus prioritaires. Dans ce cas, on parle de phénomène de famine (ou starvation).

Qu'ils soient statiques ou dynamiques, les interblocages peuvent aussi être résolus à l'aide de l'anticipation de paquet. Il s'agit de re-router ou de supprimer les paquets impliqués dans un potentiel blocage. Les paquets détruits devront être retransmis par la source. Cette technique s'applique le plus souvent dans des architectures de réseaux indirects.

2.4 Le contrôle de flux

Les protocoles de contrôle de flux ont pour but de garantir un transport de paquets sûr depuis une IP source jusqu'à'à l'IP destination. La notion de sûreté réfère ici à la garantie qu'aucun paquet ne risque pas d'être perdu à cause d'un dépassement de capacité de l'un des buffers parcourus. Aussi appelés protocoles de régulation au niveau liaison de données, ils ont pour but d'assurer la cohérence de la bufférisation afin de gérer le déséquilibre entre le taux d'injection des paquets et le taux effectif

auquel ils sont éjectés du réseau. Comme pour les stratégies de gestion de la bufférisation, il existe plusieurs protocoles de contrôle de flux [PAB05], nous détaillerons dans ce qui suit les principaux protocoles :

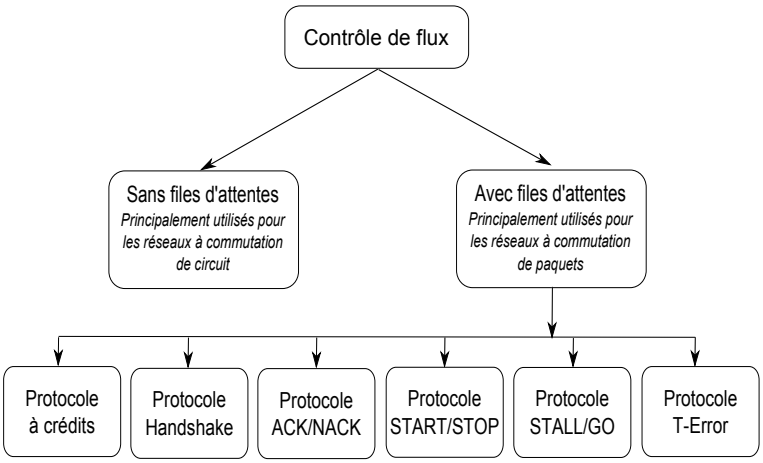

FIG. 21 – Les protocoles de contrôle de flux.

2.4.1 Le protocole à crédits

Dans le protocole à crédits[13], le port émetteur en amont garde l'information relative à l'état du buffer récepteur en aval (le crédit), cette information n'est autre que le nombre d'espaces disponibles qui est équivalent au nombre de flits pouvant être émis en séquence. À chaque fois qu'un espace se libère un crédit est envoyé afin que l'émetteur mette à jour la valeur locale du crédit.

2.4.2 Le protocole Handshake

Quand le routeur émetteur envoie des données, il active un signal pour informer le routeur récepteur de la disponibilité de ces données. Lorsque le routeur récepteur est prêt, il récupère les données et active un signal d'acquittement ACK. Le protocole handshake présente l'avantage d'être simple, peu coûteux et facile à implémenter dans un système GALS.

2.4.3 Le protocole ACK/NACK

C'est un protocole très simple où la logique de contrôle de flux se charge d'acquitter chaque flit reçu du coté du port récepteur. Le port émetteur quand à lui doit donc conserver chaque flit émis jusqu'à ce qu'il ait été acquitté. Un acquittement est envoyé une fois le flit écrit dans le buffer local, si aucun espace n'est disponible à sa réception un NACK est notifié.

2.4.4 Le protocole START/STOP

Ici le port émetteur peut émettre tant que le signal STOP contrôlé par le port récepteur est au niveau bas le cas échéant aucune émission n'est possible. Le signal de contrôle doit être positionné sur STOP par le port récepteur, dès que le taux de remplissage du buffer atteint un certain seuil, cela cause bien sûr une sous utilisation du buffer. Cette valeur seuil dépend principalement des caractéristiques temporelles (délais de transport) du lien reliant les deux ports.

13. en anglais, Credit Based

2.4.5 Le protocole STALL/GO

Dans le protocole STALL/GO, deux fils sont utilisés pour le contrôle de flux entre chaque paire d'émetteur (producteur) et de récepteur(consommateur). Quand il ya de l'espace vide dans la file d'attente, un signal GO est activé, sinon un signal STALL est activé. Le principal inconvénient de STALL/GO est qu'il n'a aucun dispositif pour le traitement des pannes. En cas de flit corrompue, un protocole complexe de haut niveau doit être déclenchée. Aucune implémentation de NoCs n'a actuellement eu recours à ce protocole de contrôle de flux.

2.4.6 Le protocole T-Error

Le protocole T-Error est très complexe par rapport à d'autres protocoles de contrôle de flux. Il vise à améliorer la performance au détriment de la fiabilité. Les systèmes d'exploitation à temps réel dans un environnement bruyant doivent éviter l'utilisation de ce protocole de contrôle de flux. Aucune implémentations de NoCs n'a utilisé cette technique de contrôle de flux.

Sous des conditions de trafics équilibrés (injection/éjection), où les capacités des buffers ne sont jamais dépassées les protocoles ACK/NACK, START/STOP et à crédits sont fonctionnellement équivalents ; c.à.d. qu'ils permettent d'avoir exactement les mêmes débits de données. Cependant sous un régime de trafic plus soutenu, des disparités apparaissent au niveau des performances; en effet sous de telles conditions de trafic les mécanismes de synchronisation sont activés plus souvent et induisent naturellement plus de pénalités sur le débit de données utiles.

2.5 L'arbitrage

Lorsqu'au sein d'un routeur, au moins deux paquets souhaitent sortir par le même port et ce, en même temps, un arbitrage doit être fait afin de choisir celui qui sortira le premier. L'arbitrage est utilisé pour garantir l'accès à un port de sortie lorsqu'un ou plusieurs ports d'entrée requièrent simultanément une connexion.
Bien qu'il existe différentes techniques pour arbitrer les paquets en conflit, la caractéristique la plus commune est l'impartialité. En effet, l'arbitre doit être capable de fournir un accès équitable aux ports de sorties si les paquets ont la même priorité. Les principales politiques d'arbitrage que l'on peut trouver dans les NoCs sont le Time- Division Multiple Access (TDMA [14]), la réservation de Time-slot, le tourniquet ou Round-Robin[15], et la priorité fixe [PBB03][Ler07].

2.5.1 TDMA

Le TDMA, ou Accès Multiple à Répartition dans le Temps, est un mode de multiplexage consistant à diviser le temps en périodes courtes appelées plages de temps, ou time-slots, et à attribuer à chaque élément communiquant une ou plusieurs plages. Ainsi, durant toute la durée d'une plage, la ressource partagée sera entièrement réservée à l'élément auquel le time-slot est associé. Cependant, la plage de temps doit être minutieusement alignée au risque d'introduire des latences supplémentaires [Ler07].

14. Time Division Multiple Access

15. L'expression provient du français ruban rond, modifié par idiotisme. Au XVIIe siècle ou XVIIIe siècle, lorsque des paysans français souhaitant se plaindre au roi lui faisaient remettre une pétition, la réaction habituelle du souverain était de faire arrêter et exécuter les deux ou trois premiers signataires. Dans ces conditions, on comprend que personne ne voulût figurer en premier sur la liste. Pour échapper à l'arbitraire de la punition, noms et signatures furent apposés circulairement en bas de la pétition (à la façon d'une guirlande), de sorte qu'il n'y ait plus de premier de liste et que tous les pétitionnaires soient tenus solidairement responsables.

2.5.2 La réservation de time-slot

La politique par réservation de time-slot est en fait une politique de type TDMA dans laquelle la réservation de la plage de temps est répétée périodiquement.

2.5.3 Le tourniquet ou Round-Robin

L'arbitrage de type tourniquet ou round-robin représente la politique d'arbitrage la plus impartiale. En effet, à chaque cycle, elle donne l'accès à un paquet provenant d'un port d'entrée différent [GZX06]. C'est-à-dire que le port le plus prioritaire deviendra le moins prioritaire au coup suivant. Cette méthode garantie que toutes les requêtes seront traitées et évite un conflit de type "starvation" (les données restent bloquées dans le buffer du routeur).

2.5.4 La priorité fixe

Enfin, dans la technique par priorité fixe, chaque paquet possède une priorité fixe qui lui a été attribuée par l'émetteur avant son entrée dans le réseau. Cette priorité est utilisée tout au long du cheminement du paquet entre sa source et sa destination.

2.6 La qualité de service

A l'instar des réseaux d'ordinateurs, les applications au sein d'un NoC peuvent avoir besoin d'une certaine qualité de service (QoS[16]) pour assurer des contraintes sévères en débits et/ou en latence. Les autres services qui sont considérés comme essentiels sont l'intégrité des données, l'absence de perte des données et l'ordonnancement des données [MVS05].
Afin de fournir tous ces services, des adaptations doivent être apportées au niveau du protocole de communication. Ces adaptations vont principalement concerner le routage et l'arbitrage des paquets (ou des flits).
Dans les NoCs, il existe essentiellement deux classes de QoS qui sont directement liées aux trafics associés. Il s'agit des trafics de type Best Effort (BE) ou de "Au mieux" et des trafics de type Guaranteed Services (GS) ou "Service Garanti".
Dans le cas du BE, aucune garantie de performance ne peut être donnée. Cependant, il offre un service garantissant que les toutes les données seront transmises et ce, correctement. L'avantage principal de cette classe de QoS est d'optimiser l'utilisation moyenne des ressources du réseau.
Quant au GS, il est implanté grâce à des mécanismes permettant de réserver des ressources de communication pour garantir un débit et/ou une latence fixés, quelque soit la charge du réseau. Ainsi, cela implique qu'une connexion doit être établie entre l'émetteur et le récepteur.

Assurer une qualité de service revient à trouver un bon compromis entre les services imposés par l'application visée et le coût des ressources matérielles.

2.7 Exemples de NoCs

Il existe actuellement un nombre important d'implémentations de réseaux-sur-puce aussi bien académiques, destinés à la recherche, qu'industriels à vocation commerciale. Dans ce qui suit nous présenterons quelques exemples d'architectures de NoCs choisis afin de bien refléter les tendances architecturales.

16. Quality Of Service

2.7.1 SPIN

SPIN[17] est un réseau d'interconnexion à commutation de paquets pour systèmes intégrés sur puce développé au LIP6 [18] qui employait initialement une topologie régulière d'arbre élargi (en anglais fat tree). Cette technologie fournit un mécanisme de communication très général entre les différents composants virtuels connectés dans le système. De plus, la bande passante croit linéairement avec le nombre de processeurs intégrés [CEG03].

Si la topologie fat tree permet d'utiliser efficacement les ressources matérielles d'un réseau sur puce, elle n'est toutefois pas bien adaptée pour interconnecter les différents composants d'un Système sur Puce. Pour remédier à cette limitation, les deux nouvelles versions de SPIN emploient une topologie en forme de grille et l'algorithme de routage déterministe distribué *X-first* [DS87]. La première nouvelle version de ce réseau sur puce, appelée DSPIN[19] [MGS06], est constituée de nœuds de routage qui possèdent chacun leur propre horloge de synchronisation et qui sont reliés par des FIFOs bi-synchrones. Une restriction de cette implémentation est d'introduire des latences importantes pour synchroniser les communications entre les différents domaines d'horloge des nœuds de routage. La deuxième nouvelle version de SPIN, appelée ASPIN[20] [SMG07], est composée de nœuds de routage asynchrones qui communiquent entre eux en utilisant un protocole à poignée de main insensible aux délais. Un avantage de cette implémentation est de réduire la latence de transmission d'un paquet car seules ses interfaces réseau nécessitent d'être synchronisées avec des composants synchrones.

2.7.2 Æthereal

Æthereal [GDR05] est un réseau sur puce synchrone développé par Philips ayant une topologie non régulière et utilisant un algorithme de routage par la source. Les interfaces réseau de Æthereal supportent différents protocoles de communication standard (OCP, AXI et DTL). La caractéristique principale de Æthereal est d'avoir des routeurs qui offrent deux niveaux de qualité de service différents. Le niveau *service garanti* assure des garanties sur les débits et sur les latences contrairement au niveau *meilleur effort* qui n'en offre aucune. Les paquets ayant un niveau de qualité de service meilleur effort sont acheminés par les routeurs en utilisant la technique de commutation Wormhole. Une technique de commutation de circuit basée sur du multiplexage temporel ou TDM[21] est utilisée pour transporter les paquets de type service garanti. Cette technique de commutation consiste à créer un circuit logique en réservant des périodes de temps dans des routeurs. Ce mécanisme permet d'assurer que les paquets qui empruntent un de ces circuits arriveront à destination sans entrer en conflit avec d'autres paquets. Le principal désavantage de cette approche est que la latence garantie est inversement proportionnelle au débit garanti. Ce mécanisme de commutation ne permet donc pas d'établir une connexion ayant un faible débit et une faible latence. Ce type de connexion est toutefois important car il permet de supporter correctement les communications liées aux interruptions.

2.7.3 MANGO

MANGO[22] [Bje05] est un réseau sur puce asynchrone développé par l'Université Technique du Danemark qui adopte une topologie en forme de grille et qui possède des interfaces réseau conformes à la norme OCP. Comme Æthereal, les nœuds de routage de MANGO fournissent les deux niveaux de qualité de services meilleur effort et service garanti. Une connexion à service garanti est établie en réservant une suite de canaux virtuels et en appliquant une politique d'arbitrage appelé ALG[23] [BS05].

17. Scalable Programmable Integrated Network
18. Laboratoire d'Informatique de Paris 6
19. Distributed Scalable Predictable Interconnect Network
20. Asynchronous Scalable Predictable Interconnect Network
21. Time Division Multiplexing
22. Message-passing Asynchronous Network-on-Chip providing Guaranteed services over OCP interfaces
23. Asynchronous Latency Guarantees

L'objectif de cette politique d'arbitrage est d'attribuer l'accès à un lien de communication de manière à respecter les latences et les débits associés aux canaux virtuels connectés à ce lien de communication. Le principal avantage de cette technique par rapport au multiplexage temporel employé par Æthereal est que les garanties sur les latences sont découplées des garanties sur les débits. Une limitation de cette technique est de ne pas permettre d'utiliser complètement la bande passante disponible pour les connections à service garanti.

2.7.4 Chain

Chain[24] [BF02] est un réseau sur puce conçu par l'Université de Manchester ayant une topologie non régulière et utilisant un routage par la source. Chain définit un ensemble de composants de base (arbitres, routeurs, . . .) qui doivent être assemblés les uns avec les autres pour former un réseau sur puce complet. Les travaux de recherche sur ce réseau sur puce ont donné naissance à la start-up silistix 5 qui propose des outils automatisant l'assemblage des composants de base de Chain. Ces outils permettent de réaliser rapidement un réseau sur puce adapté à une application donnée.

2.7.5 HERMES

Hermes est développé à l'université catholique de Rio Grande do Sol du Brésil. Ce réseau utilise la commutation trou de ver (Wormhole) dans une grille à deux dimensions avec un routage XY. Le switch ou routeur contient cinq ports bidirectionnels : Nord, Sud, Est, Ouest, et Local. Chaque IP est couplé à un switch à l'aide du port local (figure 22), l'interconnexion avec les voisins se fait à travers les quatre autres ports [MCM04].

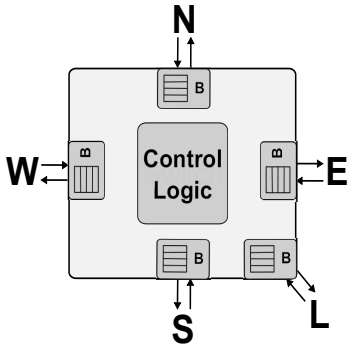

Fig. 22 – Le switch du réseau Hermes.

Chaque routeur contient principalement : la logique de contrôle (composée de la logique de routage et d'arbitrage) et les ports de communications. Chaque port d'entrée mémorise les données reçues dans une FIFO[25]. La communication asynchrone entre les routeurs est faite à l'aide d'un protocole de poignée de main (Handshake), alors que dans les routeurs les transactions sont synchrones (GALS). Un arbitrage à priorité tournante (Round Robin) est utilisé pour arbitrer l'accès aux ports. Les flits reçus ou bloqués sont stockés dans un buffer à FIFO circulaire dans le port d'entrée (ceci limite la chute des performances).

La figure 23 décrit l'interface entre deux routeurs. Quand un routeur a des données à envoyer à un de ses voisin, il active le signal tx et attend l'acquittement de l'autre partie sur la ligne ack_tx. Suite à cet échange, le routeur peut procéder à l'envoi des données. La logique de contrôle est l'ensemble de la

24. CHip Area INterconnect
25. First In First Out

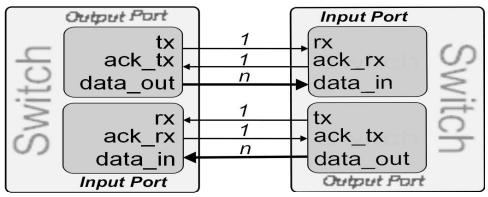

FIG. 23 – L'interface entre deux switchs du réseau Hermes.

logique de routage et d'arbitrage. Lors de la présence d'un flit à envoyer sur un port d'entrée, ce dernier envoie une requête à la logique d'arbitrage. Lorsque celle-ci termine l'arbitrage selon la politique Round Robin, elle transférera les données nécessaires à la logique de routage. Cette dernière décidera alors du port de sortie selon l'adresse et l'état du tableau du routage. Si le port est occupé, les flits de ce paquet sont bloqués sur le port d'entrée et le signal de requête à la logique d'arbitrage reste actif. Au cycle suivant le processus recommencera jusqu'au moment où le message pourra obtenir le port de sortie.

3 Notre proposition

3.1 Les mécanismes de routage dans la topologie 2D mesh

Dans un NoC, le routage joue un rôle important : le meilleur algorithme de routage donne la meilleure performance. L'algorithme de routage est donc soigneusement élaboré par les concepteurs. Il est nécessaire de faire des compromis entre une utilisation optimale des liens de communication et un algorithme simple qui peut être facilement mis en œuvre sur le silicium.

Selon les propositions d'architecture NoC, la topologie prédominante est la topologie de maillage simple à deux dimensions (2D mesh). Les raisons en sont ses avantages tels que la facilité d'implémentation dans les technologies actuelles, la simplicité des stratégies de routage et l'évolutivité du réseau. Les routeurs liés à cette topologie (comme d'ailleurs pour toute topologie NoC) doivent assurer le transport des paquets au sein du réseau en gérant au mieux les flux et les congestions sur le réseau. Ils utilisent pour cela des mécanismes de routage, d'arbitrage, de contrôle de flux, ainsi que les informations de contrôle fournies avec chaque paquet. Ces informations de contrôle sont regroupées généralement dans l'en-tête du paquet qui contient par la suite la charge utile c'est à dire le message à transmettre.

C'est l'algorithme de routage qui décide du chemin à emprunter pour envoyer le message de sa source à sa destination. Comme on l'a déjà détaillé dans le chapitre précédant, il existe plusieurs algorithmes de routage utilisés dans la conception de NoC avec des propriétés différentes. Si le chemin choisi ne dépend que de la source et de la destination, l'algorithme de routage est déterministe, il est adaptatif s'il tient compte de la congestion des liens. Comme exemple de routage déterministe dans un réseau de topologie 2D mesh, le routage XY, qui consiste à avancer d'abord sur la direction X puis ensuite sur la direction Y. Ce type d'algorithme de routage très simple permet de distribuer le contrôle au sein du réseau sans générer d'interblocage entre les paquets.

Le routage adaptatif augmente le nombre de chemins de routage possibles utilisables par un paquet pour arriver à sa destination. Toutefois, les situations de blocage peuvent se produire dans tout algorithme adaptatif global, ce qui limite son utilisation. Glass et Ni dans [GN92] ont proposés des algorithmes de routage partiellement adaptatifs à commutation wormhole et sans interblocages pour des réseaux 2D mesh. Leurs algorithmes n'utilisent pas de canaux virtuels et sont connus sous le nom de *turn model*. Dans [Chi00], un algorithme de routage appelé $Odd-Even$ a été proposée par Chiu. Basé sur le concept de turn model, il restreint les endroits où certains virages peuvent avoir lieu de telle sorte que les interblocages peuvent être évités. Un algorithme de routage appelé $DYAD$ [26] a été proposée dans [HM04]. Cet algorithme est la combinaison d'un algorithme de routage déterministe XY et un algorithme de routage adaptatif $Odd-Even$. Avec l'approche de routage $DyXY$ [27] [LZJ06], chaque paquet ne circule que le long du chemin le plus court entre la source et la destination (ce qui garantit que l'algorithme est sans interblocages statiques). Si plusieurs chemins (les plus courts) sont disponibles, le choix se fera en fonction de l'état de congestion du réseaux.

Notre proposition est inspirée des algorithmes de $Odd-Even$ de Chiu [Chi00] et de $DyXY$ de Ming Li [LZJ06]. Elle combine les deux dans le but de fournir un algorithme adaptatif, optimal et sans interblocages en tenant compte de l'encombrement local du routeur dans une topologie de maillage simple à deux dimensions.

Avant de détailler l'approche que nous proposons, nous allons donner une description succincte des algorithmes qui nous ont servies dans notre proposition.

26. DYnamic Adaptive Deterministic switching
27. Dynamic XY

3.2 Le concept turn model

Ce modèle fournit un concept méthodologique et principal afin de développer des algorithmes de routage adaptatifs partiels dans une topologie 2D mesh. Comme il a été expliqué auparavant, les interblocages statiques se manifestent lorsque le chemin emprunté par un paquet comprend un cycle. Donc, s'il n'existe pas de dépendance circulaire entre les canaux, il n'y aura pas d'interblocage. La notion de base sur laquelle repose le turn model consiste à éviter la création de cycles dans le chemin emprunté par un paquet.

Lors des passages entre les nœuds de routage dans un réseau 2D mesh, un paquet peut suivre quatre directions: Est, Ouest, Nord et Sud. Huit virages distincts sont possibles dans le chemin suivi par un paquet, comme le montre la figure 24(a). Un virage dans ce contexte se réfère à un changement de 90 degrés du sens de déplacement pour un paquet. Si on interdit au moins deux virages (lignes en pointillés dans la figure 24(b)), cela représente une condition suffisante pour implémenter des algorithmes sans interblocages. Un algorithme sans restrictions est nommé adaptatif global, mais s'il présente la moindre restriction il est nommé adaptatif partiel. Les algorithmes adaptatifs globaux devraient toujours faire face aux problèmes d'interblocages.

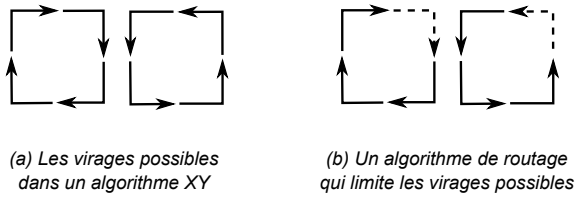

FIG. 24 – Le concept turn model.

Afin de bien illustrer le concept turn model, nous présentons quatre algorithmes de routage, l'un déterministe XY et trois partiellement adaptatifs *West First*, *Negative First* et *North Last*.

Les coordonnées de la source et de la destination sont identifiées ci-après par l'utilisation de la notations (X_S,Y_S) et (X_D,Y_D), respectivement.

L'algorithme XY est déterministe. Les flits sont d'abord acheminés dans la direction X, jusqu'à atteindre la coordonnée Y_D, et ensuite dans la direction Y, comme le montre la figure25. Si certains liens sont utilisés par un autre paquet, le flit reste bloqué dans le nœud de routage jusqu'à ce que le chemin se libère. Comme illustré dans la figure 25, les virages où les paquets viennent de la direction Y sont interdits (lignes en pointillés).

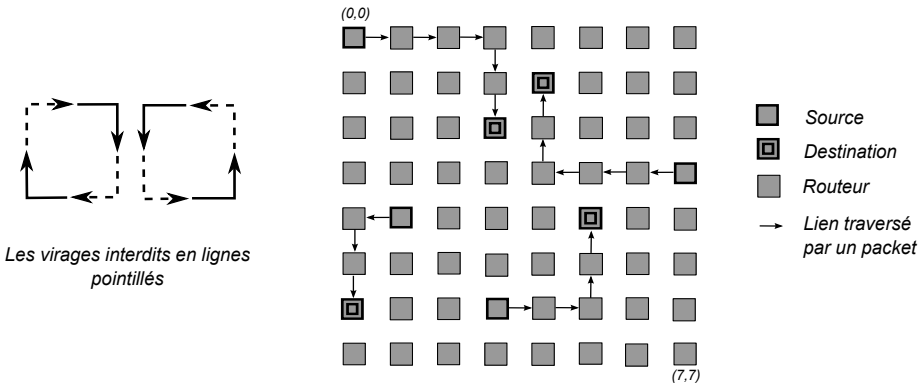

FIG. 25 – L'algorithme XY.

Dans l'algorithme de *West First* (Ouest premièrement), si $X_D \leqslant X_S$, les paquets sont routés de façon déterministe, comme dans l'algorithme XY. Si $X_D > X_S$, les paquets peuvent être acheminés de manière adaptative vers les directions Est, Nord ou vers le Sud (figure 26).

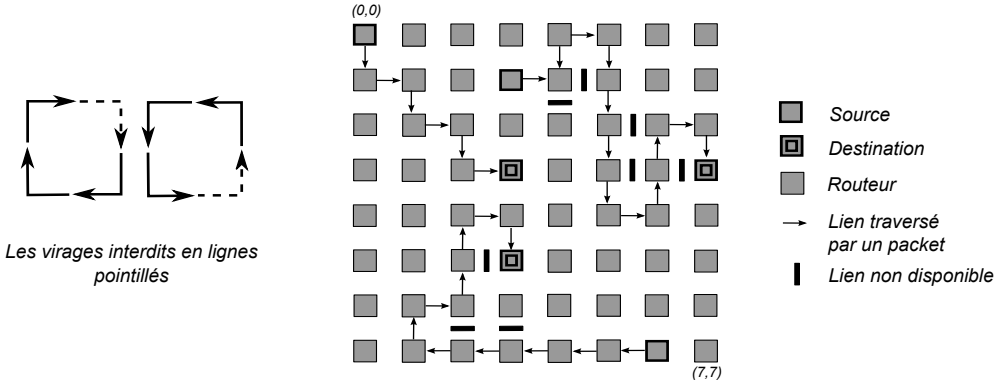

FIG. 26 – L'algorithme West First.

Les paquets sont acheminés premièrement vers les directions négatives quand on utilise l'algorithme *Negative First*, c'est à dire, vers le Nord ou l'Ouest. Si ($X_D \leqslant X_S$ et $Y_S \geqslant Y_D$) ou ($X_D \geqslant X_S$ et $Y_D \leqslant Y_S$) les paquets sont routés d'une manière déterministe, comme illustré sur la figure 27. Toutes les autres conditions permettent une certaine forme de routage adaptatif.

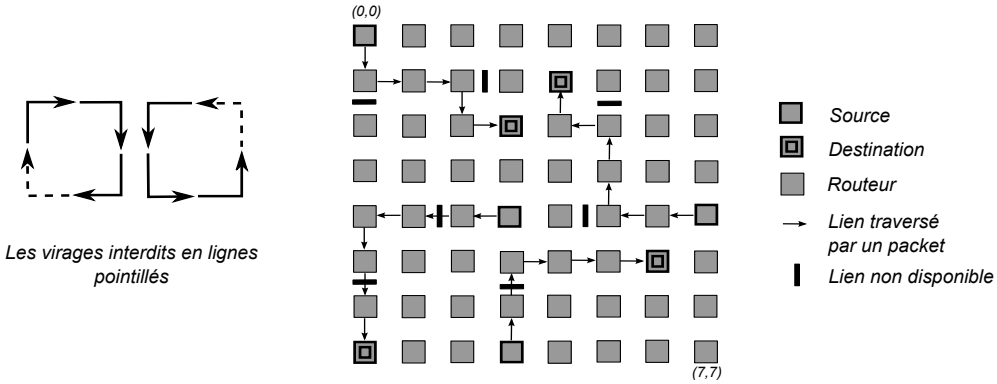

FIG. 27 – L'algorithme Negative First.

Dans l'algorithme *North Last* si $Y_D \leqslant Y_S$, les paquets sont acheminés de façon déterministe. Si $Y_D > Y_S$, les paquets peuvent être acheminés de manière adaptative vers les directions Ouest, Est, ou Sud (figure 28).

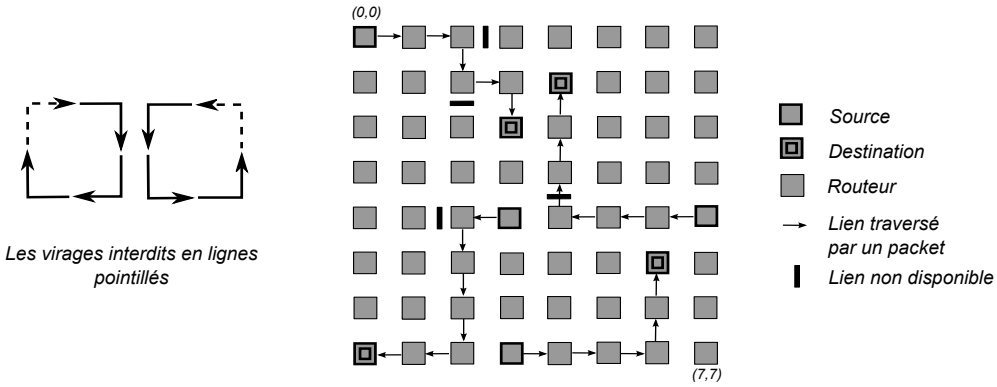

FIG. 28 – L'algorithme North Last.

3.3 Les algorithmes Odd-Even et DyXY

$Odd-Even$ est un algorithme de routage adaptatif basé sur le concept turn model. En le comparant à d'autres algorithmes adaptatifs qui n'utilisent pas de canaux virtuels, il est le plus adaptatif. Il limite les endroits où certains virages peuvent être pris pour s'assurer que des dépendances circulaires ne se produisent pas.

Dans ce modèle de routage, les colonnes d'un maillage 2D sont alternativement désignées comme impair (Odd) et paire (Even), comme le montre la figure 29 pour un maillage de 4×4 à deux dimensions.

Expliquer certaines définitions est nécessaires afin de présenter cet algorithme :

◇ Dans un réseau maillé à deux dimensions avec des dimensions $N\times N$, chaque nœud X est identifié par ses coordonnées (X_0,X_1) tel que X_0 et X_1 représentent l'abscisse et l'ordonnée de X respectivement.

◇ Une colonne est appelée impaire (odd) si son abscisse est un nombre impair, sinon elle appelée paire (Even).

◇ Un virage est appelé virage NE s'il s'agit d'un paquet qui vient du Nord et qui essaye de tourner vers l'Est. De même, nous pouvons définir les sept autres types de virages, à savoir EN, WS, WN, SE, SW, ES, et NW, où E, W, S, et N indiquent Est, Ouest, Sud et Nord respectivement.

L'idée de base du modèle $Odd-Even$ est de restreindre les endroits où certains virages peuvent se produire de telle sorte qu'un virage EN et un virage NW ne peuvent être pris au niveau des nœuds de la même colonne. De même aussi pour les virages ES et SW. La figure 29 illustre les virages interdits dans l'algorithme d'$Odd-Even$.

Plus précisément, le modèle $Odd-Even$ est régie par les deux règles suivantes :

Règle 1 : Aucun paquet n'est autorisé à faire un virage EN dans un nœud situé sur une colonne paire et aucun paquet n'est autorisé à faire un virage NW dans un nœud situé sur une colonne impaire.

Règle 2 : Aucun paquet n'est autorisé à faire un virage ES dans un nœud situé sur une colonne paire et aucun paquet n'est autorisés à faire un virage SW dans un nœud situé sur une colonne impaire.

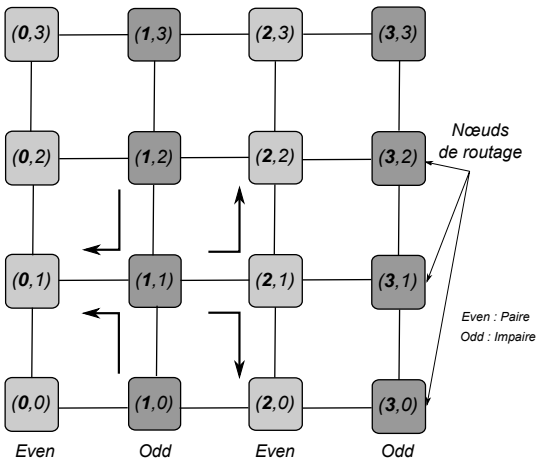

FIG. 29 – Odd Even dans un 2D mesh.

DyXY est une approche d'un algorithme de routage dynamique. Le détail de cet algorithme est résumé comme suit:

- Lire la destination d'un paquet entrant.
- Comparer les adresses de la destination et du routeur actuel.
- Si la destination est l'IP local du routeur actuel, envoyer le paquet à l'IP locale.
- Sinon
  - Si la destination a la même adresse x (ou y) que le routeur actuel, envoyez le paquet au routeur voisin sur l'axe des y (ou axe des x) vers la destination.
  - Sinon, vérifiez les valeurs de stress des routeurs voisins du routeur actuel et envoyer le paquet au voisin avec la plus petite valeur de stress.

La valeur de stress désigne le nombre de cellules occupées dans toutes les files d'attentes d'entrée d'un routeur. Elle est considérée comme un paramètre représentatif de la congestion d'un routeur.

3.4 Description de l'algorithme

Bien que le routage déterministe soit plus simple à mettre en œuvre et à valider, dans un souci d'efficacité on s'oriente de plus en plus vers des méthodes de routage adaptatif car le temps total pour livrer un paquet peut être réduit si le paquet peut, dans certains cas, faire des virages pour échapper à des situations de blocage.

L'objectif principal de notre travail est de développer une stratégie de routage adaptatif afin de sélectionner le canal de sortie qui permet au paquet d'être acheminé vers sa destination avec une latence minimale. C'est dans cette optique que nous proposons un nouvel algorithme de routage dynamique basé sur le turn model et dédié aux topologies de type 2D mesh. Son intérêt réside principalement dans la flexibilité de son aspect dynamique qui permet de contourner des points de congestion en utilisant des chemins alternatifs.

Dans cette section, nous présentons en détail le fonctionnement de notre algorithme qui est organisé en deux phases principales :

la première phase est basée sur le concept $Odd-Even$ et a pour objectif de concevoir un algorithme adaptatif partiel sans interblocages statiques. Fondamentalement, à n'importe quel nœud de routage intermédiaire, l'algorithme doit d'abord déterminer l'ensemble des directions vers lesquelles un paquet peut être transmis pour atteindre le nœud suivant. L'application de l'une des deux

règles principales de l'approche $Odd-Even$ est nécessaire pour éviter les interblocages statiques ou "deadlocks".
Le choix d'$Odd-Even$ pour cette première phase a été motivé par le fait que ce modèle fondé sur la restriction des endroits où certains virages peuvent être pris permet d'empêcher la création des dépendances circulaires, offre un degré d'adaptabilité meilleur et présente des performances supérieures par rapport aux autres algorithmes de routage basés sur le turn model de Glass et Ni à travers différents scénarios de trafic. A la fin de cette phase, l'ensemble des directions non interdites pour la transmission d'un paquet sont disponibles.

la deuxième phase a pour entrée l'ensemble des directions autorisées à un paquet pour sortir du nœud actuel et aller vers un de ses voisins afin d'atteindre sa destination. En effet, il est possible pour un paquet d'avoir de multiples directions valides qu'il peut suivre à partir de son emplacement actuel. Pour assurer des transferts avec une faible énergie, les directions des chemins minimaux ont toujours une priorité plus élevée, mais cela peut, dans certains cas engendrer des zones de congestions dans le réseaux. Le but de cette deuxième phase est de faire un choix intelligent en fonction de l'encombrement des routeurs. Si, toutefois, plusieurs directions valables existent à partir du routeur actuel vers la destination, l'objectif est de donner une priorité plus élevée à la direction qui permet de sélectionner le routeur voisin le moins congestionné. Pour le faire, on a (introduit) instauré un paramètre représentatif de l'encombrement d'un routeur qu'on a appelé l'*Indice de Congestion* . Le calcul de la valeur de cet indice utilise le nombre de flits occupés dans toutes les files d'attentes d'entrée d'un routeur (figure 30). Chaque routeur conserve instantanément les valeurs des indices de congestion de ses voisins, et chaque valeur est mise à jour si un changement intervient. Pour chacune des directions disponibles, le mécanisme de sélection consiste à vérifiez les valeurs de l'indice de congestion des routeurs voisins liés à ses directions et à envoyer le paquet au voisin possédant le plus petit indice. Si deux directions ont le même *Indice de Congestion*, on privilégera les directions Nord et Sud par rapport à l'Est et l'Ouest.

La figure 30 montre un exemple d'un NoC en grille 2D de taille 3×3. Elle montre également les valeurs d'*Indice de Congestion* pour les routeurs (1), (2), (3), (4) et (5) qui représentent le routeur actuel et ses quatre voisins respectifs. Dans le routeur (1) le flit qui se trouve dans la file d'attente d'entrée du port Sud (entouré par un cercle dans la figure 30) a été désigné par l'arbitre pour être transféré via une sortie valide.

En supposant que dans cet exemple, l'ensemble des directions de sortie autorisées au flit du port Sud du routeur (1) sont les sorties Est(E), Ouest(O) et Nord (N) étant donné que la sortie Sud est interdite car c'est la direction de provenance du paquet.
L'algorithme de routage va aider le paquet à choisir une sortie en sélectionnant celle qui correspond au routeur voisin avec la plus petite valeur d'*Indice de Congestion*, en l'occurrence c'est la sortie Est qui sera désignée.

Notre algorithme est conçu pour réduire considérablement le temps de traversée du réseau lors des encombrements. Il offre un routage adaptatif basé sur l'état de congestion de la proximité. La diversité de chemins offerte permet la déviation des paquets qui ne peuvent temporairement atteindre leur destination en évitant de les bloquer dans des files d'attente au sein des routeurs et d'engendrer des situations d'interblocages.

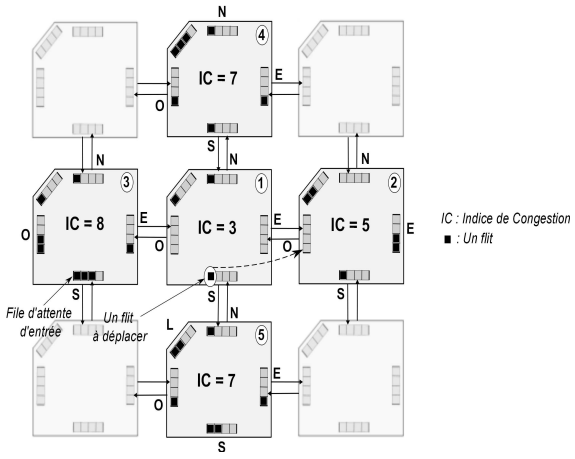

FIG. 30 – Indice de Congestion dans un 2D mesh.

Notre algorithme peut être résumé comme suit :

◎ Quand un routeur reçoit un nouvel en-tête de paquet, l'algorithme d'arbitrage Round-Robin est exécuté pour garantir l'accès à un port de sortie lorsqu'un ou plusieurs ports d'entrées requièrent simultanément une connexion.

◎ Une fois la requête est accordée par l'algorithme d'arbitrage, lire l'adresse de la destination du packet sélectionné.

◎ Comparer les adresses de la destination et du routeur actuel ou courant.

- Si l'adresse de la destination est égale à celle du routeur actuel, le packet est envoyé à l'IP locale attachée a ce routeur.

- Sinon :

  ⋆ Utiliser l'algorithme $Odd - Even$ pour calculer l'ensemble des directions non interdites pour la transmission du paquet.

  ⋆ Pour chacune des directions disponibles, calculer les valeurs de l'*indice de congestion* des routeurs voisins liés à ses directions et envoyer le paquet au voisin possédant le plus petit indice. Si deux directions ont le même *Indice de Congestion*, la priorité est donnée aux directions Nord et Sud par rapport à l'Est et l'Ouest.

  ⋆ Mettre à jour les valeurs d'*indice de congestion* des routeurs concernés après chaque envoi.

4 Implémentation et résultats

4.1 L'architecture cible

L'architecture utilisée pour tester et évaluer notre proposition est inspirée de l'architecture Hermes de Moraes [MCM04]. Elle est organisée en neuf nœuds de routage (mesh 3x3) dont chacun est composé de cinq ports bidirectionnels Est, Ouest, Nord, Sud et Local comme le montre la figure 31. Le port Local est relié à l'IP alors que les autres ports sont reliés aux nœuds voisins.

Chaque port d'entrée possède un buffer à FIFO circulaire pour stocker les données provisoirement afin de limiter la chute des performances. Le bloc de contrôle est formé d'un arbitre à priorité tournante (Round Robin) pour l'accès aux ports de sortie qui garantit que toutes les requêtes seront traitées et évite que les données restent bloquées dans le buffer du nœud (problème de starvation) et d'un routeur utilisant notre algorithme de routage dynamique (OEC[28]) décrit dans le chapitre précédant.

Le calcul de la valeur de l'Indice de Congestion qui utilise le nombre de flits occupés dans toutes les files d'attentes d'entrée d'un nœud de routage est géré par le bloc de contrôle qui diffuse instantanément cette information à tous les nœuds voisins.

Les communications asynchrones entre les routeurs sont gérées par le protocole de poignée de main ou Handshake qui présente l'avantage d'être simple, peu coûteux et facile à implémenter dans un système GALS.

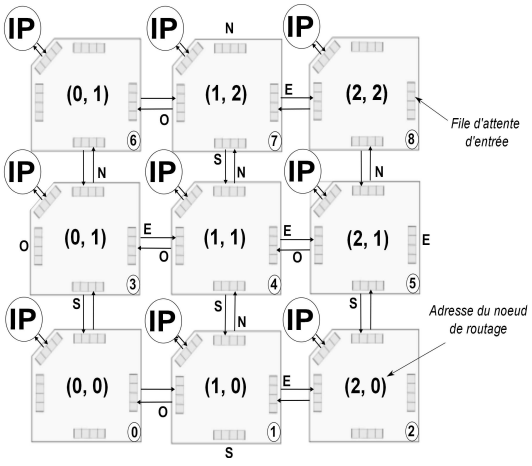

FIG. 31 – L'architecture envisagée.

Le format du paquet détaillé est montré sur la figure 32. Chaque paquet est constitué d'un en-tête(Header) et d'un corps (Payload) qui indique la charge utile de ce paquet. L'en-tête comprend deux parties (deux premiers flits): la première contient l'adresse du nœud source ou émetteur et l'adresse du nœud destination ou récepteur et la deuxième partie renseigne la taille du corps du paquet en flits.

Par exemple, 2200 0005 0001 0002 0003 0004 0005 désigne un paquet qui a comme source le nœud avec l'adresse (2, 2) et comme destination le nœud avec l'adresse (0, 0). La taille du payload dans cet exemple est de 5 flits.

28. Odd Even Congestion

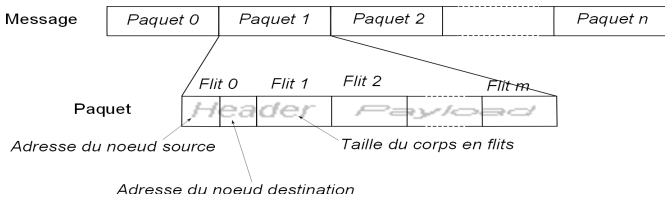

FIG. 32 – Format du paquet de l'architecture envisagée.

4.2 Environnement de simulation

Dans cette section, nous allons présenter la liste des outils qui nous ont servi dans la mise en œuvre et la validation de notre architecture. La plate-forme de simulation a été employée pour vérifier le comportement fonctionnel des composants (nœuds de routage, liens, ..) décrits en Verilog, s'assurer que les paquets étaient bien acheminés à leur destination en respectant l'algorithme de routage et évaluer les latences du réseau.

Système d'exploitation : Windows 7.

Langage de programmation : Verilog 2005.

Editeur : Notepad++ v5.8.6 de Notepad++ team.

Simulateur : Modelsim PE Student Edition 10.0a de Mentor Graphics.
Revision: 2011.02
Date: Date Feb 20 2011

ModelSim PE Student Edition est une version gratuite du simulateur ModelSim destiné à être utilisé par les étudiants dans le cadre de leur cursus universitaire et des projets d'enseignement de base. Elle est compatible aux plates-formes x86/Windows seulement, supporte à la fois le Verilog et le VHDL mais ne permet pas d'avoir des conceptions mixtes. La capacité de cette version est limitée à 10 000 lignes de code exécutable et le taux de performance est de 30% par rapport à une version PE commerciale. Ce taux passe à seulement à 1% une fois la capacité dépassée.

4.3 Simulations et résultats

Les composants matériels du notre architecture maillée 3x3 et la génération de trafic ont été développés en Verilog et validés par une simulation fonctionnelle. La génération des paquets qui modélisent le trafic (sous forme de fichiers d'entrée) est décrite par un programme C++.

Cette section présente des résultats de simulations Verilog obtenus avec l'outil ModelSim de Mentor Graphics. Nous commençons par donner quelques concepts fondamentaux avant de présenter les résultats de simulation.

Le délai des paquets ou les temps de latence se réfèrent à la période de temps (en cycles d'horloge) entre l'injection du flit d'en-tête dans le réseau par le nœud source et la réception du dernier flit composant le paquet par le nœud de destination. En réalité, la latence des paquets peut être décomposée en deux parties: la latence de transfert et le temps d'attente.

La latence de transfert est le temps passé pour transmettre le paquet. En dehors des cas de congestion du réseau, elle reste fixe. Cependant, le *temps d'attente*, qui est le temps d'attente de passage d'autres paquets, dépend complètement de l'état du réseau. En cas de congestion, la latence des paquets augmente rapidement jusqu'à atteindre des centaines de cycles. Au meilleur des cas, quand il n'y a pas de congestion le paquet traverse le réseau de manière fluide.

La charge offerte par un trafic simulé est définit comme le pourcentage de la bande passante utilisée par chaque nœud de routage [CEG03]. On atteint une charge de 100% lorsque tous les nœud envoient continuellement des données, sans interruption entre les paquets successifs. La période entre la création de deux paquets permet de générer différentes charges réseau. Dans des situations réelles, la charge est beaucoup plus petite.

Dans notre expérimentation, afin d'analyser le comportement du NoC en présence de collisions et d'évaluer notre algorithme de routage, nous utilisons deux types de trafic pour simuler les performances du notre architecture:

- Trafic uniforme : Sous ce modèle, un nœud de routage envoie un message à un autre nœud avec une probabilité égale.

- Trafic transpose1 : Sous ce modèle, un nœud de routage à l'adresse (i,j) n'envoie que des messages au nœud avec l'adresse (j,i) tel que $(i,j \in [0,E[)$ et $E = 3$.

La taille du réseau lors de la simulation est fixée à $3x3$ avec 9 sources et 9 destinations, chaque source émet 100 paquets avec une longueur égale à 16 flits ou mots.

En considérant des buffers FIFO de 16 flits, la figure 33 résume les valeurs de latences moyennes du réseau avec différentes charges et sous les deux modèles de trafic utilisés. Globalement, les résultats montrent que sous les deux trafics, les valeurs de latence obtenues avec l'algorithme de routage OEC augmentent linéairement avec la charge du réseau. Quand la charge du réseau est faible (50 et 25), les latences obtenues avec le trafic transpose1 sont un petit peu meilleures que celles du trafic uniforme. D'autre part, on peut remarquer que pour des charges du réseau plus élevée les latences sont importantes avec le trafic transpose1.

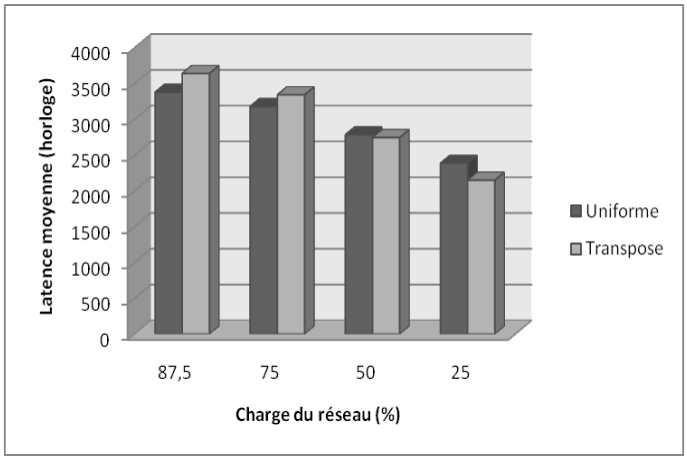

FIG. 33 – Latence moyenne d'OEC en fonction de la charge du réseau.

Pour évaluer l'efficacité de notre algorithme de routage OEC, nous l'avons comparé à deux autres algorithmes que nous avions testé, le routage déterministe XY et le routage adaptatif OE. Les algorithmes suivants ont été considérés dans l'étude de comparaison : le routage XY et le modèle adaptatif d'Odd-Even.

Les résultats de la figure 34 montrent que sous un trafic uniforme, l'algorithme de routage déterministe XY est plus rapide que les deux autres algorithmes partiellement adaptatifs OE et OEC.

Cela vient du fait que les méthodes de routage fixe visent une optimisation globale à long terme, alors que les méthodes adaptatives ont pour ambition de satisfaire à tout instant un ou plusieurs critères d'optimalité. Les algorithmes partiellement adaptatifs ont tendance à accélérer le temps pour livrer des paquets individuels (en sélectionnant des chemins de routage basé sur des informations à court terme), mais fournissent une performance globale moins bonne que l'algorithme XY pour des profils de trafic uniformes ou réguliers. Ce résultat est cohérent avec d'autres résultats rapportés dans la littérature comme [Chi00] et [HM04].

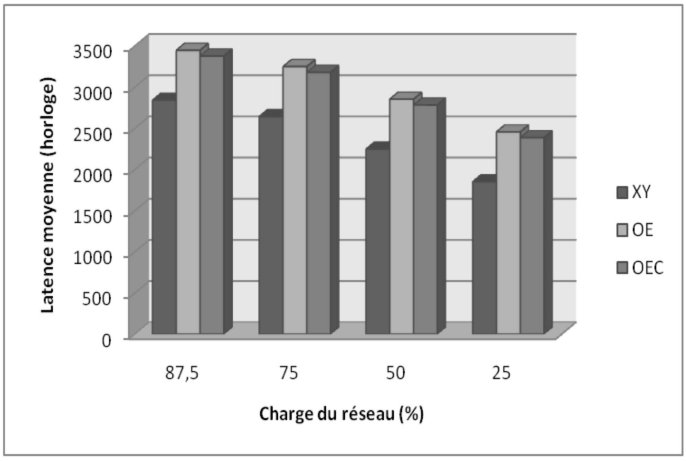

FIG. 34 – Latence moyenne en fonction de la charge avec un trafic uniforme.

Avec le trafic transpose1 qui reflète les échanges de la plupart des applications du monde réel ou les nœuds communiquent plus fréquemment avec certains nœuds comparé à d'autres, notre algorithme OEC est plus performant et présente des latences moyennes plus courtes par rapport à celles réalisées par les deux autres algorithmes (OE et XY), comme on peut l'observer sur la figure 35.

Cette étude expérimentale et ces résultats de simulation confirment l'intérêt de notre approche qui réside principalement dans la flexibilité de son aspect dynamique qui permet de contourner des points de congestion dans le réseau en utilisant des chemins alternatifs.

Un Autre modèle de trafic non uniforme transpose2 a été simulé et les résultats étaient similaires à celui du modèle de trafic transpose1. Sous le modèle transpose2, un nœud de routage à l'adresse (i,j) n'envoie que des messages au nœud avec l'adresse $(E-1-j,E-1-i)$ tel que $(i,j \in [0,E[$ et $E=3$.

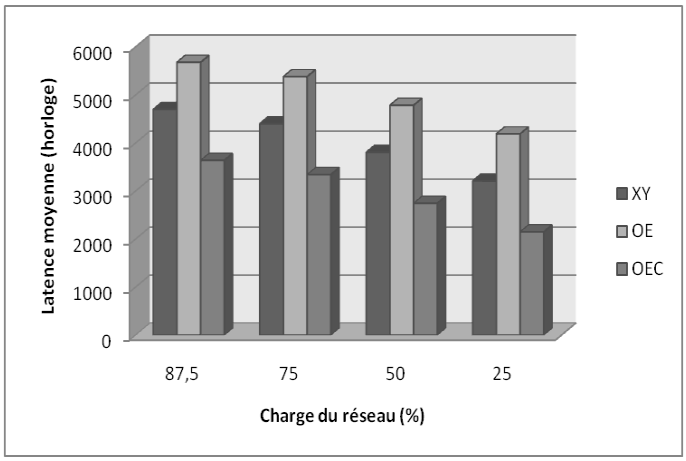

FIG. 35 – Latence moyenne en fonction de la charge avec un trafic transpose1.

Conclusion

Le paradigme de NoC a été conçu pour être une solution efficace pour surmonter les problèmes de communication entre les différentes ressources des futurs SoCs. Le rôle du mécanisme de routage au sein du NoC est de définir le chemin que doit suivre un paquet pour atteindre sa destination avec une latence minimale.

Dans ce mémoire, nous avons proposé un nouvel algorithme de routage dynamique appelé *OEC* dédié aux topologies de type 2D mesh, qui fournit un routage adaptatif et sans interblocages basé sur les conditions de la congestion du réseau dans la proximité.

Pour valider notre approche, nous avons réalisé un certain nombre d'expérimentations et nous l'avons comparé à des approches existantes. L'étude empirique que nous avons menée nous a permis de constater que les résultats de simulations de OEC présentent des latences moyennes plus basses que celles des algorithmes XY et Odd-Even sous des charges de trafic différentes.

Ces premiers résultats obtenus sont encourageants et nous laissent envisager de nouvelles améliorations ainsi que des expérimentations beaucoup plus poussées. Ils nous ouvrent la voie vers de nombreuses perspectives de recherches et de développement. En effet, il est possible de jouer sur certains paramètres comme la taille des buffers et la taille des paquets et d'en observer les effets sur les performances afin d'assurer des résultats optimaux. Il serait également possible d'envisager de simuler différentes tailles du réseau.

Il nous semble également intéressant de réaliser un prototype afin de caractériser et d'évaluer la consommation en surface de notre NoC. Une étude sur les éventuelles optimisations de notre algorithme de routage et de son extension tolérante aux fautes serait aussi instructive.

A Les machines à états finis

L'objectif de cet annexe est d'illustrer la description de la machine à états finis et de ses différents types sous forme algorithmique en langage Verilog [29].

Une machine à états finis ou en anglais FSM [30] est un modèle de comportement composé d'un nombre fini d'états, de transitions entre ces états, et d'actions. Une transition est un ensemble d'actions qui commence à partir d'un état et se termine soit dans le même état soit dans un autre état. Une transition est déclenchée par un déclencheur qui pourrait être un événement ou une condition.

Un automate ou machine à états finis constitue un modèle d'architecture des circuits numériques dans lequel un registre interne mémorise l'état courant du processus, tandis qu'une logique combinatoire permet de calculer l'état suivant à appliquer, au front d'horloge s'il s'agit d'un système synchrone, ou sur modification d'une entrée lorsqu'il s'agit d'un système asynchrone. A cette logique de génération des états suivants s'ajoute la logique de détermination des sorties. Selon que cette dernière dépend ou non des entrées, on parle de machine de Mealy ou de machine de Moore.

Dans le cas de circuits mettant en œuvre des traitements numériques plus ou moins complexes, une machine d'état est souvent utilisée comme description du fonctionnement d'une unité de contrôle du chemin de données [31].

Fondamentalement, une machine à états finis se compose de trois parties comme indiqué dans la figure 36.

1 - Une logique combinatoire : qui est utilisé pour décider du prochain état de FSM.
2 - Une logique séquentielle : qui est utilisé pour stocker l'état actuel de FSM.
3 - Une logique de sortie : qui est un mélange de la logique combinatoire et séquentielle.

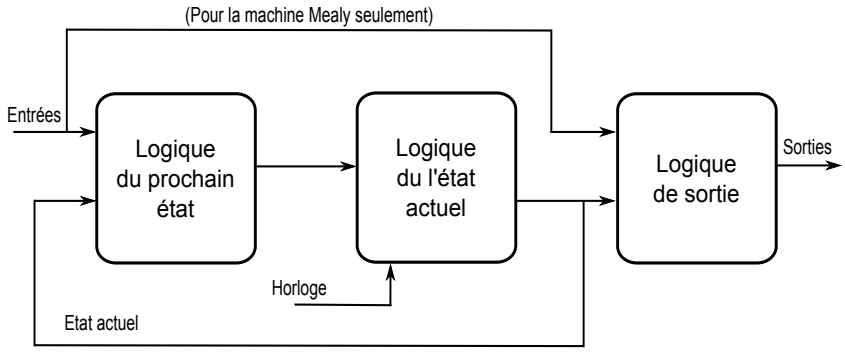

FIG. 36 – Structure de la machine à états finis.

29. Est un langage de description matériel de circuits logiques en électronique, utilisé pour la conception d'ASICs (Application-Specific Integrated Circuits, circuits spécialisés) et de FPGAs (Field-Programmable Gate Array)
30. Finite State Machine
31. Dans un système logique, un chemin de données est un ensemble d'unités fonctionnelles électroniques, telles que des unités arithmétiques et logiques, des multiplicateurs, des registres, qui effectuent des opérations de traitement de données. Par exemple la plupart des processeurs sont composés d'un chemin de données et d'un séquenceur, ce dernier ayant pour rôle de commander le chemin de données et réguler ses interactions avec la mémoire vive.

A.1 Types de machines d'état

Il y a plusieurs façons de coder ces machines d'état, mais avant d'étudier les styles de codage, nous allons d'abord définir les bases des deux types de machines d'état : la machine de Mealy et la machine de Moore.

A.1.1 Machine a états finis de Mealy

Dans la machine de Mealy, la sortie dépend de l'état actuel et les entrées courant. Dans la figure 37, les sorties dépendent à la fois de l'état courant et des variables d'entrée. A la différence de la machine de Moore, la sortie peut changer lors d'un changement de l'entrée et ce indépendamment de l'horloge. Une machine de Mealy est donc asynchrone, mais on peut avoir une version synchrone équivalente. Les Machines de Mealy permettent une description plus complexe et une prise en compte immédiate d'un changement en entrée.

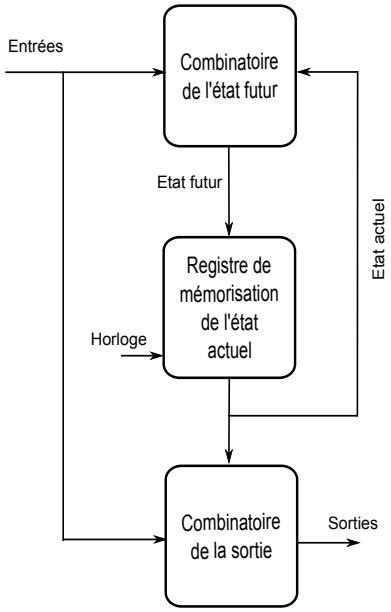

FIG. 37 – Structure de la machine a états finis de Mealy.

A.1.1.1 Exemple de machine de Mealy Dans cet exemple, une machine de Mealy reconnaissant la séquence 10 est présentée. Pour le diagramme d'état (figure 38), comme les sorties évolues en même temps que les entrées et l'état actuel, la valeur des sorties est indiquée sur la transition par un séparateur : condition en entrée/valeur en sortie(s).

A.1.1.2 Description d'une machine de Mealy Dans cette section, on présente la description en Verilog de la machine de Mealy de l'exemple précédant avec 3 process.

1 - Un process séquentiel de mise à jour de l'état présent par l'état futur sur les fronts montant d'horloge qui est utilisé pour décider du prochain état de FSM:

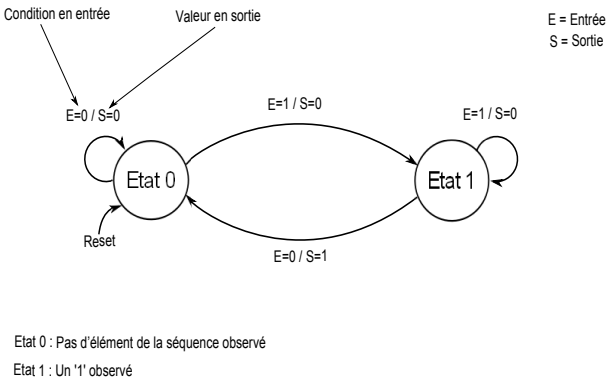

FIG. 38 – Machine de Mealy reconnaissant la séquence 10.

Listing 1: Code Verilog du process séquentiel de l'état actuel

```
parameter [1:0] Etat0 = 2'd0, Etat1 = 2'd1;
reg [1:0] Etat_actuel, Etat_futur;

always @ (posedge clock or negedge reset)

begin

  if (!rst)
    Etat_actuel <= Etat0;
  else
    Etat_actuel <= Etat_futur;

end
```

2 - Un process combinatoire de calcul de l'état futur à partir des entrées et de l'état présent :

Listing 2: Code Verilog du process combinatoire de l'état futur

```
process @ (E, Etat_actuel)

begin

  case (Etat_actuel)
    Etat0 : if (E = 1'b1)
              Etat_futur = Etat1;
            else
              Etat_futur = Etat0;

    Etat1 : if (E = 1b'1)
              Etat_futur = Etat1;
            else
              Etat_futur = Etat0;
  endcase

end
```

3 - Un process combinatoire de calcul des sorties à partir des entrées et de l'état présent :

Listing 3: Code Verilog du process combinatoire des sorties

```
process @ (E, Etat_actuel)
begin

  case (Etat_actuel)
    Etat0 : if (E = 1b'1)
              S = 1b'0;
            else
              S = 1b'0;

    Etat1 : if (E = '0')
              S = 1b'1;
            else
              S = 1b'0;
  endcase

end
```

Si les 2 process combinatoires possèdent la même liste de sensibilité, ils peuvent alors être regroupés en un seul process afin d'abréger l'écriture, la description se faisant en deux process seulement (un process séquentiel et un process combinatoire).

A.1.2 State Machine de Moore

Dans la machine de Moore, la sortie dépend seulement de l'état actuel comme on peut le constater sur la figure 39 représentant la structure d'une machine d'états finie de Moore. L'état futur est calculé à partir des entrées et de l'état actuel. Les sorties qui ne dépendent que des états actuels ou courants changent de manière synchrone sur un front d'horloge.

Le fonctionnement synchrone d'une machine de Moore basé seulement sur des transitions du signal d'horloge facilite son utilisation dans tout circuit synchrone et rends donc cette machine plus avantageuse que la machine de Mealy.

A.1.2.1 Exemple de machine de Moore L'exemple suivant décrit une machine de Moore reconnaissant la séquence 10. Dans le diagramme d'état (figure 40), les sorties évoluent après l'activation de la transition, leurs valeurs sont t indiquées dans les cases du diagramme.

A.1.2.2 Description d'une machine de Moore Dans cette section, on présente la description en Verilog de la machine de Moore de l'exemple précédant avec 3 process.

1 - Un process séquentiel de mise à jour de l'état présent par l'état futur sur les fronts montant d'horloge qui est utilisé pour décider du prochain état de FSM:

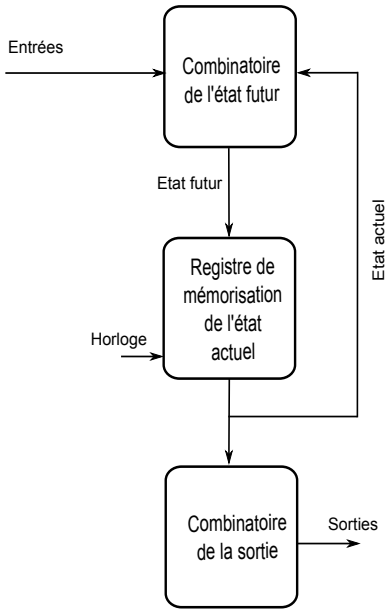

FIG. 39 – Structure de la machine a états finis de Moore.

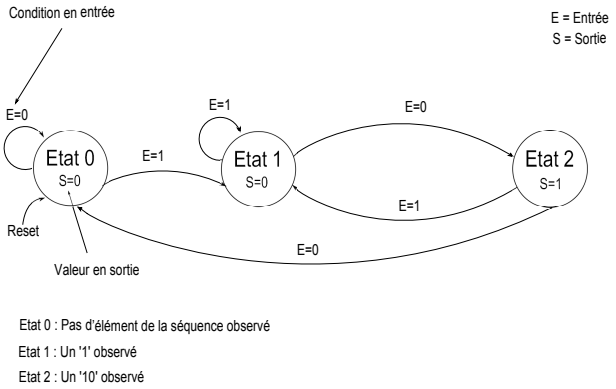

FIG. 40 – Machine de Moore reconnaissant la séquence 10.

Listing 4: Code Verilog du process séquentiel de l'état actuel

```
parameter [1:0] Etat0 = 2'd0, Etat1 = 2'd1;
reg [1:0] Etat_actuel, Etat_futur;

process @ (posedge clock or negedge reset)

begin

  if (!reset)
    Etat_actuel <= Etat0;
  else
    Etat_actuel <= Etat_futur;

end
```

2 - Un process combinatoire de calcul de l'état futur à partir des entrées et de l'état présent :

Listing 5: Code Verilog du process combinatoire de l'état futur

```
process @ (E, Etat_actuel)
begin

  case (Etat_actuel)
    Etat0 : if (E = 1b'1)
              Etat_futur = Etat1;
            else
              Etat_futur = Etat0;


    Etat1 : if (E = 1b'0)
              Etat_futur = Etat2;
            else
              Etat_futur = Etat1;

    Etat2 : if (E = 1b'1)
              Etat_futur = Etat1;
            else
              Etat_futur = Etat0;

  endcase

end
```

3 - Un process combinatoire de calcul des sorties à partir des entrées et de l'état présent :

Listing 6: Code Verilog du process combinatoire des sorties

```

process @ (Etat_actuel)
begin

  case (Etat_actuel)
    Etat0 : S = 1b'0;
    Etat1 : S = 1b'0;
    Etat2 : S = 1b'1;
  endcase

end
```

A.1.3 Exemples d'implémentation Verilog des FSMs

Cette section présente quelques exemples d'implémentation en Verilog des machines à états finis de Mealy et de Moore. Ces exemples sont tirés de la suite des tests PREP [32] de [MFA93][Gol94].

32. PRogrammable Electronics Performance corporation.

A.1.3.1 Machine de Mealy prep3 est une machine à états finis de Mealy avec 8 états et 12 transitions. Elle a 8 entrées et 8 sorties comme le montre la figure 41.

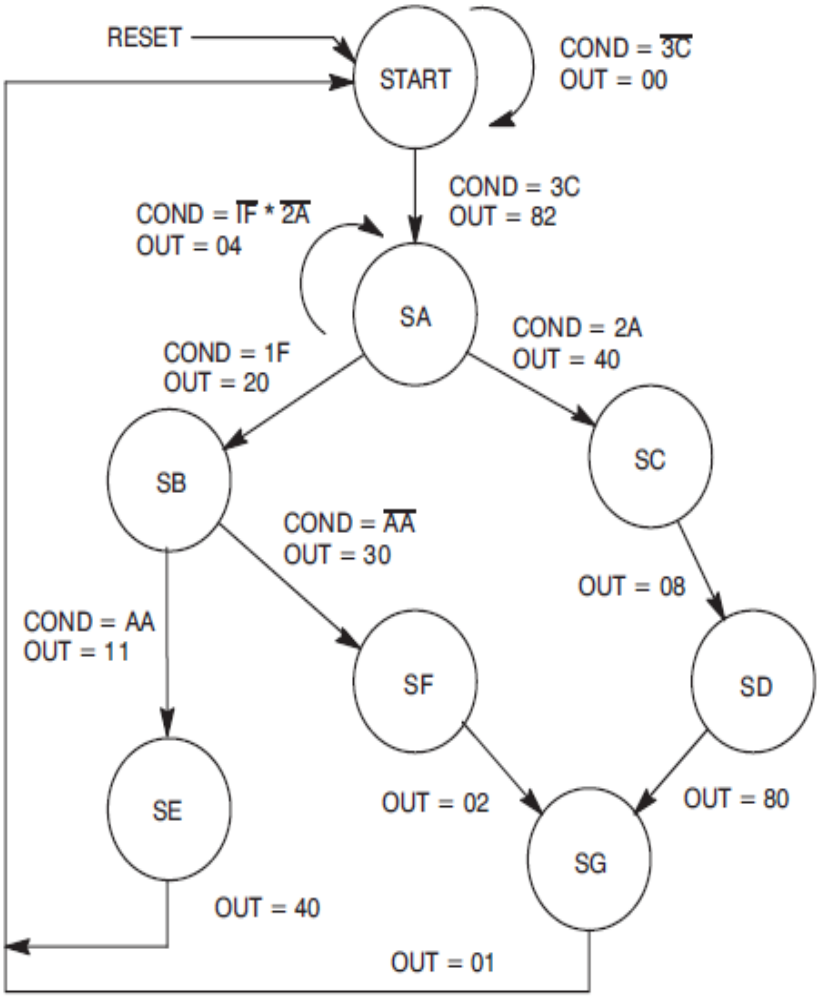

FIG. 41 – Diagramme de transition d'états.

Listing 7: Code verilog d'une FSM de Mealy

```
module prep3 (clk, rst, in, out) ;
input clk, rst ;
input [7:0] in ;
output [7:0] out ;

parameter [2:0] // synopsys enum code
START = 3'd0 ,
SA = 3'd1 ,
```

```
SB = 3'd2 ,
SC = 3'd3 ,
SD = 3'd4 ,
SE = 3'd5 ,
SF = 3'd6 ,
SG = 3'd7 ;

// synopsys state_vector state
reg [2:0] // synopsys enum code
state, next_state ;
reg [7:0] out, next_out ;

always @ (in or state) begin

// default values
next_state = START ;
next_out = 8'bx ;

// state machine
case (state) // synopsys parallel_case full_case
START:
       if (in == 8'h3c) begin
       next_state = SA ;
       next_out = 8'h82 ;
       end
       else begin
       next_state = START ;
       next_out = 8'h00 ;
       end
SA:
       case (in) // synopsys parallel_case full_case
       8'h2a:
              begin
              next_state = SC ;
              next_out = 8'h40 ;
              end
       8'h1f:
              begin
              next_state = SB ;
              next_out = 8'h20 ;
              end
       default:
              begin
              next_state = SA ;
              next_out = 8'h04 ;
              end
       endcase
SB:
       if (in == 8'haa) begin
       next_state = SE ;
```

```
        next_out = 8'h11 ;
        end
        else begin
        next_state = SF ;
        next_out = 8'h30 ;
        end
SC:
        begin
        next_state = SD ;
        next_out = 8'h08 ;
        end
SD:
        begin
        next_state = SG ;
        next_out = 8'h80 ;
        end
SE:
        begin
        next_state = START ;
        next_out = 8'h40 ;
        end
SF:
        begin
        next_state = SG ;
        next_out = 8'h02 ;
        end
SG:
        begin
        next_state = START ;
        next_out = 8'h01 ;
        end
endcase
end

// build the state flops
always @ (posedge clk or negedge rst)
begin
if (!rst) state <= #1 START ;
else state <= #1 next_state ;
end

// build the output flops
always @ (posedge clk or negedge rst)
begin
if (!rst) out <= #1 8'b0 ;
else out <= #1 next_out ;
end

endmodule
```

A.1.3.2 Machine de Moore prep4 est une machine à états finis de Moore avec 16 états et 40 transitions. Elle a 8 entrées et 8 sorties comme le montre le diagramme de la figure 42.

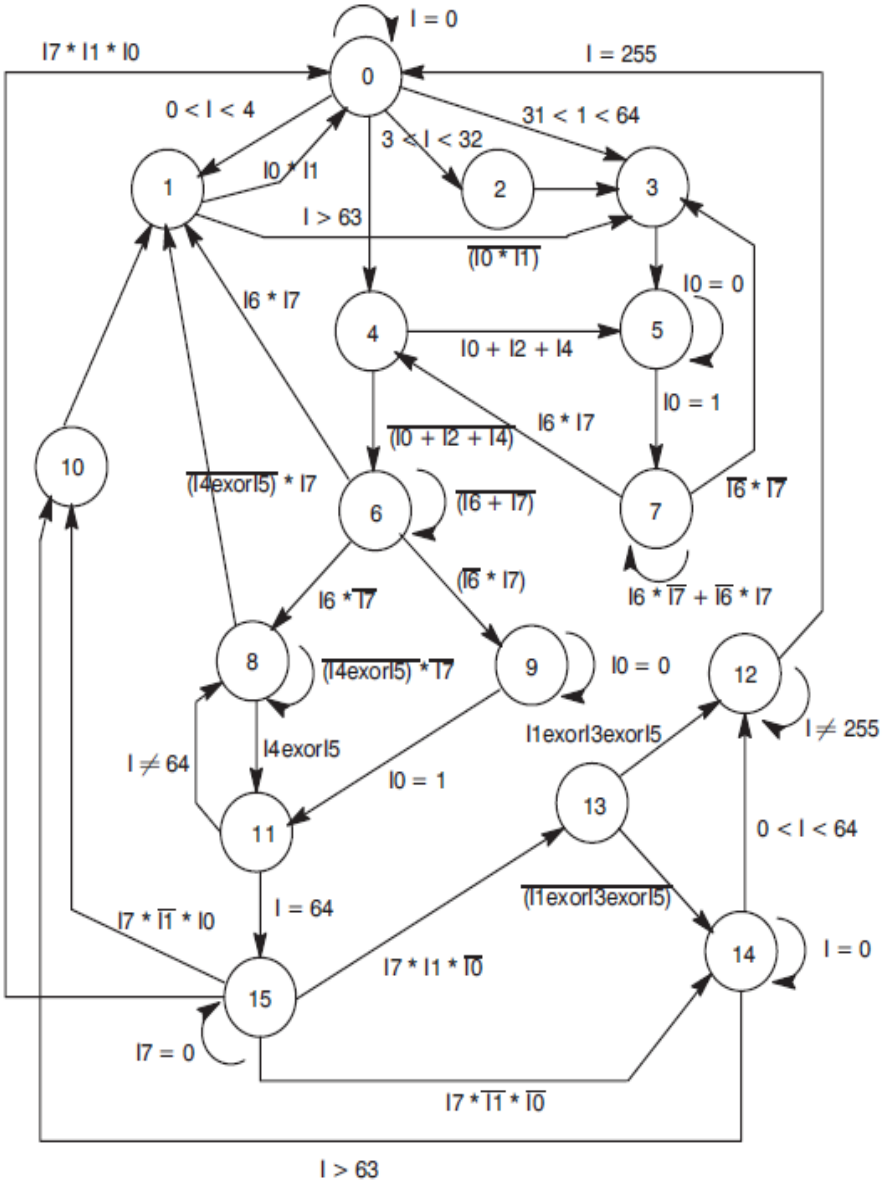

FIG. 42 – Diagramme de transition d'états..

Listing 8: Code Verilog d'une FSM de Moore

```
module prep4 (clk, rst, in, out) ;

input clk, rst ;
input [7:0] in ;
output [7:0] out ;
```

```
parameter [3:0] // synopsys enum code
S0 = 4'd0 , S1 = 4'd1 , S2 = 4'd2 , S3 = 4'd3 ,
S4 = 4'd4 , S5 = 4'd5 , S6 = 4'd6 , S7 = 4'd7 ,
S8 = 4'd8 , S9 = 4'd9 , S10 = 4'd10 , S11 = 4'd11 ,
S12 = 4'd12 , S13 = 4'd13 , S14 = 4'd14 , S15 = 4'd15 ;

// synopsys state_vector state
reg [3:0] /* synopsys enum code */ state, next_state ;
reg [7:0] out ;

// state machine

always @ (in or state) begin

// default value
next_state = S0 ; // always overridden

case (state) // synopsys parallel_case full_case

  S0: case(1'b1) // synopsys parallel_case full_case
       (in == 8'd0): next_state = S0 ;
              (8'd0 < in && in < 8'd4): next_state = S1 ;
              (8'd3 < in && in < 8'd32): next_state = S2 ;
              (8'd31 < in && in < 8'd64): next_state = S3 ;
              (in > 8'd63): next_state = S4 ;
      endcase

  S1: if (in[0] && in[1]) next_state = S0 ;
      else next_state = S3 ;

  S2: next_state = S3 ;

  S3: next_state = S5 ;

  S4: if (in[0] || in[2] || in[4]) next_state = S5 ;
      else next_state = S6 ;

  S5: if (in[0] == 1'b0) next_state = S5 ;
      else next_state = S7 ;

  S6: case(in[7:6]) // synopsys parallel_case full_case
       2'b11: next_state = S1 ;
              2'b00: next_state = S6 ;
              2'b01: next_state = S8 ;
              2'b10: next_state = S9 ;
        endcase

  S7: case(in[7:6]) // synopsys parallel_case full_case
              2'b00: next_state = S3 ;
```

```
              2'b11: next_state = S4 ;
              2'b10,
              2'b01: next_state = S7 ;
         endcase

  S8: if(in[4] ^ in[5]) next_state = S11 ;
     else if (in[7]) next_state = S1 ;
     else next_state = S8 ;

  S9: if (in[0] == 1'b0) next_state = S9 ;
     else next_state = S11 ;

  S10: next_state = S1 ;

  S11: if (in == 8'd64) next_state = S15 ;
      else next_state = S8 ;

  S12: if (in == 8'd255) next_state = S0 ;
         else next_state = S12 ;

  S13: if (in[1] ^ in[3] ^ in[5]) next_state = S12 ;
      else next_state = S14 ;

  S14: case(1'b1) // synopsys parallel_case full_case
              (in == 8'd0): next_state = S14 ;
              (8'd0 < in && in < 8'd64): next_state = S12 ;
              (in > 8'd63): next_state = S10 ;
         endcase

  S15: if (in[7] == 1'b0) next_state = S15 ;
      else
        case (in[1:0])
                 // synopsys parallel_case full_case
                 2'b11: next_state = S0 ;
                 2'b01: next_state = S10 ;
                 2'b10: next_state = S13 ;
                 2'b00: next_state = S14 ;
               endcase
  endcase
end

// outputs
always @ (state) begin

// default value
out = 8'bx ;

case (state) // synopsys parallel_case full_case
  S0: out = 8'b00000000 ;
  S1: out = 8'b00000110 ;
```

```
  S2: out = 8'b00011000 ;
  S3: out = 8'b01100000 ;
  S4: begin
        out[7] = 1'b1 ; out[0] = 1'b0 ;
      end
  S5: begin
        out[6] = 1'b1 ; out[1] = 1'b0 ;
      end
  S6: out = 8'b00011111 ;
  S7: out = 8'b00111111 ;
  S8: out = 8'b01111111 ;
  S9: out = 8'b11111111 ;
  S10: begin
         out[6] = 1'b1 ; out[4] = 1'b1 ;
         out[2] = 1'b1 ; out[0] = 1'b1 ;
       end
  S11: begin
         out[7] = 1'b1 ; out[5] = 1'b1 ;
         out[3] = 1'b1 ; out[1] = 1'b1 ;
       end
  S12: out = 8'b11111101 ;
  S13: out = 8'b11110111 ;
  S14: out = 8'b11011111 ;
  S15: out = 8'b01111111 ;
  endcase
end

// build the state flops
always @ (posedge clk or negedge rst) begin
  if (!rst) state <= #1 S0 ;
  else state <= #1 next_state ;
end

endmodule
```

B Les Fichiers d'entrée de notre trafic

Dans cette annexe, nous exposons les fichiers d'entrée utilisés pour modéliser le trafic uniforme et le trafic transpose. Pour faire simple, on se contentera de donner le listing des fichiers d'entrée du nœud de routage à l'adresse (1,0) avec une charge de 50%.

Trafic uniforme : listing du fichier d'entrée du routeur 10

```
   1 1002 000A 0010 0000 0000 0000 0001 0000 0065 000D 000E 000F
 141 1021 000A 0010 0000 0000 0000 0141 0000 0066 000D 000E 000F
 281 1021 000A 0010 0000 0000 0000 0281 0000 0067 000D 000E 000F
 3C1 1011 000A 0010 0000 0000 0000 03C1 0000 0068 000D 000E 000F
 501 1002 000A 0010 0000 0000 0000 0501 0000 0069 000D 000E 000F
 641 1000 000A 0010 0000 0000 0000 0641 0000 006A 000D 000E 000F
 781 1011 000A 0010 0000 0000 0000 0781 0000 006B 000D 000E 000F
 8C1 1002 000A 0010 0000 0000 0000 08C1 0000 006C 000D 000E 000F
 A01 1020 000A 0010 0000 0000 0000 0A01 0000 006D 000D 000E 000F
 B41 1001 000A 0010 0000 0000 0000 0B41 0000 006E 000D 000E 000F
 C81 1020 000A 0010 0000 0000 0000 0C81 0000 006F 000D 000E 000F
 DC1 1000 000A 0010 0000 0000 0000 0DC1 0000 0070 000D 000E 000F
 F01 1011 000A 0010 0000 0000 0000 0F01 0000 0071 000D 000E 000F
1041 1001 000A 0010 0000 0000 0000 1041 0000 0072 000D 000E 000F
1181 1012 000A 0010 0000 0000 0000 1181 0000 0073 000D 000E 000F
12C1 1021 000A 0010 0000 0000 0000 12C1 0000 0074 000D 000E 000F
1401 1001 000A 0010 0000 0000 0000 1401 0000 0075 000D 000E 000F
1541 1002 000A 0010 0000 0000 0000 1541 0000 0076 000D 000E 000F
1681 1021 000A 0010 0000 0000 0000 1681 0000 0077 000D 000E 000F
17C1 1011 000A 0010 0000 0000 0000 17C1 0000 0078 000D 000E 000F
1901 1020 000A 0010 0000 0000 0000 1901 0000 0079 000D 000E 000F
1A41 1021 000A 0010 0000 0000 0000 1A41 0000 007A 000D 000E 000F
1B81 1020 000A 0010 0000 0000 0000 1B81 0000 007B 000D 000E 000F
1CC1 1001 000A 0010 0000 0000 0000 1CC1 0000 007C 000D 000E 000F
1E01 1020 000A 0010 0000 0000 0000 1E01 0000 007D 000D 000E 000F
1F41 1022 000A 0010 0000 0000 0000 1F41 0000 007E 000D 000E 000F
2081 1001 000A 0010 0000 0000 0000 2081 0000 007F 000D 000E 000F
21C1 1020 000A 0010 0000 0000 0000 21C1 0000 0080 000D 000E 000F
2301 1000 000A 0010 0000 0000 0000 2301 0000 0081 000D 000E 000F
2441 1000 000A 0010 0000 0000 0000 2441 0000 0082 000D 000E 000F
2581 1012 000A 0010 0000 0000 0000 2581 0000 0083 000D 000E 000F
26C1 1020 000A 0010 0000 0000 0000 26C1 0000 0084 000D 000E 000F
2801 1011 000A 0010 0000 0000 0000 2801 0000 0085 000D 000E 000F
2941 1001 000A 0010 0000 0000 0000 2941 0000 0086 000D 000E 000F
2A81 1002 000A 0010 0000 0000 0000 2A81 0000 0087 000D 000E 000F
2BC1 1020 000A 0010 0000 0000 0000 2BC1 0000 0088 000D 000E 000F
2D01 1021 000A 0010 0000 0000 0000 2D01 0000 0089 000D 000E 000F
2E41 1020 000A 0010 0000 0000 0000 2E41 0000 008A 000D 000E 000F
2F81 1002 000A 0010 0000 0000 0000 2F81 0000 008B 000D 000E 000F
30C1 1011 000A 0010 0000 0000 0000 30C1 0000 008C 000D 000E 000F
3201 1020 000A 0010 0000 0000 0000 3201 0000 008D 000D 000E 000F
3341 1020 000A 0010 0000 0000 0000 3341 0000 008E 000D 000E 000F
```

```
3481 1012 000A 0010 0000 0000 0000 3481 0000 008F 000D 000E 000F
35C1 1022 000A 0010 0000 0000 0000 35C1 0000 0090 000D 000E 000F
3701 1021 000A 0010 0000 0000 0000 3701 0000 0091 000D 000E 000F
3841 1021 000A 0010 0000 0000 0000 3841 0000 0092 000D 000E 000F
3981 1011 000A 0010 0000 0000 0000 3981 0000 0093 000D 000E 000F
3AC1 1022 000A 0010 0000 0000 0000 3AC1 0000 0094 000D 000E 000F
3C01 1011 000A 0010 0000 0000 0000 3C01 0000 0095 000D 000E 000F
3D41 1002 000A 0010 0000 0000 0000 3D41 0000 0096 000D 000E 000F
3E81 1012 000A 0010 0000 0000 0000 3E81 0000 0097 000D 000E 000F
3FC1 1021 000A 0010 0000 0000 0000 3FC1 0000 0098 000D 000E 000F
4101 1022 000A 0010 0000 0000 0000 4101 0000 0099 000D 000E 000F
4241 1002 000A 0010 0000 0000 0000 4241 0000 009A 000D 000E 000F
4381 1000 000A 0010 0000 0000 0000 4381 0000 009B 000D 000E 000F
44C1 1022 000A 0010 0000 0000 0000 44C1 0000 009C 000D 000E 000F
4601 1011 000A 0010 0000 0000 0000 4601 0000 009D 000D 000E 000F
4741 1000 000A 0010 0000 0000 0000 4741 0000 009E 000D 000E 000F
4881 1012 000A 0010 0000 0000 0000 4881 0000 009F 000D 000E 000F
49C1 1011 000A 0010 0000 0000 0000 49C1 0000 00A0 000D 000E 000F
4B01 1020 000A 0010 0000 0000 0000 4B01 0000 00A1 000D 000E 000F
4C41 1022 000A 0010 0000 0000 0000 4C41 0000 00A2 000D 000E 000F
4D81 1021 000A 0010 0000 0000 0000 4D81 0000 00A3 000D 000E 000F
4EC1 1020 000A 0010 0000 0000 0000 4EC1 0000 00A4 000D 000E 000F
5001 1001 000A 0010 0000 0000 0000 5001 0000 00A5 000D 000E 000F
5141 1012 000A 0010 0000 0000 0000 5141 0000 00A6 000D 000E 000F
5281 1022 000A 0010 0000 0000 0000 5281 0000 00A7 000D 000E 000F
53C1 1012 000A 0010 0000 0000 0000 53C1 0000 00A8 000D 000E 000F
5501 1022 000A 0010 0000 0000 0000 5501 0000 00A9 000D 000E 000F
5641 1001 000A 0010 0000 0000 0000 5641 0000 00AA 000D 000E 000F
5781 1012 000A 0010 0000 0000 0000 5781 0000 00AB 000D 000E 000F
58C1 1021 000A 0010 0000 0000 0000 58C1 0000 00AC 000D 000E 000F
5A01 1020 000A 0010 0000 0000 0000 5A01 0000 00AD 000D 000E 000F
5B41 1001 000A 0010 0000 0000 0000 5B41 0000 00AE 000D 000E 000F
5C81 1002 000A 0010 0000 0000 0000 5C81 0000 00AF 000D 000E 000F
5DC1 1002 000A 0010 0000 0000 0000 5DC1 0000 00B0 000D 000E 000F
5F01 1000 000A 0010 0000 0000 0000 5F01 0000 00B1 000D 000E 000F
6041 1020 000A 0010 0000 0000 0000 6041 0000 00B2 000D 000E 000F
6181 1012 000A 0010 0000 0000 0000 6181 0000 00B3 000D 000E 000F
62C1 1012 000A 0010 0000 0000 0000 62C1 0000 00B4 000D 000E 000F
6401 1022 000A 0010 0000 0000 0000 6401 0000 00B5 000D 000E 000F
6541 1020 000A 0010 0000 0000 0000 6541 0000 00B6 000D 000E 000F
6681 1021 000A 0010 0000 0000 0000 6681 0000 00B7 000D 000E 000F
67C1 1012 000A 0010 0000 0000 0000 67C1 0000 00B8 000D 000E 000F
6901 1012 000A 0010 0000 0000 0000 6901 0000 00B9 000D 000E 000F
6A41 1011 000A 0010 0000 0000 0000 6A41 0000 00BA 000D 000E 000F
6B81 1001 000A 0010 0000 0000 0000 6B81 0000 00BB 000D 000E 000F
6CC1 1000 000A 0010 0000 0000 0000 6CC1 0000 00BC 000D 000E 000F
6E01 1002 000A 0010 0000 0000 0000 6E01 0000 00BD 000D 000E 000F
6F41 1000 000A 0010 0000 0000 0000 6F41 0000 00BE 000D 000E 000F
7081 1020 000A 0010 0000 0000 0000 7081 0000 00BF 000D 000E 000F
71C1 1000 000A 0010 0000 0000 0000 71C1 0000 00C0 000D 000E 000F
```

```
7301 1020 000A 0010 0000 0000 0000 7301 0000 00C1 000D 000E 000F
7441 1001 000A 0010 0000 0000 0000 7441 0000 00C2 000D 000E 000F
7581 1002 000A 0010 0000 0000 0000 7581 0000 00C3 000D 000E 000F
76C1 1020 000A 0010 0000 0000 0000 76C1 0000 00C4 000D 000E 000F
7801 1022 000A 0010 0000 0000 0000 7801 0000 00C5 000D 000E 000F
7941 1021 000A 0010 0000 0000 0000 7941 0000 00C6 000D 000E 000F
7A81 1020 000A 0010 0000 0000 0000 7A81 0000 00C7 000D 000E 000F
7BC1 1002 000A 0010 0000 0000 0000 7BC1 0000 00C8 000D 000E 000F
```

Trafic transpose1 : listing du fichier d'entrée du routeur 10

```
   1 1001 000A 0010 0000 0000 0000 0001 0000 0001 000D 000E 000F
 141 1001 000A 0010 0000 0000 0000 0141 0000 0002 000D 000E 000F
 281 1001 000A 0010 0000 0000 0000 0281 0000 0003 000D 000E 000F
 3C1 1001 000A 0010 0000 0000 0000 03C1 0000 0004 000D 000E 000F
 501 1001 000A 0010 0000 0000 0000 0501 0000 0005 000D 000E 000F
 641 1001 000A 0010 0000 0000 0000 0641 0000 0006 000D 000E 000F
 781 1001 000A 0010 0000 0000 0000 0781 0000 0007 000D 000E 000F
 8C1 1001 000A 0010 0000 0000 0000 08C1 0000 0008 000D 000E 000F
 A01 1001 000A 0010 0000 0000 0000 0A01 0000 0009 000D 000E 000F
 B41 1001 000A 0010 0000 0000 0000 0B41 0000 000A 000D 000E 000F
 C81 1001 000A 0010 0000 0000 0000 0C81 0000 000B 000D 000E 000F
 DC1 1001 000A 0010 0000 0000 0000 0DC1 0000 000C 000D 000E 000F
 F01 1001 000A 0010 0000 0000 0000 0F01 0000 000D 000D 000E 000F
1041 1001 000A 0010 0000 0000 0000 1041 0000 000E 000D 000E 000F
1181 1001 000A 0010 0000 0000 0000 1181 0000 000F 000D 000E 000F
12C1 1001 000A 0010 0000 0000 0000 12C1 0000 0010 000D 000E 000F
1401 1001 000A 0010 0000 0000 0000 1401 0000 0011 000D 000E 000F
1541 1001 000A 0010 0000 0000 0000 1541 0000 0012 000D 000E 000F
1681 1001 000A 0010 0000 0000 0000 1681 0000 0013 000D 000E 000F
17C1 1001 000A 0010 0000 0000 0000 17C1 0000 0014 000D 000E 000F
1901 1001 000A 0010 0000 0000 0000 1901 0000 0015 000D 000E 000F
1A41 1001 000A 0010 0000 0000 0000 1A41 0000 0016 000D 000E 000F
1B81 1001 000A 0010 0000 0000 0000 1B81 0000 0017 000D 000E 000F
1CC1 1001 000A 0010 0000 0000 0000 1CC1 0000 0018 000D 000E 000F
1E01 1001 000A 0010 0000 0000 0000 1E01 0000 0019 000D 000E 000F
1F41 1001 000A 0010 0000 0000 0000 1F41 0000 001A 000D 000E 000F
2081 1001 000A 0010 0000 0000 0000 2081 0000 001B 000D 000E 000F
21C1 1001 000A 0010 0000 0000 0000 21C1 0000 001C 000D 000E 000F
2301 1001 000A 0010 0000 0000 0000 2301 0000 001D 000D 000E 000F
2441 1001 000A 0010 0000 0000 0000 2441 0000 001E 000D 000E 000F
2581 1001 000A 0010 0000 0000 0000 2581 0000 001F 000D 000E 000F
26C1 1001 000A 0010 0000 0000 0000 26C1 0000 0020 000D 000E 000F
2801 1001 000A 0010 0000 0000 0000 2801 0000 0021 000D 000E 000F
2941 1001 000A 0010 0000 0000 0000 2941 0000 0022 000D 000E 000F
2A81 1001 000A 0010 0000 0000 0000 2A81 0000 0023 000D 000E 000F
2BC1 1001 000A 0010 0000 0000 0000 2BC1 0000 0024 000D 000E 000F
2D01 1001 000A 0010 0000 0000 0000 2D01 0000 0025 000D 000E 000F
2E41 1001 000A 0010 0000 0000 0000 2E41 0000 0026 000D 000E 000F
2F81 1001 000A 0010 0000 0000 0000 2F81 0000 0027 000D 000E 000F
```

```
30C1 1001 000A 0010 0000 0000 0000 30C1 0000 0028 000D 000E 000F
3201 1001 000A 0010 0000 0000 0000 3201 0000 0029 000D 000E 000F
3341 1001 000A 0010 0000 0000 0000 3341 0000 002A 000D 000E 000F
3481 1001 000A 0010 0000 0000 0000 3481 0000 002B 000D 000E 000F
35C1 1001 000A 0010 0000 0000 0000 35C1 0000 002C 000D 000E 000F
3701 1001 000A 0010 0000 0000 0000 3701 0000 002D 000D 000E 000F
3841 1001 000A 0010 0000 0000 0000 3841 0000 002E 000D 000E 000F
3981 1001 000A 0010 0000 0000 0000 3981 0000 002F 000D 000E 000F
3AC1 1001 000A 0010 0000 0000 0000 3AC1 0000 0030 000D 000E 000F
3C01 1001 000A 0010 0000 0000 0000 3C01 0000 0031 000D 000E 000F
3D41 1001 000A 0010 0000 0000 0000 3D41 0000 0032 000D 000E 000F
3E81 1001 000A 0010 0000 0000 0000 3E81 0000 0033 000D 000E 000F
3FC1 1001 000A 0010 0000 0000 0000 3FC1 0000 0034 000D 000E 000F
4101 1001 000A 0010 0000 0000 0000 4101 0000 0035 000D 000E 000F
4241 1001 000A 0010 0000 0000 0000 4241 0000 0036 000D 000E 000F
4381 1001 000A 0010 0000 0000 0000 4381 0000 0037 000D 000E 000F
44C1 1001 000A 0010 0000 0000 0000 44C1 0000 0038 000D 000E 000F
4601 1001 000A 0010 0000 0000 0000 4601 0000 0039 000D 000E 000F
4741 1001 000A 0010 0000 0000 0000 4741 0000 003A 000D 000E 000F
4881 1001 000A 0010 0000 0000 0000 4881 0000 003B 000D 000E 000F
49C1 1001 000A 0010 0000 0000 0000 49C1 0000 003C 000D 000E 000F
4B01 1001 000A 0010 0000 0000 0000 4B01 0000 003D 000D 000E 000F
4C41 1001 000A 0010 0000 0000 0000 4C41 0000 003E 000D 000E 000F
4D81 1001 000A 0010 0000 0000 0000 4D81 0000 003F 000D 000E 000F
4EC1 1001 000A 0010 0000 0000 0000 4EC1 0000 0040 000D 000E 000F
5001 1001 000A 0010 0000 0000 0000 5001 0000 0041 000D 000E 000F
5141 1001 000A 0010 0000 0000 0000 5141 0000 0042 000D 000E 000F
5281 1001 000A 0010 0000 0000 0000 5281 0000 0043 000D 000E 000F
53C1 1001 000A 0010 0000 0000 0000 53C1 0000 0044 000D 000E 000F
5501 1001 000A 0010 0000 0000 0000 5501 0000 0045 000D 000E 000F
5641 1001 000A 0010 0000 0000 0000 5641 0000 0046 000D 000E 000F
5781 1001 000A 0010 0000 0000 0000 5781 0000 0047 000D 000E 000F
58C1 1001 000A 0010 0000 0000 0000 58C1 0000 0048 000D 000E 000F
5A01 1001 000A 0010 0000 0000 0000 5A01 0000 0049 000D 000E 000F
5B41 1001 000A 0010 0000 0000 0000 5B41 0000 004A 000D 000E 000F
5C81 1001 000A 0010 0000 0000 0000 5C81 0000 004B 000D 000E 000F
5DC1 1001 000A 0010 0000 0000 0000 5DC1 0000 004C 000D 000E 000F
5F01 1001 000A 0010 0000 0000 0000 5F01 0000 004D 000D 000E 000F
6041 1001 000A 0010 0000 0000 0000 6041 0000 004E 000D 000E 000F
6181 1001 000A 0010 0000 0000 0000 6181 0000 004F 000D 000E 000F
62C1 1001 000A 0010 0000 0000 0000 62C1 0000 0050 000D 000E 000F
6401 1001 000A 0010 0000 0000 0000 6401 0000 0051 000D 000E 000F
6541 1001 000A 0010 0000 0000 0000 6541 0000 0052 000D 000E 000F
6681 1001 000A 0010 0000 0000 0000 6681 0000 0053 000D 000E 000F
67C1 1001 000A 0010 0000 0000 0000 67C1 0000 0054 000D 000E 000F
6901 1001 000A 0010 0000 0000 0000 6901 0000 0055 000D 000E 000F
6A41 1001 000A 0010 0000 0000 0000 6A41 0000 0056 000D 000E 000F
6B81 1001 000A 0010 0000 0000 0000 6B81 0000 0057 000D 000E 000F
6CC1 1001 000A 0010 0000 0000 0000 6CC1 0000 0058 000D 000E 000F
6E01 1001 000A 0010 0000 0000 0000 6E01 0000 0059 000D 000E 000F
```

```
6F41 1001 000A 0010 0000 0000 0000 6F41 0000 005A 000D 000E 000F
7081 1001 000A 0010 0000 0000 0000 7081 0000 005B 000D 000E 000F
71C1 1001 000A 0010 0000 0000 0000 71C1 0000 005C 000D 000E 000F
7301 1001 000A 0010 0000 0000 0000 7301 0000 005D 000D 000E 000F
7441 1001 000A 0010 0000 0000 0000 7441 0000 005E 000D 000E 000F
7581 1001 000A 0010 0000 0000 0000 7581 0000 005F 000D 000E 000F
76C1 1001 000A 0010 0000 0000 0000 76C1 0000 0060 000D 000E 000F
7801 1001 000A 0010 0000 0000 0000 7801 0000 0061 000D 000E 000F
7941 1001 000A 0010 0000 0000 0000 7941 0000 0062 000D 000E 000F
7A81 1001 000A 0010 0000 0000 0000 7A81 0000 0063 000D 000E 000F
7BC1 1001 000A 0010 0000 0000 0000 7BC1 0000 0064 000D 000E 000F
7D01 1001 000A 0010 0000 0000 0000 7D01 0000 0065 000D 000E 000F
7E41 1001 000A 0010 0000 0000 0000 7E41 0000 0066 000D 000E 000F
7F81 1001 000A 0010 0000 0000 0000 7F81 0000 0067 000D 000E 000F
80C1 1001 000A 0010 0000 0000 0000 80C1 0000 0068 000D 000E 000F
8201 1001 000A 0010 0000 0000 0000 8201 0000 0069 000D 000E 000F
8341 1001 000A 0010 0000 0000 0000 8341 0000 006A 000D 000E 000F
8481 1001 000A 0010 0000 0000 0000 8481 0000 006B 000D 000E 000F
85C1 1001 000A 0010 0000 0000 0000 85C1 0000 006C 000D 000E 000F
8701 1001 000A 0010 0000 0000 0000 8701 0000 006D 000D 000E 000F
8841 1001 000A 0010 0000 0000 0000 8841 0000 006E 000D 000E 000F
8981 1001 000A 0010 0000 0000 0000 8981 0000 006F 000D 000E 000F
8AC1 1001 000A 0010 0000 0000 0000 8AC1 0000 0070 000D 000E 000F
8C01 1001 000A 0010 0000 0000 0000 8C01 0000 0071 000D 000E 000F
8D41 1001 000A 0010 0000 0000 0000 8D41 0000 0072 000D 000E 000F
8E81 1001 000A 0010 0000 0000 0000 8E81 0000 0073 000D 000E 000F
8FC1 1001 000A 0010 0000 0000 0000 8FC1 0000 0074 000D 000E 000F
9101 1001 000A 0010 0000 0000 0000 9101 0000 0075 000D 000E 000F
9241 1001 000A 0010 0000 0000 0000 9241 0000 0076 000D 000E 000F
9381 1001 000A 0010 0000 0000 0000 9381 0000 0077 000D 000E 000F
94C1 1001 000A 0010 0000 0000 0000 94C1 0000 0078 000D 000E 000F
9601 1001 000A 0010 0000 0000 0000 9601 0000 0079 000D 000E 000F
9741 1001 000A 0010 0000 0000 0000 9741 0000 007A 000D 000E 000F
9881 1001 000A 0010 0000 0000 0000 9881 0000 007B 000D 000E 000F
99C1 1001 000A 0010 0000 0000 0000 99C1 0000 007C 000D 000E 000F
9B01 1001 000A 0010 0000 0000 0000 9B01 0000 007D 000D 000E 000F
9C41 1001 000A 0010 0000 0000 0000 9C41 0000 007E 000D 000E 000F
9D81 1001 000A 0010 0000 0000 0000 9D81 0000 007F 000D 000E 000F
9EC1 1001 000A 0010 0000 0000 0000 9EC1 0000 0080 000D 000E 000F
A001 1001 000A 0010 0000 0000 0000 A001 0000 0081 000D 000E 000F
A141 1001 000A 0010 0000 0000 0000 A141 0000 0082 000D 000E 000F
A281 1001 000A 0010 0000 0000 0000 A281 0000 0083 000D 000E 000F
A3C1 1001 000A 0010 0000 0000 0000 A3C1 0000 0084 000D 000E 000F
A501 1001 000A 0010 0000 0000 0000 A501 0000 0085 000D 000E 000F
A641 1001 000A 0010 0000 0000 0000 A641 0000 0086 000D 000E 000F
A781 1001 000A 0010 0000 0000 0000 A781 0000 0087 000D 000E 000F
A8C1 1001 000A 0010 0000 0000 0000 A8C1 0000 0088 000D 000E 000F
AA01 1001 000A 0010 0000 0000 0000 AA01 0000 0089 000D 000E 000F
AB41 1001 000A 0010 0000 0000 0000 AB41 0000 008A 000D 000E 000F
AC81 1001 000A 0010 0000 0000 0000 AC81 0000 008B 000D 000E 000F
```

```
ADC1 1001 000A 0010 0000 0000 0000 ADC1 0000 008C 000D 000E 000F
AF01 1001 000A 0010 0000 0000 0000 AF01 0000 008D 000D 000E 000F
B041 1001 000A 0010 0000 0000 0000 B041 0000 008E 000D 000E 000F
B181 1001 000A 0010 0000 0000 0000 B181 0000 008F 000D 000E 000F
B2C1 1001 000A 0010 0000 0000 0000 B2C1 0000 0090 000D 000E 000F
B401 1001 000A 0010 0000 0000 0000 B401 0000 0091 000D 000E 000F
B541 1001 000A 0010 0000 0000 0000 B541 0000 0092 000D 000E 000F
B681 1001 000A 0010 0000 0000 0000 B681 0000 0093 000D 000E 000F
B7C1 1001 000A 0010 0000 0000 0000 B7C1 0000 0094 000D 000E 000F
B901 1001 000A 0010 0000 0000 0000 B901 0000 0095 000D 000E 000F
BA41 1001 000A 0010 0000 0000 0000 BA41 0000 0096 000D 000E 000F
```

Trafic transpose2 : listing du fichier d'entrée du routeur 10

```
   1 1021 000A 0010 0000 0000 0000 0001 0000 0097 000D 000E 000F
 141 1021 000A 0010 0000 0000 0000 0141 0000 0098 000D 000E 000F
 281 1021 000A 0010 0000 0000 0000 0281 0000 0099 000D 000E 000F
 3C1 1021 000A 0010 0000 0000 0000 03C1 0000 009A 000D 000E 000F
 501 1021 000A 0010 0000 0000 0000 0501 0000 009B 000D 000E 000F
 641 1021 000A 0010 0000 0000 0000 0641 0000 009C 000D 000E 000F
 781 1021 000A 0010 0000 0000 0000 0781 0000 009D 000D 000E 000F
 8C1 1021 000A 0010 0000 0000 0000 08C1 0000 009E 000D 000E 000F
 A01 1021 000A 0010 0000 0000 0000 0A01 0000 009F 000D 000E 000F
 B41 1021 000A 0010 0000 0000 0000 0B41 0000 00A0 000D 000E 000F
 C81 1021 000A 0010 0000 0000 0000 0C81 0000 00A1 000D 000E 000F
 DC1 1021 000A 0010 0000 0000 0000 0DC1 0000 00A2 000D 000E 000F
 F01 1021 000A 0010 0000 0000 0000 0F01 0000 00A3 000D 000E 000F
1041 1021 000A 0010 0000 0000 0000 1041 0000 00A4 000D 000E 000F
1181 1021 000A 0010 0000 0000 0000 1181 0000 00A5 000D 000E 000F
12C1 1021 000A 0010 0000 0000 0000 12C1 0000 00A6 000D 000E 000F
1401 1021 000A 0010 0000 0000 0000 1401 0000 00A7 000D 000E 000F
1541 1021 000A 0010 0000 0000 0000 1541 0000 00A8 000D 000E 000F
1681 1021 000A 0010 0000 0000 0000 1681 0000 00A9 000D 000E 000F
17C1 1021 000A 0010 0000 0000 0000 17C1 0000 00AA 000D 000E 000F
1901 1021 000A 0010 0000 0000 0000 1901 0000 00AB 000D 000E 000F
1A41 1021 000A 0010 0000 0000 0000 1A41 0000 00AC 000D 000E 000F
1B81 1021 000A 0010 0000 0000 0000 1B81 0000 00AD 000D 000E 000F
1CC1 1021 000A 0010 0000 0000 0000 1CC1 0000 00AE 000D 000E 000F
1E01 1021 000A 0010 0000 0000 0000 1E01 0000 00AF 000D 000E 000F
1F41 1021 000A 0010 0000 0000 0000 1F41 0000 00B0 000D 000E 000F
2081 1021 000A 0010 0000 0000 0000 2081 0000 00B1 000D 000E 000F
21C1 1021 000A 0010 0000 0000 0000 21C1 0000 00B2 000D 000E 000F
2301 1021 000A 0010 0000 0000 0000 2301 0000 00B3 000D 000E 000F
2441 1021 000A 0010 0000 0000 0000 2441 0000 00B4 000D 000E 000F
2581 1021 000A 0010 0000 0000 0000 2581 0000 00B5 000D 000E 000F
26C1 1021 000A 0010 0000 0000 0000 26C1 0000 00B6 000D 000E 000F
2801 1021 000A 0010 0000 0000 0000 2801 0000 00B7 000D 000E 000F
2941 1021 000A 0010 0000 0000 0000 2941 0000 00B8 000D 000E 000F
2A81 1021 000A 0010 0000 0000 0000 2A81 0000 00B9 000D 000E 000F
2BC1 1021 000A 0010 0000 0000 0000 2BC1 0000 00BA 000D 000E 000F
```

```
2D01 1021 000A 0010 0000 0000 0000 2D01 0000 00BB 000D 000E 000F
2E41 1021 000A 0010 0000 0000 0000 2E41 0000 00BC 000D 000E 000F
2F81 1021 000A 0010 0000 0000 0000 2F81 0000 00BD 000D 000E 000F
30C1 1021 000A 0010 0000 0000 0000 30C1 0000 00BE 000D 000E 000F
3201 1021 000A 0010 0000 0000 0000 3201 0000 00BF 000D 000E 000F
3341 1021 000A 0010 0000 0000 0000 3341 0000 00C0 000D 000E 000F
3481 1021 000A 0010 0000 0000 0000 3481 0000 00C1 000D 000E 000F
35C1 1021 000A 0010 0000 0000 0000 35C1 0000 00C2 000D 000E 000F
3701 1021 000A 0010 0000 0000 0000 3701 0000 00C3 000D 000E 000F
3841 1021 000A 0010 0000 0000 0000 3841 0000 00C4 000D 000E 000F
3981 1021 000A 0010 0000 0000 0000 3981 0000 00C5 000D 000E 000F
3AC1 1021 000A 0010 0000 0000 0000 3AC1 0000 00C6 000D 000E 000F
3C01 1021 000A 0010 0000 0000 0000 3C01 0000 00C7 000D 000E 000F
3D41 1021 000A 0010 0000 0000 0000 3D41 0000 00C8 000D 000E 000F
3E81 1021 000A 0010 0000 0000 0000 3E81 0000 00C9 000D 000E 000F
3FC1 1021 000A 0010 0000 0000 0000 3FC1 0000 00CA 000D 000E 000F
4101 1021 000A 0010 0000 0000 0000 4101 0000 00CB 000D 000E 000F
4241 1021 000A 0010 0000 0000 0000 4241 0000 00CC 000D 000E 000F
4381 1021 000A 0010 0000 0000 0000 4381 0000 00CD 000D 000E 000F
44C1 1021 000A 0010 0000 0000 0000 44C1 0000 00CE 000D 000E 000F
4601 1021 000A 0010 0000 0000 0000 4601 0000 00CF 000D 000E 000F
4741 1021 000A 0010 0000 0000 0000 4741 0000 00D0 000D 000E 000F
4881 1021 000A 0010 0000 0000 0000 4881 0000 00D1 000D 000E 000F
49C1 1021 000A 0010 0000 0000 0000 49C1 0000 00D2 000D 000E 000F
4B01 1021 000A 0010 0000 0000 0000 4B01 0000 00D3 000D 000E 000F
4C41 1021 000A 0010 0000 0000 0000 4C41 0000 00D4 000D 000E 000F
4D81 1021 000A 0010 0000 0000 0000 4D81 0000 00D5 000D 000E 000F
4EC1 1021 000A 0010 0000 0000 0000 4EC1 0000 00D6 000D 000E 000F
5001 1021 000A 0010 0000 0000 0000 5001 0000 00D7 000D 000E 000F
5141 1021 000A 0010 0000 0000 0000 5141 0000 00D8 000D 000E 000F
5281 1021 000A 0010 0000 0000 0000 5281 0000 00D9 000D 000E 000F
53C1 1021 000A 0010 0000 0000 0000 53C1 0000 00DA 000D 000E 000F
5501 1021 000A 0010 0000 0000 0000 5501 0000 00DB 000D 000E 000F
5641 1021 000A 0010 0000 0000 0000 5641 0000 00DC 000D 000E 000F
5781 1021 000A 0010 0000 0000 0000 5781 0000 00DD 000D 000E 000F
58C1 1021 000A 0010 0000 0000 0000 58C1 0000 00DE 000D 000E 000F
5A01 1021 000A 0010 0000 0000 0000 5A01 0000 00DF 000D 000E 000F
5B41 1021 000A 0010 0000 0000 0000 5B41 0000 00E0 000D 000E 000F
5C81 1021 000A 0010 0000 0000 0000 5C81 0000 00E1 000D 000E 000F
5DC1 1021 000A 0010 0000 0000 0000 5DC1 0000 00E2 000D 000E 000F
5F01 1021 000A 0010 0000 0000 0000 5F01 0000 00E3 000D 000E 000F
6041 1021 000A 0010 0000 0000 0000 6041 0000 00E4 000D 000E 000F
6181 1021 000A 0010 0000 0000 0000 6181 0000 00E5 000D 000E 000F
62C1 1021 000A 0010 0000 0000 0000 62C1 0000 00E6 000D 000E 000F
6401 1021 000A 0010 0000 0000 0000 6401 0000 00E7 000D 000E 000F
6541 1021 000A 0010 0000 0000 0000 6541 0000 00E8 000D 000E 000F
6681 1021 000A 0010 0000 0000 0000 6681 0000 00E9 000D 000E 000F
67C1 1021 000A 0010 0000 0000 0000 67C1 0000 00EA 000D 000E 000F
6901 1021 000A 0010 0000 0000 0000 6901 0000 00EB 000D 000E 000F
6A41 1021 000A 0010 0000 0000 0000 6A41 0000 00EC 000D 000E 000F
```

```
6B81 1021 000A 0010 0000 0000 0000 6B81 0000 00ED 000D 000E 000F
6CC1 1021 000A 0010 0000 0000 0000 6CC1 0000 00EE 000D 000E 000F
6E01 1021 000A 0010 0000 0000 0000 6E01 0000 00EF 000D 000E 000F
6F41 1021 000A 0010 0000 0000 0000 6F41 0000 00F0 000D 000E 000F
7081 1021 000A 0010 0000 0000 0000 7081 0000 00F1 000D 000E 000F
71C1 1021 000A 0010 0000 0000 0000 71C1 0000 00F2 000D 000E 000F
7301 1021 000A 0010 0000 0000 0000 7301 0000 00F3 000D 000E 000F
7441 1021 000A 0010 0000 0000 0000 7441 0000 00F4 000D 000E 000F
7581 1021 000A 0010 0000 0000 0000 7581 0000 00F5 000D 000E 000F
76C1 1021 000A 0010 0000 0000 0000 76C1 0000 00F6 000D 000E 000F
7801 1021 000A 0010 0000 0000 0000 7801 0000 00F7 000D 000E 000F
7941 1021 000A 0010 0000 0000 0000 7941 0000 00F8 000D 000E 000F
7A81 1021 000A 0010 0000 0000 0000 7A81 0000 00F9 000D 000E 000F
7BC1 1021 000A 0010 0000 0000 0000 7BC1 0000 00FA 000D 000E 000F
7D01 1021 000A 0010 0000 0000 0000 7D01 0000 00FB 000D 000E 000F
7E41 1021 000A 0010 0000 0000 0000 7E41 0000 00FC 000D 000E 000F
7F81 1021 000A 0010 0000 0000 0000 7F81 0000 00FD 000D 000E 000F
80C1 1021 000A 0010 0000 0000 0000 80C1 0000 00FE 000D 000E 000F
8201 1021 000A 0010 0000 0000 0000 8201 0000 00FF 000D 000E 000F
8341 1021 000A 0010 0000 0000 0000 8341 0000 0100 000D 000E 000F
8481 1021 000A 0010 0000 0000 0000 8481 0000 0101 000D 000E 000F
85C1 1021 000A 0010 0000 0000 0000 85C1 0000 0102 000D 000E 000F
8701 1021 000A 0010 0000 0000 0000 8701 0000 0103 000D 000E 000F
8841 1021 000A 0010 0000 0000 0000 8841 0000 0104 000D 000E 000F
8981 1021 000A 0010 0000 0000 0000 8981 0000 0105 000D 000E 000F
8AC1 1021 000A 0010 0000 0000 0000 8AC1 0000 0106 000D 000E 000F
8C01 1021 000A 0010 0000 0000 0000 8C01 0000 0107 000D 000E 000F
8D41 1021 000A 0010 0000 0000 0000 8D41 0000 0108 000D 000E 000F
8E81 1021 000A 0010 0000 0000 0000 8E81 0000 0109 000D 000E 000F
8FC1 1021 000A 0010 0000 0000 0000 8FC1 0000 010A 000D 000E 000F
9101 1021 000A 0010 0000 0000 0000 9101 0000 010B 000D 000E 000F
9241 1021 000A 0010 0000 0000 0000 9241 0000 010C 000D 000E 000F
9381 1021 000A 0010 0000 0000 0000 9381 0000 010D 000D 000E 000F
94C1 1021 000A 0010 0000 0000 0000 94C1 0000 010E 000D 000E 000F
9601 1021 000A 0010 0000 0000 0000 9601 0000 010F 000D 000E 000F
9741 1021 000A 0010 0000 0000 0000 9741 0000 0110 000D 000E 000F
9881 1021 000A 0010 0000 0000 0000 9881 0000 0111 000D 000E 000F
99C1 1021 000A 0010 0000 0000 0000 99C1 0000 0112 000D 000E 000F
9B01 1021 000A 0010 0000 0000 0000 9B01 0000 0113 000D 000E 000F
9C41 1021 000A 0010 0000 0000 0000 9C41 0000 0114 000D 000E 000F
9D81 1021 000A 0010 0000 0000 0000 9D81 0000 0115 000D 000E 000F
9EC1 1021 000A 0010 0000 0000 0000 9EC1 0000 0116 000D 000E 000F
A001 1021 000A 0010 0000 0000 0000 A001 0000 0117 000D 000E 000F
A141 1021 000A 0010 0000 0000 0000 A141 0000 0118 000D 000E 000F
A281 1021 000A 0010 0000 0000 0000 A281 0000 0119 000D 000E 000F
A3C1 1021 000A 0010 0000 0000 0000 A3C1 0000 011A 000D 000E 000F
A501 1021 000A 0010 0000 0000 0000 A501 0000 011B 000D 000E 000F
A641 1021 000A 0010 0000 0000 0000 A641 0000 011C 000D 000E 000F
A781 1021 000A 0010 0000 0000 0000 A781 0000 011D 000D 000E 000F
A8C1 1021 000A 0010 0000 0000 0000 A8C1 0000 011E 000D 000E 000F
```

```
AA01 1021 000A 0010 0000 0000 0000 AA01 0000 011F 000D 000E 000F
AB41 1021 000A 0010 0000 0000 0000 AB41 0000 0120 000D 000E 000F
AC81 1021 000A 0010 0000 0000 0000 AC81 0000 0121 000D 000E 000F
ADC1 1021 000A 0010 0000 0000 0000 ADC1 0000 0122 000D 000E 000F
AF01 1021 000A 0010 0000 0000 0000 AF01 0000 0123 000D 000E 000F
B041 1021 000A 0010 0000 0000 0000 B041 0000 0124 000D 000E 000F
B181 1021 000A 0010 0000 0000 0000 B181 0000 0125 000D 000E 000F
B2C1 1021 000A 0010 0000 0000 0000 B2C1 0000 0126 000D 000E 000F
B401 1021 000A 0010 0000 0000 0000 B401 0000 0127 000D 000E 000F
B541 1021 000A 0010 0000 0000 0000 B541 0000 0128 000D 000E 000F
B681 1021 000A 0010 0000 0000 0000 B681 0000 0129 000D 000E 000F
B7C1 1021 000A 0010 0000 0000 0000 B7C1 0000 012A 000D 000E 000F
B901 1021 000A 0010 0000 0000 0000 B901 0000 012B 000D 000E 000F
BA41 1021 000A 0010 0000 0000 0000 BA41 0000 012C 000D 000E 000F
```

Références

[AIS09] A. Agarwal, C. Iskander, R. Shankar. *Survey of Network on Chip (NoC) Architectures & Contributions*. Journal of Engineering, Computing and Architecture ISSN 1934-7197, volume 3, issue 1, 2009.

[BD02] L. Benini and G. De Micheli. *Networks on Chip: A New Paradigm for Systems on Chip Design*. In Proceedings Design, Automation and Test in Europe Conference and Exhibition, , pages 418-419, 2002.

[BF02] J. Bainbridge, S. Furber. *Chain: a delay-insensitive chip area interconnect*. Micro, IEEE Volume 22, Issue 5, pages 16-23, 2002.

[BS05] T. Bjerregaard, J. Sparsø. *A Scheduling Discipline for Latency and Bandwidth Guarantees in Asynchronous Network-on-Chip*. In Proceedings of the 11th IEEE International Symposium on Asynchronous Circuits and Systems, pages 34-43, 2005.

[Bje05] T. Bjerregaard *The MANGO Clockless Network-on-Chip: Concepts and Implementation*. Phd thesis in Informatics and Mathematical Modelling, Technical University of Denmark, 2005.

[CEG03] H. Charlery, E. Encrenaz, A. Greiner, A. Andriahantenaina, L. Mortiez. *SPIN, un micro-réseau d'interconnexion à commutation de paquets respectant la norme VCI. Concepts généraux et validation*. Département ASIM (Laboratoire LIP6) Université Pierre et Marie CURIE, 2003.

[CEG03] H. Charlery, E. Encrenaz, A. Greiner, A. Andriahantenaina, L. Mortiez. *SPIN: a Scalable, Packet Switched, On-chip Micro-network*. In Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE), 2003.

[Chi00] G. Chiu. *The odd-even turn model for adaptive routing*. IEEE Transactions on Parallel and Distributed Systems, Volume 11, Issue 7, pages 729-738, 2000.

[DA93] W. Dally, H. Aoki. *Deadlock-free adaptive routing in multicomputer networks using virtual channels*. IEEE Transactions Parallel and Distributed Systems, volume 4, issue 4, pages 466-475, 1993.

[DS87] W. J. Dally et C. L. Seitz *Deadlock-Free Message Routing in Multiprocessor Interconnection Networks*. Computers, IEEE Transactions, Volume C-36, Issue 5, pages 547?553, 1987.

[DSM10] G. De Micheli, C. Seiculescu, S. Murali, L. Benini, F. Angiolini, A. Pullini. *Networks on Chips: From research to products*. 47th ACM/IEEE Design Automation Conference (DAC), pages 300-305, 2010.

[DT01] W.J. Dally, B. Towles. *Route Packets, Not Wires: On-Chip Interconnection Networks*. Design Automation Conference, pages 684-689, 2001.

[Dal90] W. Dally. *Virtual-channel fow control*. In Proceeding 17th Annual International Symposium on Computer Architecture, pages 28-31, 1990.

[FNQ07] G. Fen, W. Ning, W. Qi. *Simulation and Performance Evaluation for Network on Chip Design Using OPNET*. Department of electronic engineering, Nanjing University of Aeronautics and Astronautics, China, 2007.

[Fle02] E. Fleury. *Communication de groupe - du parallélisme au ad hoc-*. Habilitation à diriger des recherches de L'INSA de Lyon et de l'université Claude Bernard -lyon 1, 2002.

[GDR05] K. Goossens, J. Dielissen, A. Radulescu. *Æthereal Network on Chip : Concepts, Architectures, and Implementations.* Design & Test of Computers, IEEE Volume 22 ,Issue 5 ,pages 414-421, 2005.

[GN92] C. Glass, L. Ni. *The Turn Model for Adaptive Routing.* In Proceedings of the 19th Annual International Symposium on Computer Architecture, pages 278-287, 1992.

[GN92] C.J. Glass, L.M. Ni. *The Turn Model for Adaptive Routing.* In Proceedings of the 19th Annual International Symposium on Computer Architecture, pages 278-287, 1992.

[GY93] P. Gaughan, S. Yalamanchili. *Adaptive routing protocols for hypercube interconnection networks.* Computer, volume 26, issue 5, pages 12-23, 1993.

[GZX06] G. Xiaopeng, Z. Zhe, L. Xiang. *Round Robin Arbiters for Virtual Channel Router.* Computational Engineering in Systems Applications, IMACS Multiconference on, pages 1610-1614, 2006.

[Gol94] S. Golson. *State machine design techniques for Verilog and VHDL.* Synopsys Users Group Conference (SNUG San Jose 1994).

[Gro07] P. Grosse. *Gestion dynamique des tâches dans une architecture micro-électronique intégrée à des fins de basse consommation.* Ph.D. de l'Ecole Normale Supérieure de Lyon, 2007.

[HH10] K. Hamwi, O. Hammami. *Design and Implementation of MPSoC Single Chip with Butterfly Network.* VLSI System on Chip Conference (VLSI-SoC) 18th IEEE/IFIP, pages 143-148, 2010.

[HM04] J. Hu, R. Marculescu. *DyAD Smart Routing for Networks-on-Chip.* In Proceedings of the 41st Design Automation Conference, pages 260-263, 2004.

[KHM87] M. Karol, M. Hluchyj, S. Morgan. *Input Versus Output Queueing on a Space-Division Packet Switch.* IEEE Transactions on Communications, Volume 35 , Issue 12 , pages 1347-1356, 1987.

[LB09] S. E. Lee, N. Bagherzadeh. *A high level power model for Network-on-Chip (NoC) router.* Computers and Electrical Engineering, 2009.

[LZJ06] M. Li, Q. Zeng, W. Jone. *DyXY - A Proximity Congestion-Aware Deadlock-Free Dynamic Routing Method for Network on Chip.* 43rd ACM/IEEE Design Automation Conference, pages 849-852 , 2006.

[Lem06] R. Lemaire. *Conception et modélisation d'un système de controle d'applications de télécommunication avec une architecture de réseau sur puce (NoC).* Ph.D. Institut National Polytechnique de Grenoble, 2006.

[Ler07] A. Leroy *Optimizing the on-chip communication architecture of low power systems-on chip in deep sub-micron technology.* Phd thesis, Université Libre de Bruxelles, 2007.

[MCM04] F. G. Moraes, N. L. V. Calazans, A. V. de Mello, L. H. Möller, L. C. Ost. *HERMES: an Infrastructure for Low Area Overhead Packet-switching Networks on Chip.* Technical report series, Number 034, Faculdade de Inforática PUCRS - Brazil, 2004.

[MCM04] F. Moraes, N. Calazans, A. Mello, L. Moller, L. Ost. *HERMES: an infrastructure for low area overhead packet-switching networks on chip.* INTEGRATION, the VLSI journal 38, pages 69-93, 2004.

[MFA93] D. McCarty, D. Faria, P. Alfke. *PREP@ BENCHMARKS FOR PROGRAMMABLE LOGIC DEVICES.* In Proceedings of the IEEE Custom Integrated Circuits Conference, pages

7.7.1-7.7.6, 1993.

[MGS06] I. Miro Panades, A. Greiner, A. Sheibanyrad. *A Low Cost Network-on-Chip with Guaranteed Service Well Suited to the GALS Approach.* In Proceedings of the first International Conference ON Nano-Networks, Lausane, pages 1-5, 2006.

[MHK99] T. Meincke, A. Hemani, S. Kumar, P. Ellervee, J. Oberg, T. Olsson, P. Nilsson, D. Lindqvist, H. Tenhunen. *Globally asynchronous locally synchronous architecture for large high-performance ASICs.* In Proceedings of the 1999 IEEE International Symposium on Circuits and Systems, ISCAS '99, 1999.

[MS07] L. MA , Y. SUN . *On-Chip Network Design Automation with Source Routing Switches.* Institute of Microelectronics, Tsinghua University, Beijing, China, 2007.

[MVS05] P. Marchal, D. Verkest, A. Shickova, F. Catthoor, F. Robert, A. Leroy. *Spatial division multiplexing : a novel approach for guaranteed throughput on NoCs.* In Third IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis CODES+ISSS '05, pages 81-86, 2005.

[Mur09] J. J. Murillo. *HW-SW components for parallel embedded computing on NoC based MPSoCs.* Phd thesis in computer science, Universitat Autonoma de Barcelona, 2009.

[NM93] L. Ni, P. McKinley. *A survey of wormhole routing techniques in direct networks.* IEEE Computer, volume 26, pages 62-67, 1993.

[NO06] E. Nilsson, J. Óberg. *PANACEA - A case study on the PANACEA NoC - a Nostrum Network on Chip prototype.* Electronic, Computer, and Software Systems, Information and Communication Technology, Royal Institute of Technology, Sweden, 2006.

[NW06] C. Neeb, N. Wehn . *Designing Efficient Irregular Networks for Heterogeneous Systems-on-Chip.* University of Kaiserslautern, Germany. In Proceedings of the 9th EUROMICRO Conference on Digital System Design(DSD'06) IEEE, 2006.

[PAB05] A. Pullini, F. Angiolini, D. Bertozzi, L. Benini. *Fault Tolerance Overhead in Network on-Chip Flow Control Schemes.* In 18th Symposium Integrated Circuits and Systems Design, pages 224-229, 2005.

[PBB03] F. Poletti, D. Bertozzi, L. Benini, A. Bogliolo *Performance Analysis of Arbitration policies for SoC Comunication Architectures.* Journal of Design Automation for Embedded Systems, Volume 8, pages 189-210, 2003.

[RAR09] M.A.U. Rahman, I. Ahmed, F. Rodriguez, N. Islam. *Efficient 2DMesh Network on Chip (NoC) Considering GALS Approach.* IEEE Fourth International Conference on Computer Sciences and Convergence Information Technology. ICCIT '09, pages 841-846, 2009.

[RGR03] E. Rijpkema, K. Goossens, A. Radulescu, J. Dielissen, J. van Meerbergen, P. Wielage, E. Waterlander. *Trade-offs in the design of a router with both guaranteed and best-effort services for networks on chip.* IEE Proceedings Computers and Digital Techniques, Vol. 150, No. 5, pages 294-302, 2003.

[RS97] J.A. Rowson, A. Sangiovanni-Vincentelli. *Interface-Based Design.* In Proceedings of the 34th Design Automation Conference, pages 178-183, 1997.

[Rab00] J. Rabaey. *Busses and Networking.* Computer Science 252, Spring 2000.

[SMG07] A. Sheibanyrad, I. Miro Panades, A. Greiner. *Systematic Comparison between the Asynchronous and the Multi-Synchronous Implementations of a Network on Chip Architecture..*

In Proceedings of the conference on Design, automation and test in Europe, pages 1090-1095, 2007.

[SMN08] A. M. Shafiee, M. Montazeri, M. Nikdast. *An Innovational Intermittent Algorithm in Networks-On-Chip (NOC)*. World Academy of Science, Engineering and Technology, 2008.

[Wik05] D. Wiklund. *Development and Performance Evaluation of Networks on Chip*. Dissertation. Department of Electrical Engineering Linköping University, Sweden, 2005.

[ZG09] J. Zhang, H. Gu. *A Partially Adaptive Routing Algorithm for Benes Network on Chip*. 2nd IEEE International Conference on Computer Science and Information Technology, ICCSIT, pages 614-618, 2009.

ملخص

في الشبكة على الرقاقة، يلعب التوجيه دورا هاما حيث ان خوارزمية التوجيه هي التي تقرر الطريق الذي يتوجب اتخاذه لارسال الرسالة من مصدرها الى وجهتها. في هذه المذكرة نقوم بعرض و تقييم خوارزمية جديدة تسمي *OEC* (*OddEvenCongestion*) مخصصة لطبولوجيات ذات شبكات بسيطة ثنائية الابعاد و التي توفر توجيها مكيفا خال من الجمود مع الاخذ بعين الاعتبار ظروف ازدحام الشبكة في المنطقة المجاورة. تكمن اهمية التوجيه المقترح في ديناميكيته التي تعتمد على مؤشر الازدحام الذي يسمح بتجاوز نقاط الاحتقان باستخدام مسارات بديلة. و قد تم تطوير بنية لتقييم اداء خوارزمية التوجيه المقترح. و لقد لوحظ ان نتائج المحاكاة للتوجيه *OEC* فيما يتعلق بالكمون المتوسط هي الافضل مقارنة بالتي تحصلنا عليها باستعمال خوارزمات التوجيه *OddEven* و *XY* .

الكلمات الرئيسية : شبكة على الرقاقة، نظام على الرقاقة، موجه، ازدحام.

Résumé

Dans un NoC ou réseau sur puce, le routage joue un rôle très important. C'est l'algorithme de routage qui décide du chemin à emprunter pour envoyer le message de sa source à sa destination. Dans ce mémoire, nous présentons et évaluons un nouvel algorithme de routage appelé *OEC* (Odd Even Congestion) dédié aux topologies de maillage simple à deux dimensions, qui fournit un routage adaptatif et sans interblocages en tenant compte des conditions de la congestion du réseau dans la proximité. L'intérêt de notre stratégie de routage réside dans son aspect dynamique basé sur l'*Indice de Congestion* qui permet de contourner les points de congestion en utilisant des chemins alternatifs. Une architecture a été développée afin d'évaluer les performances de notre algorithme. Il a été observé que les résultats de simulation de *OEC* en terme de latence moyenne sont meilleurs, par rapport à ceux obtenus par les algorithmes *OddEven* et *XY*.

Mots clés : Réseau sur puce, système sur puce, routeur, congestion.

Abstract

In a NoC or Network on Chip, routing plays an important role. A routing algorithm defines a route which message traverses to get to destination. In this paper, we present and evaluate a novel deadlock-free routing algorithm called *OEC* (Odd Even Congestion) for a two dimensional mesh topologies without virtual channels, which provides an adaptive routing based on the network's congestion conditions. The point of our dynamic routing strategy based on the turn model approach is its simplicity and inherent freedom from deadlock. It use an alternate route to bypass the congestion points. An architecture was developed to evaluate the performance of our algorithm. Experimental and simulation results indicate that the proposed OEC routing scheme provides better average latencies compared to *OddEven* and deterministic *XY* routing algorithms.

Keywords : Networks on Chip, System on Chip, router design, congestion.