Conception d'un micro-réseau intégré NOC tolérant les fautes multiples statiques et dynamiques

Yi Gang

UNIVERSITÉ GRENOBLE ALPES

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES

Spécialité : **« Micro et Nano Électronique»**

Arrêté ministériel : 7 août 2006

Présentée par

Yi GANG

Thèse dirigée par **Mme. Lorena ANGHEL** et
codirigée par **M. Mounir BENABDENBI**

préparée au sein du **Laboratoire TIMA**
dans **l'École Doctorale d'Électronique, Électrotechnique, Automatique, et Traitement du Signal (EEATS)**

Conception d'un micro-réseau intégré NOC tolérant les fautes multiples statiques et dynamiques

Thèse soutenue publiquement le **05 novembre 2015**,
devant le jury composé de :

M. Habib MEHREZ
Professeur, Université Paris 6 (Président & Rapporteur)
M. Ian O'CONNOR
Professeur, Ecole Central de Lyon (Rapporteur)
Mme. Lorena ANGHEL
Professeur, Institut polytechnique de Grenoble (Directrice de thèse)
M. Mounir BENABDENBI
Maître de Conférences, Institut polytechnique de Grenoble (Co-encadrant de thèse)

TITRE
Conception d'un micro-réseau intégré NOC tolérant les fautes multiples statiques et dynamiques

---

RESUME
Les progrès dans les technologies à base de semi-conducteurs et la demande croissante de puissance de calcul poussent vers une intégration dans une même puce de plus en plus de processeurs intégrés. Par conséquent les réseaux sur puce remplacent progressivement les bus de communication, ceux-ci offrant plus de débit et permettant une mise à l'échelle simplifiée.

Parallèlement, la réduction de la finesse de gravure entraine une augmentation de la sensibilité des circuits au processus de fabrication et à son environnement d'utilisation. Les défauts de fabrication et le taux de défaillances pendant la durée de vie du circuit augmentent lorsque l'on passe d'une technologie à une autre. Intégrer des techniques de tolérance aux fautes dans un circuit devient indispensable, en particulier pour les circuits évoluant dans un environnement très sensible (aérospatial, automobile, santé, ...).

Nous présentons dans ce travail de thèse, des techniques permettant d'améliorer la tolérance aux fautes des micro-réseaux intégrés dans des circuits évoluant dans un environnement difficile. Le NoC doit ainsi être capable de s'affranchir de la présence de nombreuses fautes. Les travaux publiés jusqu'ici proposaient des solutions pour un seul type de faute.

En considérant les contraintes de surface et de consommation du domaine de l'embarqué, nous avons proposé un algorithme de routage adaptatif tolérant à la fois les fautes intermittentes, transitoires et permanentes. En combinant et adaptant des techniques existantes de retransmission de flits, de fragmentation et de regroupement de paquet, notre approche permet de s'affranchir de nombreuses fautes statiques et dynamiques. Les très nombreuses simulations réalisées ont permis de montrer entre autre que, l'algorithme proposé permet d'atteindre un taux de livraison de paquets de 97,68% pour un NoC 16x16 en maille 2D en présence de 384 liens défectueux simultanés, et 93,40% lorsque 103 routeurs sont défaillants. Nous avons étendu l'algorithme aux topologies de type tore avec des résultats bien meilleurs.

Une autre originalité de cette thèse est que nous avons inclus dans cet algorithme une fonction de gestion de la congestion. Pour cela nous avons défini une nouvelle métrique de mesure de la congestion (Flit Remain) plus pertinente que les métriques utilisées et publiées jusqu'ici. Les expériences ont montré que l'utilisation de cette métrique permet de réduire la latence (au niveau du pic de saturation) de 2,5 % à 16,1 %, selon le type de trafic généré, par rapport à la plus efficace des métriques existante.

La combinaison du routage adaptatif tolérant les fautes statiques et dynamiques et la gestion de la congestion offrent une solution qui permet d'avoir un NoC et par extension un circuit beaucoup plus résilient.

MOTS CLEFS
MPSoCs, NoC, routage adaptif, tolérance aux fautes, injection des fautes

---

TITRE

Design of a Network on chip (NoC) that tolerates multiple static and dynamic faults

---

ABSTRACT

The quest for higher-performance and low-power consumption has driven the microelectronics' industry race towards aggressive technology scaling and multicore chip designs. In this many-core era, the Network-on-chip (NoCs) becomes the most promising solution for on-chip communication because of its performance scaling with the number of IPs integrated in the chip.

Fault tolerance becomes mandatory as the CMOS technology continues shrinking down. The yield and the reliability are more and more affected by factors such as manufacturing defects, process variations, environment variations, cosmic radiations, and so on. As a result, the designs should be able to provide full functionality (e.g. critical systems), or at least allow degraded mode in a context of high failure rates. To accomplish this, the systems should be able to adapt to manufacturing and runtime failures.

In this thesis, some techniques are proposed to improve the fault tolerance ability of NoC based circuits working in harsh environments. As previous works allow the handling of one type of fault at a time, we propose here a solution where different kinds of faults can be tolerated concurrently.

Considering constraints such as area and power consumption, a fault tolerant adaptive routing algorithm was proposed, which can cope with transient, intermittent and permanent faults. Combined with some existing techniques, like flit retransmission and packet fragmentation, this approach allows tolerating numerous static and dynamic faults. Simulations results show that the proposed solution allows a high packet delivery success rate: for a 16x16 2D Mesh NoC, 97.68% in the presence of 384 simultaneous link faults, and 93.40% with the presence of 103 simultaneous router faults. This success rate is even higher when this algorithm is extended to NoCs with Tore topology.

Another contribution of this thesis is the inclusion of a congestion management function in the proposed routing algorithm. For this purpose, we introduce a novel metric of congestion measurement named Flit Remain. The experimental results show that using this new congestion metric allows a reduction of the average latency of the Network on Chip from 2.5% to 16.1% when compared to the existing metrics.

The combination of static and dynamic fault tolerant and adaptive routing and the congestion management offers a solution, which allows designing a NoC highly resilient.

KEYWORDS

Network on Chip (NoC), MPSoCs, adaptive routing, fault tolerance, congestion

---

Remerciements

Je tiens tout d'abord à remercier Mme Lorena Anghel, Professeur de Grenoble INP, pour avoir accepté d'être le directeur de cette thèse. Je tiens à lui exprimer ma reconnaissance pour sa disponibilité, ses conseils et sa gentillesse.

Je remercie Mr. Mounir Benabdenbi, Maître de Conférence de Grenoble INP durant ma thèse, pour m'avoir proposé cette thèse, dirigé et soutenu durant ces dernières années. Son encadrement, ses encouragements, sa confiance, ses conseils et ses critiques m'ont permis de progresser et mener à bien ce travail. Qu'il trouve ici l'expression de ma sincère reconnaissance.

Mes gratitudes s'adressent également à Mr Habib Mehrez, Professeur à l'Université Paris 6, pour l'honneur qu'il accepté d'être président et rapporteur de ma thèse. Merci à Mr. Ian O'Connor, Professeur de l'Ecole Central de Lyon qui a rapporté également cette thèse. Je les remercie profondément pour leurs remarques, leurs conseils et leurs critiques constructives.

Je tiens à remercier M. Michael Dimopolos pour ses suggestions et supports au cours de l'avancement de cette thèse. J'aimerais remercier Mr. Alexandre Chagoya pour son assistance au CIME.

Je remercie les Directeurs successifs du laboratoire TIMA pour m'avoir accueilli, Mme. Dominique Borrione, Professeur de l'Université Grenoble Alpes et Mr. Salvador Mir, directeur actuel de TIMA. Je voudrais remercier les chercheurs et personnel administratif et du Laboratoire TIMA pour m'avoir accueilli. Je pense à Anne-Laure, Laurence, Marie-Christine, Youness, Sophie, et le personnel du service informatique, Nicolas, Frederic et Ahmed pour leur disponibilité permanente et leur aide à l'égard de mon travail.

Merci à tous mes collègues des Laboratoires TIMA et avec qui j'ai eu le plaisir de travailler, très particulièrement Claudia Rusu, Hai Yu, Gilles Bizot, Diarga Fall, Hamayun Muhammad, Fabien Chaix, Vladimir Paska, Adrian Evans, Wassim Mansour, Saif Ur Rehman, Michael Dimopolous, Mihai Iordache et Amir Charif.

Enfin, je remercie ma famille et mes amis pour leur soutien et leurs encouragements. Finalement je réserve mon dernier merci, spécial et sincère à ma famille et mes parents pour leur soutien et leurs encouragements.

Table des matières

Liste des figures

Liste des abréviations

BPR	Bypass Packet Recovery
BW	Buffer Write
ECC	Error-correcting code
FB	Free Buffer
FIFO	First In First Out
Flit	Flow control digit
FR	Flit Remain
MPSOC	Multiprocessors System on Chip
NACK	Negative Acknowledgment
NI	Network Interface
NOC	Network on Chip
PE	Processing Element
Phit	Physic Digit
QoS	Quality of Service (Qualité de service)
RC	Route Calculation
SA	Switch allocation
SAF	Save and Forward
SBST	Software-Based Self-Test
SOC	System on Chip
SPR	Splitting Packet Recovery
ST	Switch Transfer
UTP	Unique Token Protocol
VCA	Virtual Channel Allocation
VC	Virtual Channel (Canal virtuel)
VCT	Virtual Cut Through
VN	Virtual Network
VS	Virtual Source
WH	Wormhole

Introduction

La demande continue de circuits hautes performances et basse consommation gouverne l'industrie des semi-conducteurs ces dernières décennies. Grâce ou à cause de la loi de Moore, on intègre de plus en plus de blocs dans une seule puce. Les systèmes sur puce (System on Chip, SoC) peuvent contenir ainsi un microprocesseur, de la mémoire, des périphériques, des capteurs et encore d'autres composants permettant ainsi de répondre à la demande du marché de l'électronique grand public.

Cependant augmenter les performances tout en baissant ou en maitrisant la consommation est devenu un problème majeur. De ce fait, la course à la fréquence n'étant plus possible, la solution privilégiée pour garantir de meilleures performances consiste à intégrer plusieurs processeurs dans une seule puce. Un SoC avec plusieurs processeurs intégrés est appelé système multiprocesseurs sur puce (MPSoC).

Selon l'étude d'ITRS [ITR11a], le développement des « Smartphones » en particulier, et des technologies multimédia en général a créé de nouveaux besoins: appareil photo, streaming de médias (décodage vidéo), les jeux en trois dimensions interactives, GPS, capteurs biologiques, etc. Ces applications nécessitent de plus en plus de puissance de calcul comme représenté sur la figure 0.1. L'intégration d'un certain nombre de processeurs dans une seule puce est donc une solution potentielle pour faire face à la puissance de calcul exigée en permanence par des applications embarquées avec une contrainte de temps de mise sur le marché (Time to Market) toujours aussi forte. Dans ces MPSoCs, le nombre d'unité de traitement (Process Element, PE) devrait augmenter de manière significative dans les prochaines années [ITR11a] (figure 0-1) et atteindre « mille cœurs» en 2023.

Cependant, la conception basée sur l'agrégation de plusieurs blocs (IPs) dans un MPSoC génère un certain nombre de défis. L'un des plus importants est le problème de communication inter IPs. Des interconnexions implémentées avec un ou des bus classiques ne sont plus suffisantes. Ces architectures ne peuvent plus fournir la bande passante nécessaire et posent aussi des problèmes de mise à l'échelle.

Pour pallier ces limitations, la solution étudiée et préférée dans le monde des architectures MPSoC depuis plus d'une décennie est celle basée sur le paradigme des réseaux

sur puce (Network on Chip, NoC) [Benini 2002]. Un NoC offre en effet une solution de communication « scalable » et modulaire et répond au problème du haut débit tout en ayant une consommation maîtrisée.

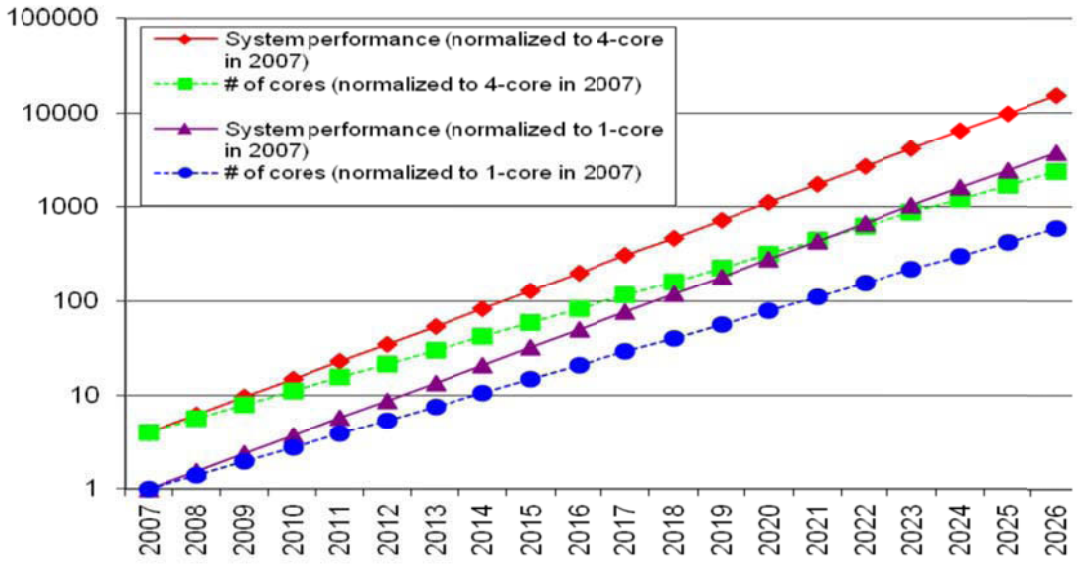


Figure 0.1 Tendance des besoins en capacité de calcul pour les SoCs (ITRS 2011)

Avec la capacité d'intégration croissante de la technologie CMOS, les systèmes sur puce sont de plus en plus sensibles à différents facteurs qui influent considérablement sur leur rendement de fabrication et leur fiabilité lors de la mise en service. A côté des défauts dits catastrophiques (pannes permanentes), les défauts paramétriques notamment dus aux problèmes de variabilité, sont de plus aigus [ITR11]. Ces défauts paramétriques liés à la variabilité des paramètres des transistors amènent à un comportement similaire à celui induit par des défauts catastrophiques. Ainsi la fiabilité des circuits intégrés est menacée par les variations de tension et de fréquence [Weiser 1994], par le vieillissement, les effets des phénomènes de radiation [Ziegler 1979][Ziegler 1981][Ziegler 1996], les variations thermiques et électriques, le bruit intrinsèque, la diaphonie, etc.

Pour illustrer cela, la figure 0.2 nous montre l'évolution du taux de défaillance de trois circuits CMOS couramment utilisés (cellule SRAM, verrou et inverseur) pour différents nœuds technologiques [ITR11].

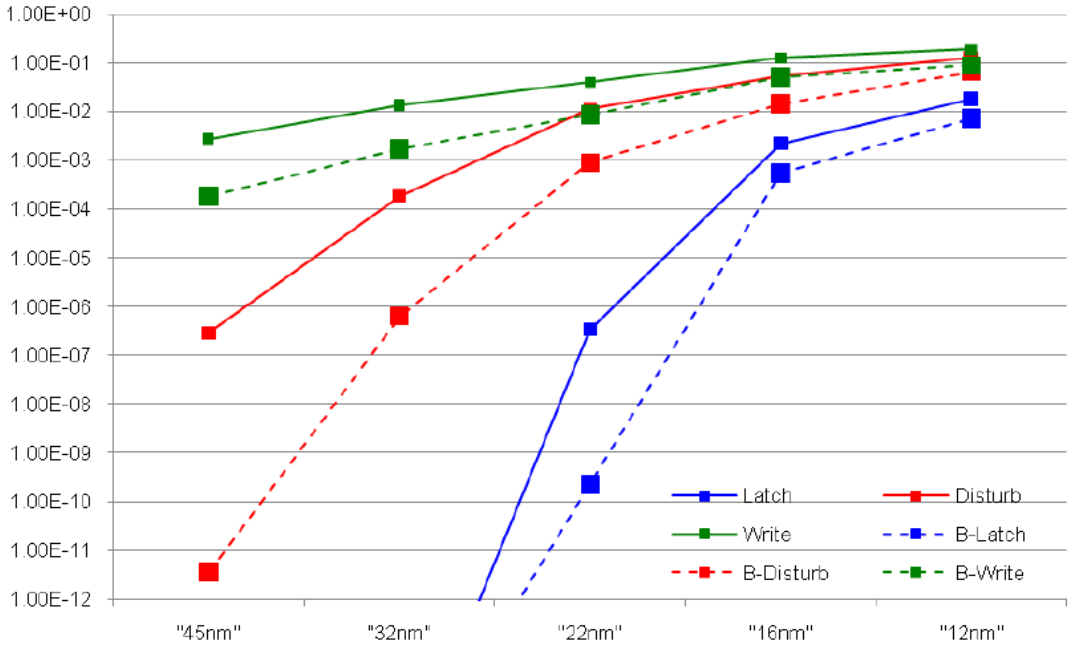


Figure 0.2 Taux de défaillance induit par les problèmes de variabilité pour trois types de circuits ITRS 2011)

Il est donc indispensable de prendre en compte dès la phase de spécification l'aspect gestion des défaillances pour assurer un fonctionnement correct et continu du circuit dans son environnement, et cela même lorsque le taux de défaillance est élevé.

Pour des applications critiques (aérospatial, automobile, biomédical, …) différentes techniques de tolérance aux fautes sont utilisées et cela à plusieurs niveaux (circuit/système et matériel/logiciel).

Le travail décrit dans ce document s'inscrit dans ce contexte et cible principalement la tolérance aux fautes des réseaux sur puces. Ce travail fait suite, et vient en complément, à la thèse de Vladimir Pasca [Pasca 2011] qui a proposé pour les NoCs des solutions de tolérance aux fautes au niveau circuit et au niveau lien.

Pour compléter cela, de manière orthogonale, nous proposerons dans ce manuscrit des solutions au niveau algorithme de routage et système, qui permettent de tolérer des défauts nombreux, de différents types et origines, et qui tiennent compte de la congestion (due à la variation du trafic ou induite par les défauts).

Dans cette thèse, un algorithme de routage adaptatif tolérant aux fautes est présenté. Sa spécificité est qu'il permet de tolérer à la fois des fautes permanentes, transitoires et intermittentes. Une nouvelle technique de recouvrement de paquet est utilisée permettant de garantir un haut niveau de fiabilité dans un contexte de taux de défaillances élevée. Pour

réduire la latence et router les paquets de manière plus efficace vers des zones moins congestionnées, une nouvelle métrique de congestion est introduite. Un algorithme exploitant cette métrique et les états de congestion des liens et des routeurs est proposé et implémenté.

Des simulations intensives sur des plateformes virtuelles ont été effectuées avec des injections de fautes permanentes, transitoires, et intermittentes, pour représenter de manière réaliste les défauts pouvant survenir dans un NoC. Les résultats en terme de tolérance aux fautes, taux de défaillance et latence démontrent l'efficacité des solutions proposées.

Ce mémoire est organisé comme suit :

Dans le premier chapitre, les problématiques liées à la tolérance aux fautes et à la congestion des NoC seront exposées.

Une étude des différentes techniques existantes traitant ces deux points sera décrite dans le chapitre 2 « Etat de l'art ».

Dans le chapitre 3, les algorithmes tolérants aux fautes et de gestion de la congestion seront présentés ainsi que la microarchitecture du routeur réalisant ces fonctionnalités.

La plateforme d'expérimentation, le simulateur de réseau et les mécanismes d'injection de fautes seront décrits dans le chapitre 4. Dans ce même chapitre les résultats de simulation seront analysés et discutés.

Enfin le dernier chapitre permettra de tirer les conclusions sur ce travail et ouvrira sur les améliorations à apporter et les perspectives.

Chapitre 1 Contexte et Problématique

Ce chapitre introduit brièvement la thématique des réseaux sur puce en présentant d'abord quelques notions de bases comme la topologie, le routage, le protocole de communication, le contrôle de flux, etc. Nous pourrons ensuite mieux définir les deux problèmes liés à la congestion dans le réseau et la tolérance aux fautes qui sont les deux points majeurs traités dans cette thèse.

1.1 Terminologie des réseaux sur puce

Un réseau sur puce, en anglais Network on Chip (NoC), est proposé comme une infrastructure d'interconnexion pour les communications dans un système sur puce (SoC). Appelé aussi micro réseau, par rapport au macro réseau qui relie un certain nombre d'ordinateurs par des câbles et des routeurs, il emprunte les modèles des macros réseaux au niveau de sa conception.

Topologie

La topologie définit l'arrangement des nœuds et des liens du réseau, en termes de placement et de connectivité des nœuds les uns par rapport aux autres.

En termes de connectivité, les topologies sont usuellement réparties en trois classes : directes, indirectes et hybrides. Dans le réseau direct, appelé aussi réseau point-à-point, chaque nœud est connecté à un certain nombre de nœuds voisins. Chaque nœud se compose d'un routeur, qui est connecté d'une part directement aux routeurs des nœuds voisins, et d'autre part à l'unité de traitement locale via une interface réseau (en anglais, Network Interface, NI). Dans un réseau indirect, la connexion entre les nœuds est établie par l'intermédiaire d'un certain nombre de commutateurs (switches). Le réseau hybride est une combinaison des deux types précédents.

Commutation

La commutation définie la façon dont est établie la connexion entre une paire de nœuds source-destination ainsi que la manière dont est acheminée un message. Il existe principalement deux modes de commutation :

Commutation de circuit : le chemin entre la paire source-destination doit être établi avant l'envoi d'un message. Les ressources allouées aux préalables ne peuvent être libérées

qu'une fois la transmission terminée. La commutation de circuit garantit une latence minimale une fois le chemin établi, mais la phase d'initialisation introduit un délai important. De plus, au niveau global, le débit est limité parce que le canal est réservé pendant la durée d'acheminement du message, bloquant ainsi l'envoi d'autres messages.

Commutation de paquet : dans ce mode de commutation, le message est divisé en plusieurs paquets (eux même divisés en flit, en anglais flow control digit). Le paquet est injecté dans le réseau sans allouer un chemin entre la paire source-destination contrairement à la commutation de circuit. Chaque paquet contient les informations nécessaires pour le routage, par exemple, l'adresse de destination, la taille du paquet en unité de flit, etc. Un paquet arrive dans un tampon d'entrée du routeur, et après calcul de routage, le paquet est envoyé vers un port de sortie correspondant sa destination. Il y a trois types de commutation de paquet en termes de stratégie de gestion de la propagation : Store and Forward (SAF), Virtual Cut-Through (VCT) et Wormhole (WH).

SAF : le paquet est le quantum de base pour une transmission. Ainsi un paquet ne peut être envoyé vers le routeur aval que s'il y a un tampon libre de taille suffisante pour stocker l'ensemble du paquet.

VCT améliore la latence par rapport à SAF en permettant d'envoyer le paquet vers le routeur aval, sans attendre, dès que celui-ci possède suffisamment de place pour accueillir le paquet. Le routeur courant n'est pas obligé de stocker tout le paquet avant de le transmettre. Cependant VC, comme SAF nécessite de prévoir un tampon dans chaque routeur au moins égal à la taille du paquet.

WH : la commutation de paquet de type Wormhole considère le flit comme unité de transmission. Cela a pour conséquence de réduire les besoins en espace mémoire pour les stockages intermédiaires. Le paquet peut avancer vers le routeur aval dès que celui-ci a dans ses tampons d'entrées/sorties un espace mémoire au moins équivalent à la taille d'un flit. Un paquet est composé de trois parties: un flit d'en-tête, des flits de corps (charge utile, en anglais payload) et un flit de queue. Le flit d'en-tête découvre le chemin vers la destination et réserve en même temps les ressources nécessaires pour le reste du paquet. Au passage du flit de queue, les ressources sont libérées et deviennent disponibles pour d'autres paquets.

La commutation de paquet de type Wormhole offre de meilleures performances que celles de SAF et VCT, mais WH introduit des risques d'inter-blocages des messages circulant

dans le réseau. Ces problèmes d'inter-blocages peuvent être résolu en utilisant un routage adéquat. Pour les systèmes sur puce la commutation de paquet de type Wormhole est préférée aux autres non seulement à cause des performances supérieures qu'elle permet mais aussi à cause d'une implémentation moins couteuse en surface. Nous nous focaliserons donc dans la suite de ce document sur les NoCs utilisant ce type de stratégie de communication.

Routage

Le routage définit quel chemin va être utilisé par un message/paquet du nœud source au nœud destination. Dans le cas Wormhole, l'en-tête d'un paquet (flit d'en-tête) contient les informations nécessaires à son routage. Lorsque le flit d'en-tête arrive à un nœud, le chemin vers le prochain nœud est défini par la fonction de routage. On distingue les routages déterministes et les routages adaptatifs [Duato 2003].

Le routage déterministe définit toujours le même chemin à suivre pour une paire (source, destination) donnée. Un exemple est le routage par ordre de dimensions, tel que le routage XY souvent employé dans un réseau sur puce en maille 2D (2D Mesh). Le paquet avance vers sa destination en traversant toujours en priorité la dimension X puis la dimension Y. La distance entre la source et la destination est minimale. Parmi les routages déterministes on trouve aussi les routages South Last, North Last, Negative First [Kariniemi 2004], …, dans lesquels un sens de propagation des paquets est supprimé pour éviter les problèmes de « deadlock » (inter-blocages). L'avantage des routages déterministes réside dans la simplicité d'implémentation, cependant ils manquent de flexibilité pour les nœuds / zones congestionnées ou défaillantes. Ce problème peut être résolu par un routage adaptif qui peut se référer à des critères comme la santé des liens de sortie, l'état d'un nœud ou de ses voisins, le trafic du réseau, etc. lors de la décision de routage. Avec ces critères, les paquets peuvent éviter les zones du réseau congestionnées ou défaillantes et par conséquent diminuer la latence. Cependant, l'implémentation est plus complexe que celle du routage déterministe. En plus le routage adaptif peut induire d'autres phénomènes dynamiques comme les « Live-Lock, » qui correspondent à une situation où un paquet continue à parcourir le réseau mais sans jamais arriver à destination, situation à laquelle le routage déterministe est naturellement immun.

Protocole de contrôle de flux

Le contrôle de flux détermine comment les ressources du réseau (canaux physiques/ virtuels, tampons, …) sont allouées aux paquets qui circulent dans ce réseau. Pour la commutation Wormhole, le contrôle de flux est généralement basé sur le mécanisme de crédit d'émission. Le routeur en amont possède des compteurs qui mesurent dans le routeur aval le nombre de tampons d'entrée libres pour chaque canal. Lorsque le routeur amont transmet un flit dans un canal, le compteur correspondant est décrémenté. Si un compteur est nul, cela signifie que le tampon en aval correspondant est plein et aucun autre flit ne peut être transmis jusqu'à ce que de la place se libère. Lorsque le routeur aval envoie à son tour un flit, il libère une partie du tampon associé, il renvoie un crédit vers le routeur en amont et le compteur correspondant est incrémenté.

Canal virtuel

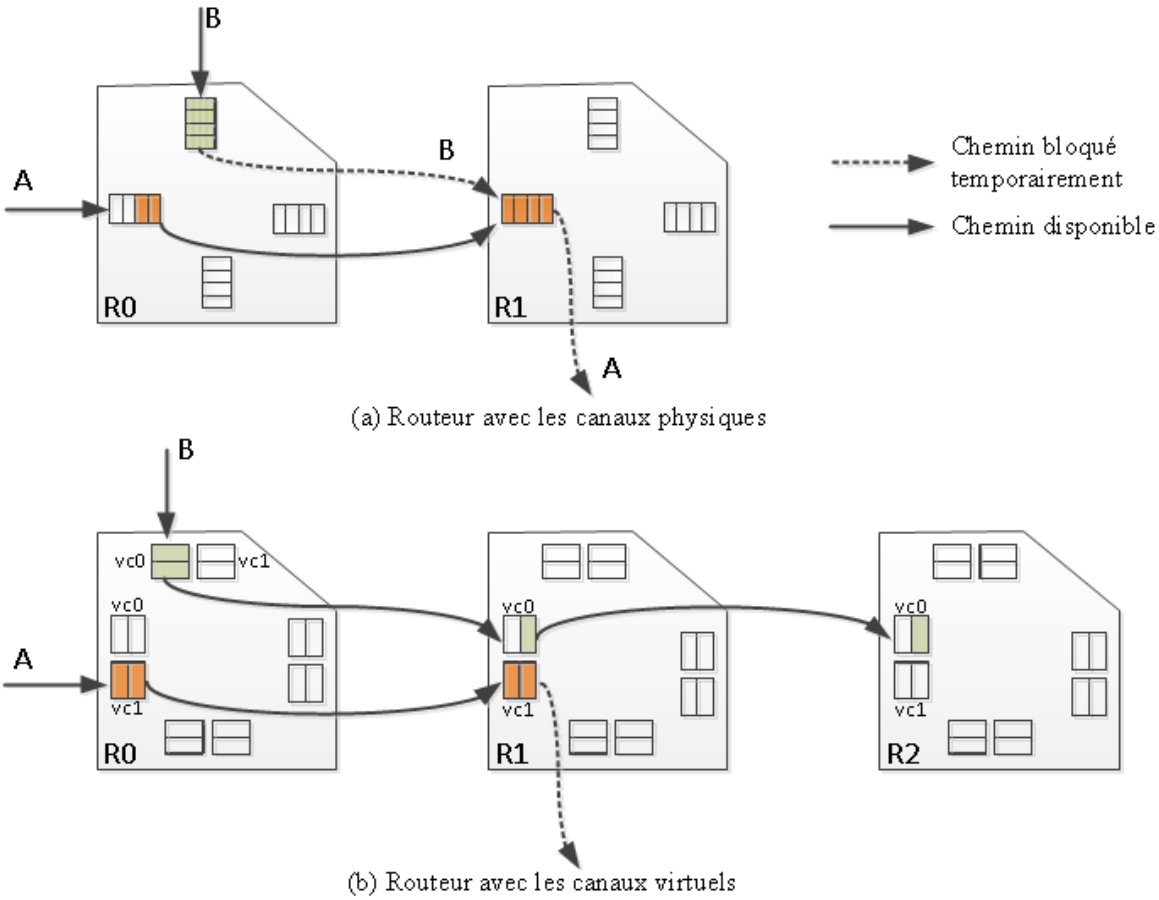


Figure 1.1 Routeurs avec et sans canaux virtuels

Dans la commutation WH le flit d'en-tête réserve un chemin physique et le reste des flits suivent. A un instant donné, on peut voir apparaître des problèmes de concurrence d'accès à un tampon d'entrée de routeur, de paquets provenant de plusieurs routeurs. Les tampons fonctionnent généralement comme des FIFOs. Ainsi, dès qu'un paquet occupe un tampon d'un canal physique, aucun autre paquet ne peut accéder à ce canal physique.

Sur l'exemple de la figure 1.1.a, un paquet A prend le canal physique (port d'entrée Est du routeur R1), et A est bloqué temporairement dans R0 et R1. En même temps, le paquet B doit aussi être envoyé vers le port d'entrée Est de R1. Bien que le lien physique soit libre, le paquet B ne peut pas traverser car le canal physique est pris par le paquet A.

Pour éviter cela, la notion de canal virtuel a été introduite. Ainsi un canal physique peut supporter plusieurs canaux virtuels (en anglais Virtual Channel, VC) multiplexés sur ce canal physique. En effet, l'intérêt des canaux virtuels est de découpler les canaux physiques des tampons mémoire. Cela permet à de multiples paquets de partager un même canal physique. Dans la figure 1.1.b, nous reprenons l'exemple précédent mais cette fois avec un routeur ayant 2 VCs pour chaque port d'entrée. Le paquet A entre dans le port Est du routeur R0 en occupant VC1 puis est envoyé vers le port est du routeur R1 toujours sur le canal VC1. Le paquet B entre en même temps sur le port Est de R0 en occupant le canal VC0. Comme le tampon d'entrée du canal VC0 du routeur R1 est libre, ainsi que le lien physique entre R0 et R1, le paquet B peut transiter vers R1 puis R2. Grâce aux canaux virtuels, les paquets peuvent traverser le même lien en multiplexant le temps d'accès, le débit est augmenté en cas de congestion et par conséquent la latence moyenne est diminuée.

Comme dit précédemment la notion de canal virtuel a aussi été introduite pour éviter l'inter blocage qui correspond à une situation où tous les paquets sont bloqués parce qu'ils s'attendent les uns les autres, attendant que les ressources qu'ils demandent se libèrent [Dally 1992]. L'utilisation des canaux virtuels permet d'avoir une plus grande diversité de routages possibles et permet de choisir la stratégie de qualité de service (QoS) adéquate en mettant les priorités sur les bons canaux et en allouant la bande passante pour les trafics prioritaires [Felicijan 2004][Rostislav 2005][Beigne 2005].

Réseau virtuel

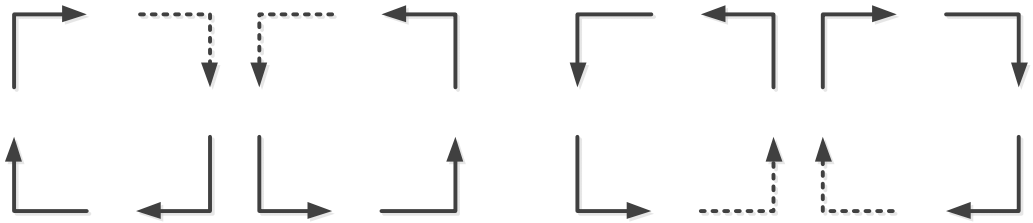


Figure 1.2 Routage South-Last (à gauche) / North-Last (à droite)

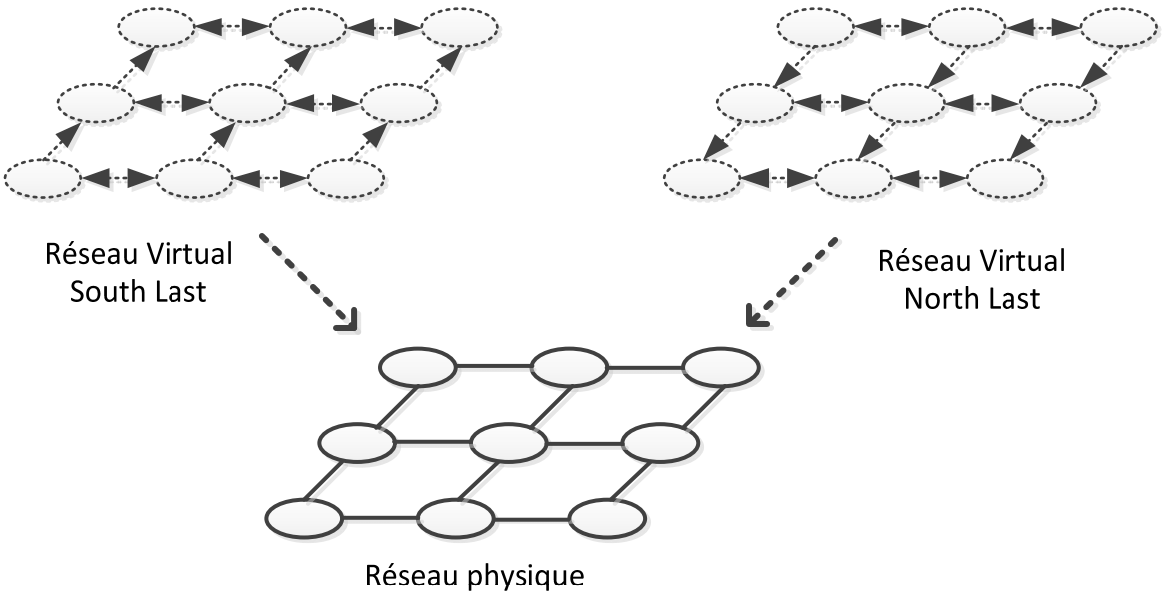


Figure 1.3 Un exemple de réseau virtuel

En 1995 [Cunningham 1995] la notion de « Réseau virtuel » (en anglais Virtual Network, abrégé VN) a été introduite. Un VN est un réseau logique projeté sur le réseau physique qui utilise les canaux virtuels. La figure 1.3 représente deux VNs (SouthLast VN et NorthLast VN) projetés sur le réseau physique qui a 4 canaux virtuels. A chaque VN est attribué 2 canaux virtuels. Chaque VN dans cet exemple possède son propre algorithme de routage. Les messages dont la destination est au sud de la source sont injectés dans le VN NorthLast, et les messages dont la destination est au nord de la source sont injectés dans le VN SouthLast. L'utilisation de plusieurs VNs augmente ainsi la diversité des routages effectifs.

Les couches d'un réseau sur puce

En 2002 Benini a proposé des piles protocolaires adaptées au réseau sur puce, comme schématisé sur la figure 1.4 [Benini 2002]. Le réseau sur puce peut être vu comme une couche physique, une couche architecture et contrôle et une couche logicielle.

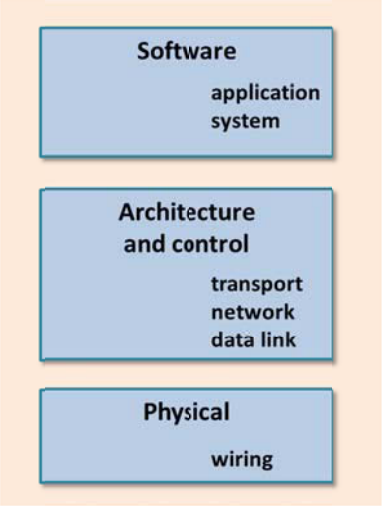


Figure 1.4 Les couches de protocole d'un NoC

La couche physique correspond aux fils physiques interconnectant les routeurs. La couche architecture et contrôle se décompose en trois niveaux : le niveau lien qui lie les nœuds et permet la transmission de messages; le niveau réseau qui détermine le parcours des données et l'adressage logique ; et le niveau transport qui permet d'établir la connexion entre les nœuds du réseau.

Critères guidant les choix de conception

Pour mesurer et quantifier les performances d'un réseau sur puce, nous avons besoin de métriques. L'un des critères les plus importants est la latence. Celle-ci correspond au délai entre le moment où un paquet est injecté dans le réseau sur puce par le nœud source et le moment où il est consommé par le nœud destination.

Un autre critère souvent considéré est le débit effectif ou bande passante utile, qui quantifie le volume de données transitant dans le réseau par unité de temps. Deux autres métriques non moins importantes sont la surface de silicium et le profil énergétique qui vont rendre compte de l'efficacité d'une conception de réseau sur puce par rapport à une autre. Enfin, pour ce qui concerne la fiabilité, et plus particulièrement la tolérance aux fautes, le taux de réussite de paquets donne une bonne indication sur la stratégie choisie. Ce taux correspond

au nombre de paquets arrivant à destination par rapport au nombre total de paquets injectés, cela pour un type de trafic donné, un débit donné et pour un certain taux de défaillance.

Ces différentes métriques aident le concepteur à choisir une stratégie par rapport à une autre selon ses contraintes, ses priorités (surface, rapidité, basse consommation ou robustesse)

1.2 Gestion de la congestion

La performance d'un NoC est liée à la gestion de la congestion en cas de trafic élevé. Le problème peut être décrit comme suit : avec une charge du réseau basse ou moyenne, le taux de trafic accepté est égal au taux d'injection. Cependant, si le trafic augmente et atteint ou dépasse un certain niveau (le point de saturation), le taux de trafic accepté diminue et la latence augmente fortement et par conséquent, le débit effectif maximal ne répond plus aux attentes. La raison en est que lorsque le trafic est élevé, plusieurs paquets entrent en concurrence pour accéder aux mêmes ressources (canaux physiques ou virtuels, ports de sortie, etc.). Cependant seul un paquet peut réserver une ressource, les autres restent dans une file d'attente et bloquent à leur tour d'autres paquets amonts. Une telle situation peut conduire à la saturation du réseau et fortement dégrader les performances.

Plusieurs travaux sur les comportements des réseaux, avec des trafics correspondant à des applications réelles, montrent que ces trafics sont irréguliers, en rafales, et le trafic-pic va jusqu'à saturer le réseau [Flich 1999] [Martinez 1999] [Silla 1998].

La gestion de la congestion est donc incontournable et sa mise en œuvre est multi contraintes: large bande passante demandée avec en même temps une implémentation faible en surface et basse consommation.

Pour traiter le problème de la congestion, trois étapes doivent être observées:

1. Mesure de la congestion : il est nécessaire dans un premier temps de définir des métriques et des moyens de mesurer la congestion dans le réseau en surveillant par exemple la latence ou les états du réseau.
2. Notification/propagation de l'information de la congestion : une fois que la congestion est détectée, il faut signaler/transmettre cette information au bloc décisionnaire local ou global.
3. Action pour la congestion : il s'agit ici de traiter l'information de congestion et d'appliquer la stratégie de désengorgement choisie. Plusieurs travaux proposent

une régulation du trafic. Lors de la détection de la congestion, les nœuds sources réduisent le taux d'injection de messages dans le réseau [Gerla 1980] [Jacobson 1988] [Jain 1992], voire stoppe l'injection complètement. D'autres travaux ont proposé l'utilisation d'un routage non-minimal (au contraire du routage minimal en termes de nombre de sauts/hops) pour équilibrer la charge sur le réseau [Dally 1993]. Des solutions introduisant l'ajout de tampons supplémentaires ont été proposées, cela afin de ralentir les messages et réduire le trafic du réseau [Smai 1998].

Cependant toutes les techniques proposées précédemment sont adaptées à des NoC sans défauts. Hors, pour les futurs MPSoCs, les taux de défaillance vont aller crescendo. Le nombre de défauts, dus entre autre au vieillissement, va s'accroitre de manière significative entrainant de manière inéluctable des problèmes de congestion, avant même que le NoC soit saturé par un trafic élevé.

Les questions qui peuvent se poser sont les suivantes :

a) Les métriques de mesure de la congestion existantes sont elles optimales et adaptées ?

b) Peut on mettre en place une solution de traitement de la congestion qui tienne compte non seulement du trafic mais aussi des ressources du NoC qui sont ou deviennent défectueuses ?

Une première contribution de cette thèse va consister à répondre à ces questions.

1.3 Gestion des fautes

1.3.1 Types de fautes

Les défauts dans un circuit intégrés peuvent être modélisés par ce que l'on appelle des fautes et ces dernières peuvent être rangées dans trois catégories selon leur comportement : permanentes, transitoire et intermittentes.

Les fautes permanentes correspondent à des changements physiques irréversibles dans le circuit dus par exemple à des perturbations dans le processus de fabrication ou aux phénomènes de vieillissement du circuit chez l'utilisateur. Sans traitement adéquat, une faute permanente ne peut donc être éliminée par une simple réinitialisation du circuit.

Les fautes transitoires sont pour leur part induites par des événements environnementaux externes temporaires. Par exemple les radiations solaires créent un flux de particules qui peuvent modifier l'état d'un nœud du circuit. Ces fautes ne sont pas destructrices. Ainsi, une erreur due à une faute transitoire peut être éliminée par la réinitialisation du matériel. Les fautes transitoires apparaissent généralement pendant un temps très court.

Les fautes intermittentes sont dues quant à elles au fonctionnement du circuit dans un environnement instable : variations de température, de tension d'alimentation, etc. Similaire aux fautes transitoires, une faute intermittente a aussi une durée de vie courte, mais elle apparaît et disparaît aléatoirement dans le temps sans période fixe. Pour leur modélisation, elles sont souvent regroupées sous la forme d'une rafale de fautes d'une durée de plusieurs cycles [Yu 2012]. Comme les fautes transitoires, les erreurs dues à une faute intermittente peuvent être réglées par la réinitialisation du circuit.

En bref, sauf pour les fautes permanentes qui endommagent physiquement la puce, les fautes transitoires et intermittentes peuvent être éliminées par la réinitialisation. Cependant, cette procédure est longue et pendant cette période le système n'est pas disponible. L'application doit être stoppée complètement, et tous les résultats obtenus précédemment sont perdus si des mécanismes de sauvegarde et de restauration du système n'ont pas été prévus, sans compter que ces mécanismes augmentent eux aussi le temps de non-disponibilité du système ainsi que sa complexité.

Dans la littérature scientifique, les solutions permettant de tolérer la présence de fautes dans un NoC sont nombreuses. Cependant, chaque solution proposée ne cible qu'un type de faute : permanente, transitoire ou intermittente, ce qui ne reflète que partiellement la réalité des circuits actuels et futurs où l'on retrouve un mix de ces trois types de fautes.

Ainsi une autre question à laquelle va essayer de répondre ce travail est :

c) Peut on proposer une architecture de NoC permettant de tolérer simultanément des fautes multiples et de type différents à savoir permanentes, transitoires et intermittentes ?

1.3.2 Modèles de fautes dans un NoC

Outre les types de fautes cités précédemment, on peut se poser la question de la granularité du défaut considéré. En effet la solution choisie pour tolérer un défaut sera plus ou moins couteuse et complexe selon la granularité de l'erreur à « réparer ».

Les routeurs dans un NoC sont interconnectés par des paires de liens unidirectionnels (connexion bidirectionnelle). Chaque lien unidirectionnel regroupe des fils parallèles dont le nombre est égal à la taille d'un phit (digit physique). Les NoCs ciblés par ce travail sont des micro-réseaux qui seront utilisés dans des systèmes massivement multi-processeurs et dont les taux de défaillance seront élevés. Il semble donc judicieux dans un premier temps, pour des raisons de simplicité, de privilégier deux modèles de faute à gros grain :

- un lien unidirectionnel est défectueux. Cela peut être du à un défaut des fils reliant les routeurs ou à un défaut sur les ports d'entrée/sortie de ces mêmes routeurs. Les liens partiellement défectueux, dans lesquels seulement une partie des fils du lien est fonctionnelle ne seront pas considérés dans ce travail. Ainsi un lien complet est considéré comme défectueux lorsque ses deux liens unidirectionnels sont considérés comme fautifs.
- Un nœud est défectueux. Avec le modèle de faute de lien décrit précédemment, un nœud est considéré comme un nœud défaillant lorsque tous ses liens sont défectueux.

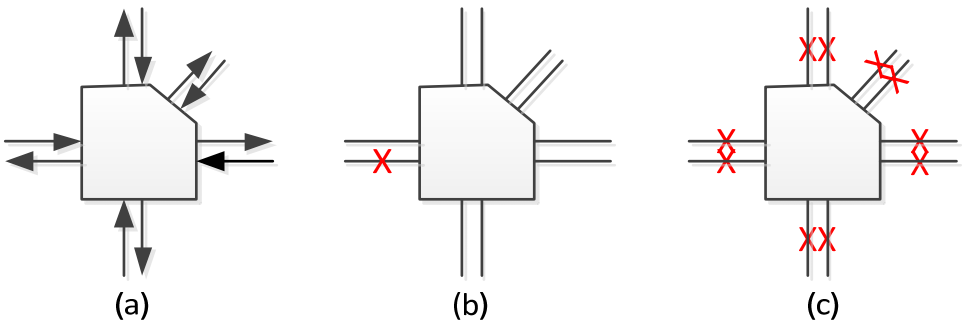


Figure 1.5 Modèles de faute : (a) un routeur avec 5 ports, deux liens unidirectionnels pour chaque port. (b) modèle de faute de lien unidirectionnel. (c) modèle de faute de nœud.

La figure 1.5 montre les différents modèles de faute considérés dans cette thèse, avec un routeur avec 5 ports (sud, nord, est, ouest et local). Nous considérons que tous les liens sont défaillants simultanément pour un nœud défectueux.

1.3.3 Tolérance aux fautes multi-niveaux

Différentes techniques de tolérance aux fautes existent selon le niveau/la couche d'abstraction où elles interviennent. Elles sont le plus souvent basées sur l'introduction de redondance de données, de redondance temporelle ou spatiale.

Au niveau circuit, on peut trouver des techniques comme GRAAL [Nicolaidis 2007], qui permettent de tolérer des fautes transitoires de types « soft error » en effectuant un double échantillonnage.

Au niveau lien, pour détecter et corriger des fautes, il est possible d'introduire des techniques à base de codes de détection et correction d'erreurs (ECC) [Pasca 2011], de retransmettre et/ou de sérialiser les données [Nicolaidis 2011].

Au niveau du système ou de l'application, des techniques logicielles peuvent permettre de détecter, stopper et relancer une application en cas de défaillance du système : techniques de Checkpoint & Rollback [Rusu 2010], migration des tâches vers des ressources saines, …

Au niveau routage, les solutions proposées sont basées soit sur un routage déterministe, soit sur un routage adaptatif. Comme indiqué précédemment le routage adaptatif est préférable pour des systèmes où la disponibilité de service est un critère primordial, le système n'ayant pas besoin d'être réinitialisé en cas de défaillance.

Cependant il n'y a pas de solution idéale qui permette à elle seule de détecter et tolérer tous les types de faute possibles. En fait ces techniques sont complémentaires. Selon la criticité de l'application finale et de la technologie utilisée, une combinaison de solutions multi-niveaux devra être mise en place : par exemple ECC (lien) + retransmission (système) + routage adaptatif.

1.4 Les contributions

L'objectif de ce travail de thèse est d'améliorer les performances d'un NoC en termes de latence et débit pour un trafic élevé et en présence de fautes nombreuses et de différents types. Pour cela, on se propose à la fois d'améliorer le traitement de la congestion et de tolérer des fautes qui peuvent être permanentes, transitoires ou intermittentes. Les solutions proposées jusqu'ici permettent de traiter chaque type de faute indépendamment des autres et

pour des taux de défaillance qui ne correspondent pas à ce que l'on va rencontrer avec des densités d'intégration toujours plus grande. Le contexte dans lequel nous nous plaçons dans ce travail de thèse cible les applications s'exécutant sur du matériel soumis à un taux de défaillances élevées et où le critère de disponibilité est un paramètre clé. Ainsi les techniques de gestion de trafic et de tolérance aux fautes que nous proposons par la suite s'exécuterons en ligne, c'est-à-dire en cours d'exécution de l'application.

Les contributions originales qui vont être décrites et étudiées dans la suite de ce manuscrit sont :

1) Proposition et étude d'une nouvelle métrique de mesure de la congestion.
2) Amélioration, adaptation et implémentation de techniques de tolérance aux fautes permanentes, transitoires et intermittentes.
3) Intégration des propositions 1) et 2) pour proposer une microarchitecture du routeur correspondant.
4) Extension de la proposition aux topologies de type Tore.
5) Des simulations extensives faisant varier différents paramètres nous permettront d'évaluer l'efficacité et la viabilité de ces nouvelles approches.

Chapitre 2 Etat de l'art

Dans ce chapitre nous allons dans un premier temps restreindre les champs de recherche en fonction des objectifs initiaux. Suite à cela les travaux existant les plus pertinents concernant la gestion de la congestion puis la tolérance aux fautes dans les NOCs seront présentés. Nous étudierons leurs limites en termes d'efficacité, de coût et de complexité. Nous décrirons ensuite comment nous avons sélectionné une plateforme de simulation parmi les différentes existantes.

Caractéristiques du NoC étudié

Pour répondre aux questions soulevées précédemment tout en ciblant des NoCs pour systèmes massivement multiprocesseurs et dans un contexte de défaillances élevées, il faut au préalable définir les caractéristiques des NoCs qui feront l'objet de cette étude. Les deux contraintes primordiales qui vont guider les différents choix de conception sont la surface de silicium et la consommation additionnelles.

Topologie : 2D Mesh. Pour tenir compte de la mise à l'échelle nous choisissons de travailler sur des réseaux en maille 2D puis en tore 2D. Les solutions trouvées pourront alors plus facilement être étendues aux réseaux 3D.

Commutation : Wormhole. Le type de commutation choisie est la commutation de paquet WormHole car elle nécessite moins de mémoires tampons que SAF et VCT. De plus WH permet d'avoir une latence peu élevée, ce qui correspond à un de nos objectifs. Par conséquent, le contrôle de flux le plus approprié est celui basé sur le crédit [Dally 2004]. Néanmoins, la commutation de type WormHole introduit le risque de création d'interblocages. Il faut donc prévoir des techniques de prévention ou bien de recouvrement de l'interblocage.

Routage : Nous souhaitons tolérer des fautes en ligne, sans avoir à réinitialiser le système, le routage adaptif est donc le plus adapté, le routage déterministe ne pouvant pas modifier en cours d'exécution le routage en présence de liens/routeurs défectueux ou congestionnés. Les contraintes dues à l'embarqué, notamment la surface de silicium, font qu'on préfèrera un routage mis en œuvre grâce à des machines à états finis (FSM) plutôt qu'à l'utilisation de tables de routages. Enfin, pour éviter l'interblocage, nous utiliserons des canaux virtuels.

Architecture type du routeur

Il découle de ces choix et contraintes pour les NoCs embarqués, une architecture de routeur présentée sur la figure 2.1. Ce routeur est constitué de tampons en entrée, le contrôle de flux est à base de crédits et la commutation est de type WormHole. L'architecture du routeur est un pipeline à 4 étages [Dally 2004]. Ces étages correspondent à l'écriture dans les tampons (Buffer Write, abrégé BW), au calcul du routage (Route Computation, abrégé RC), à l'allocation du canal virtuel (Virtual Channel Allocation, abrégé VA), à l'allocation de commutation (Switch Allocation abrégé SA) et au transfert de commutation (Switch Transfer abrégé ST).

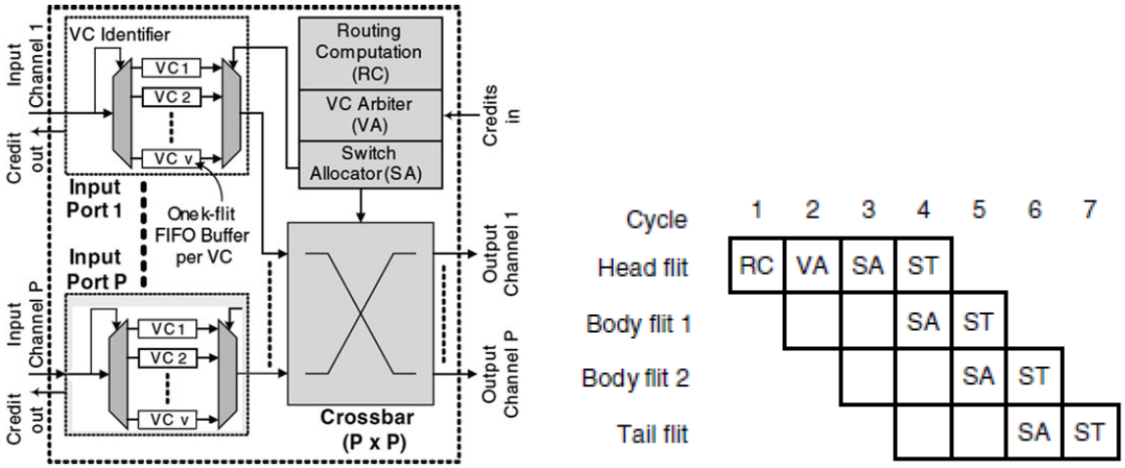


Figure 2.1 Structure d'un routeur à canaux virtuels pour réseau sur puce

Quand un flit d'en-tête arrive à un port d'entrée P, il est stocké dans le canal virtuel VC correspondant. L'adresse de destination stockée dans le flit d'en-tête est lue par le bloc RC qui calcule par quel port va sortir ce paquet (Nord, Sud, Est, Ouest). Dans le cycle suivant, le bloc VA attribue un canal virtuel correspondant, dans le routeur aval. Si en même temps plusieurs paquets demandent le même canal, le canal est attribué à un paquet selon une politique de type Round Robin ou FCFS (First-come, first-served). Lorsque le canal virtuel est alloué, dans le cycle suivant (ST), le paquet demande son transfert vers le tampon du routeur aval. Le paquet à qui est attribué le lien peut avancer d'un flit vers sa destination.

La littérature sur les réseaux d'interconnexion d'une manière générale est très vaste et fournie et son intégration dans les circuits a accéléré le nombre de travaux de recherche sur le sujet depuis la fin des années 1990. Une conférence leur est même dédiée : le symposium NoCs depuis 2007.

Nous ne ferons donc pas dans la suite un état de l'arte exhaustif des techniques de gestion de la congestion et de tolérance aux fautes. Seules les techniques correspondant aux NoCs dont les caractéristiques ont été citées plus haut seront décrites.

2.1 Congestion de réseau sur puce

La congestion dans un réseau sur puce de type commutation de paquet correspond à un état où la performance est dégradée due à la saturation des ressources du réseau, tels que les liens, les tampons, les canaux virtuels, etc. Les effets résultant des problèmes de congestion incluent : retard de livraison de messages (latence), réduction du débit (bande passante), voire paralysie, toutes les communications s'arrêtent.

Les approches existantes pour la gestion de la congestion peuvent être classées en deux catégories : la prévention de la congestion et le recouvrement [Yang 1995].

- Prévention

La prévention nécessite pour chaque nœud du circuit d'avoir l'autorisation d'injecter un paquet dans le réseau. Basé sur les connexions établies, le taux d'injection de messages est régulé en fonction des ressources réservées. L'inconvénient de ce type de gestion est que le débit du réseau est limité et cela d'autant plus que la qualité de service est élevée.

- Recouvrement

Les techniques de recouvrement sont basées sur la surveillance du réseau et déclenchent des actions si la congestion est détectée. Cette approche consiste en trois étapes.

Premièrement, la congestion doit être détectée localement. Certaines stratégies [Hyatt 1997][Smai 1998] mesurent le temps d'attentes des messages bloqués, alors que d'autres [Ascia 2008] [Gratz 2008] [Ma 2011] regardent l'état d'utilisation des ressources de réseau.

Deuxièmement, la congestion doit être notifiée aux nœuds du réseau. Nous pouvons classer ces techniques en deux catégories en termes de champ de notification : local/régional ou global.

Troisièmement, des actions doivent être appliquées pour résoudre le problème de congestion. Localement [Dally 1993] [Baydal 2005], en arrivant vers un nœud ou une région, un paquet peut être amené à contourner la zone, selon le degré de congestion et cela avec l'aide d'un algorithme de routage adaptif. Globalement, au niveau du réseau, le taux

d'injection de nouveaux messages dans les différents nœuds peut être ajusté à la baisse [Baydal 2005] [Thottethodi 2001] [Towles 2003]. Ces deux techniques, contournement et régulation du trafic, sont complémentaires et orthogonales et peuvent être utilisées ensembles

Nous présentons dans la suite trois approches différentes de gestion de la congestion par recouvrement qui utilisent des techniques de contournement

1. NoP

Dans un réseau utilisant un contrôle de flux basé sur le crédit, le routeur de chaque nœud a la possibilité de connaitre les états de congestion de ses voisins (nombres de canaux virtuels, nombres de cases disponibles des tampons dans le routeur aval). Cette connaissance est appelée conscience de la congestion locale (en anglais Local Congestion Awareness, abrégé LCA).

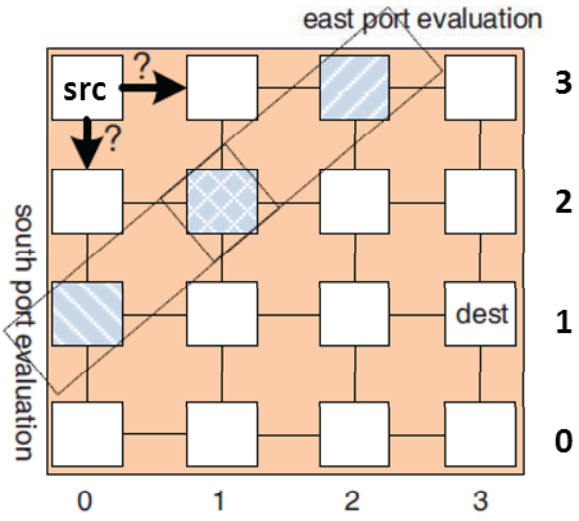


Figure 2.2 Technique NoP : src : nœud source (0,0), dest : nœud destinataire (3,2) [Ma 2011]

Afin d'éviter la congestion, NoP (en anglais Neighbors on Path, abrégé NoP) évalue la congestion de nœuds distants de deux rangs (hops) par rapport au nœud courant en utilisant la LCA de ses voisins au lieu de ne prendre en compte que sa propre LCA [Ascia 2008].

La figure 2.2 présente un exemple de NoC utilisant l'approche NoP. Lorsqu'un paquet est injecté dans le nœud de source (0,3), le routage adaptatif donne 2 directions possibles vers sa destination, l'un est l'est vers le nœud (1,3) et l'autre est le sud vers le nœud (0,2). Le choix doit être fait en comparant les états de congestion sur ces deux directions. NoP calcule,

anticipe les directions de sortie pour le paquet qui arrive dans le nœud (1,3) ou dans le nœud (0,2) :

1, Mesure de congestion : la métrique de congestion « nombre de tampons libres dans les canaux» (« Free Buffer ») est mesurée pour chaque possibilité de routage (0,2) et (1,3). Chaque mesure prend en compte la mesure des tampons libres des nœuds directement voisins pour chaque cas, ici (0,1) et (1,2) pour le premier cas et (1,2) et (2,3) pour le second.

2, Propagation de la métrique de congestion : les mesures de congestion pour les deux voisins (0,2) et (1,3) sont envoyées au nœud (0,3).

3, Agrégation des métriques de congestion : la métrique pour la direction vers le nœud (0,2) est la somme des « Free Buffer » pour les canaux allant vers les nœuds (0,1) et (1,2) ; de même, la métrique pour la direction vers le nœud (1,3) est la somme des « Free Buffer » pour les canaux allant vers les nœuds (1,2) et (2,3). Le paquet sera envoyé alors vers le nœud dont la mesure est la plus faible.

Les résultats observés pour cette technique montrent des performances (latence) bien meilleures que le cas où seulement LCA (1 hop) est considérée. Cependant seule la congestion sur une petite région (une distance de 2 hops) peut être prise en compte, ce qui est limitant lorsqu'on vise des réseaux de grandes tailles. Un autre inconvénient est que le surcoût au niveau du silicium pour implanter cette technique est non négligeable.

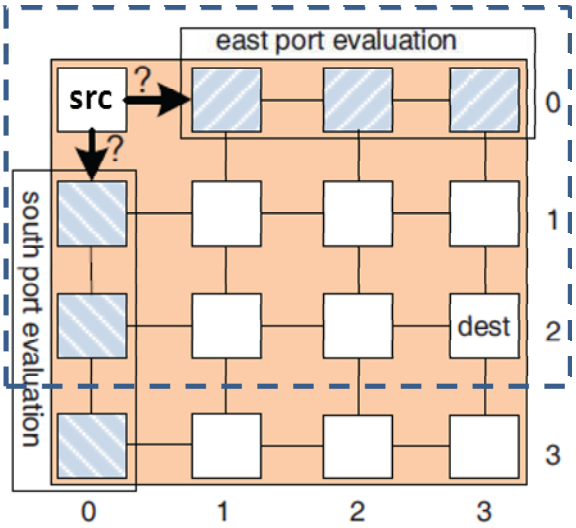


Figure 2.3 Exemple de RCA : src : nœud source (0,0), dest : nœud destinataire (3,2)

2. RCA

NoP prend donc en compte seulement les informations de congestion à une distance de 2 hops. RCA (en anglais Regional Congestion Awareness, abrégé RCA) [Gratz 2008] résout ce problème par la propagation de l'état de congestion le long des axes (figure 2.3).

1, Mesure de congestion : dans chaque nœud, la métrique de congestion LCA est mesurée pour tous les canaux liés.

2, Propagation de la métrique de congestion : les mesures de congestion sont propagées le long de toutes les dimensions (par exemple, pour une topologie maille 2D, la propagation est horizontale et verticale.)

3, Agrégation de la métrique de congestion : la métrique pour le nœud (0,2) est la somme des métriques de tous les nœuds verticaux du nœud (0,3) jusqu'au nœud (0,0); et la métrique pour le nœud (1,3) est la somme des métriques de tous les nœuds horizontaux du nœud (3,0) jusqu'au nœud (0,0).

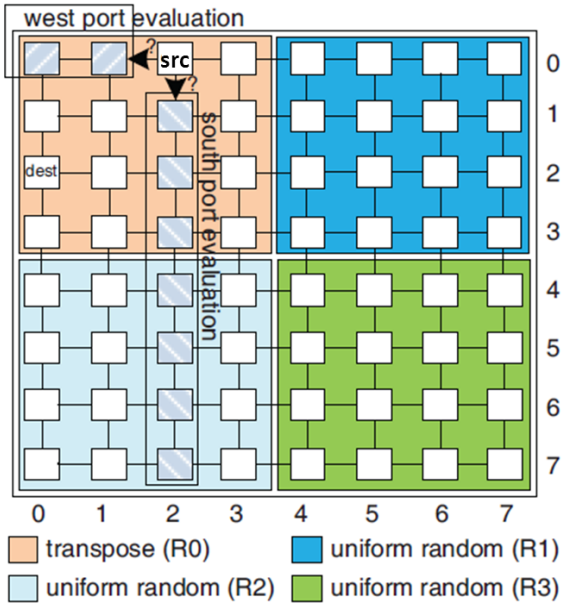


Figure 2.4 Exemple de RCA dans un NoC avec plusieurs applications [Ma 2011]

RCA peut donc évaluer la congestion globale sur les axes de propagation, et cela à chaque cycle au fur et à mesure que le paquet avance. Cependant, cette méthode inclut dans son calcul de routage des nœuds / régions non pertinents. Ainsi dans la figure 2.3, RCA évalue la congestion dans tout le réseau au lieu de s'arrêter à la zone de taille 3x4 qui est

définie par les positions de la paire source et destination. Cela peut entrainer de mauvais choix de routage. Par exemple, dans le cas où le MPSoC est prévu pour exécuter plusieurs applications et que le NoC est partitionné en conséquence mais sans prévoir de technique d'isolation ou de filtrage, on peut se retrouver dans la situation illustrée sur la figure 2.4. Dans ce cas la mesure de congestion va entrainer un routage du paquet vers la mauvaise direction.

3. DBAR

Une solution remédiant à ce problème a été mise au point. DBAR ([Ma 2011] prend en compte la position du nœud destination (zone de parcours minimale). Il résout le problème « d'interférence » dans RCA en filtrant les mesures de congestion au-delà de la zone du parcours minimal défini par la paire source-destination.

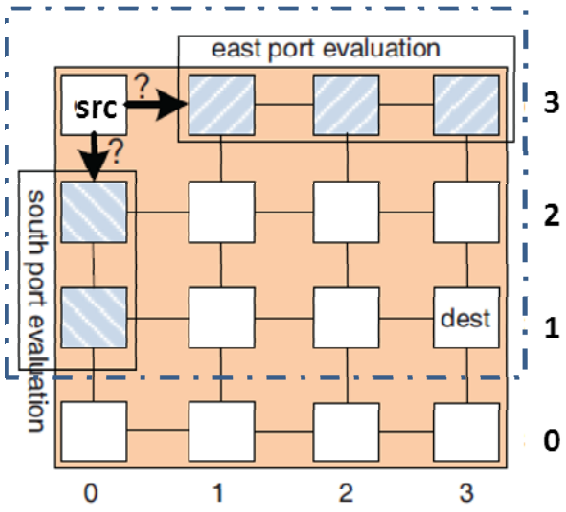


Figure 2.5 Exemple de DBAR : src : nœud source (0,0), dest : nœud destinataire (3,2)

DBAR évalue l'état de la congestion en suivant les principes suivants:

1, Mesure de congestion : dans chaque nœud, la métrique de congestion LCA est mesurée, comme précédemment.

2, Propagation de la métrique de congestion : Comme pour l'approche RCA, les mesures de congestion sont propagées suivant les axes X et Y.

3, Agrégation de la métrique de congestion : ici la différence de DBAR par rapport à RCA est que DBAR sur un axe donné ne fait la somme des mesures de congestions que jusqu'au niveau de la destination et pas sur tout l'axe. Cette technique définit bien la zone

connexe ou l'information de congestion est pertinente. De plus, elle permet de résoudre le problème soulevé par RCA lorsqu'un système est multi-applicatif et que le NoC doit être partitionné.

Inconvénients majeurs de ces techniques (NoP, RCA, DBAR)

Ces techniques pour la propagation/notification des mesures de congestion nécessitent un nombre important de fils supplémentaires entre les routeurs. Les fonctions d'agrégation (sommation) et de calcul du routage ne sont pas négligeables en terme de surcoût en silicium. Enfin, la propagation de notification dure un cycle pour chaque hop. Les informations sur les états de congestion deviennent assez rapidement obsolètes lorsqu'un paquet doit voyager sur une longue distance. Ces dernières deviennent inutiles et donnent une vision erronée de la congestion d'une zone, entrainant une possible augmentation de la congestion.

Métriques de mesure de la congestion

Les différentes approches décrites précédemment n'utilisent pas les mêmes métriques de mesure de la congestion. NoP utilise le nombre de cases de tampons libres en unité de flit. RCA exploite des métriques combinées parmi lesquelles le nombre de tampons libres, le nombre de canaux virtuels libres et le nombre de requêtes de transfert dans le crossbar vers un port de sortie donné. Quant à DBAR, l'approche mesure le nombre de canaux virtuels libres.

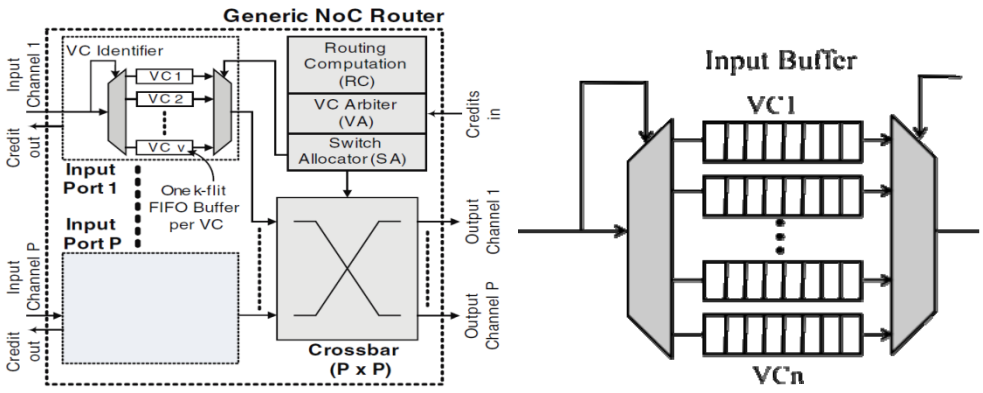


Figure 2.6 Structure d'un routeur de référence et le tampon d'entrée

Une première raison au fait que les métriques utilisées ne sont pas les mêmes est qu'elles ne se basent pas sur la même architecture de routeur.

Pour un routeur à canaux virtuels, le nombre de canaux libres pour un port d'entrée signifie que ce port peut recevoir plusieurs nouveaux paquets. Pour un routeur à canaux

physiques, ce nombre est soit 0 soit 1. Le nombre de cases libres est un meilleur indicateur sur l’état de congestion.

Mais de manière générale, aucune des métriques utilisées jusqu’ici ne peut donner une information précise sur l’état de congestion. Si nous prenons le routeur à canaux virtuels comme routeur de référence, il est clair que si moins on a de canaux virtuels libres ou si plus on a des tampons bien remplis, plus les canaux sont congestionnés. Cependant le nombre de canaux virtuels libres n’informe pas sur le fait qu’un tampon est bien rempli, à moitié ou peu rempli. De même le nombre de tampons libres, ne donne pas d’information sur l’utilisation des canaux virtuels. Mais ceci dit, un paquet ne pourra pas avancer s’il n’y pas de canal virtuel libre en aval même si tous les tampons des canaux virtuels sont peu remplis.

Ces observations nous amènent à penser que les métriques utilisées bien qu’utiles ne sont pas suffisantes pour donner une évaluation précise de la congestion sur un nœud, même lorsqu’elles sont combinées entre elles comme dans l’approche RCA.

Nous avons montré dans cette section les principales méthodes de gestion de la congestion dans un NoC. De manière générale, pour une architecture donnée on doit se poser les questions suivantes :

- Métrique de congestion : quelle est la plus pertinente ?
- Propagation de la congestion et champ de notification : que propager et jusqu’où ?
- Agrégation et choix du routage : quel algorithme de routage doit être mis en œuvre pour guider les paquets de manière efficace du point de vue de la congestion ?

Dans le chapitre suivant, nous allons répondre à ces questions en analysant le trafic dans le réseau. Nous proposerons une nouvelle métrique de mesure de la congestion plus représentative que les métriques existantes. Cette métrique pourra être appliquée aux techniques discutées dans ce chapitre.

2.2 Gestion de fautes

Cette section a pour but de présenter quelques notions de base sur la tolérance aux fautes, puis de décrire quelques approches spécifiques au type de NoC visé dans cette thèse.

2.2.1 Notion de tolérance aux fautes

D'après la définition dans [Avizienis 2004], la sûreté d'un système est représentée principalement par ses trois attributs :

La disponibilité : il s'agit de savoir si le système est toujours disponible et fournit la fonction correcte.

La fiabilité : le système a t'il un taux de défaillance élevé ? Ce dernier peut rendre le système hors service partiellement ou complètement.

La maintenabilité : quelle est la difficulté pour mettre à jour le système ou le réparer après une défaillance ?

En terme de mesure, nous pouvons qualifier la fiabilité d'un système par son taux de défaillances, le temps moyen entre deux défaillances consécutives et le temps moyen de remise en service après une défaillance.

La fiabilité d'un système est mise en péril par des fautes, des erreurs et des défaillances. Le lien entre ces trois notions est illustré dans la figure 2.7.

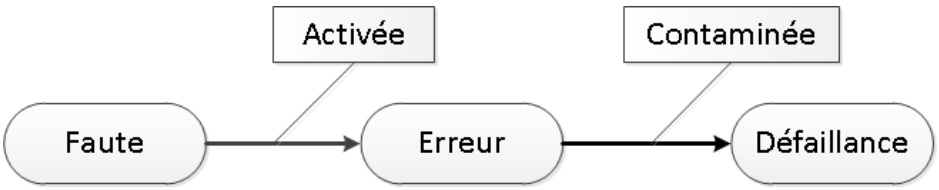


Figure 2.7 Relation entre la faute, l'erreur et la défaillance

Une faute est un modèle mathématique représentant le comportement d'un ensemble de défauts physiques pour un système donné. Par exemple, pour les circuits VLSI, le modèle le plus utilisé est le modèle de collage (stuck-at-1/0) correspondant à une affectation permanente à 0 ou 1 d'une entrée ou sortie de porte logique. Il existe d'autres modèles faisant intervenir entre autre une notion de temporalité/durée de la faute.

Une erreur correspond à l'activation de la faute lors de l'exécution. Elle correspond à un décalage entre le comportement souhaité et le comportement réel. Elle a lieu à l'intérieur du système, et correspond par exemple à une donnée erronée ou un état invalide.

Une défaillance correspond à la propagation de l'erreur vers les sorties du système (contamination). Elle peut être d'importance plus ou moins grande selon la criticité de l'application, selon qu'elle modifie les données seulement ou stoppe l'exécution du système.

Ainsi une faute non masquée et activée conduit à une erreur, qui peut entrainer une défaillance.

En ce qui concerne les méthodes permettant d'améliorer la fiabilité d'un système, elles peuvent être classées en quatre catégories : prévention de fautes, prévision de fautes, suppression de fautes, et tolérance aux fautes. Ces quatre approches peuvent cohabiter et coopérer afin de rendre un système plus fiable. Cependant, les trois premières (prévention, prévision, et suppression) n'interviennent que dans la phase de conception et de fabrication. Par exemple, les structures de durcissement contre les effets des radiations sont définies et ajoutées dans la phase de conception et les circuits pour lesquels des fautes ont été détectées sont enlevés pendant la phase de test de fabrication.

Pendant l'utilisation du système, des fautes peuvent apparaître et cela de manière aléatoire. Ces fautes peuvent être dues aux effets du vieillissement, des variations de fréquence, de température et de tension (DVFS), aux effets de diaphonie, du bruit, etc.

Il en résulte que pour améliorer la fiabilité, la quatrième catégorie d'approche, la tolérance aux fautes, est nécessaire, voire obligatoire pour des systèmes à applications critiques.

Les techniques de tolérance aux fautes, sont généralement rangées dans deux familles, les techniques dites statiques et les techniques dites dynamiques.

- La tolérance aux fautes statique introduit par exemple de la redondance matérielle (duplication voire triplication d'unités de traitement NMR [Lyons 1962]), ou de la redondance d'information (introduction de codes détecteurs et correcteurs d'erreurs ECC [Hamming 1950])

- La tolérance aux fautes dynamique pour être mise en œuvre s'appuie sur trois étapes critiques : détection et localisation de la faute, isolation de la faute et recouvrement.

1. Détection et localisation de fautes

Les techniques matérielles existantes peuvent être exécutées de manière **continue ou périodique**. Par exemple, Dual Modular Redundancy (DMR) [Siewiorek 1998] et DIVA [Austin 1999] permettent de détecter des défauts dans le silicium en exécutant en parallèle les calculs et en comparant le résultat.

Pour les techniques de détection périodique, une technique bien connue et largement utilisée est l'approche Sauvegarde-Restauration (Checkpoint-Rollback [Rusu 2010]). Ici l'état du système est testé et sauvegardé périodiquement. Le système valide l'exécution après la réussite de la vérification périodique, sinon un échec conduit à une reprise du dernier état sauvegardé.

Parmi les techniques de détection en ligne, on trouve des solutions matérielles basées sur du BIST [Bushnell 2000] mais aussi des solutions logicielles. [Constantinides 2007] propose par exemple une technique enrichissant le jeu d'instruction avec des instructions pour l'accès et le contrôle (ACE, Access-Control Extension) des états intérieurs de processeur. Cette technique ajoute aussi un firmware spécial qui permet de suspendre périodiquement le processeur pour exécuter des tests utilisant ces instructions ACE pour la détection et le diagnostic de fautes.

2. Isolation de la faute

L'isolation (confinement) de la faute a pour objectif de ne pas laisser propager son ou ses effets potentiels. Au niveau des solutions, en matériel, il est possible de masquer la faute voire désactiver les composants détectés comme défectueux. En logiciel, des solutions comme [Pullum 2001] permettent d'isoler les applications afin d'éviter une propagation d'une défaillance système générale due à une ou plusieurs fautes.

3. Le recouvrement de faute

Il peut être vers l'avant ou vers l'arrière. Le recouvrement de faute vers l'arrière abandonne l'exécution courante si une erreur est détectée. Le système essaie de restaurer le dernier état correct sauvegardé au préalable (partie Rollback de l'approche Checkpoint & Rollback). Le recouvrement avant quant à lui n'arrête pas l'exécution si une erreur est détectée. Cette erreur est compensée/corrigée par des techniques de redondance matérielle (NMR avec voteur) ou d'information ECC (Error Code Correction).

Il est à noter que pour un système complexe, là encore selon la criticité de l'application, il est possible d'intégrer différentes techniques de tolérance aux fautes en matériel et/ou en logiciel.

Les différentes approches de tolérance aux fautes et leur classification s'appliquent aussi aux micro-réseaux embarqués. Nous allons par la suite restreindre notre étude seulement à ce type de composant.

2.2.2 Techniques de tolérance aux fautes dans un NoC

Comme indiqué au début de ce chapitre, au vu des objectifs de ce travail de thèse, les caractéristiques du type de NoC tolérant aux fautes que nous étudions sont : (1) topologie 2D mesh; (2) commutation wormhole et contrôle de flux basé sur les crédits ; (3) routage adaptif. Nous nous intéresserons donc qu'aux techniques de tolérance aux fautes dites « dynamiques ». Dans le chapitre précédent (section 1.3.3) nous avons vu que des solutions pouvaient améliorer la tolérance aux fautes à différents niveaux : niveau circuit, niveau lien, niveau routage et niveau système. Nous focaliserons notre attention sur le niveau routage.

Problème étudié :

Notre objectif est de proposer une solution qui permettent de tolérer les fautes apparaissant en cours de fonctionnement (on line), que le défaut corresponde à une faute transitoire de type « soft error » ou permanente (due au vieillissement par exemple). Mais avant cela on peut se poser la question de savoir quelle est la conséquence d'une faute dynamique apparaissant dans le NoC pendant le transfert d'un paquet ? Dans le meilleur des cas la valeur d'un flit de donnée va être corrompue mais dans le pire des cas le paquet ne pourra plus atteindre sa destination, sera perdu (supprimé) ou provoquera un blocage du réseau en ne libérant pas les ressources réservées. Et si un routage adaptatif est prévu, que faire des morceaux de paquets passés et re-routés ? Ainsi se pose la question du contrôle de flux et du routage de paquets en présence de fautes dynamiques.

Nous allons décrire ci-dessous les techniques apportant une première réponse à ces problèmes et identifierons leurs inconvénients. Ces techniques serviront de base aux solutions que nous proposons dans le chapitre suivant.

2.2.2.1 Contrôle de flux et fragmentation de paquet

UTP

Au niveau du contrôle de flux, [Dally 1994] avec le « Reliable Router » propose un nouveau contrôle de flux nommé Protocole à jeton unique (en anglais Unique Token Protocol, UTP) pour garantir les transferts de paquets en présence de fautes de lien ou de routeur.

UTP se base sur la commutation de paquet de type wormhole. L'idée consiste à ajouter un flit à la fin du paquet, derrière le flit de queue. Ce flit, appelé flit jeton, peut avoir deux valeurs différentes correspondant à deux états du paquet: (1) l'état jeton unique, qui signifie qu'un seul exemplaire du paquet est en circulation dans le réseau; (2) l'état jeton réplique, qui signifie que le paquet courant est une copie d'une partie du paquet voire une copie complète du paquet. Si le réseau doit propager un paquet par plusieurs chemins lorsque le paquet est fragmenté, le flit jeton unique est modifié en jeton réplique. Si au cours des fragmentations un paquet ne contient pas de flit de jeton réplique, celui-ci est ajouté. L'objectif de cette technique est de s'assurer qu'en cas de présence de faute une copie du paquet est toujours disponible quelque part et que les fragments même s'ils ont des parcours différents sont assemblés au nœud destination. Enfin l'insertion de flit de queue (jeton) permet de désallouer les ressources réservées par un paquet.

UTP fait en sorte :

1 Qu'une copie de du paquet courant est toujours sauvegardée dans le routeur amont.

2 Qu'il n'y a pas de flit de donnée derrière un flit de jeton.

3 Que le flit d'en-tête est dupliqué dans le tampon de chaque nœud lors de son passage. Rappelons que le flit d'en-tête sert au calcul du routage du paquet.

4 Que le flit de jeton (unique ou réplique) libère les ressources réservées par le flit d'en-tête. Ce dernier est supprimé lors de cette libération.

Pour mieux comprendre cette technique UTP voyons le scénario de propagation d'un paquet dans un réseau sans faute (figure 2.8), puis avec fautes (figure 2.9)

Soit un paquet partant du nœud A vers le nœud D. Ce paquet est constitué d'un flit d'entête H (head), de deux flits de données D1 et D2 et d'un flit de queue T (tail). UTP ajoute en queue un flit de jeton tk.

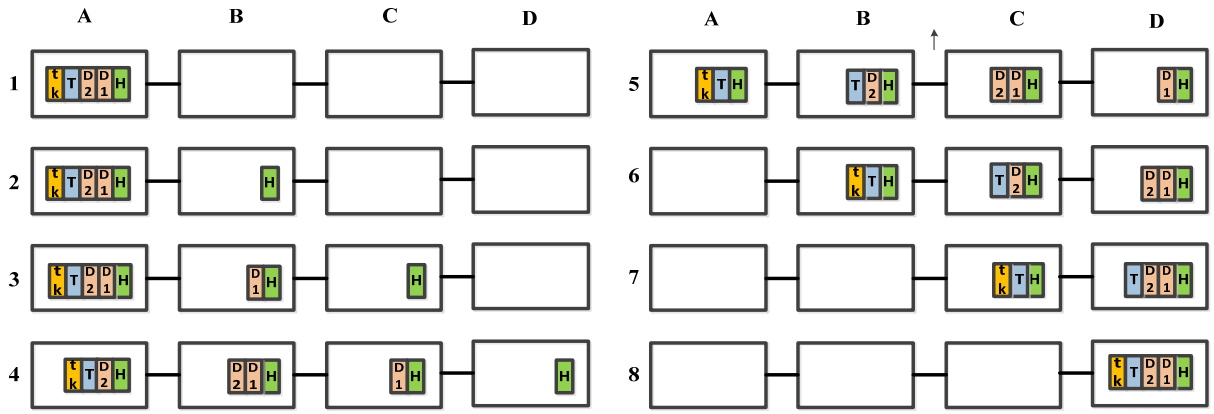


Figure 2.8 Un exemple d'un avancement d'un paquet par UTP

La progression est la suivante :

1. Le paquet est injecté dans le routeur A.
2. H arrive en B. A garde une copie de H
3. H arrive en C. D1 arrive en B. A garde une copie de D1.
4. H arrive en D. D1 arrive en C. D2 arrive en B. A garde une copie de D2. Comme le routeur B possède une copie de D1, A supprime le flit D1.
5. D1 arrive en D. D2 arrive en C. T arrive en B. A garde une copie de T. Comme le routeur B possède une copie de D2, A supprime le flit D2.
6. D2 arrive en D. T arrive en C. tk arrive en B. Le flit de jeton tk libère le canal réservé dans le routeur A et provoque la suppression du flit d'en-tête H.
7. T arrive en D. Le jeton tk arrive en C. Comme en 6 le flit de jeton tk libère le canal réservé dans le routeur B et provoque la suppression du flit d'en-tête H.
8. Tous les flits du paquet arrivent en D.

Voyons maintenant un scénario correspondant à un réseau avec fautes dynamiques équipé d'un mécanisme de détection de fautes et d'un routage adaptatif. Ce scénario est illustré figure 2.9.

Soit un paquet constitué de trois flits de données (D1, D2 et D3), qui est injecté dans le réseau par le nœud S(0,2) et de destination le nœud D(2,0).

(a) : le paquet progresse vers sa destination en D(2,0) en suivant le chemin (0,2) (0,1), (0,0), (1,0).

(b) : Au cours du transfert, les liens reliant d'une part (0,2) à (0,1) et (1,2) à (1,1) tombent en panne. Le paquet est alors fragmenté en deux parties. La première partie contient D1 et D2. La deuxième contient D2, D3. Le protocole UTP effectue les opérations suivantes :

- En (0,2) le flit de jeton unique est transformé en jeton réplique.
- Dans le routeur (0,1), un flit de jeton réplique est ajouté en queue de la première partie du paquet.

(c) : Un re-routage du paquet en (0,2) est effectué. H est envoyé vers (1,2) pour contourner les liens défectueux. Le fragment de paquet situé en (0,1) continue lui sa progression vers sa destination D.

(d) : La première partie du paquet arrive à destination en suivant le chemin (0,1), (0,0), (1,0), (2,0) et la deuxième arrive en suivant le chemin (1,2), (2,2), (2,1) et (2,0). A l'arrivée, le routeur voyant qu'il a des jetons « réplique » saura qu'il faut reconstituer le paquet à partir des deux fragments.

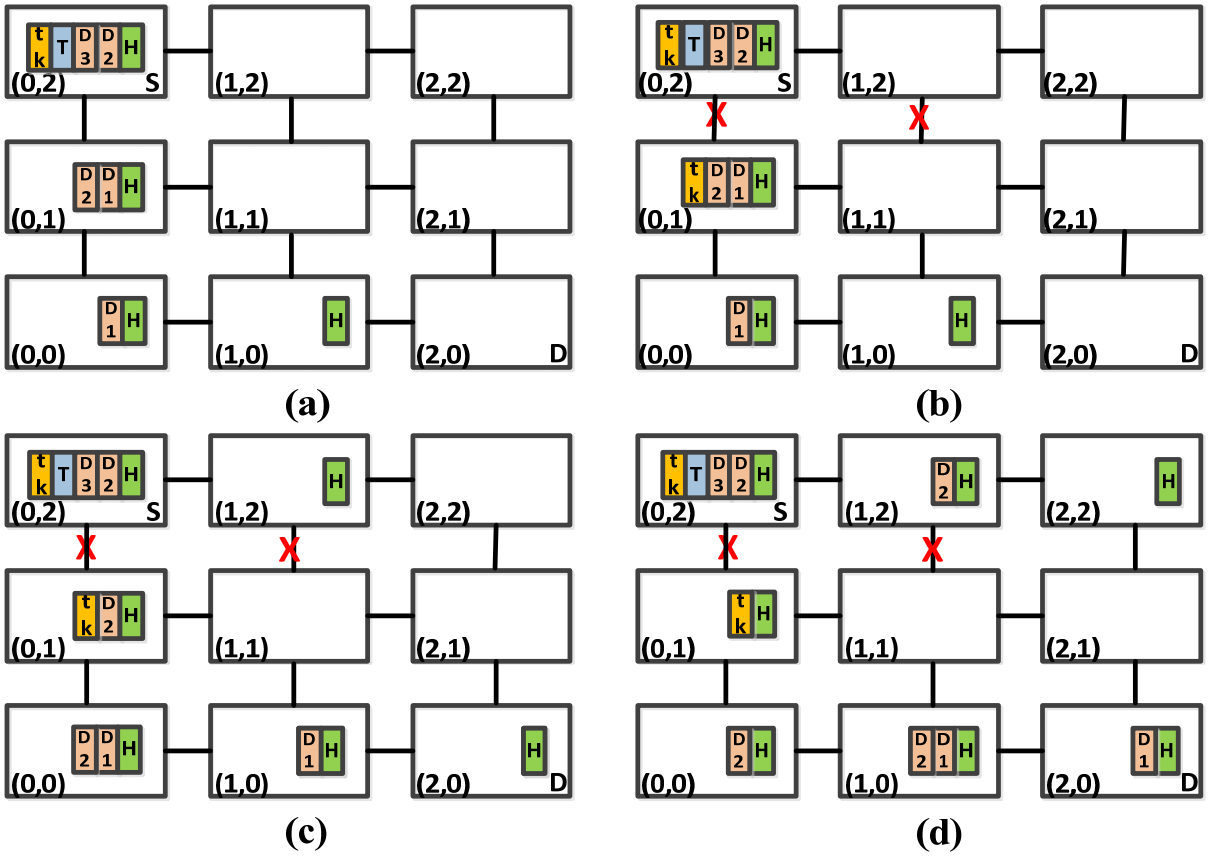


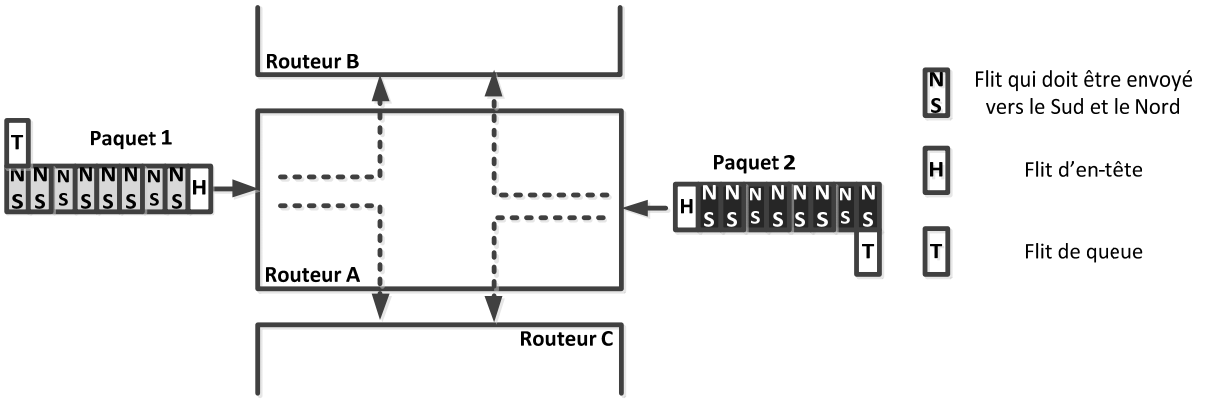
Figure 2.9 Tolérance à une faute par UTP

Pour résumer, en cas de lien défectueux, le nœud amont du lien peut re-router le paquet avec une copie du flit d'en-tête et le nœud en aval peut ajouter un jeton réplique pour libérer les ressources allouées.

Cette technique de fragmentation est intéressante cependant elle est limitée. En effet, elle ne permet pas de tolérer deux fautes de routeurs voisins car dans ce cas il y a une perte d'information, une seule copie est faite alors qu'il en faudrait deux. Mais l'inconvénient majeur est que pour réaliser les différentes copies de paquet, un nombre important de tampons est nécessaire et cela a un impact sur la surface additionnelle et le débit du réseau.

Fragmentation dynamique contre l'interblocage

La technique de fragmentation de paquet a aussi été utilisée pour résoudre des problèmes d'interblocage. En effet, dans [Kang 2009], les auteurs adressent le problème d'interblocage dans un réseau à commutation de paquet de type wormhole pour des applications « multicast ». Dans ces applications un paquet est envoyé d'une source vers plusieurs destinations, en même temps. Même si dans ce travail de thèse nous nous concentrons sur des applications « unicast » (une seule source vers une seule destination) il est intéressant d'étudier cette technique.



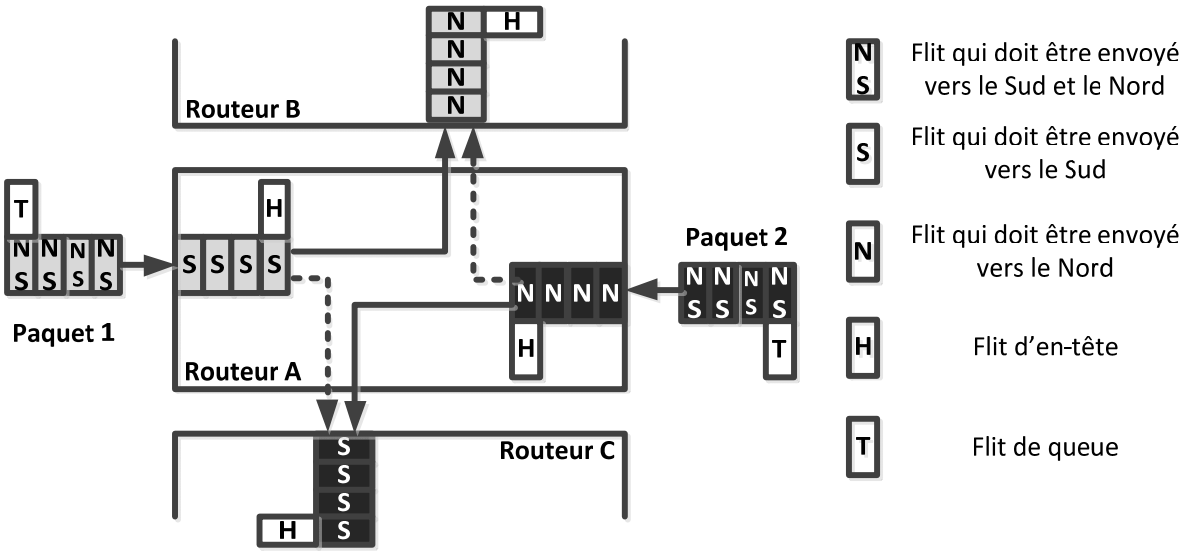


Figure 2.10 L'interblocage créé par deux paquets multicast

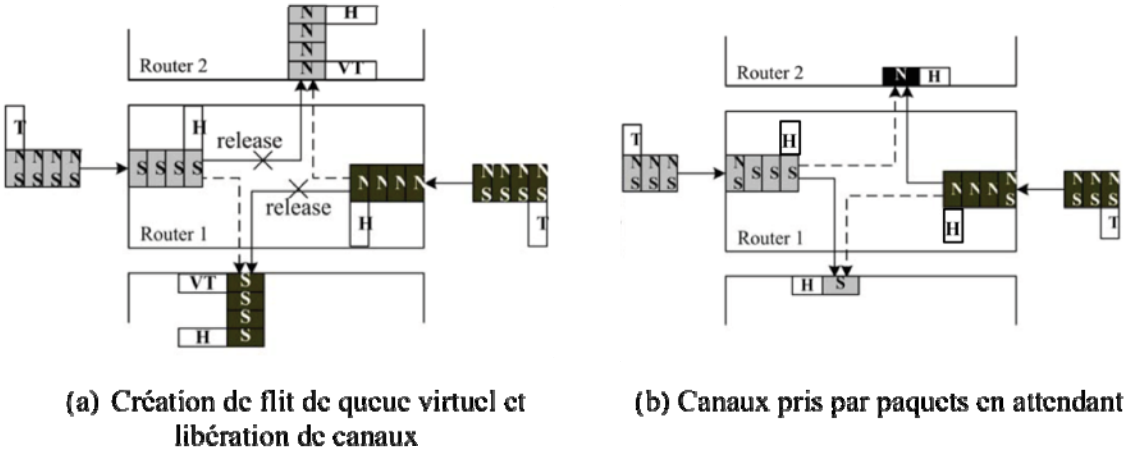


Figure 2.11 Fragmentation de paquet multicast

Pour empêcher cette situation d'arriver les auteurs de cette technique proposent une solution basée sur la fragmentation de paquet (figure 2.11) Lorsque la situation d'interblocage est identifiée, le paquet est scindé en deux paquets en insérant un flit virtuel de queue pour libérer le canal. Ainsi le paquet 1 en gris clair stocké dans le routeur B est augmenté d'un flit VT (Virtual Tail flit). Dans le routeur A, pour ne pas perdre l'information sur la destination, les flits restants du paquet 1 sont concaténés avec une copie du flit d'entête H. Les mêmes opérations étant réalisées pour le paquet 2 (en noir), les canaux sont libérés et il n'y a plus de boucle de dépendance. A chaque fois qu'un paquet traverse un routeur, son flit d'entête est sauvegardé. Avec cette solution, même si l'interblocage apparaît sur plusieurs nœuds du chemin, le paquet sera fragmenté plusieurs fois mais les flits arriveront à destination et cela

dans le bon ordre (in order delivery). L'interface réseau du nœud destination se charge de reconstituer le paquet.

Cette solution est donc implémentée dans un contexte de déblocage de paquets multicast, pour des réseaux à commutation de paquet de type Wormhole. Elle ne fait pas intervenir de changements dans le contrôle de flux basé sur l'échange de crédits. Cette technique ne permet pas de tolérer la présence de fautes mais possède un potentiel pour être adaptée à cette problématique.

Ainsi, en se basant sur les deux solutions vues ci-dessus utilisant la fragmentation dynamique de paquet, nous allons proposer une approche permettant de tolérer les fautes dynamiques (permanentes et/ou temporaires).

2.2.2.2 Routage

Au niveau du routage, pour les raisons vues au début de ce chapitre, nous orientons notre étude vers les solutions de tolérance aux faute basées sur un routage adaptatif, sans tables de routage et utilisant des canaux virtuels.

Un problème à résoudre lorsqu'un nœud devient défaillant (lien ou routeur) est celui de la perte de connectivité et de régularité dans le réseau. Pour les routages déterministes, on peut par exemple reconfigurer les routeurs voisins de celui fautif pour créer des contournements de zone [Zhang 2008][Zhang 2010]). Le problème pour ce type de techniques est qu'il peut être nécessaire de sacrifier des routeurs sains cela afin de garder une zone régulière. Mais surtout la reconfiguration du réseau nécessite de stopper puis de relancer l'application ce qui n'est pas l'objectif visé ici.

Une réponse au problème de perte de connectivité pour des routages adaptatifs consiste à introduire de la redondance matérielle. C'est le cas de MICOF [Ebrahimi 2012] et BFT-NoC [Tsai 2011]

La première vise la tolérance aux fautes dans les routeurs en utilisant un algorithme adaptif nommé « MiCoF ». L'architecture du routeur est modifiée afin d'inclure des fils supplémentaires reliant chaque routeur à ses voisins. Ces fils ajoutés sont utilisés comme un lien traversant (bypass) dans le cas où le routeur est défaillant. Le routeur devient transparent et la connectivité du réseau reste intacte.

BFT-NoC vise quant à elle les fautes au niveau lien. La technique se base sur l'ajout de liens bidirectionnels reconfigurables dynamiquement afin de conserver la connectivité entre les routeurs lors des défaillances de liens.

L'algorithme de routage adaptatif en tant que tel permet de s'affranchir de ces problèmes de connectivité et de régularité. Si des nœuds ou des liens sont défaillants, les paquets doivent pouvoir continuer à circuler en contournant les parties fautives. La limite de la tolérance aux fautes est alors le cas où les défaillances créent un partitionnement complet du réseau.

Pour cas d'étude et algorithme de référence nous nous intéressons à l'algorithme tolérant aux fautes défini par Cunningham et Avresky [Cunningham 1995][Avresky 1999], implémenté et optimisé au laboratoire TIMA par Fabien Chaix [Chaix 2010]. Cet algorithme, que nous appellerons Variant B, répond de manière adéquate aux objectifs visés par cette thèse.

Variant B est un algorithme de routage tolérant aux fautes qui garantit l'absence d'interblocages (Deadlocks) et l'absence de paquets bouclants sans fin (LiveLock).

Variant B nous sert de référence pour deux raisons :

Premièrement, Variant B n'utilise pas de tables de routage, il peut donc être adapté à un NoC de grande taille sans pour autant nécessiter une grande surface de silicium. Il combine deux algorithmes de routage adaptatifs North-Last et South-Last qui intègrent chacun des restrictions de parcours évitant ainsi les risques d'interblocage [Glass 1992].

Deuxièmement, Variant B offre un routage de type adaptabilité complète : le paquet n'arrive pas forcément à sa destination par le chemin le plus court, autrement dit le paquet peut prendre une direction non optimale. Deux réseaux virtuels (VN, voir chapitre 1.1) séparés sont utilisés, l'un pour le routage North-Last et l'autre pour le routage South-Last. Quatre canaux virtuels (VCs) sont attribués à chaque port et deux d'entre eux sont attribués à chaque VN. La sélection d'un VN dépend de la position relative de la destination par rapport à la source. Si la destination se situe au le nord (sud), le VN South-Last (North-Last) est choisi.

Variant B intègre la notion de source virtuelle (abrégé VS, Virtual Source) attachée à chaque nœud.

Source virtuelle

Pour illustrer la notion de source virtuelle considérons le schéma de la figure 2.12 ou un paquet ne peut rejoindre sa destination à cause de liens inter routeurs défectueux.

(a) Dans ce NoC 3x3, un paquet doit être envoyé du nœud S(0,2) au nœud D(2,0). Comme le nœud D se situe au sud du nœud S, le paquet est injecté dans le réseau virtuel NorthLast. Le paquet passe en (0,1), (0,0) et arrive au nœud (1,0). D'après l'algorithme de routage NorthLast, il n'y a pas de sorties possibles (le paquet ne peut pas tourner à 180° vers l'ouest, le lien lié à la sortie Est est défaillant et le nord est interdit pour le routage NorthLast).

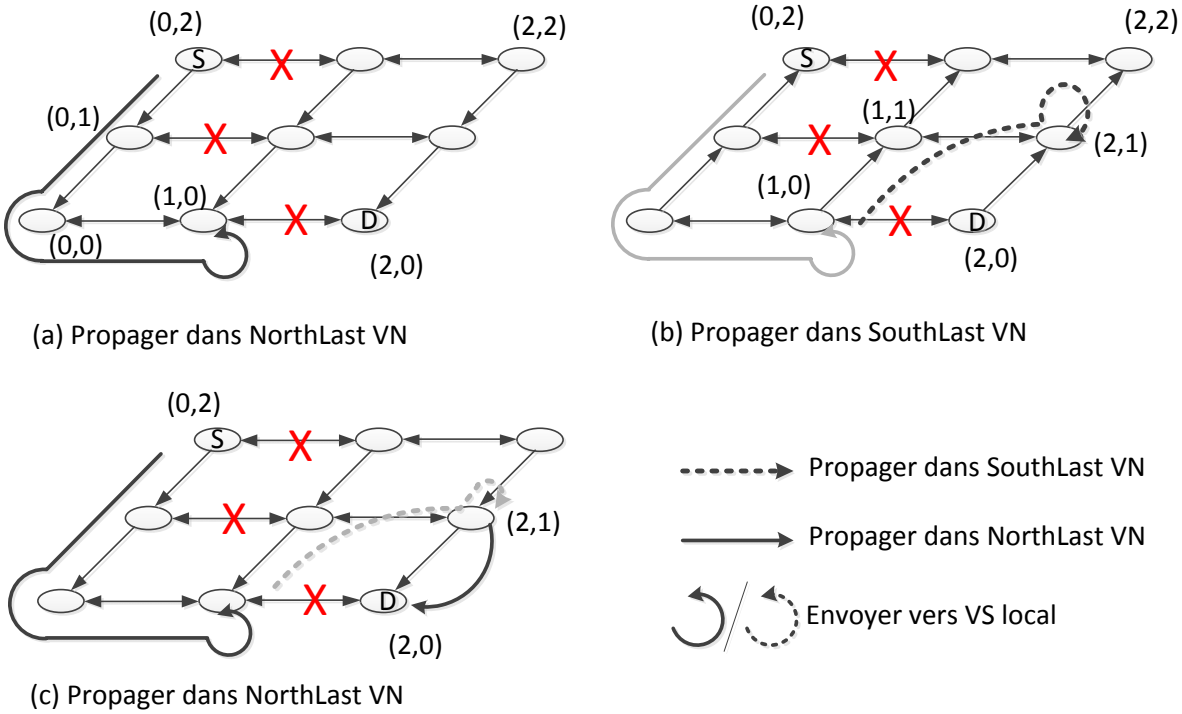


Figure 2.12 Un exemple d'application de source virtuelle

L'idée de la source virtuelle (en anglais Virtual Source, abrégé VS) consiste à consommer le paquet dans un VN où le paquet n'a pas de sortie possible, dans un nœud intermédiaire, puis le réinjecter dans un autre VN où le paquet pourra continuer à avancer vers sa destination initiale.

(b) Ainsi le paquet est consommé par la VS dans le nœud (1,0) et est injecté dans le VN SouthLast. Le paquet peut avancer alors vers le nœud (1,1) puis (2,1). Pour atteindre la destination en D(2,0), il faut que le paquet soit propagé dans le VN NorthLast. Le paquet est à nouveau consommé par une nouvelle VS en (2,1).

(c) Le paquet est injecté dans le VN NorthLast, et peut arriver finalement en D(2,0).

Les sources virtuelles permettent donc d'augmenter la diversité des routages et par conséquent d'améliorer le taux de livraison de paquets, sans introduction d'inter blocages.

Variant B utilise donc cette technique et ajoute dans chaque routeur un mécanisme empêchant la création de source virtuelle plus de deux fois pour chaque nœud. Cela garantie l'absence de « live lock », situation où un paquet continue à traverser le réseau sans jamais arriver à destination).

Bien que Variant B ait le potentiel de tolérer les fautes permanentes avec son adaptabilité, l'approche ne permet pas de tolérer des fautes temporaires et dynamiques. Supposons qu'un message soit envoyé d'un nœud S vers un nœud D en utilisant l'algorithme de routage Variant B et qu'une faute affecte un ou des flits du paquet pendant son transit. Nous pouvons avoir les scénarios suivants :

1. Le flit d'en-tête n'est pas affecté et arrive à destination. Le reste des flits n'arrivent pas en D ou un ou plusieurs flits arrivent mais en étant corrompus. Si le mécanisme est prévu, D enverra un message de corruption à S par un retour de non accusé de réception (en anglais Negative Acknowledgment, abrégé NACK) après un délai prédéfini (timeout).

2. Le flit d'en-tête est corrompu ou le chemin réservé par le flit d'en-tête est devenu défaillant à cause d'une défaillance intermittente sur le lien / nœud. Ce cas conduira à la perte du message et à l'échec de la transmission. Le problème ici est que ni la source ni la destination n'ont conscience de la non-transmission du paquet.

Nous proposons dans le chapitre suivant de pallier le problème des fautes dynamiques en enrichissant Variant B des fonctionnalités suivantes :

- Conscience des défaillances des nœuds voisins : chaque routeur connait l'état de santé de ses propres liens et des liens de ses routeurs voisins [Feng 2013]

- Retransmission de paquet combinée avec un nouveau schéma de recouvrement de message.

Variant B sera enrichi aussi d'une régulation du trafic / gestion de la congestion, ce dont il n'est pas pourvu.

2.3 Plateforme virtuelle d'expérimentation

Pour mener notre étude nous avons besoin d'une plateforme de modélisation et de simulation. Nous appellerons cet ensemble plateforme virtuelle pour modéliser et simuler l'architecture matérielle relative au réseau sur puce : microarchitecture de routeur, interface réseau, etc.

Dans le domaine de la microélectronique, il existe plusieurs approches pour modéliser un système matériel en fonction du niveau d'abstraction et de la précision requise. Le niveau transfert de registre RTL correspond au modèle le plus précis mais le temps de simulation est très long surtout quand le système est complexe. Ici nous visons un système avec plusieurs dizaines/centaines de nœuds. Ce niveau de modélisation/simulation est donc mis de côté. Cette dernière décennie, plusieurs travaux ont proposé des plateformes de modélisation du matériel basés sur le langage C++, et son extension tels que SystemC [Web SystemC]. En haussant le niveau d'abstraction et en baissant le niveau de précision, SystemC permet de faciliter la modélisation de ces systèmes et surtout permet de réduire grandement le temps de simulation.

Dans [Web Networkonchip], un résumé des plateformes virtuelles existantes est présenté. Voici ci-dessous les trois plateformes les plus en lien avec nos objectifs.

- Booksim [Jiang 2013] est un simulateur de réseau précis au niveau cycle, les modèles étant écrits en C++. Il permet de simuler une large gamme de topologies telles que la maille, le tore, etc. Il fournit divers algorithmes de routage et de nombreuses options pour personnaliser la microarchitecture des routeurs à simuler.

- Garnet [Agarwal 2009] est un simulateur de NoC qui s'intègre dans l'environnement GEM5 [Binkert 2011] plateforme modulable utilisée dans la recherche d'architectures de systèmes de calcul. Garnet fournit diverses options pour personnaliser la microarchitecture de routeurs pipeline. Cependant, Garnet offre peu de choix en ce qui concerne les algorithmes de routage. Il n'offre pas par exemple la possibilité d'utiliser un algorithme adaptatif.

- Iris [Wang 2014] est un simulator de réseau sur puce écrit en C++ basé sur la simulation à événement discret. Iris permet d'avoir une vitesse de simulation très élevée, même pour un système avec des centaines de nœuds. Iris s'intègre dans la plateforme SST

(the Structural Simulation Toolkit) [Rodrigues 2011] utilisée dans la recherche d'architecture de supercalculateurs. De plus SST fournit un environnement de simulation parallèle et distribuée sur plusieurs PCs.

Nous avons fait le choix du simulateur Iris, en raison de son avantage en terme de vitesse de simulation, mais aussi en prévision d'une future utilisation de la plateforme SST pour modéliser et simuler un système complet en ne se limitant pas au NoC seulement.

Iris, comme les autres d'ailleurs, n'intègre pas tout ce dont on a besoin. Il faut l'enrichir de nouvelles fonctionnalités : mécanisme d'injection de fautes, génération automatique de configurations de simulation, etc. Ces fonctionnalités seront décrites dans le chapitre suivant.

2.4 Conclusions

Notre objectif étant de tolérer tous types de fautes, statiques et dynamiques, pouvant survenir dans un NoC de grande taille, nous avons spécifié les caractéristiques que devra avoir ce NoC : réseau à commutation de paquet de type Wormhole, routage adaptatif basé sur des FSMs et utilisant de canaux et de réseaux virtuels.

En ce qui concerne le traitement de la congestion, nous avons vu les principaux mécanismes à mettre en place, quelques approches illustrant ces mécanismes ainsi que leurs limitations.

Pour ce qui est de la tolérance aux fautes, après une brève description des notions de base, nous avons ciblé le problème du recouvrement de paquet en l'illustrant avec des techniques de fragmentations de paquet. Nous avons décrit l'approche dite Variant B, solution qui avec son algorithme de routage fournit une solution partielle au problème qui nous préoccupe. En effet, Variant B permet de tolérer les fautes statiques mais pas les fautes dynamiques. Nous nous proposons de l'enrichir dans le chapitre suivant et Variant B nous servira de référence.

Enfin nous avons brièvement introduit le choix du simulateur Iris, simulateur qui va nous servir pour faire toutes nos mesures et discuter les solutions proposées.

Chapitre 3 Algorithmes de routage tolérant aux fautes avec prise en compte de la congestion

Dans ce chapitre nous allons décrire les techniques que nous avons mises en œuvre dans cette thèse pour répondre aux problèmes abordés dans les chapitres précédents. Une nouvelle approche du routage tolérant aux fautes est donc présentée. Cette approche, nommée CAFTA, est constituée de la combinaison de trois techniques inspirées de l'état de l'art et adaptées à un réseau sur puce embarqué, en tenant compte des contraintes associées aux circuits intégrés:

1. un mécanisme de retransmission de paquet intégrant des schémas de recouvrement basés sur la fragmentation/assemblage de paquet
2. un algorithme de routage tolérant les fautes où les nœuds sont informés de l'état de santé de leurs voisins directs (1 hop)
3. l'intégration dans l'algorithme de routage de la gestion de la congestion en utilisant une nouvelle métrique mesurant celle-ci

Les deux premières techniques seront traitées dans la section tolérance aux fautes, la troisième dans une section dédiée à la congestion.

3.1 Tolérance aux fautes

3.1.1 Hypothèse

Nous nous concentrons dans ce travail uniquement à la tolérance aux fautes, ainsi nous faisons les hypothèses de départ suivantes sur lesquelles s'appuie notre travail:

- Les liens bidirectionnels sont constitués de deux liens unidirectionnels.
- La détection de fautes est considérée existante et réalisée par d'autres mécanismes. Des techniques d'autotest matériel ou logiciel peuvent être utilisées pour la détection de fautes sur les éléments de traitement (en anglais Processing Elements, abrégé PEs), les routeurs ou les liens. Par exemple, la méthode d'autotest par logiciel (en anglais Software-Based Self-Test, abrégé SBST) peut être utilisée pour détecter des fautes permanentes dans les PEs tandis que les fautes temporaires sur les liens, les routeurs et les PEs peuvent être détectées et corrigées grâce au double échantillonnage présent dans GRAAL [Nicolaidis 2007]. De même pour les routeurs, un mécanisme de détection de fautes basé sur l'envoi périodique de messages de type « alive » ou

« heart beat » [Aguilera 1997]) peut être utilisé. Si un routeur particulier échoue à envoyer ses battements de cœur, il est alors considéré comme défaillant.

- Les fautes permanentes statiques existantes n'entrainent pas le partitionnement du NoC en deux ou plusieurs sous réseaux disjoints. Autrement dit, il existe au moins un chemin qui connecte une paire source-destination donnée. Cette hypothèse est nécessaire afin de pouvoir évaluer l'algorithme de routage.

  Si le cas se présente, un mécanisme doit être présent pour sélectionner une partition de nœuds sains, sur laquelle sera déployée l'exécution de l'application. Des solutions permettant d'effectuer les migrations de tâches et désactivant la partition inaccessible peuvent être utilisées, au niveau application ou au niveau système d'exploitation. [DeOrio 2011].

- Les routeurs sont équipés d'un mécanisme de retransmission de paquet et les liens intègrent de la redondance de type ECC [Hamming 1950]. En cas d'erreur(s) non corrigeable(s) (selon le niveau de correction choisi), la retransmission des flits (FLow control digIT, abrégé FLIT) est lancée par le routeur amont.
- Une fois qu'un routeur est détecté comme défaillant, il s'arrête de fournir le service au réseau (fail-silent node [Avizienis 2004])
- Les routeurs voisins du routeur ayant un défaut permanent statique sont informés de son état et sont reconfigurés afin d'empêcher tout trafic vers ce routeur défectueux (par exemple, ils peuvent désactiver les ports de sortie menant à ce routeur). Dans le cas où le routeur défaillant est la destination de certains messages, la transmission échoue. Un mécanisme de renvoi du paquet par le nœud source doit être implémenté.
- Pour ce qui est d'une faute de lien, les routeurs à chaque extrémité sont informés de son état et sont reconfigurés afin d'interdire le trafic à travers ce lien (le routeur en amont désactive le port de sortie lié à ce lien, tandis que le routeur aval désactive le port d'entrée associé).
- Chaque PE, son interface réseau (NI) et les liens reliant le routeur au NI sont considérés comme sains. Si tel n'est pas le cas, le nœud correspondant est éliminé de la liste des nœuds source ou destination mais le routeur peut quant à lui fonctionner normalement.

Il convient de noter que pendant le laps de temps comprenant la détection de la faute, la propagation de cette information et la reconfiguration du réseau, il y a des paquets en

transit et les parcours sélectionnés par la logique de calcul du routage peuvent ne plus être valides. Cette situation peut créer des dépendances entre les anciens paquets (avant l'arrivée de la faute) et les nouveaux paquets (après son apparition). Ces dépendances peuvent conduire à des phénomènes d'interblocage paralysant le réseau. Les techniques que nous proposons ont aussi pour but de résoudre de telles situations.

3.1.2 Le recouvrement de paquet

Pendant le fonctionnement du NoC, lorsqu'une faute apparaît (de type permanent, transitoire ou intermittente), elle peut avoir différentes conséquences selon la nature de la faute et sa localisation : corruption d'un paquet, interblocage de paquets, perte du paquet, …

Pour cibler à la fois les fautes statiques et dynamiques, permanentes et temporaires nous proposons d'intégrer dans chaque routeur :

1. Un mécanisme de retransmission à k tentatives pour le port de sortie sélectionné par l'étape de calcul de routage (RC). Cela résout le problème des fautes dynamiques temporaires non corrigées par les techniques d'ECC (fautes sur plusieurs bits).
2. Un algorithme de routage adaptatif tolérant aux fautes lié au mécanisme de retransmission. Après k tentatives de réémission sans succès (ici pour des raisons pratiques nous prenons k=2), l'algorithme sélectionne un autre port de sortie pour acheminer le paquet. Cela résout en partie le problème des nœuds ou liens qui sont le siège d'une faute permanente statique, et permet au paquet de contourner le nœud/lien défectueux.
3. Un mécanisme de fragmentation/assemblage de paquets. Ce mécanisme vise à tolérer les fautes dynamiques (permanentes ou temporaires) qui modifient la validité du chemin emprunté pendant le transit du paquet. Par exemple un routeur ou un lien devient subitement un trou au milieu du chemin alors que tous les flits ne sont pas passés, créant ainsi des sous-paquets qui ne peuvent plus arriver tous à destination.

Pour ce qui est du mécanisme de fragmentation nous proposons d'étudier deux solutions ayant des stratégies d'assemblage différentes.

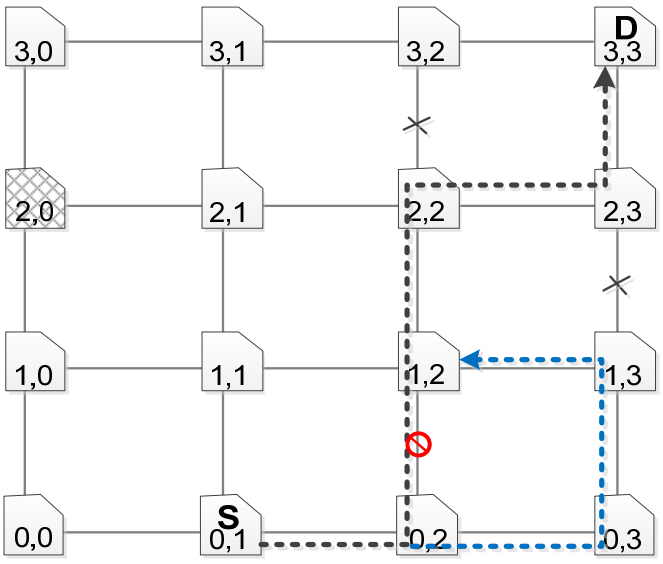
La première, que nous appelons BPR, fait en sorte que le fragment de paquet rejoigne le chemin initial après avoir contourné le nœud fautif. La seconde, SPR, crée pour le second

fragment un nouveau chemin vers la destination finale: le paquet dans ce cas est reconstitué dans le nœud destination à partir des fragments issus de chemins différents.

Ces solutions s'inspirent des techniques de fragmentation vues dans le chapitre précédent. Il s'agit ici de les compléter et de les adapter aux contraintes de l'intégré.

1ère stratégie: BPR: (Bypass Packet Recovery)

Pour identifier le problème à résoudre, considérons le scénario décrit par la figure 3.1 où un paquet doit être acheminé de S (0,1) à D (3,3). Le flit d'en-tête arrive à destination et la charge utile (payload) est guidée vers D par la voie {(0,1), (0,2), (1,2), (2,2), (2,3), (3,3)}. Une faute arrive soudainement sur le lien (0,2)↔(1,2) et divise le paquet. Le sous-paquet 1 va arriver au nœud destination par les ressources réservées tandis que le sous-paquet 2 est coincé dans le nœud (0,2) et ne pourra pas atteindre la destination D. Sachant que les ressources réservées par le flit d'en-tête ne seront libérées que par le passage du flit de queue, les ressources occupées (canaux (1,2), (2,3) et (3,3)) ne seront jamais libérés puisque le sous paquet 2 ne peut les emprunter. Au final la transmission échoue, et le réseau est paralysé.



Nœud (2,0) défectueux, liens fautifs : {(1,3)↔(2,3)}, {(2,2)↔(3,2)}
Faute dynamique arrive sur le lien {(0,2)↔(1,2)}

Figure 3.1 BPR : recouvrement lors d'une rupture de paquet

Pour résoudre ce problème, un pseudo flit de queue est créé dans le routeur (1,2) et un temporisateur est initialisé à kl cycles (kl : un nombre défini par l'utilisateur). Parallèlement, une seconde retransmission est tentée par le routeur (0,2). Après kl tentatives de retransmission, si le lien de sortie est encore défectueux et donc pas disponible, un pseudo flit

d'en-tête est créé dans le routeur (0,2) et envoyé par une autre voie {(0,2), (0,3), (1,3) (1,2)} avec l'objectif de contourner le lien défaillant et rétablir le parcours. Autrement dit, nous fixons comme destination de ce pseudo flit d'en-tête le nœud (1,2) afin que le sous-paquet 2 puisse rejoindre le sous-paquet 1 au nœud (1,2). Par sécurité, si le pseudo flit d'en-tête créé en (0,2) échoue à renouer le chemin rompu au bout de kl cycles, le pseudo flit de queue qui a été créé dans le routeur (1,2) est envoyé immédiatement à la destination (3,3) pour libérer les ressources réservées sur le chemin. Cela évite de paralyser le système en cas de fautes multiples.

Si le contournement réussit, les pseudos flits d'en-tête et de queue sont supprimés dans le routeur en aval du lien / routeur défaillant (ici 1,2). Comme le sous paquet 1 a déjà réservé le chemin vers D, la propagation du sous paquet 2 continue vers le nœud de destination. Ce nœud destination n'est à aucun moment averti de la rupture instantanée du paquet (pas de pseudo flit reçu). Le mécanisme est transparent pour lui. Cela a l'avantage de ne pas nécessiter de modification de l'interface réseau associée à chaque routeur (NI).

Limites de BPR :

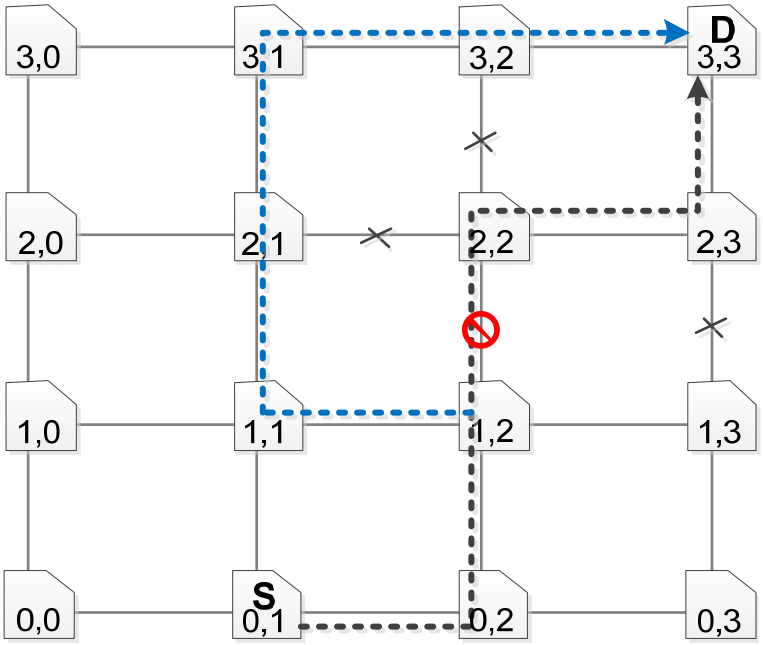
La technique présentée ci-dessus est intéressante et résout un certain nombre de problèmes mais elle n'est pas toujours efficace surtout quand les fautes sont nombreuses (ce que l'on vise dans cette thèse). Lorsque le routeur en aval du lien défaillant est inaccessible par le pseudo flit d'en-tête, à cause d'autres fautes voisines, BPR échoue. L'exemple de la figure 3.2 illustre le cas ou le routeur (2,3) est inaccessible par le pseudo flit d'en-tête. Ici tous les liens permettant d'accéder au routeur (2,3) sont défaillants.

2ème stratégie: SPR

Pour tolérer ce cas de figure, qui peut très bien arriver dans un environnement très perturbé, nous proposons une nouvelle stratégie que nous appelons SPR (Split Packet Recovery). La stratégie ici consiste à attribuer à chaque fragment de paquet un chemin qui lui est propre, avec comme contrainte que l'interface réseau (NI) du nœud destination a comme tâche de reconstruire le paquet correctement.

Dans la figure 3.2 le paquet doit être acheminé de S (0,1) à D (3,3) via le parcours {(0,1), (0,2), (1,2), (2,2), (2,3), (3,3)}. Une faute dynamique sur le lien (1,2) ↔ (2,2) apparaît et divise le paquet. Comme dans BPR, un pseudo flit de queue est créé dans le routeur (2,2) et un temporisateur est initialisé à kl cycles (kl : nombre défini par l'utilisateur). Dans le même

temps, une seconde retransmission est tentée par le routeur (1,2). Si le lien de sortie est encore défectueux et n'est pas disponible après kl tentatives de retransmission, un pseudo flit d'en-tête est créé dans ce routeur (1,2) et envoyé en suivant un chemin alternatif sélectionné par l'algorithme de routage adaptif tolérant les défauts: {(1,2), (1,1), (2,1), (3,1), (3,2), (3,3)} Après kl cycles, le pseudo flit de queue créé dans le routeur (2,2) est envoyé en (3,3) pour libérer les ressources occupées.



Nœud (2,0) défectueux, liens fautifs : {(1,3)↔(2,3)}, {(2,1)↔(2,2)}, {(2,2)↔(3,2)}
Une défaillance d'exécution sur le lien (1,2)↔(2,2)

Figure 3.2 Recouvrement de paquet utilisant la technique SPR

L'interface réseau du nœud (3,3) va être chargée de fusionner les deux paquets. La complexité de la stratégie est ainsi reportée vers le NI du nœud destination. Dans le cas où ce dernier est défaillant, les mécanismes de plus haut niveau doivent alors prendre le relais (réémission coté source, migration de tâches sur des nœuds sains, …).

Cette stratégie est semblable à l'approche UTP présentée en Figure 2.8. En effet dans les deux cas les fragments suivent des chemins distincts et sont reconstitués à l'arrivée. Cependant, contrairement à UTP, SPR peut tolérer des fautes dynamiques ou permanentes sur des routeurs ou liens voisins et surtout, dans SPR contrairement à UTP, on ne duplique pas les flits du paquet. Dans un contexte de réseau intégré, c'est un argument important, la taille des éléments de mémorisation étant un critère critique.

SPR (associé à l'algorithme de routage décrit ci-après) permet d'éviter les phénomènes d'inter blocages. Mais l'avantage de SPR par rapport à [Kang 2009] est qu'elle a le mérite de permettre la gestion de fautes pour tous les types de communication et pas seulement l'envoi de paquets multicast.

3.1.2 Routage pour la Maille 2D

Les mécanismes de recouvrement de paquets présenté ci-dessus peuvent être efficaces pour les MPSoCs embarquant des applications multimédias dans lesquels de grands volumes de données sont transférés entre les différents nœuds. Cependant il faut qu'un algorithme de routage adéquat puisse leur servir de support

L'algorithme de routage que nous avons appelé CAFTA a été pensé pour un réseau sur puce de topologie maille 2D (2D-Mesh). Nous verrons plus tard son extension aux topologies de type Tore 2D. A travers des exemples, nous montrerons comment CAFTA peut tolérer les fautes permanentes / transitoires / intermittentes et en même temps contourner les zones congestionnées.

CAFTA est basé sur l'algorithme présenté dans [Chaix 2010], appelé Variant B.

Pour rappel, ce dernier est un algorithme de routage tolérant aux fautes qui garantit l'absence d'interblocage et de boucles sans fin. Il n'utilise pas de tables de routage. Il combine deux algorithmes de routage adaptatifs North-Last et South-Last qui utilisent des restrictions de direction pour éviter les interblocages [Dally 1987]. Deux réseaux virtuels (VN) séparés sont utilisés, l'un pour le routage North-Last et l'autre pour le routage South-Last. Quatre canaux virtuels (VCs) sont associés à chaque port (deux pour chaque VN). Le choix du VN à parcourir dépend de la position relative de la destination par rapport à la source. Si la destination se situe sur le nord (sud), le VN South-Last (North-Last) est choisi. La figure 3.3 décrit plus formellement cet algorithme de sélection.

Nous avons vu dans le chapitre précédent que l'algorithme Variant B ne permet pas de tolérer les fautes dynamiques et temporaires qui peuvent arriver pendant le transit d'un paquet. Ces fautes peuvent potentiellement conduire à des interblocages du réseau. Ce problème est résolu en intégrant les mécanismes de recouvrement de paquet décrits dans la section précédente.

Pour une paire de source et destination données, S : source (xs,ys), D : destination(xd, yd)
VNi : est le réseau virtuel dans lequel le paquet M est en train d'acheminer vers sa destination D
dx=xd-xs, dy=yd-ys;
(1) dy>0, SouthLast VN
(2) dy<0, NorthLast VN
(3) dy=0, South-Last VN ou North-Last VN aléatoire d'après la disponibilité des canaux

Figure 3.3 Partitionnement de réseaux virtuels pour la Maille 2D

L'algorithme CAFTA associe donc Variant B, le recouvrement de paquet et la gestion de la congestion. Il est énoncé dans la figure 3.4. Pour mieux comprendre cet algorithme prenons l'exemple suivant :

Soit M un paquet qui est envoyé d'un nœud source S au nœud destination D. M arrive dans le nœud courant x. L'objectif de algorithme de routage CAFTA est d'aiguiller M vers sa destination D avec ou sans présence de fautes ou de congestion.

- Au nœud x on va d'abord prendre en compte l'état des nœuds voisins et l'état des liens de sortie. Nous obtenons ainsi une liste de nœuds Q qui peuvent recevoir le paquet M.
- Si le nœud courant x correspond à la destination D, M est alors consommé (éjecté) localement. Sinon, M peut être aiguillé vers l'un des nœuds de la liste Q. Si pour cela M doit prendre une direction interdite vers le nœud w dans VNi, il faut alors créer une source virtuelle pour faire transiter M dans le réseau VNj. Il est à noter que les directions interdites dans VNi/VNj sont complémentaires. Ainsi M est consommé dans le NI local puis réinjecté dans le VNj via le nœud w.
- En cas de fautes dynamiques apparaissant pendant l'exécution, les mécanismes de retransmission et recouvrement de paquets peuvent alors être exploités comme vu précédemment.

Les deux algorithmes de routage adaptatifs North-Last et South-Last utilisent les restrictions de direction de façon à éviter les interblocages, les sources virtuelles et le recouvrement de paquets n'introduisent pas de dépendance de canaux. CAFTA se basant sur ces techniques garantit donc par construction l'absence d'interblocages. De même que l'algorithme VariantB, CAFTA utilise la technique de marquage des routeurs parcourus pour éviter le phénomène de boucle sans fin (Livelock). Ainsi un paquet ne peut pas être envoyé vers les nœuds qu'il a déjà traversés et donc le Livelock est évité.

Algorithme de Routage (x,D,M):

Pour une paire de source et destination donnée (S : source, D : destination)
VN_i : est le réseau virtuel dans lequel le paquet M transite vers sa destination D
x : nœud courant, *w* : nœud prochain dans le parcours
Initialisation:
w=∅
Q : la liste de nœuds disponibles pour accueillir le paquet M prochainement // Avec la conscience de faute des voisins
If x=D *then* éjecter vers le port local et signaler « succès » // M est arrivé à destination
else begin
  If ($Q \neq \emptyset$) *then begin*
    If (sizeof(*Q*) = 1) // seulement un choix disponible
      then mettre $w = Q_i \in Q$.
    else begin //plusieurs choix possibles
      Mettre $w = Q'_i \in Q$, //où Q'_i a un parcours minimal vers la destination. En cas d'égalité ou tout le $Q'_i \in Q$, a un parcours minimal ou non-minimal, la métrique de congestion peut être utilisée comme second critère de classement.
    end
  end

  If $w \neq \emptyset$ *then begin* // Acheminer le paquet M vers le nœud *w*
    If w n'est pas valide dans VN_i *then begin*
      Ejecter M vers *Network Interface* (NI) du nœud courant
      Réinjecter M dans le réseau virtuel VN_j (i != j, w est valide dans VN_j) et tenter de l'aiguiller vers *w*.
    end
    If accès à w bloqué par une faute *then begin //runtime* défaillance
      Essayer une retransmission et si elle échoue, essayer soit *BPR* soit *SPR*
      If w est encore bloqué *then return* échec // destination *inaccessible*!
    end
  end
  else return failure // destination inaccessible!

end

Figure 3.4 Pseudocode de l'algorithme de routage CAFTA

3.1.3 Routage pour le Tore 2D

Pour voir l'impact de la topologie sur l'efficacité de CAFTA, nous avons décidé d'étendre l'algorithme au NoC de type Tore 2D. Avec une plus grande connectivité, la topologie Tore offre plus de parcours possibles entre une paire (source, destination).

Pour une paire de source et destination données, S : source (xs,ys), D : destination(xd, yd), dans un réseau de radix k
VNi : est le réseau virtuel dans lequel le paquet M est en train d'acheminer vers sa destination D
dx=xd-xs dy=yd-ys;
(1) 0< dy< k/2 or dy< -(k/2) , SouthLast VN
(2) -(k/2)<dy<0 or dy> k/2 , NorthLast VN
(3) dy=0, South-Last VN ou North-Last VN aléatoire d'après la disponibilité des canaux

Figure 3.5 Partitionnement de réseaux virtuels pour le tore 2D

La figure 3.5 décrit comment pour un Tore 2D on choisit le réseau virtuel à parcourir selon la localisation de la destination. Les réseaux sur puce en Tore 2D peuvent être partitionnés en deux réseaux virtuels : (a) SouthLast ; (b) NorthLast. Comme CAFTA pour Maille 2D, les algorithmes de routage SouthLast et NorthLast sont appliqués dans les deux réseaux virtuels respectivement. Cependant, bien que l'algorithme de routage soit le même que pour les réseaux 2D Mesh, le fait qu'il y ait des liens supplémentaires va introduire des cas d'interblocage (dead-lock) et de paquets en transit sans fin (live-lock).

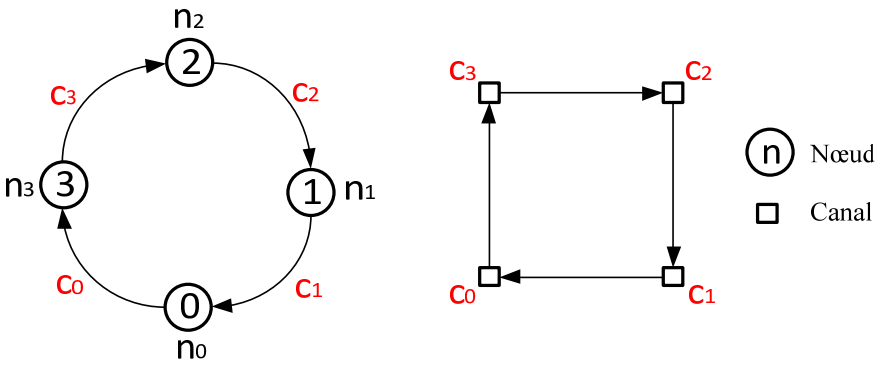
Routage et interblocage dans le tore 2D

Avant d'étendre CAFTA à ce type de réseau, il faut d'abord pouvoir garantir que l'algorithme est sans dead-lock et dans le cas contraire proposer une solution pour y remédier.

Pour déterminer si un algorithme de routage introduit un deadlock ou pas, nous nous basons sur le théorème de Dally et Seitz [Dally 1987] : une fonction de routage est garantie sans inter-blocages si et seulement si son graphe de dépendance de canal est acyclique.

Prenons le cas général d'un réseau avec des liens unidirectionnels comme indiqué sur la figure 3.6, avec des nœuds N={n0, n1, n2, n3} et des liens (canaux) C={c0, c1, c2, c3}. Le

graphe d'interconnexion I est à gauche tandis que le graphe de dépendance correspondant D est représenté à droite. On voit sur D que le parcours des quatre canaux unidirectionnels peut être fait de manière infinie. Il y a une boucle de dépendance de canaux, le routage peut donc amener à une situation de dead-lock.

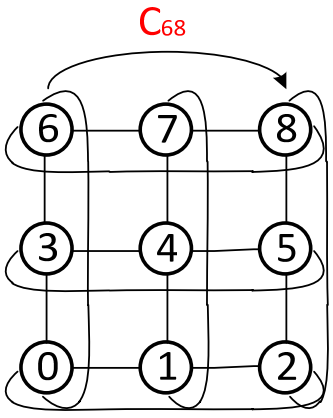


I: Graphe d'interconnexion

D: Graphe de dépendance

Figure 3.6 Un exemple de 3x3 Tore 2D

Etudions maintenant le graphe de dépendance de canaux dans le cas d'un tore 2D. La figure 3.7 présente un tore 2D de taille 3x3. Cij représente un canal avec un nœud i comme entrée, et un nœud j comme sortie.



C68 signifie le canal unidirectionel de nœud 6 vers nœud 8

Figure 3.7 Un exemple de 3x3 Tore 2D

La différence entre un NoC de topologie 2D Mesh et un NoC en Tore 2D réside dans l'ajout de liens enveloppants au niveau des nœuds en bordure du réseau. L'algorithme Variant B que nous utilisons est prouvé sans dead-lock pour un 2D Mesh, la question qui se pose alors est : les liens enveloppants introduisent-ils des interblocages ?

Pour répondre à cette question nous nous intéresserons d'abord aux liens enveloppants verticaux, aux liens enveloppants horizontaux et enfin à tous les liens enveloppants.

1. Liens enveloppants verticaux

Dans Variant B, nous avons deux réseaux virtuels distincts, NorthLast et SouthLast. La figure 3.9 montre le graphe de connexion pour les noeuds 0, 3 et 6 pour le réseau North Last. On voit que dans ce réseau, un paquet peut emprunter les canaux : C06 -> C63 -> C30 -> C06 ou C60 -> C63 -> C30 -> C60. Le graphe de dépendance correspondant introduit des cycles, le routage va donc créer des situations d'interblocage de paquet.

Pour remédier à cela nous proposons d'introduire une restriction de parcours en supprimant sur le graphe de dépendance le lien entre C30 et C06. Cela rend le graphe acyclique, le routage est alors garanti sans dead-locks (figure 3.8)

Concrètement cela revient à interdire au nœud 0 d'envoyer un paquet vers le nœud 6. La seule façon de réaliser ce transfert consiste à créer une source virtuelle dans le nœud 0 et transiter vers le nœud 6 en utilisant le réseau virtuel dual à savoir ici le réseau South Last. Par contre un paquet peut être envoyé du nœud 0 au nœud 6 sans changer de réseau (et donc sans création de VS).

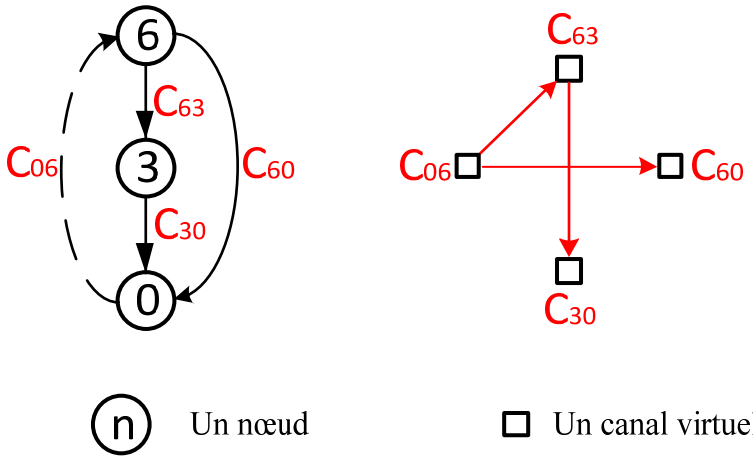


Figure 3.8 Restriction appliquée au réseau virtuel NorthLast

De manière complètement symétrique, le même raisonnement s'applique pour un parcours de paquets sur le réseau virtuel South Last, comme on peut le voir sur la figure 3.9. En supprimant sur le graphe de dépendance de canaux le lien C36-C60, le graphe est rendu acyclique et le routage se trouve donc garanti sans risques d'inter-blocages.

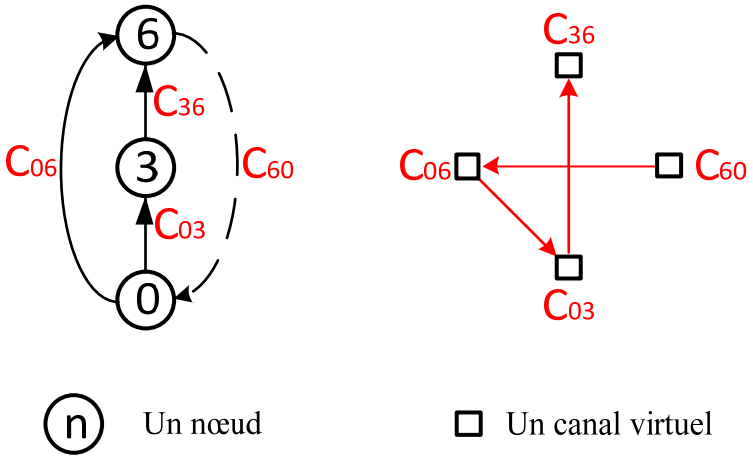


Figure 3.9 Restriction appliquée au réseau virtuel SouthLast

Pour les liens verticaux on peut élargir et extrapoler cette restriction au Tore 2D NxN :

Restriction 1 :

- Pour le réseau virtuel North Last, pour chaque nœud situé à la frontière **sud** :
  - Si le nœud est une source alors le nouveau paquet reste dans le même réseau virtuel et est envoyé par le port sud en utilisant le lien enveloppant.
  - Si le nœud n'est qu'un nœud de transit, une source virtuelle est créée en ce nœud et le paquet est envoyé par le port sud mais sur le réseau virtuel dual.
- Pour le réseau virtuel South Last, pour chaque nœud situé à la frontière **nord** :
  - Si le nœud est une source alors le nouveau paquet reste dans le même réseau virtuel et est envoyé par le port nord en utilisant le lien enveloppant.
  - Si le nœud n'est qu'un nœud de transit, une source virtuelle est créée en ce nœud et le paquet est envoyé par le port nord mais sur le réseau virtuel dual.

Illustrons cette restriction par les deux cas de figure qui peuvent apparaître pour les nouveaux paquets injectés dans le réseau South Last:

1) Dans la figure 3.10, le paquet est injecté dans le nœud (0,1) qui est un nœud à la frontière sud, il peut donc emprunter le lien enveloppant directement ;

2) Dans la figure 3.11, le paquet est injecté dans le nœud (1,1), quand il arrive en (0,1), il ne peut emprunter directement le lien enveloppant. Il ne pourra le faire qu'en changeant de réseau virtuel via le nœud (0,1) qui servira de source virtuelle (VS).

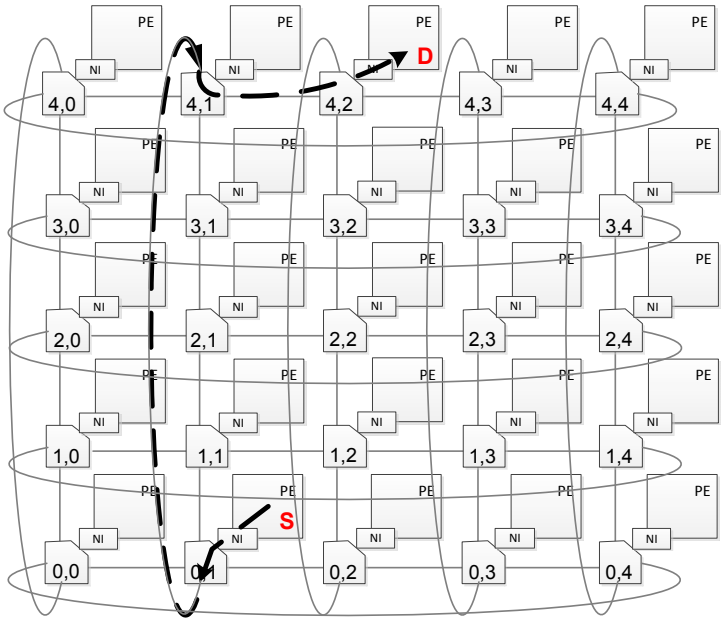


Figure 3.10 Cas 1 un paquet injecté par une source à la frontière sud

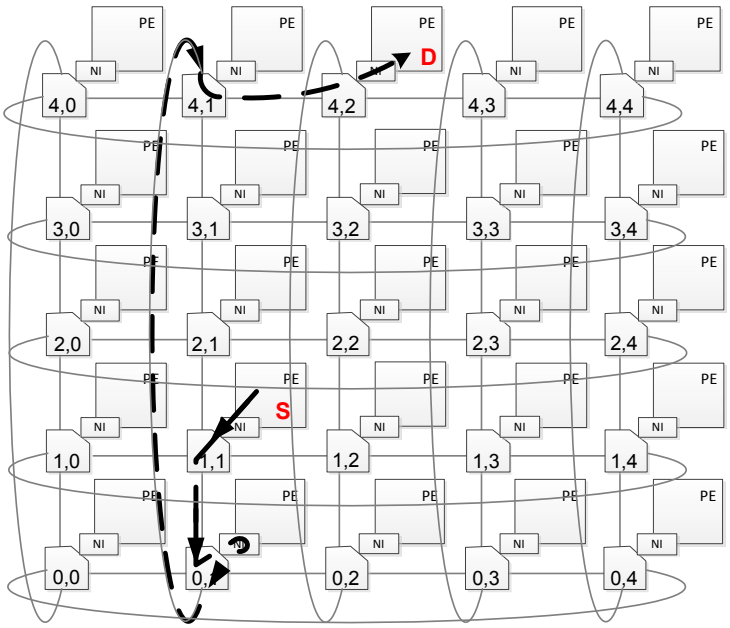


Figure 3.11 Cas 2 un paquet est injecté par une source virtuelle

2. Liens enveloppants horizontaux

Si on s'intéresse cette fois aux liens enveloppants horizontaux, on retrouve le même problème de dépendance de canaux que pour les liens verticaux. On appliquera la même technique, comme on peut le voir sur la figure 3.12. Ici le fait que l'on soit sur un réseau virtuel ou l'autre n'a pas d'importance. Les algorithmes de routage NorthLast et SouthLast ayant les mêmes restrictions pour les liens horizontaux, nous ne distinguerons pas les deux cas.

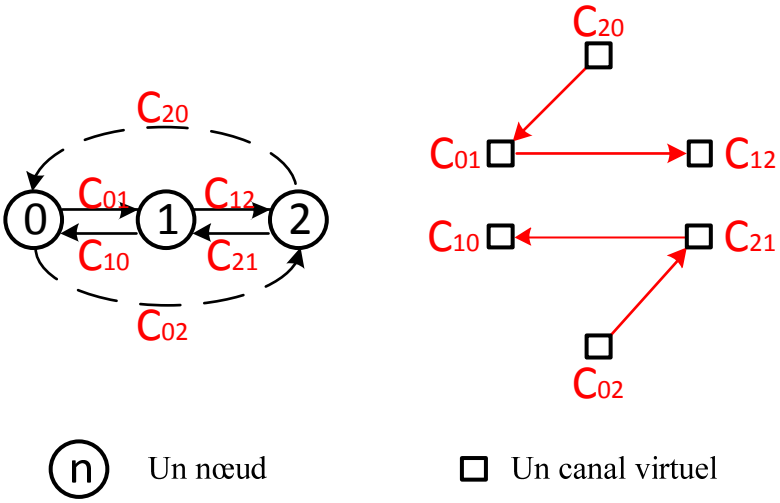


Figure 3.12 Dépendance de canal acyclique entre les liens enveloppants horizontaux

Sans restrictions, plusieurs cycles apparaissent dans de le graphe de dépendance de canaux : C20->C01->C12->C20, C02->C21->C10->C02, …. Pour rompre ces cycles nous proposons de supprimer les dépendances entre les liens C12-C20 et C10-C02.

En suivant le même raisonnement que précédemment et en extrapolant à un tore 2D NxN, nous obtenons la restriction suivante :

Restriction 2 :

Pour chaque nœud situé à la frontière Est ou Ouest :

- Si le nœud est une source alors le nouveau paquet peut utiliser le lien enveloppant normalement sans changer de réseau virtuel.
- Si le nœud n'est qu'un nœud de transit, une source virtuelle est créée en ce nœud et le paquet est envoyé par le port correspondant (est ou ouest) sur le lien enveloppant mais sur le réseau virtuel dual.

3. Liens verticaux et horizontaux

Le dernier cas à vérifier est celui concernant les liens enveloppants reliant les 4 routeurs aux extrémités du NoC. La figure 3.13 nous permet de voir qu'en absence de restriction le graphe de dépendance peut être cyclique. Or, si on applique les restrictions 1 et 2 vues précédemment, le graphe ne peut plus contenir de cycles et il n'y a donc pas de risques d'inter-blocages. Sur cette figure les dépendances entre les liens C06-C68 et C82-C20 sont supprimées.

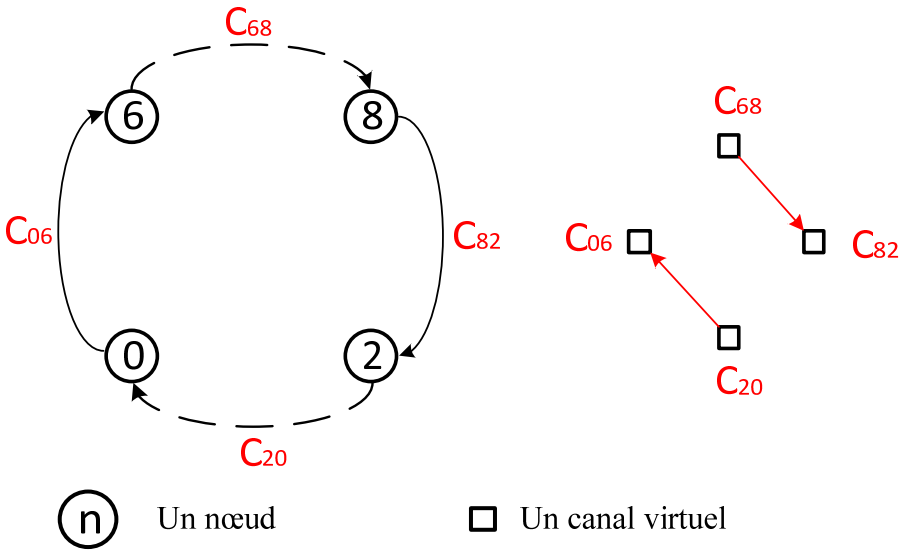


Figure 3.13 Dépendance de canal acyclique pour les 4 liens enveloppants

En tenant compte des restrictions 1 et 2 citées précédemment, nous pouvons maintenant utiliser l'algorithme de routage CAFTA et l'appliquer aux NoCs à topologie Tore en garantissant l'absence de dead-lock. En intégrant le marquage des paquets dans un routeur, on se prémunit aussi des live-lock.

Le pseudo-code de l'algorithme de routage CAFTA pour le Tore est décrit dans la figure 3.14.

Algorithme de Routage pour Tore (x,D,M):

Pour une paire de source et destination donnée (S : source, D : destination)
VN_i : est le réseau virtuel dans lequel le paquet M transite vers sa destination D
x : nœud courant, w : nœud prochain dans le parcours
Initialisation :
$w=\varnothing$
Q : la liste de nœuds disponibles pour accueillir le paquet M prochainement // Avec la conscience de faute des voisins
If x=D *then* éjecter vers le port local et signaler « succès » // M est arrivé à destination
else begin
  If ($Q \neq \varnothing$) *then begin*
    If (sizeof(Q) = 1) // seulement un choix disponible
      then mettre $w=Q_i \in Q$
    else begin //plusieurs choix possibles
      Mettre $w= Q'_i \in Q$, //où Q'_i a un parcours minimal vers la destination. En cas d'égalité ou tout $Q'_i \in Q$, a un parcours non-minimal, la métrique de congestion peut être utilisée comme second critère de classement
    end
  end
  If $w \neq \varnothing$ *then begin* // Acheminer le paquet M vers le nœud w
    If w n'est pas valide dans VN_i *then begin*
      Ejecter M vers *Network Interface* (NI) du nœud courant
      Réinjecter M dans le réseau virtuel VN_j (i !=j, w est valide dans VN_j) et tenter de l'aiguiller vers w.
    end
    If accès à w bloqué par une faute *then begin* *//runtime* défaillance
      Essayer une retransmission et si ele échoue, essayer soit *BPR* soit *SPR*
      If w est encore bloqué *then return* échec // destination *inaccessible*!
    end
  end
  else return échec *// destination inaccessible!*
  If un lien enveloppant horizontal est nécessité pour w, *then begin*
    if M vient d'être injecté localement, *then* aiguiller M vers w.
    else begin
      Éjecter M vers *Network Interface (NI)* de nœud en courant
      Réinjecter M dans le réseau virtuel VN_i et l'aiguiller vers w.
  end
  If un lien enveloppant vertical est nécessité pour w, *then begin*
    if (*M* vient d'être injecté localement
      or x est sur la borne du nord dans *VN North-Last*
      or x est sur la borne du sud dans *VN South-Last*)
      then aiguiller M vers w.
    else begin
      Éjecter *M* vers *Network Interface (NI)* de nœud en courant
      Réinjecter *M* dans le réseau virtuel VN_i et l'aiguiller vers w.
  end
end

Figure 3.14 Pseudo-code de l'algorithme de routage CAFTA

CAFTA pour Tore 2D : exemple de scénario

Pour mieux illustrer cette extension de CAFTA aux tores 2D, nous proposons grâce à la figure 3.15 les étapes permettant l'envoi d'un paquet d'une source en (1,6) vers sa destination en (6,0).

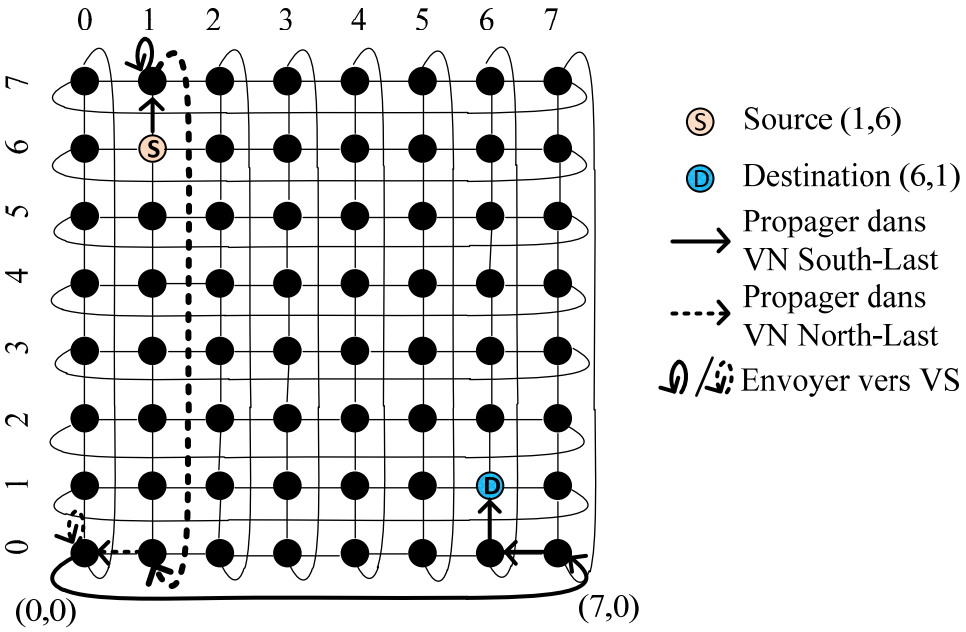


Figure 3.15 Exemple de routage CAFTA dans un tore 2D 8x8

0) D'après la politique de sélection du réseau virtuel décrit par la figure 3.5, ce paquet doit être injecté dans le VN0 correspondant au réseau virtuel SouthLast.

1) Il peut avancer indifféremment soit vers (0,6) soit vers (1,7). Supposons que le paquet choisisse d'aller vers le nœud (1,7) à cause de la présence d'une faute ou d'un état de congestion en (0,6)

2) En raison des restrictions de direction, pour aller au sud il doit changer de VN. Pour cela une source virtuelle est créée en (1,7) et il est consommé localement par le NI.

3) Le paquet est réinjecté dans le réseau virtuel VN1 et peut ainsi traverser le lien enveloppant qui lie les nœuds (1,7) et (1,0).

4) Arrivé en (1,0), la direction optimale en terme de distance (hops) vers sa destination (6,0) est l'ouest vers le nœud (0,0).

5) Comme en 2) à cause des restrictions de directions pour les nœuds à la frontière, pour emprunter le lien enveloppant qui lie les nœuds (0,0) et (0,7), il faut qu'il change de réseau en étant consommé localement puis réinjecté dans le réseau grâce à une source virtuelle.

6,7) Le paquet est réinjecté dans le réseau virtuel VN0 (le réseau virtuel SouthLast) et arrive à destination en (6,0) via (7,0).

3.2 Congestion de réseau sur puce

Nous avons vu jusqu'ici comment tolérer la présence de fautes dans un réseau sur puce de topologies Mesh ou Tore. L'autre objectif de CAFTA est de router les paquets en tenant compte de la congestion. Comme dit plus tôt dans cette thèse, il existe deux types d'approches dans la littérature. La première est basée sur la régulation du taux d'injection de paquets par les nœuds source. Avec la prise en compte du type d'application et de la charge du réseau, ce mécanisme de régulation de trafic peut être très efficace.

La deuxième approche repose sur l'utilisation d'un algorithme de routage adaptif pouvant contourner les zones congestionnées.

Le point commun entre ces deux approches est que dans les deux cas il est nécessaire de surveiller dynamiquement l'état du réseau. L'efficacité du contrôle de congestion dépend donc de l'efficacité et de la précision de la mesure de congestion.

Nous avons vu auparavant que les métriques utilisées dans les différents travaux publiés possèdent des limitations. Dans cette section, nous proposons ainsi une nouvelle métrique de mesure de la congestion qui a priori devrait être plus intéressante.

3.2.1 Nouvelle métrique de congestion – Flit Remain (FR)

Il y a des métriques qui utilisent les informations locales, comme le nombre de VCs libres [Dally 1993], la disponibilité des tampons des nœuds voisins FB (Free Buffer) [Kim 2005], et les métriques comme RCA [Gratz 2008], DBAR [Ma 2011] qui utilisent un ensemble d'informations régionales (voir chapitre 2). Ces métriques ont été proposées et utilisées principalement dans un environnement sans fautes. Cependant, les usages de ces métriques peuvent être exploités afin d'aider la sélection des trajets de propagation aussi dans

un environnement défectueux. Dans un tel environnement, l'objectif primaire pour un algorithme de routage tolérant aux fautes est la sélection minutieuse de trajets sans-faute vers la destination. La métrique de congestion ne peut être alors utilisée que comme un critère secondaire pour la sélection du parcours.

Dans CAFTA, les informations sur l'état des nœuds et des liens proviennent d'une zone délimitée par le nœud courant et ses voisins directs (1 hop). Nous appellerons cette zone la région de conscience de fautes. La congestion étant un critère secondaire, la zone de surveillance et propagation des informations de congestion sera inclus dans la région de conscience de fautes (1 hop aussi).

Une métrique simple de mesure du trafic/congestion nommée FR (en anglais, Flits-Remain) est présentée ici.

Dans un contrôle de flux de type Wormhole, le flit d'en-tête d'un paquet définit le chemin le long duquel les données (payload) se suivent, de la source vers la destination. Ce trajet reste actif jusqu'à le flit de queue du même paquet passe et libère les ressources. Généralement, le flit d'en-tête, outre la destination, contient la taille du paquet auquel il appartient. Cette taille, en unité de flits, correspond au nombre de flits qui suivront le flit d'en-tête. Dans notre architecture de routeur, un compteur est ajouté dans chaque port pour suivre les valeurs actives de FR. Quand le flit d'en-tête d'un paquet choisit l'un des ports de sortie de ce routeur, la valeur FR de ce port est initialisée d'un montant égal à la taille de son paquet (en unité de flit). Après cela, chaque fois qu'un flit de donnée de ce paquet sort de ce port, la valeur FR de ce port est décrémentée de 1.

3.2.2 Etude de cas : comparaison de deux métriques FB et FR

Nous allons ici comparer les conséquences du choix d'utilisation d'une métrique ou une autre. Nous comparerons seulement la nouvelle métrique FR avec la métrique FB. La métrique basée sur le nombre de canaux virtuels libres (VCs) nous semble moins précise et moins performante et ne sera donc pas traitée ici. Cette hypothèse sera confirmée ou infirmée dans le chapitre suivant, expérimentalement.

Considérons la situation décrite dans Figure 3.16 où nous avons trois routeurs I, J et K, I étant le routeur courant, chaque routeur possède des tampons d'entrée. Pour illustrer simplement, nous considérons qu'il y a deux canaux virtuels pour chaque port et les tampons d'entrée sont de profondeur égale à deux flits.

Avant l'entrée du paquet P3 dans le routeur I, les trafics existant sont (Figure 3.16.a) :

1) un paquet P1 entrant du port W du routeur I qui se propage vers le port N du routeur K. Il reste encore 2 flits de P1 qui doivent traverser le routeur I ;

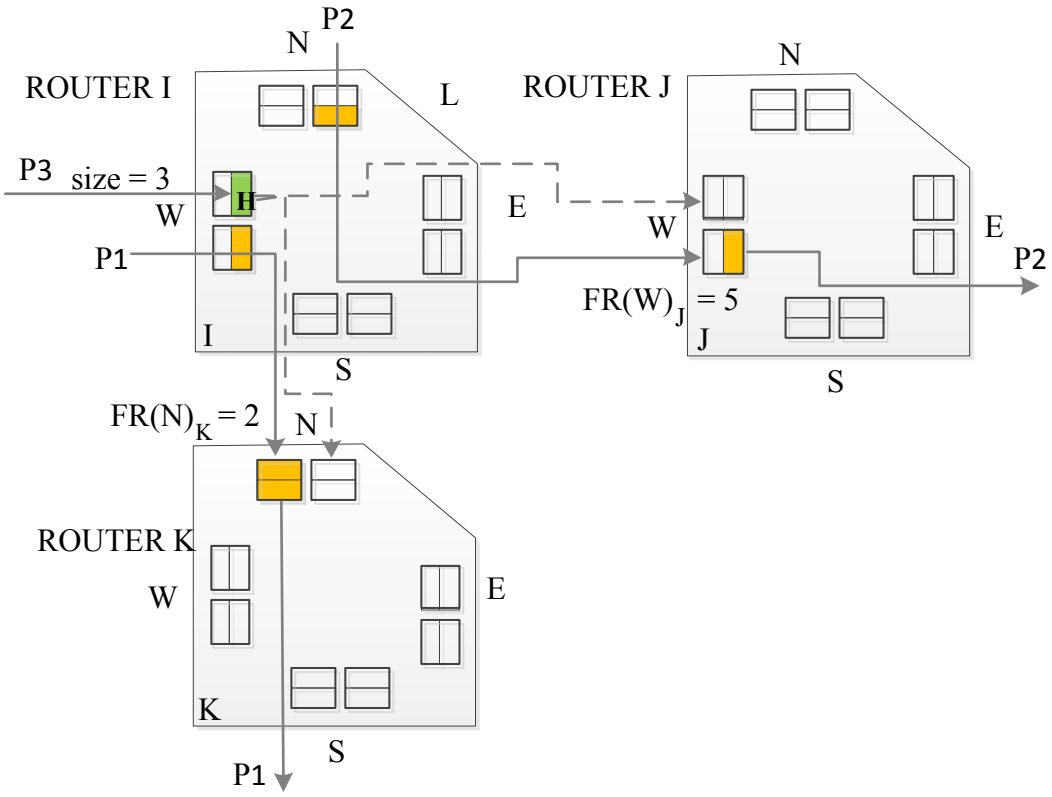
2) un autre paquet P2 entrant du port N du routeur I qui se propage vers le port W du routeur J. Il reste encore 5 flits de P2 qui doivent traverser le routeur I.

Le flit d'en-tête d'un paquet P3 avec une taille de payload de 3 flits entre dans le routeur I par le port W. L'étage de calcul du routage (RC-stage) pour le flit d'en-tête propose au choix deux ports de sortie qui sont : 1) le port est vers le routeur J ; 2) le port sud vers le routeur K.

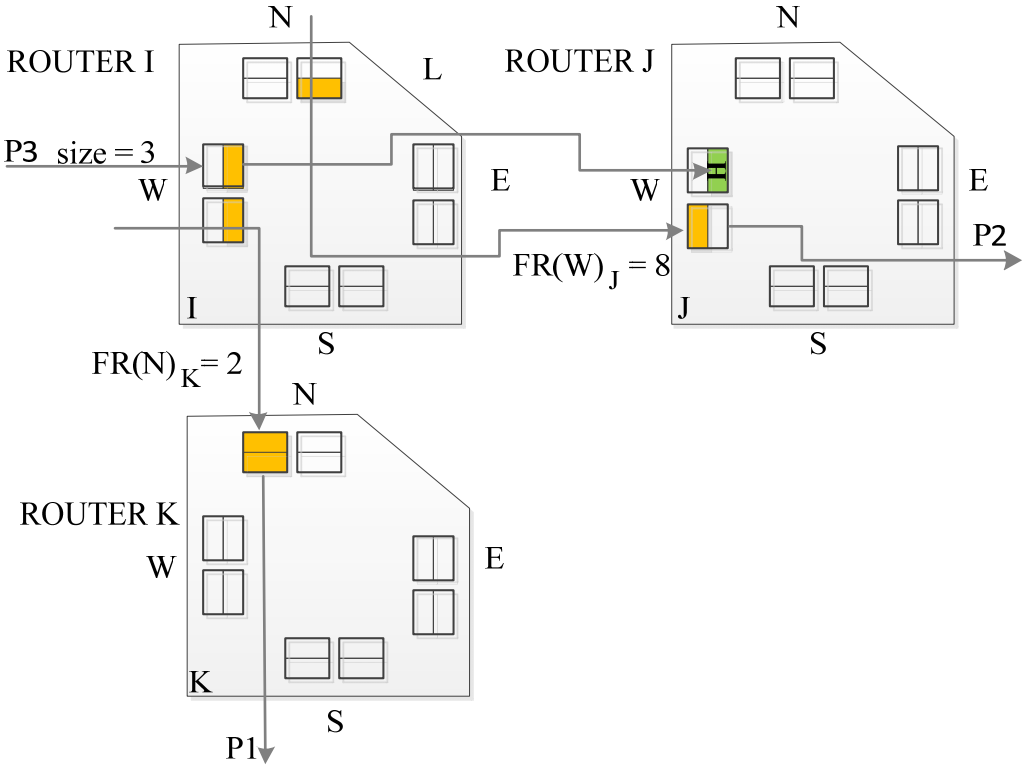
Si on considère comme paramètre de sélection du port la métrique classique FB (en anglais FreeBuffer), qui correspond au nombre de cases libres en unité flit pour un port donné, nous avons : $FB(N)_K = 2 < 3 = FB(W)_J$. Le port W du Routeur J a plus de tampons libres que ceux du port N du Routeur K. Donc, la décision du routage pour P3 est de prendre le port W du Routeur J.

Si nous considérons cette fois la métrique FR comme critère de sélection nous avons : $FR(N)_K = 2 < 5 = FR(W)_J$. La valeur FR du port nord du routeur K est inférieure à celle du port ouest du routeur J. Le port du routeur avec la valeur FR la plus petite sera choisi pour la propagation du paquet, ici le port N du routeur K

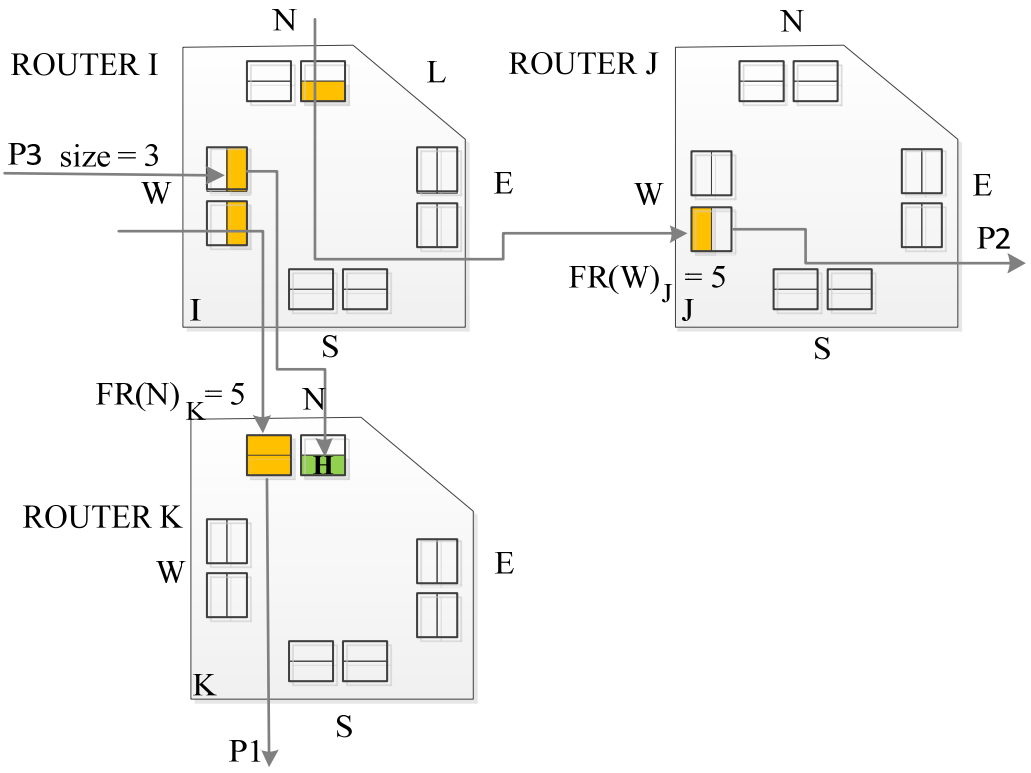
Pour comparer les conséquences des décisions faites en utilisant soit FB, soit FR, nous mesurons la nouvelle valeur FR pour chaque choix. Les figures 3.16.b, et 3.16.c nous montrent l'état du réseau après que le flit d'en-tête de P3 ait été propagé au routeur K. Nous voyons que la première option mène à avoir $FR(N)_K = 2$ et $FR(W)_J = 8$. Quant à la seconde option, elle amène à équilibrer les trafics vers les deux canaux car $FR(N)_K = 5 = FR(W)_J$. Sachant qu'un trafic équilibré sur le réseau conduit à une latence moyenne plus petite, la métrique FR devrait permettre d'obtenir une gestion de la congestion plus efficace qu'en utilisant la métrique FB.



(a) Sélection de la direction de propagation basée sur la métrique FR



(b) Mise à jour de la valeur FR dans le cas du choix basé sur « Free Buffer ».



(c) Mise à jour de la valeur de FR dans le cas du choix basé sur « Flit Remain ».

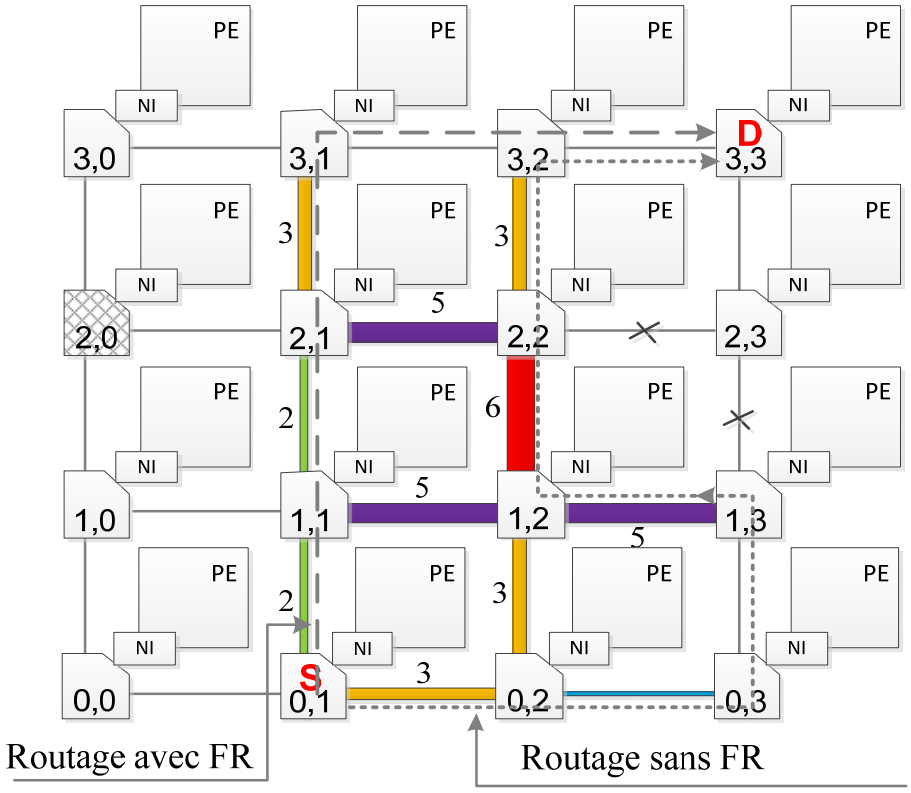
Figure 3.16 Exemple du calcul de la mise à jour de la métrique FR métrique

Si par la suite un paquet P4 devait transiter par un autre port d'entrée de I vers les routeurs K ou J, la sélection du port de sortie serait faite aléatoirement, les métriques FR associées étant égale, $FR(N)_K = 5 = FR(W)_J$.

3.2.3 Etude de cas : tolérance aux fautes & congestion avec FR

L'exemple suivant a pour but de montrer l'efficacité de notre solution CAFTA : Tolérance aux fautes + métrique FR.

Supposons que nous avons la situation présentée dans figure 3.17 où nous devons envoyer un paquet de S(0,1) à D(3,3). Dans le même temps, il existe d'autres trafics dans le réseau, indiqués par l'épaisseur et la couleur des liens ; plus l'épaisseur du lien est grande, plus de trafic à travers ce lien est grand. Généralement, dans les NoCs on observe souvent un fort trafic dans la région centrale du réseau.



Nœud (2,0) est défectueux, et deux liens fauteif: {(2,2)↔(2,3)},{(1,3)↔(2,3)}
Les chiffres sur les liens indiquent les valeurs de FR.

Figure 3.17 Exemple d'un paquet contournant la zone congestionnée grâce à la métrique FR

Si on applique l'algorithme Variant B d'origine le chemin choisi est le suivant : {(0,1), (0,2), (0,3), (1,3), (1,2), (2,2), (3,2), (3,3)}; on constate que le paquet doit traverser la zone congestionnée.

Si on applique CAFTA avec la métrique FR le chemin est alors le suivant: {(0,1), (1,1), (2,1), (3,1), (3,2), (3,3)}. Le paquet évite alors la zone congestionnée.

Dans l'algorithme proposé, les trajets de longueur minimale sont favorisés, et les flits avancent progressivement. Pour rompre les cas d'égalité (pas de fautes de nœud ou de lien sur les routeurs voisin), la métrique FR est utilisée comme critère secondaire pour sélectionner le port adéquat. Le port dont la valeur de FR est la plus petite est choisit. Ainsi sur la figure, le paquet transite du nœud (0,1) au nœud (1, 1) lorsque CAFTA est implémenté.

3.2.3 Amélioration de la nouvelle métrique FR

La métrique « Flit Remain » est à priori plus efficace que la métrique « Free Buffer ». Cependant utiliser cette seule mesure ne donne pas forcément le choix optimal en ce qui concerne le port de sortie.

Pour illustrer cela prenons l'exemple de la figure 3.18.

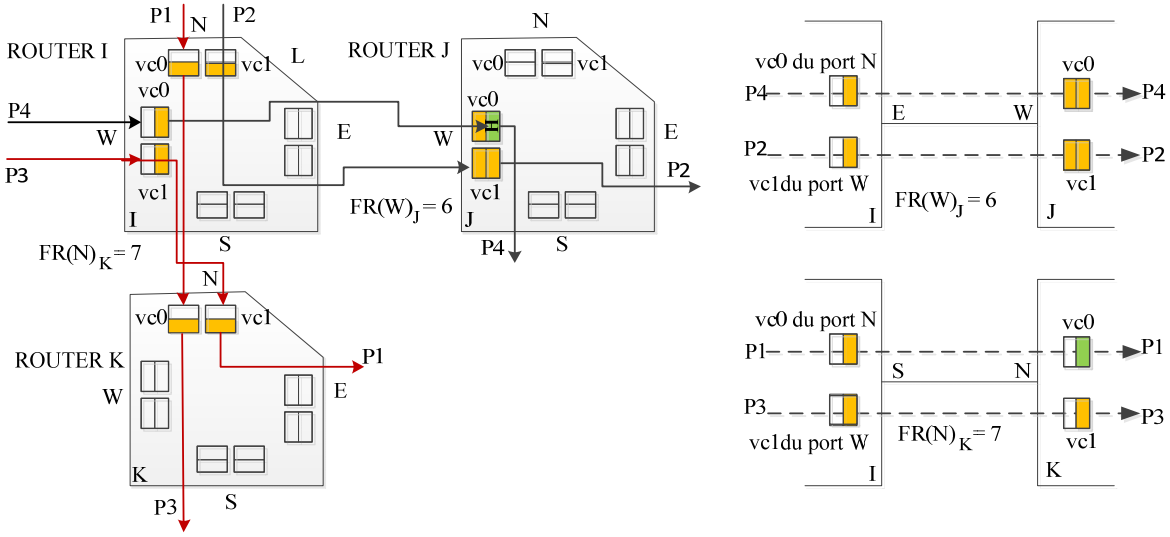


Figure 3.18 Cas d'inadéquation de la métrique FR

Nous avons trois routeurs I, J, K équipés de tampons d'entrée (pas de tampons de sorties), I étant le routeur courant. Supposons qu'un paquet soit injecté localement dans le routeur I et après calcul du routage (RC-stage) pour le flit d'en-tête, les deux ports de sortie possibles sont le port Est vers le routeur J, et le port Sud vers le routeur K. Nous voulons évaluer l'état de congestion de ces deux candidats.

Dans le routeur I sont en cours de transit quatre paquets P1, P2, P3, et P4. P1 et P3 sortent du routeur I par le port Sud. P2 et P4 sortent du routeur I par le port Est. La métrique FR pour le port Est est $FR(W)_J = 6$ tandis que pour le port Sud $FR(N)_K = 7$. Le port Est devrait donc être sélectionné.

Si nous regardons de plus près et que nous tenons aussi compte de l'état d'occupation des tampons d'entrée des routeurs J et K, le choix du port de sortie risque d'être différent. La figure 3.18 droite nous montre les deux choix de ports de sortie possibles.

Considérons maintenant comme métrique celle qui correspond à attendre le moins de temps avant d'envoyer un nouveau paquet vers un port. Pour le choix 1 (port Est), il faut attendre $FR(W)_J + 4$ (flits présents dans les tampons d'entrée du routeur J) = 10 flits.

Pour le choix 2 (port Sud), il faut attendre la libération de $FR(N)_K + 2 = 9$ flits.

Dans ce cas c'est le port Sud qui sera sélectionné comme port de sortie pour le nouveau paquet.

Cette métrique est donc plus précise que FR utilisée seule. Ainsi dans CAFTA nous utiliserons au final la métrique que l'on appellera FR*:

FR* = X + Y = FR + occupation des tampons en aval.

X : pour un port de sortie, la somme des flits qu'il reste à envoyer avant libération des ressources.

Y : les flits stockés dans les tampons d'entrée du routeur aval auquel le port est connecté;

Ces deux informations pouvant être disponibles localement dans le routeur pour chaque port, il est possible de calculer FR* pour chaque port de sortie du routeur.

3.3 Microarchitecture du routeur proposé

Dans cette section, nous détaillons la mise en œuvre des techniques que nous proposons dans une nouvelle architecture de routeur.

L'architecture de base de notre routeur est présentée sur la Figure 3.19. Il est équipé de tampons mémoire sur les ports d'entrée, le contrôle des échanges entre routeurs est à base de crédits et le routage est de type « Wormhole ». L'architecture est une architecture typique de pipeline à 4 étages : écriture dans un tampon (Buffer Write, abrégé BW), calcul du routage (Route Computation, abrégé RC), allocation du canal virtuel (Virtual Channel Allocation, abrégé VA), Switch allocation (SA) et Switch Transfer (ST).

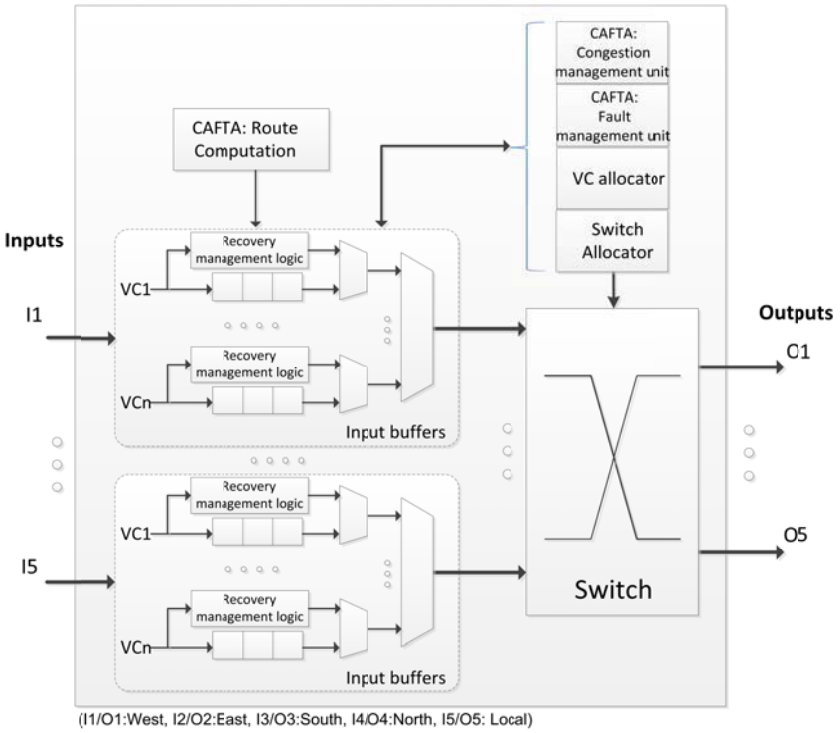


Figure 3.19 Architecure du routeur Ri,j

Le routeur contient également :

- Une unité de gestion des fautes, chargée de la propagation (réception) de signaux correspondant à l'état de défaillance vers (de) les routeurs voisins. Ces informations de conscience de faute sont utilisées par la fonction RC pour sélectionner le bon port de sortie vers un routeur sans fautes. Le port de sortie connecté à un routeur défaillant est invalidé.

- Une unité de gestion de la congestion, qui effectue le calcul de la métrique FR ou FR*. Comme dit dans la section précédente, cette information est utilisée comme critère secondaire pour classer les ports de sorties éligibles. Cette information est une entrée de la fonction logique RC.

- Une unité de recouvrement de fautes constituée d'une part de la logique intégrant le mécanisme de retransmission/réémission et d'autre part de la logique de recouvrement pour chaque canal virtuel.

Unité de gestion des fautes

Chaque routeur met à jour les informations sur l'état de santé de ses liens et ceux de ses routeurs voisins (à une distance d'un hop) dans un registre d'état de faute. Cette technique

est similaire à celle présentée dans [Feng 2013], mais contrairement à celle-ci les fautes considérées sont celles de liens unidirectionnels. Un bit est utilisé pour représenter l'état du lien : 1 pour défectueux, 0 pour pas d'erreur. De nouveaux signaux sont ajoutés sur l'interface du routeur pour implémenter le mécanisme de propagation d'états de fautes entre les routeurs du NoC. Le routeur enregistre donc les états de santé de tous les liens.

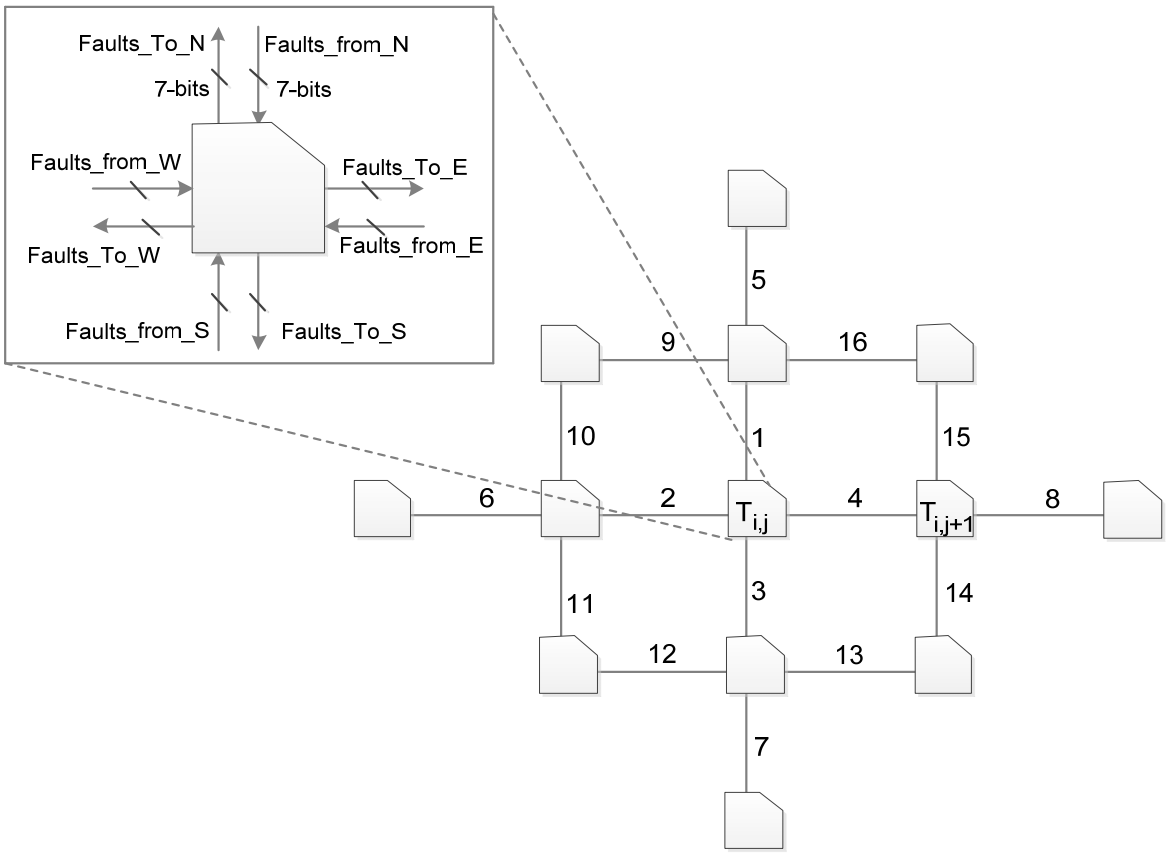


Figure 3.20 Le mécanisme de la conscience de fautes pour le routeur Ri,j

Pour illustrer cela, prenons l'exemple du routeur Ti,j de la figure 3.20. La taille du registre d'état des faute est de 32 bits (32 liens unidirectionnels = 16 liens bidirectionnels numérotés (de 1 à 16 sur la figure 3.20) x 2). Le Routeur Ti,j, transmet (reçoit) les états vers (de) ses routeurs voisins. Par exemple, le Routeur Ti,j envoie au routeur Ti,j+1 sept bits composés de : un bit (1 pour défectueux, 0 pour pas d'erreur) correspondant à l'état du lien de sortie (lien 4 dans Figure 3.17), et six bits (trois paires de bits) qui correspondent aux états des paires de liens 1, 2, 3. De manière symétrique, le routeur Ti,j reçoit sept bits d'information du routeur Ti,j+1 qui représentent : un bit pour l'état du lien d'entrée 4 (lien de sortie du routeur Ti,j+1) et six bits qui correspondent à l'état des liens 14, 15, 8.

Au départ, une faute détectée sur un lien va être enregistrée en tant que faute temporaire. Si le lien est détecté comme défaillant k fois d'affilée, alors la faute est considérée comme permanente et enregistrée comme telle. Ce lien sera désactivé localement et l'information transmises aux routeurs voisins. Lors de la prochaine reconfiguration du réseau, le lien pourra par exemple être matériellement désactivé.

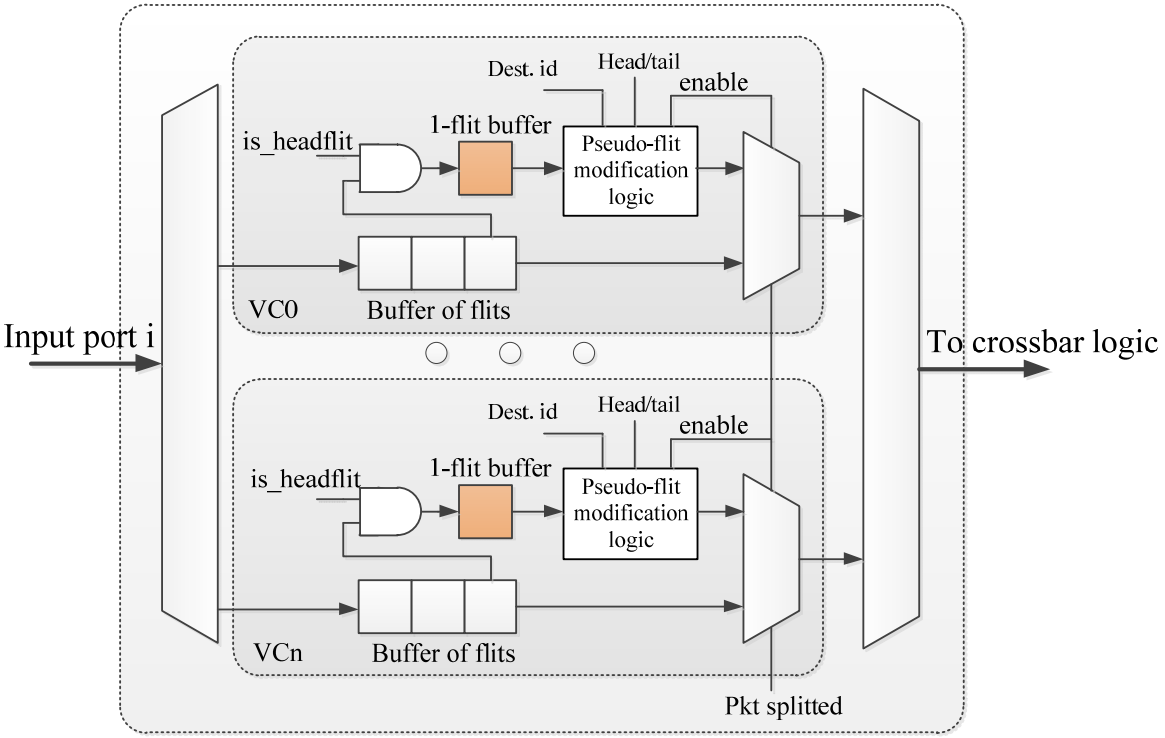


Figure 3.21 Schéma d'un port d'entrée avec la logique dédiée au recouvrement de paquet

Unité de recouvrement de fautes

La figure 3.21 permet de faire un zoom sur le module en charge de la fragmentation, mécanisme qui est déclenché lorsqu'une faute dynamique est détectée. Ce mécanisme comme indiqué dans les sections précédentes intègre la création de pseudos flits d'entête ou de queue.

D'après ce mécanisme, lorsqu'un nouveau paquet arrive sur un port d'entrée du routeur, une copie du flit d'en-tête est stockée dans un tampon de taille un flit jusqu'au moment où le flit de queue du même paquet libère les ressources occupées. Si pendant le transit du paquet du nœud source vers le nœud destination une faute est détectée, il y a alors fragmentation de paquet. Les ressources occupées ne peuvent pas être libérées parce que les flits de queue et charges utiles (payload) sont obligés de suivre la voie tracée par le flit d'en-tête, le chemin est bloqué, et le réseau est alors rapidement paralysé.

Pour éviter le blocage du chemin, le bloc « pseudo-flit modification logic » est activé dès qu'une fragmentation de paquet est réalisée. Un nouveau flit d'en-tête est créé à partir de la copie du flit d'en-tête d'origine. Ce nouveau flit d'entête va avoir une nouvelle destination et un nouveau port (à travers les étapes RC/VC/SW allocation) qui va lui être attribué pour permettre sa propagation.

Si on considère la solution BPR, la nouvelle destination du flit d'entête correspondra à celle d'un nœud intermédiaire où les fragments de paquet se rejoindront et fusionneront. Si c'est la solution SPR qui est choisie la nouvelle destination est celle du nœud destination d'origine, la fusion des paquets sera réalisée par le contrôleur réseau du nœud destination.

Pour ce qui est du flit de queue, dans le routeur aval de la ressource défectueuse, un pseudo flit de queue sera généré en changeant l'étiquette du flit de « en-tête » à « queue ». Ce pseudo flit de queue va suivre le dernier flit du paquet de la première partie et libérer ainsi les ressources réservées.

3.4 Conclusions

Dans ce chapitre nous avons défini un nouvel algorithme de routage tolérant aux fautes qui tient aussi compte de la congestion.

Cet algorithme que nous avons appelé CAFTA se base sur l'algorithme variant B [Chaix 2010] que nous avons enrichi :

1. D'un mécanisme de fragmentation de paquets pour tolérer les fautes dynamiques. Deux techniques ont été proposées selon le nombre de fautes à tolérer BPR (Bypass Packet Recovery) et SPR (Split Packet Recovery)
2. D'un mécanisme de gestion de la congestion basé sur une nouvelle métrique appelée FR (Flit Remain). Une optimisation de cette métrique (FR*) a été proposée.

Pour chaque routeur, la gestion du routage est effectuée en fonction de l'état des liens et des routeurs directement voisins (un hop). Lorsque ceux-ci ne sont pas défectueux, la gestion du routage est faite en fonction des valeurs de FR (ou FR*) des routeurs directement voisins.

Nous avons aussi dans ce chapitre, proposé une extension de l'algorithme CAFTA aux NoCs de topologie Tore. Pour cela nous avons défini des restrictions de routage à apporter aux nœuds en bordure du NoC (liens enveloppants), cela afin d'avoir un algorithme sans deadlocks.

Enfin nous avons donné une description globale de la microarchitecture du routeur nécessaire à la mise en œuvre des différentes fonctionnalités citées ci-dessus.

Chapitre 4 Expérimentations et Résultats

Ce chapitre a pour but d'évaluer les techniques que nous avons proposées dans le chapitre précédent. Nous allons implémenter l'algorithme de routage tolérant aux fautes (CAFTA) dans une plateforme de simulation pour réseau sur puce.

Des simulations intensives nous permettent d'évaluer l'efficacité de la nouvelle métrique de mesure de la congestion par rapport à celles existantes. Nous comparerons la capacité à tolérer des fautes des techniques de fragmentations de paquets BPR et SPR. L'algorithme CAFTA dans son ensemble sera comparé à Variant B et son efficacité sera mesurée dans le cas d'un réseau de topologie 2D-Mesh et de topologie Tore.

4.1 Plateforme d'expérimentation

Nos expérimentations vont consister à effectuer des campagnes d'injections de fautes et de comparer notre approche avec les algorithmes de référence et cela pour différents types de trafics synthétisés. Les critères de performance pertinents que nous étudierons dans ce chapitre sont la latence moyenne et le ratio de paquets livrés avec succès.

Avant d'analyser les résultats, nous allons d'abord présenter le simulateur utilisé et voir comment il a été mis en œuvre. Nous décrirons ensuite les différents trafics générés et le type de fautes injectées.

4.1.1 Simulateur utilisé :

Iris [Web Iris NoC] est un simulateur incluant un modèle de réseau sur puce écrit en C++. Iris est lui-même intégré dans une plateforme de simulation multi composants, multi niveaux nommée Manifold [Wang 2014]. Le simulateur Iris est basé sur la méthodologie de simulation à événements discrets. Iris permet une vitesse de simulation très rapide même pour un réseau avec des centaines de nœuds et il est précis au niveau cycle grâce à la modélisation comportementale (en langage C++) des composants du NoC.

Le micro réseau que nous étudions est composé de quatre types d'éléments : une interface réseau (Network Interface Controller, NIC), un routeur, un lien et un bloc de génération/consommation de paquets (processeur + mémoire).

- Le composant interface réseau est connecté d'un coté à un routeur et de l'autre à un processeur ou à une mémoire. L'interface réseau possède deux rôles :

- assembler et ordonnancer les flits reçus depuis le routeur et les transmettre au processeur et/ou à la mémoire.
- Construire un paquet de flits à partir des données envoyées par le processeur et les envoyer au routeur auquel il est attaché.

- Le routeur fourni par Iris est un routeur classique avec un pipeline de 4 étages (RC/IB : Routage Calcul / Input Buffer, VCA : VC Allocation, SWA : Switch Allocation et ST : Switch Transfer) supportant des canaux virtuels. Les paramètres du routeur sont configurables : nombre de canaux virtuels, taille des tampons, nombre de ports, etc.
- Le bloc de génération de paquets est modélisé comme un générateur de trafic (pour émuler un processeur) et un réservoir de paquets (pour émuler une mémoire).

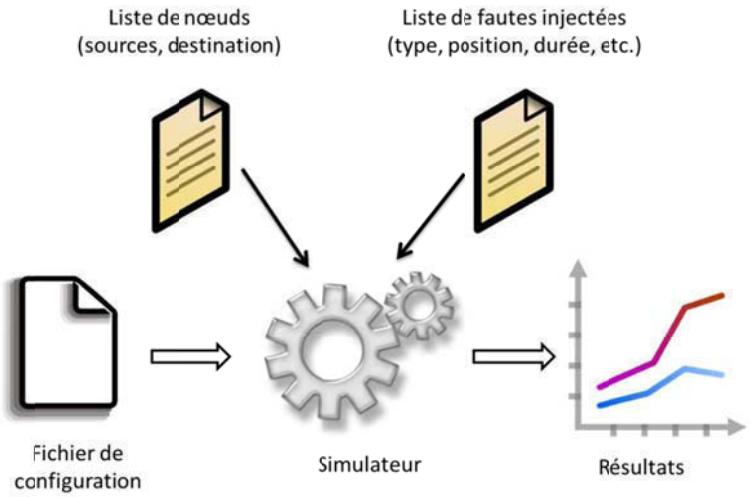


Figure 4.1 Flot de simulation

Le flot de simulation est représenté dans la figure 4.1. Le fichier de configuration est une entrée du simulateur. Parmi les paramètres nécessaires on retrouve :

- Les paramètres du NoC :
  - Topologie
  - Taille du NoC
- Les paramètres du routeur
  - Architecture : physique / canaux virtuels
  - Nombre de ports
  - Nombre de canaux virtuel

- Taille des tampons d'entrée
- Politique d'arbitrage et d'allocation de canal / crossbar
- Algorithme de routage
- Métrique de congestion

- Les paramètres de simulation
  - Type de trafic généré
  - Période d'injection de paquets dans le réseau (intervalle de temps entre deux injections de paquets)
  - Faute injectée (modèle, type, position, durée, etc.)
  - Nombre de paquets injectés et reçus (durée de la simulation)

Comme l'un de nos objectifs est d'étudier la tolérance aux fautes pour un algorithme de routage donné, il nous a fallu adapter le fichier de configuration pour y inclure la possibilité d'injecter des fautes. Ainsi pour permettre l'automatisation des campagnes d'injection de fautes, nous avons créé un générateur de fichiers de configuration.

Topologies	Maille / Tore 2D
Taille du NoC	10x10, 12x12, 16x16
Architecture du routeur :	Support de canaux virtuels
Arbitration d'allocation de canal crossbar	Round-robin
Nombre de ports	5 (Est, Sud, Ouest, Nord, Local)
VNs (Réseaux virtuels)	2
Nombre de VCs pour chaque port	4 / 2 per VN
Taille des tampons	8 flits
Taille des paquets	10 flits
Algorithme de routage	VarientB, CAFTA
Métrique de congestion	Free buffer, Free VC, FR, xbar (nb de demandes pour crossbar), FR*
Trafics générés	Uniform, random, Transpose, Bit complement
Modèles de fautes injectées	Fautes gros grains sur les liens unidirectionnels et les routeurs
Taux d'injection de faute (%)	0, 2, 5, 10, 20, 30, 40
Nombre de paquets tracés par nœud source	5000
Nombre d'itération	1000

Tableau 4.1 Configuration de simulation de réseau

Le tableau 1 synthétise les configurations possibles du simulateur pour une campagne d'injection de fautes. Les simulations commencent par une période d'initialisation (2000 cycles), correspondant à positionner le NoC avec une certaine charge de trafic. Les nœuds sources commencent à injecter les 5000 paquets, chacun possédant son étiquette (id de 0~4999). Ces nœuds envoient ensuite des paquets sans étiquette, pour conserver le même niveau de trafic dans le réseau, jusqu'à la fin de la simulation caractérisée par l'arrivée des 5000 paquets aux nœuds destinations.

Pour chaque simulation, nous définissons un certain nombre de nœuds sources qui injectent les paquets et un certain nombre de nœuds destination qui consomment / éjectent les paquets reçus. Le nombre de nœuds source/destination sera le paramètre qui nous permettra de faire varier la charge du réseau et mesurer la latence correspondante. De même, on fera varier le nombre, le type et la durée des fautes pour mesurer le taux de succès et la latence du NoC. Pour que les simulations soient déterministes, représentables et répétables, nous générons une liste de nœuds source / nœuds destination / positions des fautes injectées (type de faute, durée de faute). A chaque itération, le simulateur prend en entrée des sources, destinations et fautes injectées différentes dans ces listes. Dans nos expérimentations, pour une configuration donnée, chaque résultat (point d'une courbe) de simulation correspond à une moyenne obtenue sur 1000 itérations. La durée d'une itération de simulation varie entre 30 secondes et 3 minutes selon le taux de trafic et les fautes injectées.

4.1.2 Génération de trafics synthétisés

Trois différents trafics sont utilisés pour l'évaluation et l'analyse de l'algorithme que nous avons proposé : « Uniform Random », « Transpose » et « Bit complement »

Pour décrire ces trafics synthétisés, nous définissons s_i (d_i) comme le i-ème bit de l'adresse du nœud source (nœud destination) en binaire. La longueur en nombre de bits d'une adresse est $a=\log_2 N$, ou N est le nombre de nœuds dans le réseau. Les fonctions qui calculent les adresses de destination des paquets seront différentes selon le type de trafic utilisé :

- Uniform Random: chaque source envoie une quantité égale de trafic vers une destination

- Transpose: chaque source envoie des messages uniquement à une destination dont la valeur d'adresse correspond à sa propre adresse transposée : $d_i = s_{(i+a/2) \bmod a}$ (par exemple, pour un NoC 4x4, d=5=b0111 lorsque s=b1101=13)

- Bit complement: chaque nœud envoie des messages uniquement à une destination dont l'adresse est le complément à un de sa propre adresse : di = /si (si si = 1, di = 0 ; si si = 0, di = 1, par exemple, pour un NoC 4x4, d=b0111=7 lorsque s=b1000=8)

Il existe d'autres types de trafics synthétisables comme « Hotspot », « Tornade » [Jiang 2010], etc. Nous nous limiterons dans ce qui suit à l'utilisation des trois types de trafic Uniform Random , Transpose et Bit complement qui permettent de modéliser les principaux trafics que l'on peut voir dans une majorité d'applications. Ils permettent aussi de se comparer plus facilement avec d'autres techniques issues de la littérature.

4.1.3 Génération de fautes

Grâce au simulateur à événements discrets, nous pouvons modéliser les injections de fautes comme des événements et les mettre dans les queues de simulation. Avec le modèle de fautes gros grain que nous avons défini, nous pouvons utiliser deux types d'événement « désactivation » et « réactivation » pour injecter une faute permanente, une faute transitoire et une faute intermittente:

- Faute permanente : au temps T, création d'un événement d'injection de faute pour désactiver un lien (faute de lien) ou tous les liens d'un routeur (faute de routeur)
- Faute transitoire : au temps T, création d'un événement d'injection de faute pour désactiver un lien (faute de lien) ou tous les liens d'un routeur (faute de routeur) ; puis au temps T+1 cycle, création d'un événement d'enlèvement de la faute pour réactiver le lien (faute de lien) ou tous les liens d'un routeur (faute de routeur).

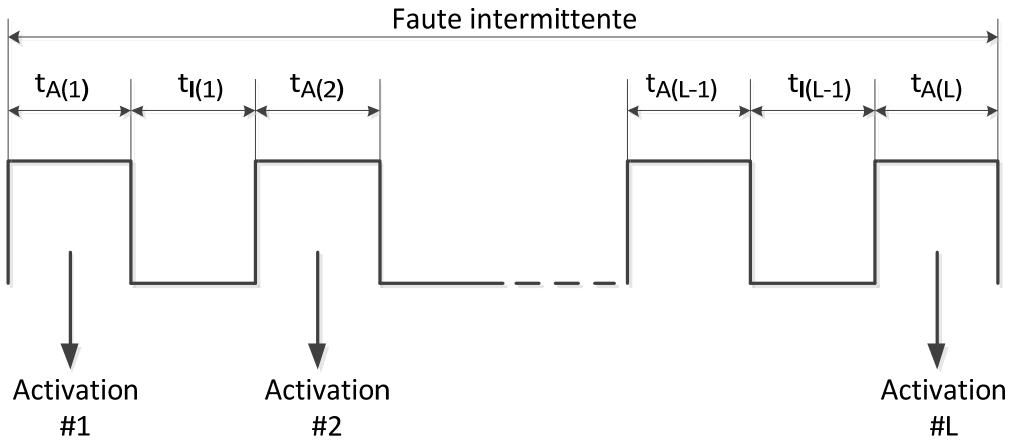


Figure 4.2 Mise en œuvre de l'injection d'une faute intermittente

- Faute intermittente : une faute intermittente est modélisée comme une rafale de L fautes d'une durée tA avec un intervalle tI entre deux fautes actives consécutives. Ainsi on peut créer un événement d'injection de faute pour désactiver un lien (faute de lien) ou tous les liens d'un routeur (faute de routeur) au temps T + l*(tA+tI), 0<=l<L; puis au temps T+ l*(tA+tI)+tA, on créera un événement d'enlèvement de la faute pour réactiver un lien (faute de lien) ou tous les liens d'un routeur (faute de routeur) [Yu 2012]. On pourra jouer sur la durée de la faute intermittente en faisant varier L, tA et tI.

Dans cette thèse nous nous intéressons aux fautes de liens unidirectionnels et aux fautes de nœuds. Ces composants peuvent donc être le siège de fautes ayant des effets permanents, intermittents ou transitoires. Pour être représentatif de ce qui se passe dans la réalité dans un environnement difficile, nous avons, pour chaque campagne d'injection de faute, injecté les trois types de fautes en même temps. Le ratio de fautes différentes injectées en même temps est celui communément admis [Benso 2003], à savoir:

- 15% de fautes permanentes statiques (défauts de fabrication ayant pour conséquences des liens ou routeurs fautifs au lancement de la simulation).
- 5% de fautes permanentes dynamiques (fautes permanentes pouvant apparaître pendant la simulation qui peuvent être dues à un vieillissement du circuit par exemple).
- 40% de fautes transitoires (pouvant être dues aux conséquences du bombardement de particules de type SEU, SET, …) d'une durée un cycle.

- 40% de fautes intermittentes ayant une durée aléatoire entre 200 et 800 cycles. Comme les fautes intermittentes se manifestent en rafale, elles sont modélisées avec une longueur de rafale aléatoire entre 1 et 10 [Yu 2012].

Dans un réseau en maille 2D NxN, le nombre de liens unidirectionnel est égal à 4n(n-1). Ce sera aussi le nombre maximal de fautes de lien. Donc, dans notre étude, il y aura 360 fautes de lien possibles pour un réseau en taille 10x10, 528 fautes de liens possibles pour un réseau 12x12, et 960 pour un 16x16. Quant au réseau en tore 2D, le nombre de fautes de liens possibles pour les tailles 10x10, 12x12 et 16x16 est de 400, 576 et 1024 respectivement.

Injection de fautes et partitionnement du réseau

Nous nous intéressons dans cette thèse aux futurs NoC qui seront potentiellement utilisés dans des environnements critiques. De ce fait nous injecterons de nombreuses fautes et cela pourrait entrainer un partitionnement du NoC en plusieurs sous réseaux. Il faut donc s'assurer qu'il existe toujours un chemin qui lie la source à la destination. Ainsi, quand nous générons la liste des fautes à injecter, nous utilisons l'algorithme « Breadth First Search » [Knuth 1998] pour vérifier la connexité du réseau et vérifier qu'il existe toujours un chemin entre chaque paire (source et destination) pour la liste de faute à injecter.

4.2 Résultats expérimentaux

Dans nos simulations, nous allons mesurer :

- la latence d'un paquet, c'est à dire le temps entre le moment où le paquet est injecté dans l'interface réseau et le moment où ce paquet est consommé dans l'interface réseau du nœud destination. La latence mesurée correspond à la latence moyenne des paquets délivrés avec succès parmi les 5000 paquets injectés.
- le taux de réussite/succès de livraison qui représente le pourcentage de paquets qui arrivent à destination par rapport aux paquets injectés.

Nous avons vu précédemment que pour chaque configuration de simulation, 1000 itérations de simulation étaient effectuées de façon à faire varier les nœuds source et

les nœuds destination. Ainsi les latences et taux de réussite qui seront présentées dans la suite de ce chapitre correspondent à des moyennes sur les 1000 itérations.

4.2.1 Comparaison des métriques de congestion

Les premières expériences menées ont consisté à évaluer la pertinence de la nouvelle métrique de mesure de congestion FR. Le NoC modélisé et simulé est un NoC sans fautes, l'algorithme de routage est celui de CAFTA sans tolérance aux fautes (donc celui de Variant B) que l'on a modifié pour tenir compte des métriques de congestion. Les différentes simulations ont été réalisées pour les trois types de trafic : Uniform Random, Transpose et Bit complement, sur un NoC de taille 12x12 (les paramètres du NoC sont ceux du tableau 4.1).

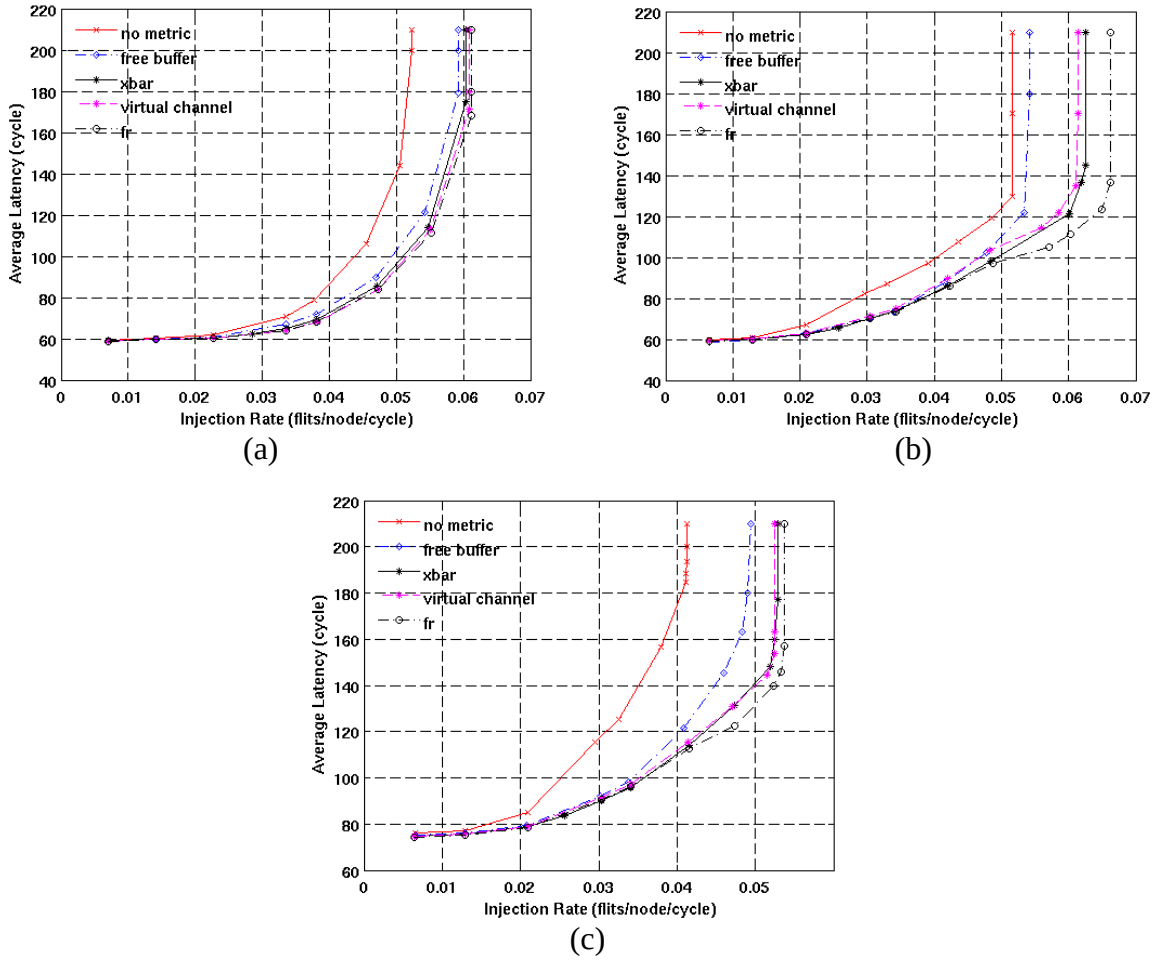


Figure 4.3 Latence moyenne avec différentes métriques de mesure de congestion pour les trafics : (a) « Uniform Random », (b) « Transpose », (c) « Bit complement », pour un NoC 12x12 sans injection de fautes

Nous avons réalisé cinq modèles de ce NoC qui diffèrent les uns des autres par la métrique de mesure de congestion utilisée :

- no metric : pas de routage fonction de la congestion. Dans ce cas, l'algorithme de routage étant Variant B, une comparaison directe peut être faite.
- free VC : métrique qui prend en compte la disponibilité des canaux virtuels libres dans le routeur aval [Dally 1993].
- free buffer : métrique qui prend en compte le nombre de cases disponibles dans les tampons en aval du routeur [Kim 2005].
- xbar : métrique qui prend en compte le nombre de demandes au crossbar pour un port de sortie dans le routeur actuel [Gratz 2008].
- FR : est la nouvelle métrique que nous proposons qui prend en compte le nombre de flits restants à faire passer dans les routeurs avals.

La figure 4.3 synthétise les nombreuses simulations pour chaque trafic. Nous pouvons observer que la prise en compte de la congestion permet comme attendu d'améliorer la latence et que le réseau est saturé pour une charge plus élevée : le taux d'injection de paquets, le pir (paquet injection ratio) correspondant à la saturation du réseau est plus élevé quel que soit le type de trafic.

On peut voir aussi sur ces figures que la latence mesurée pour la métrique FR est plus faible que celles obtenues pour les autres métriques :

- Pour le trafic « Uniforme Random », la métrique FR présente une amélioration de la latence moyenne (mesurée sur la pointe de saturation) de 2,5% par rapport à « free VC », de 3,5% par rapport à « xbar », de 14,2% par rapport à « free buffer » et de 31,3% par rapport à « no metric ».

- Pour le trafic « Transpose », la métrique FR présente une amélioration de la latence moyenne de 17,3% par rapport « free VC », de 18,2% par rapport à « xbar », de 16,1% par rapport à « free buffer » et de 23% par rapport « no metric ».

- Pour le trafic « Bit Complement », la métrique FR présente une amélioration de la latence moyenne de 7,1% par rapport « free VC », de 9,1% par rapport à « xbar », de 21% par rapport « free buffer» et de 49,2% par rapport « no metric ».

Ces simulations nous permettent de conforter nos hypothèses en ce qui concerne la pertinence de la métrique FR. Nous obtenons de meilleurs résultats (latence moyenne avant saturation plus petite, ratio d'injection de paquets à saturation plus grand) pour tous les cas étudiés. FR permet d'améliorer l'algorithme de routage en répartissant mieux le trafic qu'avec

une autre métrique. Pour les expérimentations décrites ci-après, l'algorithme utilisé est CAFTA incluant la gestion de la congestion basé sur la métrique FR.

4.2.2 Tolérance aux fautes : performance BPR et SPR

Dans le chapitre précédent, nous avons présenté deux schémas de recouvrement de paquets : BPR (Bypass Packet Recovery) et SPR (Split Packet Recovery). Les deux méthodes diffèrent l'une de l'autre de par l'endroit où les parties d'un paquet sont concaténées. BPR essaie de rassembler les deux parties du paquet dans un nœud intermédiaire en aval du routeur fautif, pour contourner un ou plusieurs liens/nœuds défaillants. Quant à SPR, le regroupement des parties de paquet se fait toujours dans l'interface réseau du nœud destination.

En terme de complexité d'implémentation, BPR nécessite d'intégrer des fonctions de regroupement de paquets dans chaque routeur, alors que pour SPR ces fonctions doivent se situer dans chaque interface réseau. Une implémentation fine doit être réalisée pour déterminer quelle solution est la moins couteuse en terme de surface silicium. Mais avant cela voyons quelle solution est la plus intéressante par rapport à nos objectifs de tolérance aux fautes.

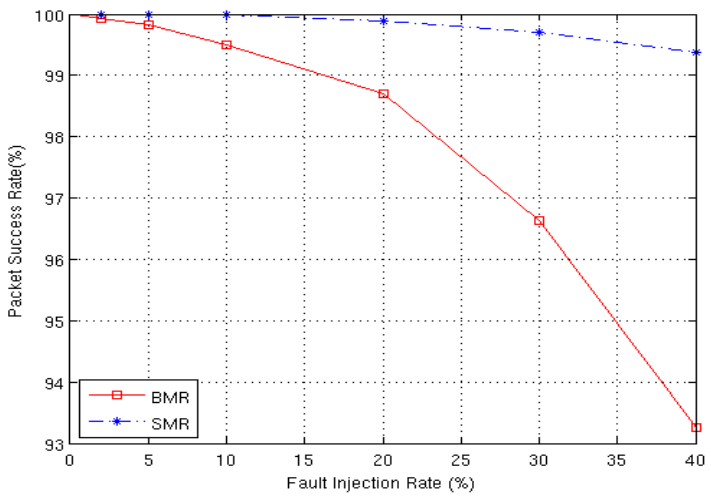


Figure 4.4 Taux de paquets délivrés avec succès avec différents ratios d'injection de faute (faute de liens) sous le trafic «Uniform Random» pour un NoC en maille16x16

Nous avons effectué un ensemble de simulations pour des NoC équipés du mécanisme BPR, d'une part et du mécanisme SPR d'autre part. Nous avons pour chaque simulation fait varier le nombre de fautes injectées (les paramètres du NoC sont ceux du tableau 4.1).

Il existe deux cas pour lesquels un paquet est considéré comme perdu et éjecté par le simulateur :

1) Un fragment de paquet arrive dans une impasse : par exemple un routeur sans sortie saine ou un routeur ou le paquet est bloqué par les restrictions de routage.
2) Un fragment de paquet arrive à atteindre l'interface réseau du nœud destination, mais les autres parties de ce paquet n'arrivent pas à le rejoindre avant un « timeout » T (une valeur empirique qui varie la taille de NoC) fixé dans le simulateur.

La synthèse pour le trafic Uniform Random présentée figure 4.4, nous montre que lorsque le taux d'injection de fautes est faible, les mécanismes BPR et SPR permettent de sensiblement tolérer les 4 types de fautes injectées (fautes permanentes statiques et dynamiques, fautes transitoires et intermittentes). Néanmoins, dès que ce taux d'injection augmente, SPR est beaucoup plus efficace. Avec des taux élevés de fautes présentes dans le micro réseau, avec SPR plus de 99 % des paquets arrivent à destination contre 93 % avec la technique BPR. Cela confirme là aussi nos hypothèses présentées dans le chapitre précédent quant aux limitations du mécanisme BPR. En effet lorsque beaucoup de fautes affectent des routeurs ou des liens voisins, le mécanisme BPR peut introduire une forte augmentation de la latence, voire bloquer le réseau. Ainsi pour la suite des expériences nous nous baserons sur l'algorithme CAFTA implémentant le mécanisme SPR.

4.2.3 Comparaison CAFTA - Variant B

L'expérience suivante nous a permis de situer notre algorithme CAFTA par rapport à Variant B. Nous voulons voir entre autres si l'introduction des mécanismes de gestion de la congestion et de tolérance aux fautes dynamiques n'entrainent pas d'effets de bord.

Nous comparons donc les deux algorithmes sur un micro réseau en maille 2D 16x16 sachant que CAFTA est une importante extension de Variant B, et devrait logiquement donner de meilleures performances.

1. Mesure de la latence

Nous avons d'abord mesuré la latence moyenne pour trois taux d'injection de fautes différents avec un trafic Uniform Random:

a) 0% des liens sont fautifs : cas sans faute qui revient à évaluer l'ajout de la gestion de congestion à Variant B (Figure 4.5.a);

b) 10% des liens sont fautifs (Figure 4.5.b)

c) 40% des liens sont fautifs (Figure 4.5.c)

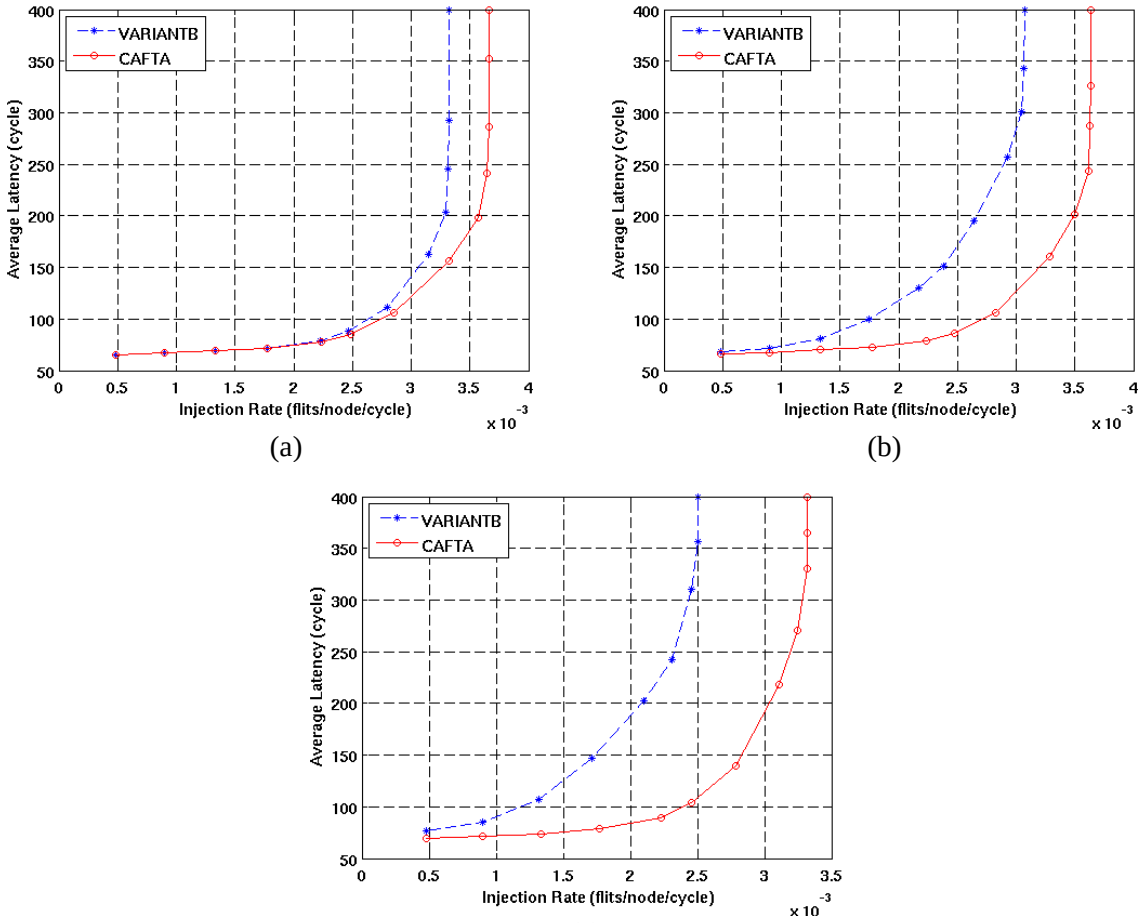


Figure 4.5 Latence moyenne de CAFTA v.s. VariantB avec différents taux d'injection de paquet sous le trafic « Uniform Random» pour un NoC 16x16 en maille 2D (a) 0% fautes de lien (b) 10% (c) 40%

Comme prévu, les fautes injectées affectent la latence moyenne du réseau. Celle-ci tend à augmenter avec l'augmentation de la charge du réseau mais aussi avec l'accroissement du nombre de fautes de lien.

On peut voir sur la figure 4.5 que la latence moyenne de Variant B augmente plus vite que celle de CAFTA si on injecte plus de trafic et/ou plus de fautes. En prenant comme référence le pic de saturation (pir) de Variant B, on constate que CAFTA permet d'améliorer

(réduire) la latence moyenne de 52%, 110% et 290% par rapport Variant B pour un taux d'injection de fautes de 0%, 10% et 40% respectivement.

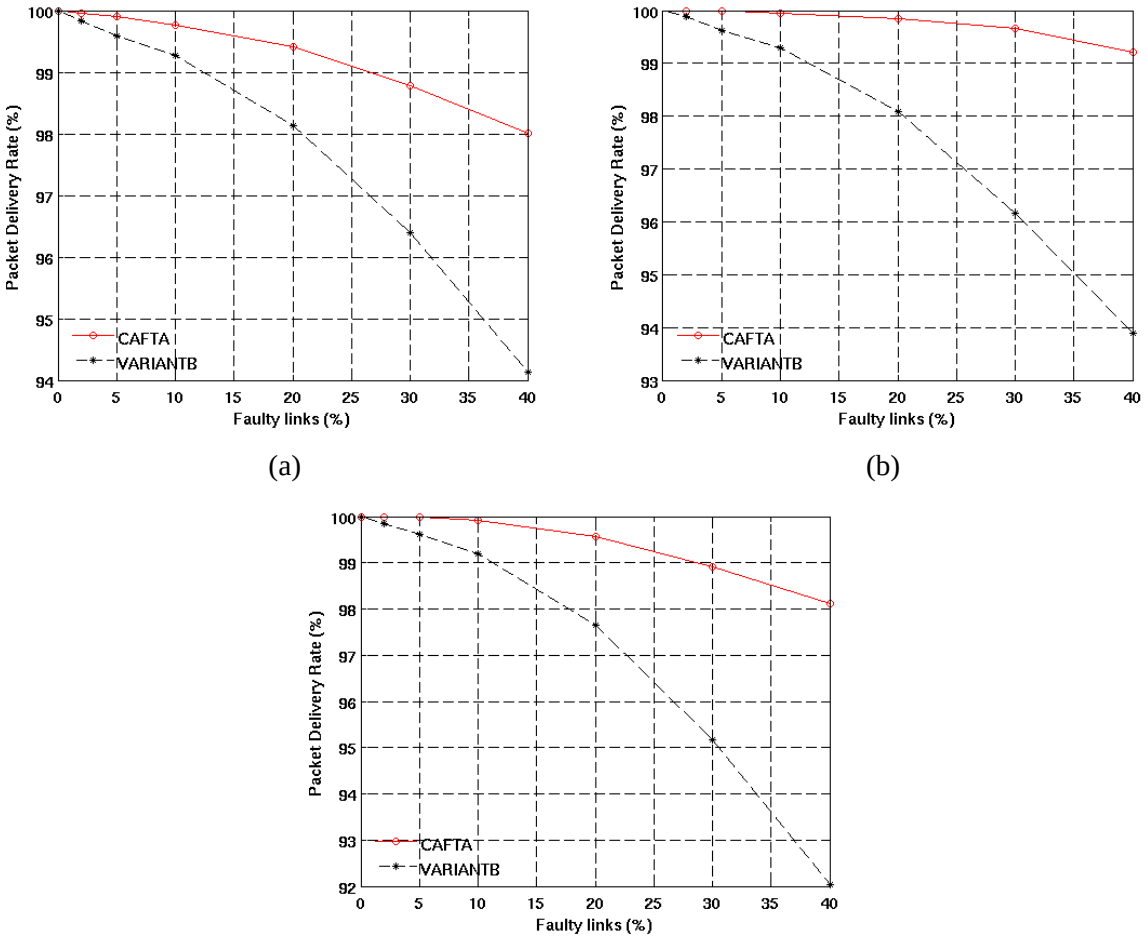


Figure 4.6 Taux de réussite avec différents ratios d'injection de faute (faute de liens) sous le trafic : (a) « Uniform Random », (b) « Transpose », (c) « Bit complement », pour un NoC en maille16x16

2. Taux de réussite de livraison de paquets

Nous avons ensuite comparé l'aspect fiabilité entre CAFTA et Variant B, en mesurant le taux de réussite de paquets envoyés. Nous reprenons les mêmes conditions expérimentales que précédemment : NoC de topologie Maille 2D 16x16, modèle de faute de lien (Figure 4.6) sous différents trafics. Nous faisons varier le taux d'injection de fautes et mesurons le taux de paquets qui arrivent à destination.

Là encore comme prévu, les résultats montrent que CAFTA offre un niveau de fiabilité plus élevée que Variant B pour les trois types de trafics. Avec 40% de liens défectueux, CAFTA réussit à acheminer plus de 98% des paquets. Quant à Variant B, pour

ce même taux, 4 fois plus de paquets échouent à atteindre leur destination (taux de réussite de 92 %).

Grâce à l'ajout des techniques de fragmentation/recouvrement de paquets et de prise en compte de la congestion, CAFTA propose clairement une amélioration des performances et de la fiabilité par rapport à Variant B.

4.2.4 Etude de l'algorithme CAFTA pour un réseau en maille 2D

Nous avons ensuite voulu étudier plus finement l'impact du type de fautes et du nombre de fautes sur la latence moyenne (et saturation du réseau) et le taux de livraison de paquets.

4.2.4.1 Saturation du réseau

Nous avons considéré dans un premier temps un NoC avec seulement des fautes de liens puis le même NoC avec uniquement des fautes de routeurs. Les conditions expérimentales restent identiques à celles présentées précédemment.

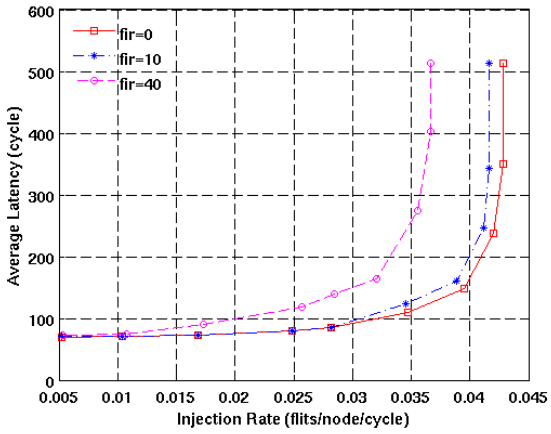
Fautes de lien

Nous avons mesuré la latence moyenne (Figure 4.7) pour trois différents trafics, le NoC étant de taille 16x16. Nous avons « dégradé » le circuit comme suit :

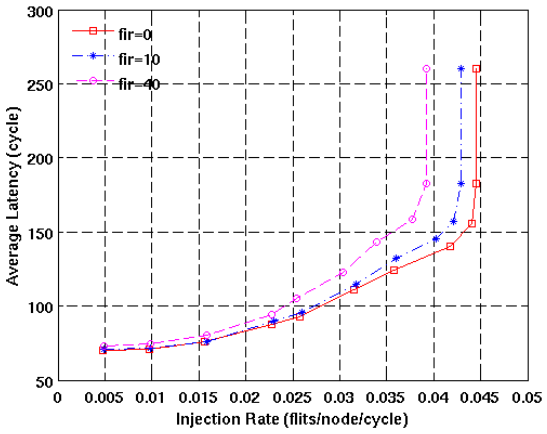
a) 0% des liens sont défaillants/fautifs : cas sans faute (Figure 4.7.a);

b) 10% des liens sont défaillants/fautifs (Figure 4.7.b)

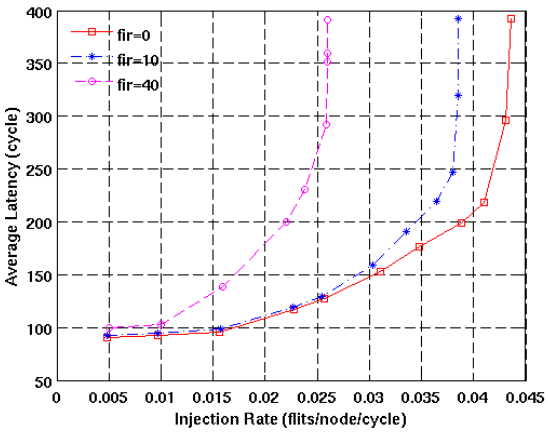
c) 40% des liens sont défaillants/fautifs (Figure 4.7.c)



(a)



(b)



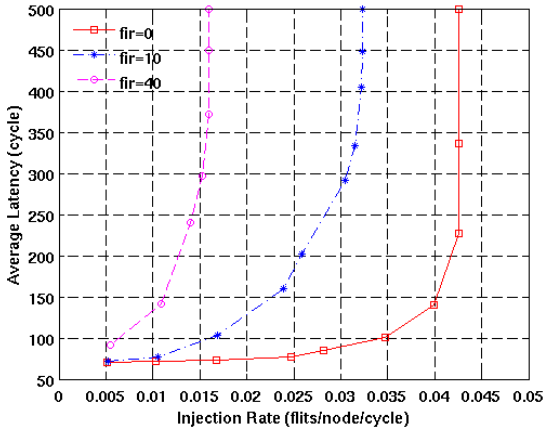
(c)

Figure 4.7 Latence moyenne de CAFTA avec différents taux d'injection de fautes (fautes de liens) sous le trafic: (a) « Uniform Random », (b) « Transpose », (c) « Bit complement », pour un NoC 16x16 en maille 2D

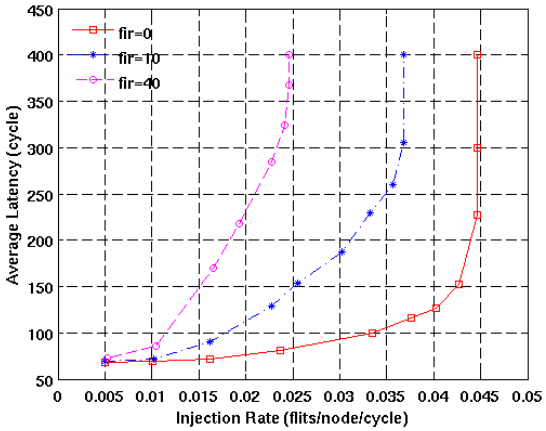
Comme prévu, l'injection de faute affecte la latence moyenne du réseau, qui augmente avec le nombre de fautes de lien et le nombre de paquets injectés dans le réseau. On constate aussi que le réseau sature sensiblement plus vite lorsque le trafic est de type Bit complement en particulier pour un taux de défaillance correspondant à 40%

Fautes de routeur

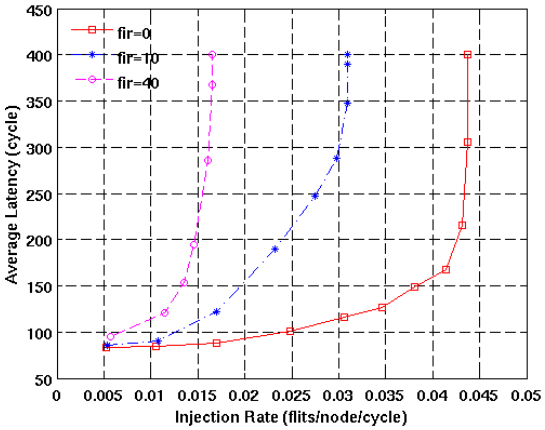
Nous avons réalisé la même série d'expérience mais cette fois-ci en prenant le modèle de fautes de routeur (Figure 4.8). A nouveau, nous avons vu une augmentation de latence moyenne fonction du nombre de fautes et du taux d'injection de paquet. Il est à noter que la latence moyenne augmente ici plus rapidement que dans le cas précédent ou seulement les liens étaient considérés comme défectueux. Cela s'explique par le modèle utilisé pour définir une faute de routeur : une faute de routeur est un routeur avec tous ses liens fautifs. Pour l'algorithme CAFTA, plus il y a de routeurs défaillants, plus il est difficile pour un paquet de trouver un chemin jusqu'à destination. On constate malgré tout qu'avec 40% des routeurs défectueux, le service est quand même rendu mais avec un taux d'injection de paquet réduit de moitié voire des deux tiers selon le type de trafic, par rapport à une situation sans fautes.



(a)



(b)

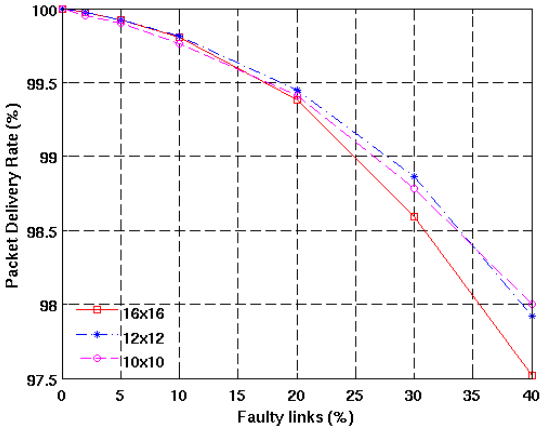


(c)

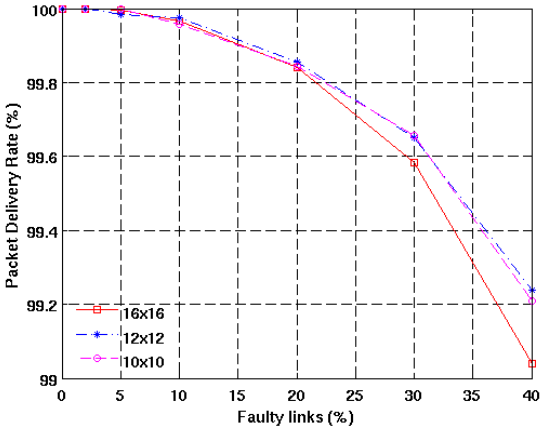
Figure 4.8 Latence moyenne de CAFTA avec différents taux d'injection de fautes (faute de routeurs) sous le trafic (a) « Uniform Random », (b) « Transpose », (c) « Bit complement », pour un NoC 16x16 2D Mesh

4.2.4.2 Taux de livraison de paquets

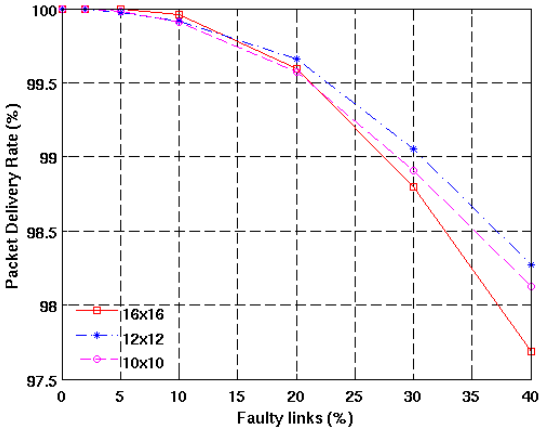
Nous avons voulu ensuite déterminer l'impact du taux de défaillance et de la taille du réseau sur le taux de réussite de livraison de paquets. Pour cela nous avons mesuré le nombre de paquets n'ayant pas atteint leur destination (ou dans un laps de temps trop long) pour trois tailles de NoC et pour différentes situations de défaillance. Les figures 4.9 et 4.10 synthétisent les nombreuses simulations réalisées pour les deux modèles de faute considérés. Nous pouvons voir comme attendu que le pourcentage de paquets livrés avec succès diminue lorsque le taux de défaillance augmente.



(a)

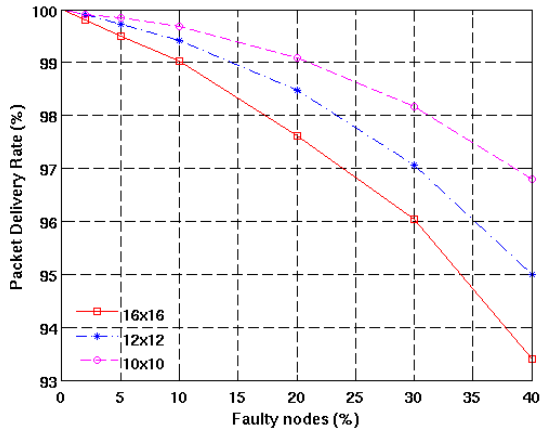


(b)

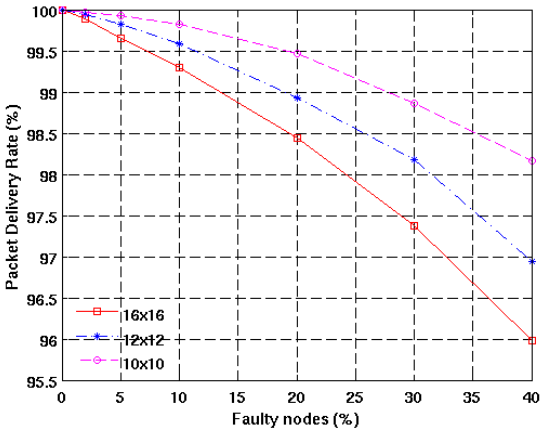


(c)

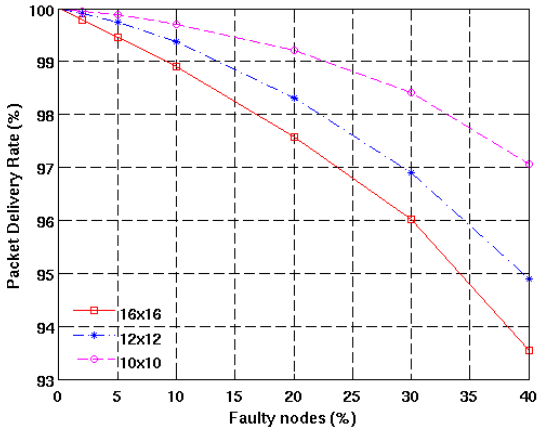
Figure 4.9 Taux de paquets délivrés avec succès pour différentes tailles de réseau et différents taux d'injection de faute (faute de liens) sous le trafic (a) « Uniform Random », (b) « Transpose », (c) « Bit complement », pour un NoC 2D Mesh



(a)



(b)



(c)

Figure 4.10 Taux de paquets délivrés avec succès pour différentes tailles de réseau et différents taux d'injection de fautes (faute de nœud) sous le trafic (a) « Uniform Random », (b) « Transpose », (c) « Bit complement », pour un NoC 2D Mesh

On voit que CAFTA permet de garantir un niveau de fiabilité élevé. Dans le cas d'un NoC 16x16, CAFTA permet de délivrer plus de 99,8% des messages lorsque 10 % des liens sont défectueux (99 fautes de lien simultanées). Avec 40% des liens défectueux, le taux de livraison avec succès varie selon le type de trafic de 97,5 à 99 %.

Lorsque l'on considère les fautes de nœud (routeur), le taux de paquets livrés avec succès décroit, comme on peut s'y attendre étant donné l'impact plus important d'une faute de routeur. Néanmoins, même dans le scénario où 40% de routeurs sont défectueux, le taux de livraison atteint 93,40% pour le trafic « Uniform Random » (Figure 4.10.a), 95,98% pour le trafic « Transpose » (Figure 4.10.b), et 93,53% pour le trafic « Bit complement» (Figure 4.10.c).

4.2.5 Performance de l'algorithme de routage CAFTA en tore 2D

Pour avoir une idée de l'influence de la topologie sur notre technique, nous avons décidé de l'appliquer à un réseau en tore 2D.

Nous avons ainsi réalisé les mêmes simulations pour le NoC en Tore que précédemment. Les figures disponibles en annexe, nous permettent de montrer que l'extension de CAFTA pour ce type de réseau est très intéressante au niveau traitement de la congestion et de la tolérance aux fautes, que ce soit pour des fautes de lien ou de routeur.

Pour aller plus loin nous avons comparé les taux de livraison associés aux deux topologies pour trois tailles de NoC. La figure 4.11 montre que pour un réseau de taille 16x16 où 40% des liens sont défectueux, le taux de livraison de paquets pour le réseau 2D est de

97,52% contre 99,6% pour le tore (20 paquets sur 5000 n'arrivent pas à destination). Ce résultat bien qu'attendu montre l'impact de la topologie sur la fiabilité du réseau. La plus grande connectivité du tore offre à l'algorithme CAFTA plus de solutions de routage.

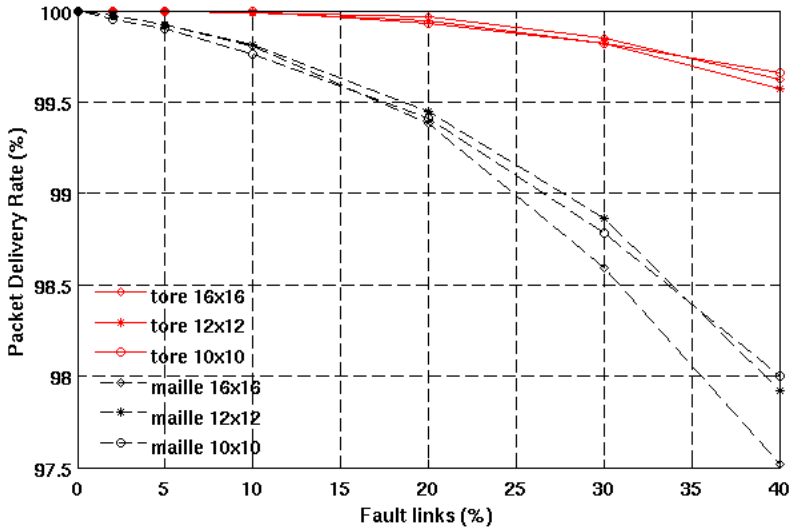


Figure 4.11 Comparaison de la fiabilité entre le tore et la maille sous le trafic « Uniform Random»

Globalement les différentes simulations nous montrent qu'une topologie tore est plus performante qu'une topologie 2D Mesh que ce soit au niveau de la latence moyenne ou du taux de livraison de message. Ainsi, si on vise une application où la tolérance à de très nombreuses fautes est prioritaire, l'utilisation d'un NoC en topologie tore est alors recommandée. Cela se fera néanmoins au prix de la surface additionnelle principalement due aux liens enveloppants.

4.3 Amélioration de la métrique de congestion

Dans le chapitre précédent, nous avons présenté une amélioration de la métrique de congestion proposée, FR* qui combine la métrique «FR» et l'occupation des tampons avals. Nous avons intégré cette nouvelle métrique et refait les simulations de la section 4.2.1 pour mesurer les effets de l'optimisation de cette nouvelle métrique.

Les résultats de simulation présentés dans la figure 4.12 nous montrent que FR* permet de réduire de manière significative la latence moyenne par rapport à FR :

- pour le trafic « Uniform Random », la métrique FR* présente à la pointe de saturation une amélioration de latence moyenne de 5.0% par rapport FR.

- pour le trafic « Transpose », la métrique FR présente à la pointe de saturation une amélioration de latence moyenne de 44,7% % par rapport FR.

- pour le trafic « Bit Complement », la métrique FR* présente à la pointe de saturation une amélioration de latence moyenne de 28,6% par rapport FR.

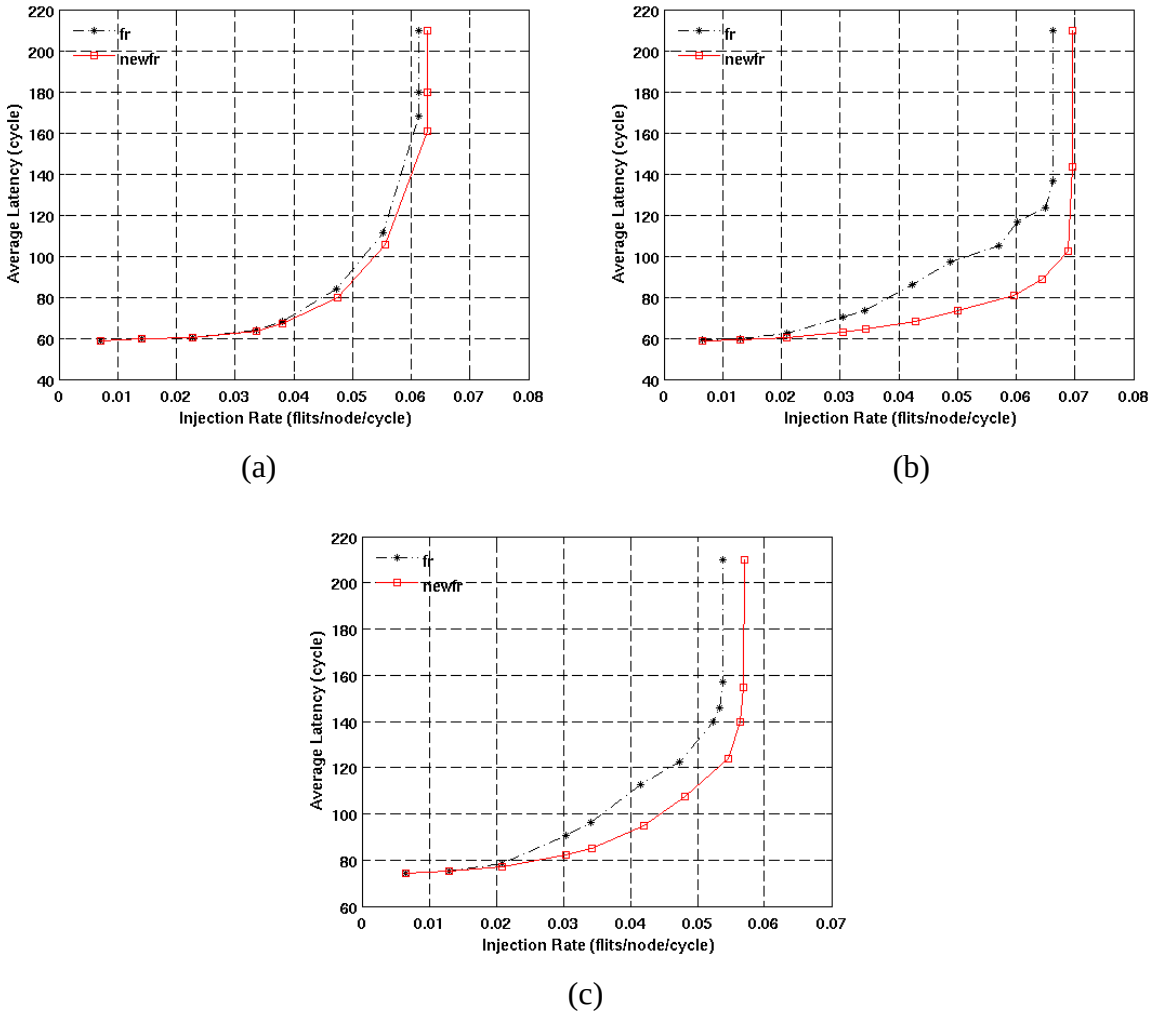


Figure 4.12 Latence moyenne avec différentes métriques de congestion sous le trafic (a) « Uniform Random », (b) « Transpose» et (c) pour un NoC 16x16 sans injection de fautes

Ces différences s'expliquent par le fait que FR* permet de donner une vision plus large, plus complète et donc plus représentative que FR de l'état de charge d'un canal.

FR* est donc beaucoup plus intéressante à prendre en compte que FR dans la gestion de la congestion, cependant son implémentation nécessitera un peu plus de surface additionnelle.

4.4 Conclusions

Dans ce chapitre nous avons dans un premier temps présenté la plateforme d'expérimentation utilisée, le type de simulations réalisées et les caractéristiques des campagnes d'injection de fautes menées.

Les très nombreuses simulations réalisées nous ont permis de valider notre algorithme CAFTA pour les NoC de type 2D mesh et son extension pour les NoC à topologie 2D Tore, en particulier le mécanisme qui permet d'éviter les « dead locks » et « live-locks » décrits dans le chapitre précédent.

Les résultats nous ont permis de montrer que la nouvelle métrique FR (Flit Remain) et sa version améliorée FR* sont plus représentatives de l'état de charge du réseau que les métriques utilisées jusqu'ici. L'intégration de FR* dans CAFTA permet d'avoir une gestion de la congestion beaucoup plus efficace (réduction de la latence moyenne) et ainsi repousser le seuil de saturation du réseau d'au moins 4,8 % par rapport à FR.

Pour ce qui est de la tolérance aux fautes, nous avons comparé les latences moyennes et les taux de livraison de paquets pour des NoCs intégrant CAFTA et VariantB. Sans surprise, les résultats de simulation montrent que CAFTA permet d'améliorer les deux métriques. CAFTA intègre un mécanisme de gestion des fautes dynamiques (fragmentation & fusion de paquets) qui le différencie aussi de VariantB. Ce mécanisme supplémentaire de tolérance aux fautes permet à CAFTA de garantir un niveau de fiabilité élevé : par exemple pour un NoC en maille 2D 16x16, 97,68% des paquets arrivent à destination lorsque 40% des liens sont défaillants simultanément et 93,4 % lorsque 40% des routeurs sont défaillants. La fiabilité est améliorée pour CAFTA appliqué à un NoC de type Tore 2D : dans les mêmes conditions 99.4% de taux de réussite avec 40% de liens défectueux et 95,2% avec 40% des routeurs défectueux.

Cependant, notre approche connait quelques limitations :

- Les modèles de faute ne sont pas assez précis. Par exemple, dans la réalité, une faute peut affecter un ou plusieurs bits d'un lien sans entrainer que tout le lien soit considéré comme défectueux.
- L'implémentation de notre approche à travers des modèles écrits en C/C++ ne nous permet pas d'avoir une estimation précise de la surface et de la consommation de l'architecture proposée.

- Les résultats concernant le taux de livraison avec succès doivent être affinés car dans nos simulations certains messages sont rejetés et considérés perdus s'ils arrivent à destination après Y cycles, Y étant un paramètre de nos simulations. On peut aussi affiner ce mécanisme par un « Time out » matériel et/ou logiciel dans lequel les nœuds envoient des messages de type « i'm alive » avec un nombre de cycles d'attente paramétrables.

Chapitre 5 Conclusions et perspectives

Dans ce chapitre, nous présentons un résumé des conclusions de cette thèse ainsi qu'une discussion sur les perspectives à ce travail.

5.1 Conclusions générales

Les progrès dans les technologies à base de semi-conducteurs et la demande croissante de puissance de calcul poussent vers une intégration dans une même puce de plus en plus de processeurs intégrés. Par conséquent les réseaux sur puce remplacent progressivement les bus de communication, ceux-ci offrant plus de débit et permettant une mise à l'échelle simplifiée.

Parallèlement, la réduction de la finesse de gravure entraine une augmentation de la sensibilité des circuits au processus de fabrication et à son environnement d'utilisation. Les défauts de fabrication et le taux de défaillance pendant la durée de vie du circuit augmentent lorsque l'on passe d'une technologie à une autre. Intégrer des techniques de tolérance aux fautes dans un circuit devient indispensable, en particulier pour les circuits évoluant dans un environnement très sensible (aérospatial, automobile, santé, …).

Nous avons présenté dans ce travail de thèse, des techniques permettant d'améliorer la tolérance aux fautes des micro-réseaux intégrés dans des circuits évoluant dans un environnement difficile. Le NoC doit ainsi être capable de s'affranchir de la présence de nombreuses fautes. Les travaux publiés jusqu'ici proposaient des solutions pour un seul type de faute.

En considérant les contraintes de surface et de consommation du domaine de l'embarqué, nous avons proposé un algorithme de routage adaptatif tolérant à la fois les fautes intermittentes, transitoires et permanentes. En combinant et adaptant des techniques existantes de retransmission de flits, de fragmentation et de regroupement de paquet, notre approche permet de s'affranchir de nombreuses fautes statiques et dynamiques. Les très nombreuses simulations réalisées ont permis de montrer entre autre que, l'algorithme proposé permet d'atteindre un taux de livraison de paquets de 97,68% pour un NoC 16x16 en maille 2D en présence de 384 liens défectueux simultanés, et 93,40% lorsque 103 routeurs sont défaillants. Nous avons étendu l'algorithme aux topologies de type tore. Dans ce cas le taux de succès

atteint 99.4% pour un NoC en tore 2D 16x16 en présence de 410 liens défectueux, et 95.2% lorsque 103 routeurs sont inutilisables.

Une autre originalité de cette thèse est que nous avons inclus dans cet algorithme une fonction de gestion de la congestion. Pour cela nous avons défini une nouvelle métrique de mesure de la congestion (Flit Remain) plus pertinente que les métriques utilisées et publiées jusqu'ici. Les expériences ont montré que l'utilisation de cette métrique permet de réduire la latence (pic de saturation) de 2,5 % à 16,1 %, selon le type de trafic généré, par rapport à la plus efficace des métriques existante (Free VC). Une optimisation de la métrique FR, FR*, a été proposée, elle permet d'atteindre des réductions de latence allant de 5% à 45 % par rapport à l'utilisation de FR.

La combinaison du routage adaptatif tolérant les fautes statiques et dynamiques et la gestion de la congestion offrent une solution qui permet d'avoir un NoC et par extension un circuit beaucoup plus résilient. Néanmoins cette solution peut être complétée par des techniques de tolérance aux fautes au niveau circuit, au niveau système/logiciel, si la criticité de l'application finale l'exige.

5.2 Perspectives

Ce travail a pour vocation à être complété, en effet de nombreux points sont à préciser, à développer à court et moyen terme.

A court terme

- Evaluation de la surface et de la consommation

Les travaux de modélisation du NoC au niveau matériel sont en cours. La synthèse va nous permettre d'estimer de manière précise le surcout en silicium et en puissance consommée induits par l'ajout de nos fonctionnalités de tolérance aux fautes et de gestion de congestion.

- Simulations avec d'autres trafics issus d'applications réelles

Les simulations ont été réalisées pour trois types de trafic : « Uniform Random », « Transpose » et « Bit Complement ». Pour vérifier dans un contexte plus industriel la validité de notre approche, il serait intéressant de réaliser ces simulations avec des trafics issus d'applications réelles, de type « benchmark ».

- 3D

  L’évolution de la technologie permet depuis quelques années de réaliser des circuits en trois dimensions, constitués d’un empilement de puces interconnectées entre elles par des interconnexions appelées TSV (Through Silicon Via). Cette tendance se confirme et on voit très bien l’intérêt d’avoir à disposition des NoC 3D. Nous pensons que notre approche est facilement extensible à ce type de NoC, le routeur serait sensiblement le même mais avec deux ports verticaux. L'extension à la 3D de ce travail sera aussi facilitée par le fait que des travaux sur le routage 3D tolérant aux fautes ont déjà été menés dans l’équipe [Rusu 2011].

A moyen et long terme

- Modèles de fautes grain plus fin

Les modèles de faute sur lesquelles se basent nos solutions, sont des modèles gros grains : faute de lien ou faute de routeur. Un modèle plus réaliste et moins radical serait de considérer au niveau du lien une faute sur un ou plusieurs bits. La désactivation complète du lien ne serait pas forcément nécessaire. Pour le routeur on pourrait là aussi envisager un modèle de faute plus fin, associé à un bloc par exemple.

- Combiner avec ECC

En ayant ces modèles de faute à grain plus fin, il serait intéressant de combiner la nouvelle approche avec les techniques de tolérance aux fautes de type codes correcteurs d’erreurs (ECC), et mesurer le gain global en terme de fiabilité.

- Intégration du NoC dans une plateforme de simulation plus large de type SST

Enfin au plus long terme, il est envisagé d’inclure ce NoC dans une plateforme massivement multiprocesseurs et d’y intégrer aussi l’ensemble des mécanismes d’auto-détection de fautes et d’autoréparation, que ce soit par matériel ou par logiciel et cela à plusieurs niveaux. Ce travail s’inscrit dans un projet de plus grande envergure.

Bibliographie

[Agarwal 2009] N. Agarwal, T. Krishna, L.-S. Peh, and N. K. Jah, "GARNET: A detailed on-chip network model inside a full-system simulator," in Proceedings of the IEEE Symposium on Performance Analysis of Systems and Software, 2009

[Aguilera 1997] M. Aguilera, W. Chen, S. Toueg, "Heartbeat: a Timeout-Free Failure Detector for Quiescent Reliable Communication," Technical Report 97-1631, Department of Computer Science, Cornell University, May 1997.

[Ascia 2008] G. Ascia, V. Catania, M. Palesi, and D. Patti, "Implementation and analysis of a new selection strategy for adaptive routing in networks-on-chip,. Computers, IEEE Transactions on, 57(6):809 –820, June 2008.

[Austin 1999] T. M. Austin. “DIVA: A reliable substrate for deep submicron microarchitecture design”. In MICRO-32, pages 196-207, 1999.

[Avizienis 2004] A. Avizienis, J. Laprie, B. Randell, C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," IEEE Trans. Dependable Secur. Comput. 1 (1) (2004) 11–33.

[Avresky 1999] D. R. Avresky, C. Cunnungham, and H. Ravichandran, "Fault-tolerant adaptive routing for two-dimensional meshes," Int. Journal of Computer Systems Science and Engineering, vol. 14, no. 6, november 1999.

[Baydal 2005] E. Baydal et al., "A family of mechanisms for congestion control in wormhole networks, " IEEE Trans. on Dist. Systems, 16, 2005.

[Beigne 2005] E. Beigne, F. Clermidy, P. Vivet, A. Clouard, and M. Renaudin, "An asynchronous NOC architecture providing low latency service and its multi-level design framework, " In Proceedings of the 11th International

Symposium on Asynchronous Circuits and Systems (ASYNC). IEEE, 54–63, 2005.

[Benini 2002] L. Benini and G.D. Micheli (2002). "Networks on Chips: A New SoC Paradigm," Computer, 35 :70–78. 2002

[Benso 2003] A. Benso and P. Prinetto, Eds., "Fault Injection Techniques and Tools for Embedded Systems Reliability Evaluation," 1st ed. Boston: Springer-Verlag, Oct. 2003.

[Binkert 2011] N. Binkert, et al., "The GEM5 Simulator," SIGARCH Computer Architecture News, vol. 39, no. 2, May 2011.

[Chaix 2010] F. Chaix, D. Avresky, N. Zergainoh, and M. Nicolaidis, "Fault-Tolerant Deadlock-Free Adaptive Routing for Any Set of Link and Node Failures in Multi-cores Systems," Proceedings of the 2010 Ninth IEEE International Symposium on Network Computing and Applications (NCA

'10), Cambridge, Massachusetts, USA, pp. 52-59, July 2010.

[Constantinides 2007] K. Constantinides, O. Mutlu, T. Austin, and V. Bertacco. "Software-based online detection of hardware defects: Mechanisms, architectural support, and evaluation." In Proceedings of the 40th International Symposium on Microarchitecture (MICRO 40), pages 97–108, Dec. 2007

[Cunningham 1995] C. M. Cunningham and D. R. Avresky, "Fault-tolerant adaptive routing for two-dimensional meshes," In HPCA ’95: Proceedings of the 1st IEEE Symposium on High-Performance Computer Architecture, page 122, Washington, DC, USA, 1995. IEEE Computer Society.

[Dally 1986] W. J. Dally and C. L. Seitz, "The torus routing chip," Journal of Distributed Computing, vol. 1, no. 3, pp. 187–196, October 1986.

[Dally 1987] W. J. Dally, C. L. Seitz, "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks," Computers, IEEE Transactions on , vol.C-36, no.5, pp.547,553, May 1987

[Dally 1993] W. J. Dally and H. Aoki, "Deadlock-Free Adaptive Routing in Multicomputer Networks Using Virtual Channels, IEEE Trans. on Parallel and Distributed Systems, vol. 4, no. 4, pp. 466–475, April 1993.

[Dally 1994] W. J. Dally, L. R. Dennison, D. Harris, K. Kan, and T. Xanthopoulos. "The reliable router: A reliable and high-performance communication substrate for parallel computers," Proc. PCRCW, 1994.

[Duato 2003] J. Duato, S. Yalamanchili, L. Ni, "Interconnection networks: an engineering approach," Morgan Kaufmann Publishers, 2003.

[Dally 2004] W. J. Dally and B. Towles, "Principles and Practices of Interconnection Networks," San Mateo, CA: Morgan Kaufmann, 2004.

[DeOrio 2011] A. DeOrio, K. Aisopos, V. Bertacco, L.-S. Peh, "DRAIN: distributed recovery architecture for inaccessible nodes in multi-core chips," in: 48th ACM/EDAC/IEEE Design Automation Conference (DAC), June 2011, pp. 912 917.

[Ebrahimi 2012] M. Ebrahimi, M. Daneshtalab, J. Plosila, H. Tenhunen, "MAFA: Adaptive Fault-Tolerant Routing Algorithm for Networks-on-Chip," DSD 2012, pp. 201-207, 2012.

[Felicijan 2004] T. Felicijan, and S. B. Furber, "An asynchronous on-chip network router with quality-of-service (QoS) support," In Proceedings IEEE International SOC Conference. IEEE, 274–277, 2004.

[Feng 2013] C. Feng, Z. Lu, A. Jantsch, M. Zhang, Z. Xing, "Addressing Transient and Permanent Faults in NoC With Efficient Fault-Tolerant Deflection Router," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol.21, no.6, pp. 1053-1066, June 2013.

[Flich 1999] J. Flich, M. P. Malumbres, P. L´opez and J. Duato, "Performance Evaluation of Networks of Workstations with Hardware Shared Memory Model Using Execution-Driven Simulation”, Proc. Int. Conf. Parallel Processing, Sep. 1999.

[Gerla 1980] M. Gerla and L. Kleinrock, "Flow Control: A Comparative Survey," IEEE Trans. on Communications, 28(4), April 1980.

[Gratz 2008] P. Gratz, B. Grot and S. W. Keckler, "Regional Congestion Awareness for Load Balance in Networks on Chip," In International Symposium on High Performance Computer Architectures (HPCA), pp. 203-214, 2008.

[Glass 1992] C. Glass, L. Ni, "The turn model for adaptive routing," in Proceedings of the 19th annual international symposium on Computer architecture (ISCA '92), New York, NY, USA, pp. 278-287, 1992.

[Hamming 1950] R. Hamming, "Error detecting and error correcting codes," Bell System Technical Journal, Vol. 29, 1950, pp. 147 160.

[Hyatt 1997] C. Hyatt and D. P. Agrawal, "Congestion Control in the Wormhole Routed Torus With Clustering and Delayed Detection," Proc. Parallel Computing, Routing, and Communication Workshop, June 1997.

[ITRS 2011] ITRS. International Technology Roadmap for Semiconductors. In Interconnect, 2011.
http://www.itrs.net.

[ITRS 2011a] ITRS. International Technology Roadmap for Semiconductors. In System Drivers, 2011.
http://www.itrs.net.

[Jacobson 1988] V. Jacobson, "Congestion Avoidance and Control," Proc. ACM SIGCOMM '88 Conf., 1988.

[Jain 1992] R. Jain, "Myths About Congestion Management in High-Speed Networks," Internetworking: Research and Experience, Vol. 3, No. 3, 1992.

[Jiang 2010] N. Jiang, G. Michelogiannakis, D. Becker, B. Towles, and W. J. Dally, "Booksim 2.0 User's Guide," Stanford University, March 2010

[Jiang 2013] N. Jiang, D. U. Becker, G. Michelogiannakis, J. Balfour, B. Towles, J. Kim and W. J. Dally, "A Detailed and Flexible Cycle-Accurate Network-on-Chip Simulator," In Proceedings of the 2013 IEEE International Symposium on Performance Analysis of Systems and Software, 2013.

[Kariniemi 2004] H. Kariniemi, J. Nurmi, "Arbitration and Routing Schemes for On-chip Packet Networks," Interconnect-Centric Design for Advanced SoC and NoC, Kluwer Academic Publishers, 2004, pages: 253–282.

[Kang 2009] Y. H. Kang, J. Sondeen, and J. Draper. "Multicast Routing with Dynamic Packet Fragmentation," In Proc. of Great Lakes Symposium on VLSI, May 2009.

[Kang 2010] Y. Kang, T. Kwon, J. Draper, "Fault-tolerant flow control in on-chip networks," in Proceedings of the 2010 Fourth ACM/IEEE International Symposium on Networks-on-Chip (NOCS '10), Washington, DC, USA, 2010, pp. 79–86.

[Kim 2005] J. Kim, D. Park, T. Theocharides, N. Vijaykrishnan, and C. R. Das, "A Low Latency Router Supporting Adaptivity for On-Chip Interconnects," In International Conference on Design Automation, pages 559–564,

2005.

[Knuth 1998] D. E. Knuth, "The Art of Computer Programming Volumes 1-3," Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 1998

[Latif 2013] K. Latif, A.-M. Rahmani, E. Nigussie, T. Seceleanu, M. Radetzki, H. Tenhunen, "Partial virtual channel sharing: a generic methodology to enhance resource management and fault tolerance in networks-on-chip," J. Electr. Test. Theor. Appl. 29 (3) (2013) 431–452.

[Lyons 1962] R. E. Lyons, W. Vanderkulk, "The Use of Triple-Modular Redundancy to Improve Computer Reliability," IBM Journal of Research and Development , vol.6, no.2, pp.200,209, April 1962

[Ma 2011] S. Ma, N. Jerger, Z. Wang, "DBAR: an efficient routing algorithm to support multiple concurrent applications in networks-on-chip," in Proceedings of the 38th annual international symposium on Computer architecture (ISCA '11), pp. 413-424, 2011.

[Martinez 1999] J.F. Martinez, J. Torrellas and J. Duato, "Improving the Performance of Bristled CC-NUMA Systems using Virtual Channels and Adaptively," Proc. ACM Int. Conf. on Supercomputing, June 1999.

[Miro-Panades 2006] I. Miro-Panades, A. Greiner, A. Sheibanyrad, "A Low Cost Network-on-Chip with Guaranteed Service Well Suited to the GALS Approach," Nano-Networks and Workshops, 2006. NanoNet '06. 1st International Conference on, vol., no., pp.1,5, Sept. 2006.

[Nicolaidis 2007] M. Nicolaidis, "GRAAL: a new fault tolerant design paradigm for mitigating the flaws of deep nanometric technologies," IEEE Internatinal Test Conference, Santa Clara, CA, pp.1-10, 2007.

[Nicolaidis 2011] M. Nicolaidis, V. Pasca, L. Anghel. "I-BIRAS: Interconnect Built-In Self-Repair and Adaptive Serialization for Inter-Die Communication in 3D Integrated Systems," Proceedings of European Test Symposium, ETS 2011, Trondheim, Norway.

[Pasca 2012] V. Pasca, S.-U. Rehman, L. Anghel, M. Benabdenbi, "Efficient Link-level Error Resilience in 3D NoCs", Proceedings of International Symposyum on Designs and Diagnosis of Electronic Circuits and Systems (DDECS 2012), pp. 135-140, 2012.

[Pasca 2013] V. Pasca, "Développement d'Architectures HW / SW Tolérantes aux Fautes et Auto-calibrantes pour les Technologies Intégrées 3D", Phd. thesis, Université de Grenoble, 2013.

[Pullum 2001] L. L. Pullum, "Software Fault Tolerance Techniques and Implementation," Norwood, MA: Artech House, 2001.

[Rodrigues 2011] A. F. Rodrigues, et al., "The structural simulation toolkit," SIGMETRICS Perform. Eval. Rev., vol. 38, no. 4, pp. 37–42, 2011.

[Rostislav 1995] D. Rostislav, V. Vishnyakov, E. Friedman, and R. Ginosaur, "An asynchronous router for multiple service levels networks on chip," In Proceedings of the 11th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC). IEEE, 44–53, 2005

[Rusu 2010] C. Rusu, L. Anghel, "Checkpoint and rollback recovery in network-on-chip based systems," Student forum at ASP-DAC 2010, Taipei, Taiwan.

[Rusu 2011] C. Rusu, L. Anghel, D. Avresky, "Adaptive Inter-Layer Message Routing in 3D Networks-on-Chip," Microprocessors and Microsystems Journal 2011.

[Siewiorek 1998] D. P. Siewiorek and R. S. Swarz. "Reliable computer systems: Design and evaluation," 3rd edition. AK Peters, Ltd, 1998.

[Silla 1998] F. Silla, M.P. Malumbres, J. Duato, D. Dai, and D.K. Panda, "Impact of Adaptively on the Behavior of Networks of Workstations under Bursty Traffic," Proc. Int. Conf. on Parallel Processing, August 1998.

[Singh 2003] A. Singh,W. J. Dally, A. K. Gupta, and B. Towles. "GOAL: A Load-Balanced Adaptive Routing Algorithm for Torus Networks," In International Symposium on Computer Architecture, pages 194–205, 2003.

[Smai 1998] A. Smai and L. Thorelli, "Global Reactive Congestion Control in Multicomputer Networks", Proc. Int. Conf. on High Performance Computing, 1998.

[Thottethodi 2001] M. Thottethodi, A. R. Lebeck, and S. S. Mukherjee. "Self-Tuned Congestion Control for Multiprocessor Networks," In International Symposium on High-Performance Computer Architecture, pages 107–118, 2001.

[Towles 2003] B. Towles, W. J. Dally, and S. Boyd. "Throughput-centric Routing Algorithm Design," In Symposium on Parallel Algorithms and Architectures, pages 200–209, 2003.

[Tsai 2011] W. Tsai, D. Zheng, S. Chen, Y. Hu, "A Fault-Tolerant NoC Scheme using bidirectional channel," 48th ACM/EDAC/IEEE Design Automation Conference (DAC), pp. 918-923, June 2011.

[Wang 2014] J. Wang, J. Beu, R. Bheda, T. Conte, Z. Dong, C. Kersey, M. Rasquinha, G. Riley, W. Song, H. Xiao, P. Xu, and S. Yalamanchili, "Manifold: A Parallel Simulation Framework for Multicore Systems," 2014 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), March 2014.

[Weiser 1994] M. Weiser, B. Welch, A. Ddmers, and S. Shenker, "Scheduling for reduced CPU energy," In 1st OSDI (Monterey, CA, USA, Nov 1994), pp. 13–23.

[Web Iris NoC] http://iris-casl.github.io/

[Web Networkonchip] https://networkonchip.wordpress.com/

[Web SystemC] http://www.systemc.org/home/

[Yang 1995] C. Yang et al. "A taxonomy for congestion control algorithms in packet switching networks," IEEE Network, 9, 1995.

[Yu 2012] Q. Yu, J. Cano, J. Flich and P. Ampadu, "Transient and permanent error control for high-end multiprocessor systems-on-chip," Proc. 6th ACM/IEEE Intl. Symp. on Networks-on-Chip, pp. 169-171, May 2012.

[Zhang 2008] Z. Zhang, A. Greiner, and S. Taktak, “A reconfigurable routing algorithm for a fault-tolerant 2-D-mesh network-on-chip,” in Proc. Design Autom. Conf. (DAC), 2008, pp. 441–446.

[Zhang 2010] Z. Zhang, A. Greiner and M. Benabdenbi. "Fully Distributed Initialization Procedure for a 2D-Mesh NoC, Including Off Line BIST and Partial Deactivation of Faulty Components," In Proceedings of the 16th IEEE International On-Line Testing Symposium (IOLTS’10) pages 194-196, Greece, 2010.

[Ziegler 1979] J. F. Ziegler and W. A. Lanford, "Effect of cosmic rays on computer memories," Science, 206: 776-788, Nov. 1979.

[Ziegler 1981] J. F. Ziegler and W. A. Lanford, "The effect of sea level cosmic rays on electronic devices," J. App. Phys., 52(6): 4305-4311, June 1981.

[Ziegler 1996] J. F. Ziegler et al., "IBM experiments in soft fails in computer electronics (1978-1994)," IBM J. Res. Dev., 40(1): 3-18, Jan. 1996.

Annexe A Résultats des simulations pour le Tore 2D

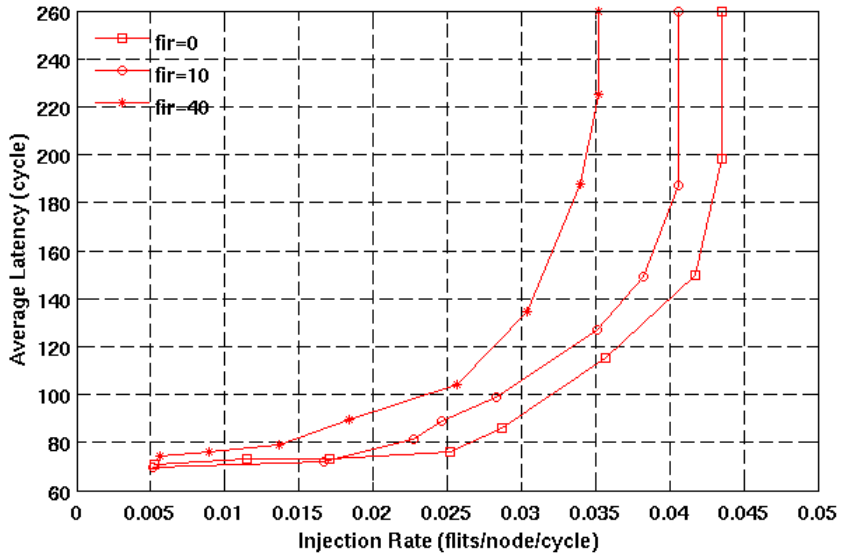


Fig. A.1.a. Latence moyenne de CAFTA avec différentes ratio d'injection de faute (faute de liens) sous le trafic « Uniform Random » pour un NoC 16x16 en tore 2D

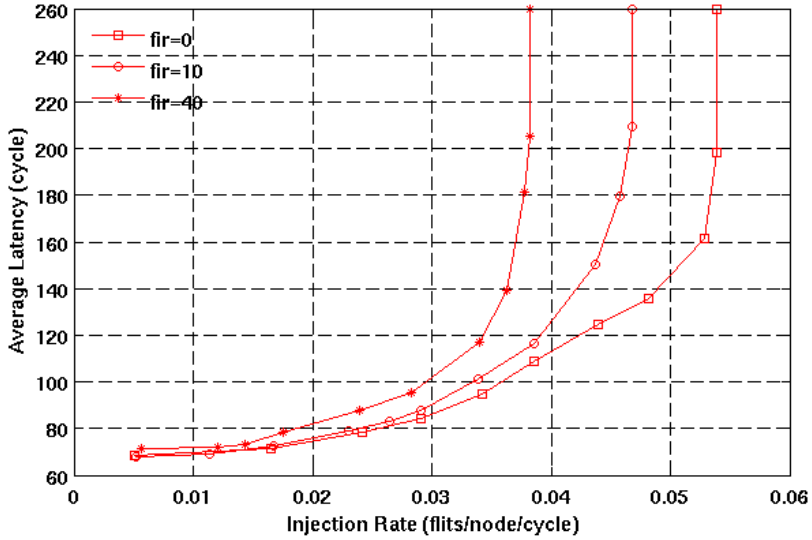


Fig. A.1.b. Latence moyenne de CAFTA avec différentes ratio d'injection de faute (faute de liens) sous le trafic « Transpose » pour un NoC 16x16 en tore 2D

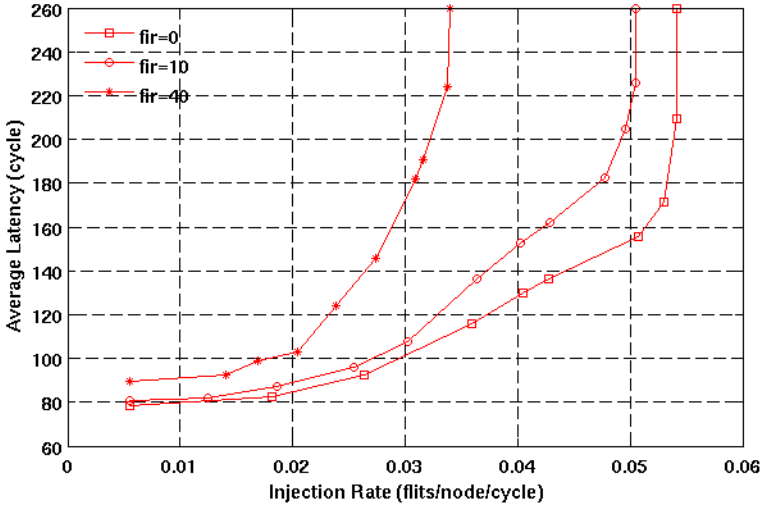


Fig. A.1.c. Latence moyenne de CAFTA avec différentes ratio d'injection de faute (faute de liens) sous le trafic « Bit complement» pour un NoC 16x16 en tore 2D

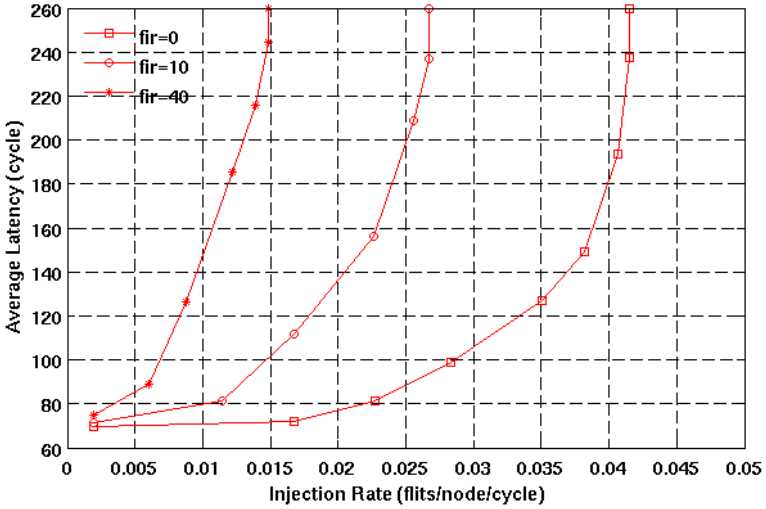


Fig. A.2.a. Latence moyenne de CAFTA avec différentes ratio d'injection de faute (faute de routeurs) sous le trafic « Uniform Random» pour un NoC 16x16 en tore 2D

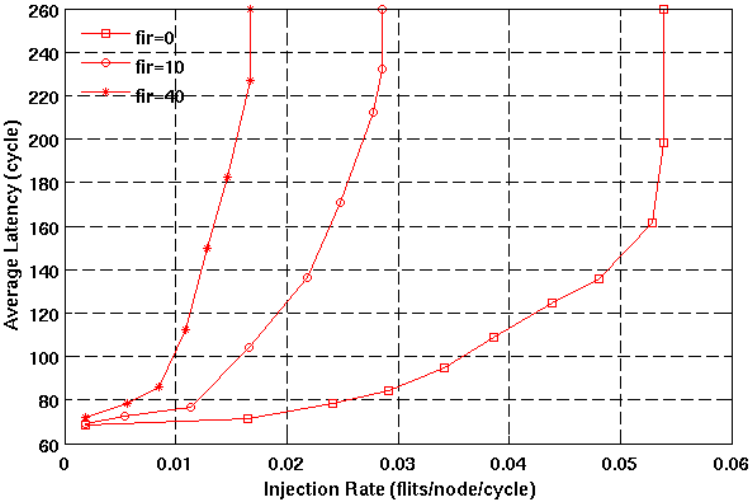


Fig. A.2.b. Latence moyenne de CAFTA avec différentes ratio d'injection de faute (faute de routeurs) sous le trafic « Transpose» pour un NoC 16x16 en tore 2D

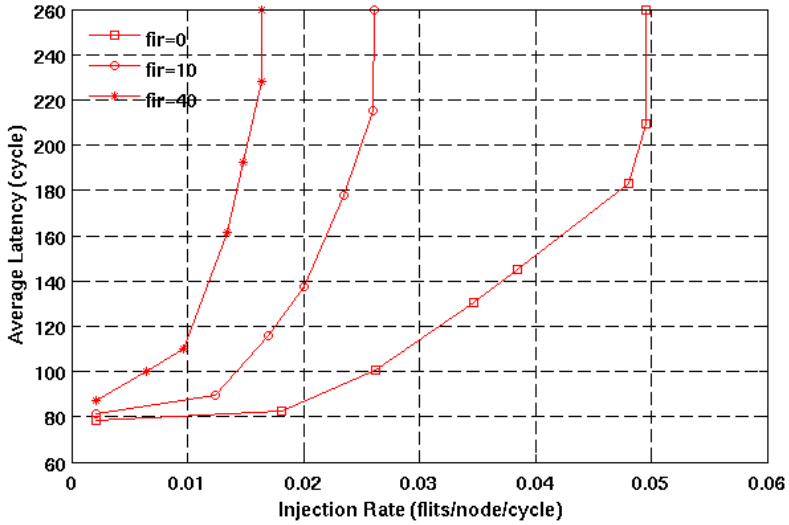


Fig. A.2.c. Latence moyenne de CAFTA avec différentes ratio d'injection de faute (faute de routeurs) sous le trafic « Bit complement» pour un NoC 16x16 en tore 2D

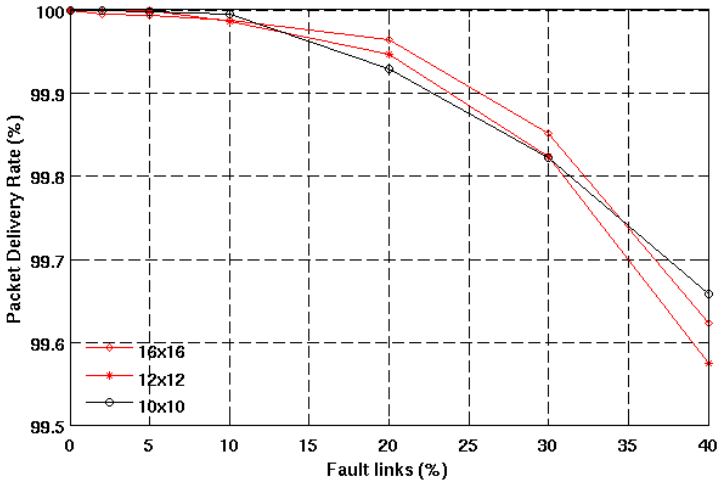


Fig. A.3.a. Ratio de paquets délivrés avec succès avec différentes taille de réseau et différents ratios d'injection de faute (faute de liens) sous le trafic « Uniform Random» pour un NoC 16x16 en tore 2D

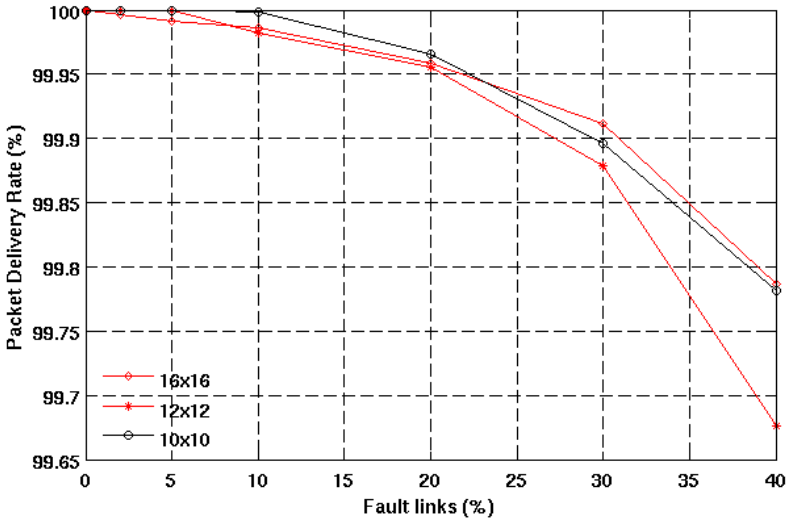


Fig. A.3.b. Ratio de paquets délivrés avec succès avec différentes taille de réseau et différents ratios d'injection de faute (faute de liens) sous le trafic « Transpose» pour un NoC 16x16 en tore 2D

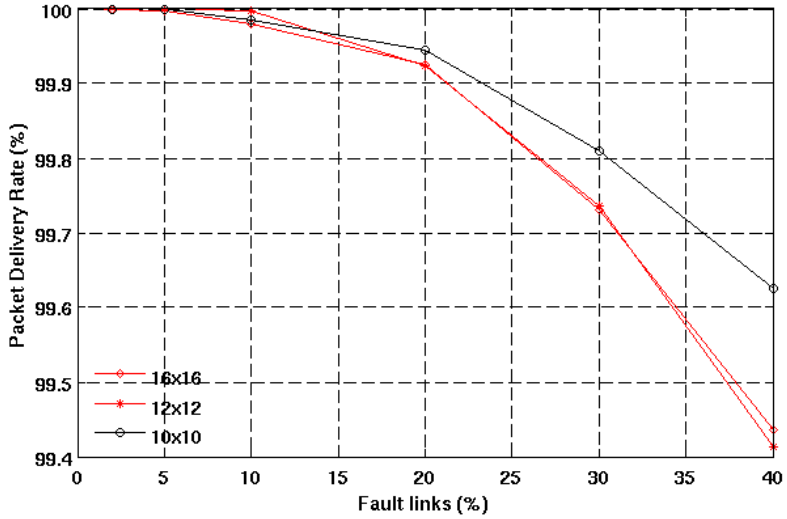


Fig. A.3.c. Ratio de paquets délivrés avec succès avec différentes taille de réseau et différents ratios d'injection de faute (faute de liens) sous le trafic « Bit complement» pour un NoC 16x16 en tore 2D

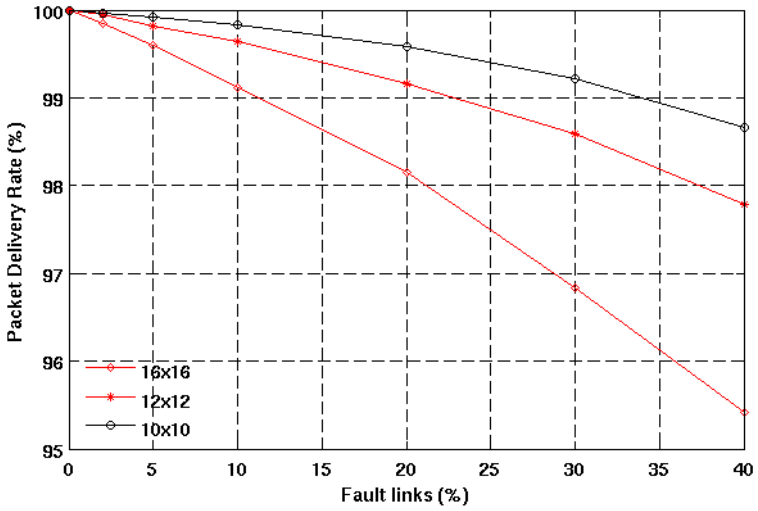


Fig. A.4.a. Ratio de paquets délivrés avec succès avec différentes taille de réseau et différents ratios d'injection de faute (faute de nœud) sous le trafic « Uniform Random» pour un NoC 16x16 en tore 2D

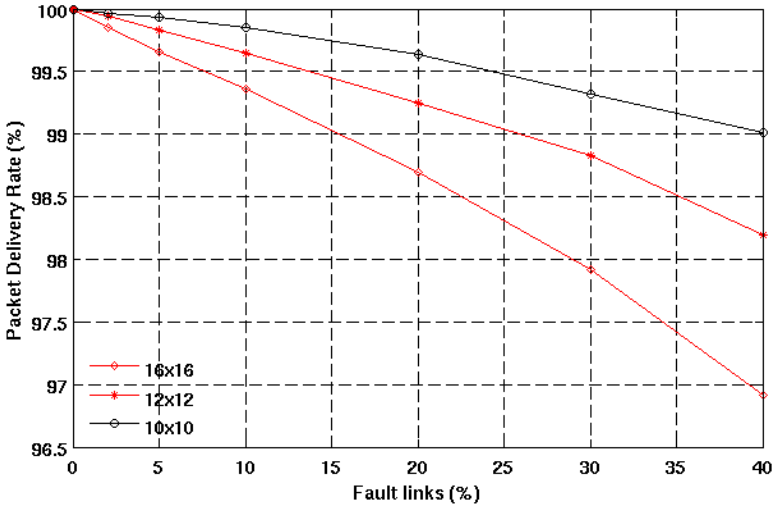


Fig. A.4.b. Ratio de paquets délivrés avec succès avec différentes taille de réseau et différents ratios d'injection de faute (faute de nœud) sous le trafic « Transpose» pour un NoC 16x16 en tore 2D

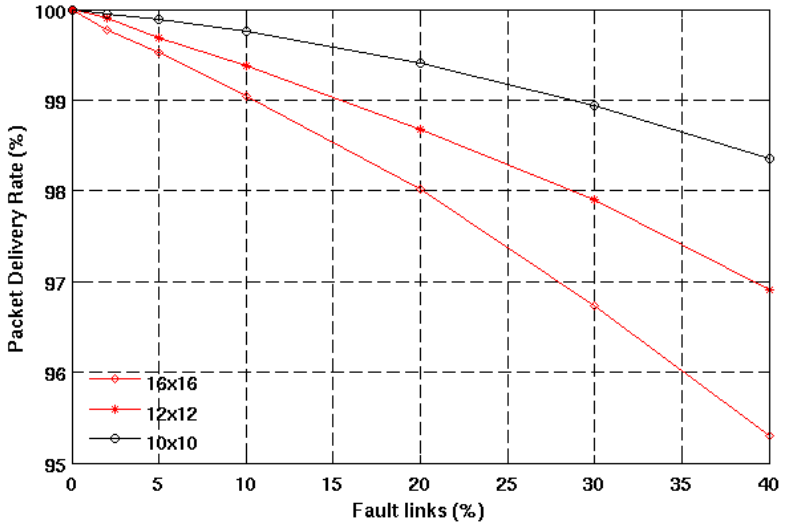


Fig. A.4.c. Ratio de paquets délivrés avec succès avec différentes taille de réseau et différents ratios d'injection de faute (faute de nœud) sous le trafic « Bit complement» pour un NoC 16x16 en tore 2D

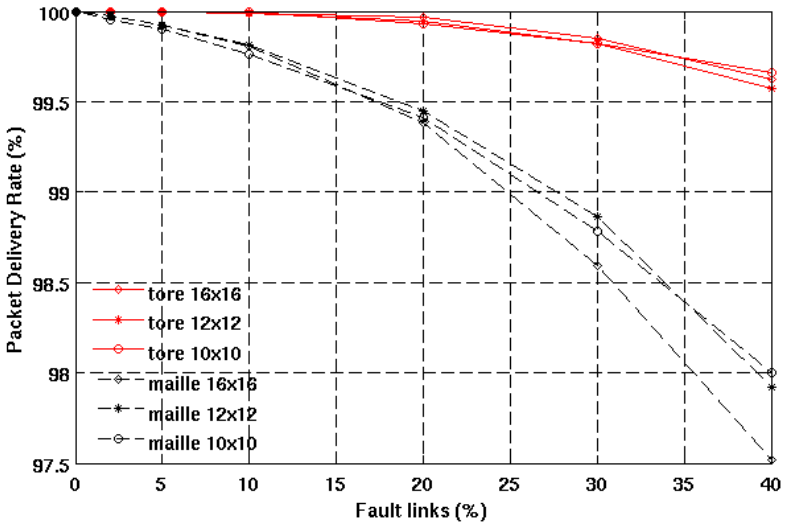


Fig. A.5. Comparaison de la fiabilité entre le tore et la maille sous le trafic « Uniform Random»

Annexe B Publications issues de ce travail de thèse

M. Dimopoulos, Y. Gang, M. Benabdenbi, L. Anghel, N. Zergainoh, M. Nicolaidis, Fault-tolerant adaptive routing under permanent and temporary failures for many-core systems-on-chip, in: IEEE 19th International On-Line Testing Symposium (IOLTS' 2013), Chania, Crete, Greece, 2013, pp. 7–12.

M. Dimopoulos, Y. Gang, M. Benabdenbi, L. Anghel, N. Zergainoh, M. Nicolaidis, Efficient Fault-Tolerant Adaptive Routing under an unconstrained Set of Node and Link Failures for Many Cores System On Chip, in Workshop on Dependable Multicore and Transactional Memory Systems (DMTM'14), (joint to HIPEAC event), Jan 2014, Vienna, Austria. pp.1-2, Proceedings

M. Dimopoulos , Y. Gang , L. Anghel , M. Benabdenbi , N. Zergainoh , M. Nicolaidis, Fault-tolerant adaptive routing under an unconstrained set of node and link failures for many-core systems-on-chip, Microprocessors & Microsystems, v.38 n.6, p.620-635, August, 2014

TITRE

Conception d'un micro-réseau intégré NOC tolérant les fautes multiples statiques et dynamiques

RESUME

Le nombre croissant d'éléments à interconnecter dans un système fait que le bus traditionnel n'est plus adéquat et constitue un goulot d'étranglement au niveau des communications, limitant ainsi les performances. Le réseau sur puce (NoC) s'est rapidement imposé comme une solution d'avenir notamment parce qu'il permet d'obtenir une bande passante plus grande et qu'il permet un passage à l'échelle simplifié. Avec une densité d'intégration toujours croissante, la fiabilité des circuits fabriqués devient un enjeu crucial. Ainsi la tolérance aux défauts apparaissant pendant la fabrication ou dans le contexte d'utilisation est désormais indispensable.

Dans cette thèse, nous nous intéressons aux micro-réseaux intégrés. Nous présentons pour ces derniers un algorithme de routage adaptatif tolérant à la fois les fautes intermittentes, transitoires et permanentes. En combinant et adaptant des techniques existantes de retransmission des flits, de fragmentation et de regroupement de paquets, l'approche proposée permet de s'affranchir de nombreuses fautes statiques et dynamiques. De plus, une nouvelle métrique de mesure de la congestion « Flit Remain » est proposée pour améliorer gestion de la congestion et diminuer ainsi la latence moyenne. La combinaison du routage adaptatif tolérant les fautes statiques et dynamiques et la gestion de la congestion offrent une solution qui permet d'avoir un NoC et par extension un circuit beaucoup plus résilient.

MOTS CLEFS
Réseau sur Puce (NoC), MPSoCs, routage adaptif, tolérance aux fautes, contournement de congestion

TITLE

Design of a Network on chip (NoC) that tolerates multiple static and dynamic faults

ABSTRACT

A fast on chip communication fabric is vital to a multiprocessor system on chip (MPSoC). Network on chip becomes a most promising interconnection solution to replace the traditional shared BUS because of its higher performance in terms of throughput and scalability. Meanwhile, as the semiconductor technology scaling down and the number of devices integrated increases, the performance of circuits become more and more susceptible to various factors (manufacturing defects, process variation, wear-out, environmental constraints and so on). Facing the increasing probability of failure, the capacity of fault tolerance becomes mandatory in such NoC based MPSoCs, MPSoCs which may include in the future thousands of cores.

In this thesis, a fault tolerant adaptive routing algorithm has been proposed, which can cope with transient, intermittent and permanent faults. Combined with some existing techniques, like flit retransmission and packet fragmentation, this approach allows tolerating numerous static and dynamic faults. Moreover, the use of a new congestion metric named "Flit Remain" helps decreasing the average latency in the case of heavy traffic.
.

KEYWORDS
Network on Chip (NoC), MPSoCs, adaptive routing, fault tolerance, congestion

INTITULE ET ADRESSE DU LABORATOIRE
Laboratoire TIMA, 46 avenue Félix Viallet, 38031 Grenoble, France.

ISBN : 978-2-11-129203-1