Table des matières

Table des figures

Liste des tableaux

Table des sigles et acronymes

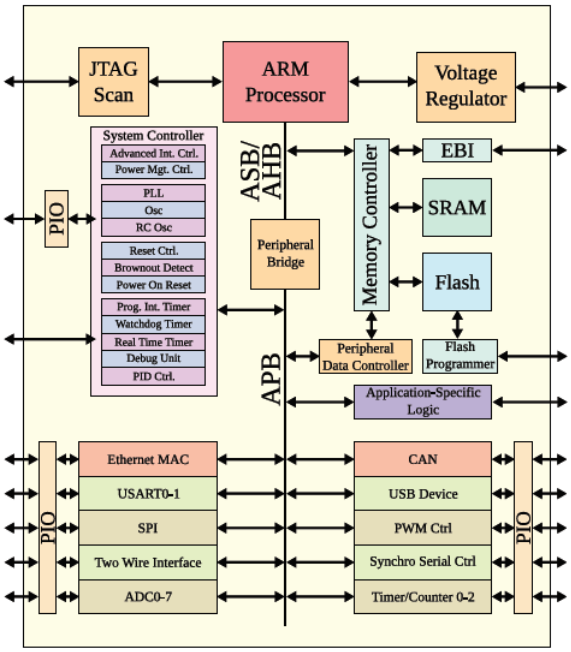
ASIC	Application-Specic Integrated Circuit
FIFO	First In First Out
Flit	Flow control digit
MPSoC	Multiprocessors System on Chip
NACK	Negative Acknowledgement
NI	Network Interface
NoC	Network on Chip
PE	Processing Element
PHit	PHysical unIT
PIR	Packet Injection Rate
FLIT	FLow control unIT
QoS	Quality of Service
IP	Intellectual Property
RC	Routing Computing
SA	Switch Allocation
SAF	Store and Forward
SoC	System on Chip
ST	Switching
VCA	Virtual Channel Allocation
VC	Virtual Channel
VCT	Virtual Cut-Through
WH	WormHole
SN	Sub-network
SSN	Stae Sub-network
ToP	Type of Packets

Introduction Générale

L'évolution technologique a permis d'intégrer plus d'un Milliard de transistors dans le même substrat de silicium, en 2025 la dimension physique de transistor franchira le seuil de 10 nm selon ITRS [1]. En effet , la possibilité d'intégrer un nombre important de transistors dans le même substrat a encouragé les concepteurs des circuits intégrés à intégrer toujours plus de fonctionnalités et d'architectures dans un même circuit afin de réduire le coût de production. Pour maintenir ce niveau rapide on parle aujourd'hui de systèmes sur puce ou System on Chip (SoC). Les composants assemblés dans un SoC varient selon l'application à réaliser. Les SoCs peuvent inclure différents blocs fonctionnels réutilisables tels que :

— Des éléments de traitements (Microprocesseurs, Microcontrôleurs, DSPs) ;
— des blocs mémoires (RAM, ROM, Flash) ;
— des structures d'interconnexions (Bus, Crossbar, NoC) ;
— des blocs d'entrées/sorties (USB, Ethernet) ;
— des convertisseurs analogique/numérique ;
— des interfaces externes (IP).

FIGURE 1 – Architecture d'un système sur puce (SoC).

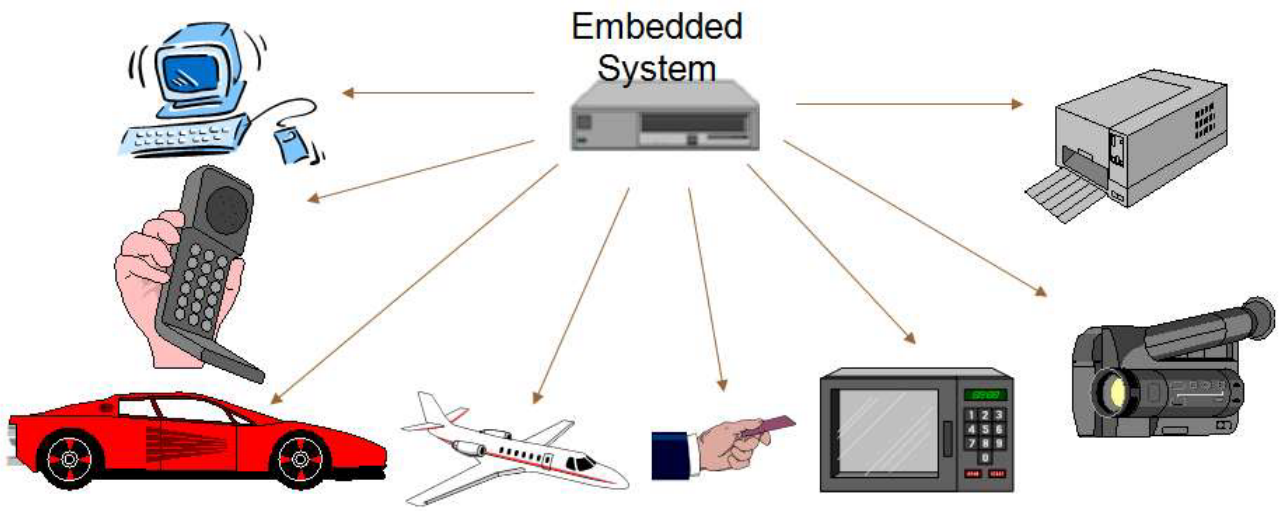


Les systèmes sur puces peuvent être trouvés dans de nombreuses catégories de produits allant des dispositifs de consommation à des systèmes industriels (figure 2) :

1. **International Technology Roadmap for Semiconductors**.

1. Les téléphones portables utilisent plusieurs processeurs programmables pour gérer les tâches de traitement du signal et de protocole requis par la téléphonie. Ces architectures doivent être conçues pour fonctionner à des niveaux de très faible puissance fournies par des piles ou des batteries.
2. Les télécommunications et les réseaux utilisent des SoCs spécialisés, tel que les processeurs de réseau, pour gérer les énormes débits de données offertes par les équipements de transmission moderne.
3. Les téléviseurs numériques et les décodeurs utilisent des multiprocesseurs sophistiqués pour effectuer des fonctions de vidéo, de décodage audio et d'interface utilisateur en temps réel.
4. Les consoles de jeux vidéo utilisent plusieurs machines de traitement parallèles complexes pour rendre l'action de jeu en temps réel.

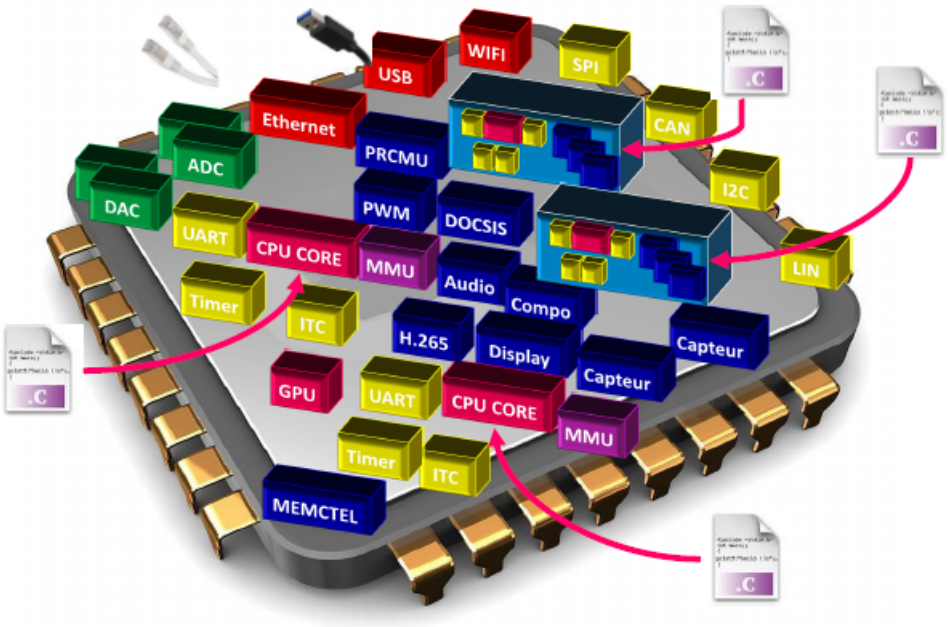
FIGURE 2 – Exemples d'applications des systèmes sur puce.



Un SoC est une combinaison de logiciel et de matériel interagissant dans le but de réaliser un ensemble de fonctions spécifiques et bien définies pour des domaines variés tels que l'automobile, l'Internet des objets –ou Internet of Things (IoT) ou encore les téléphones mobiles. Les fonctionnalités de ces systèmes doivent être fournies avec un niveau de performance satisfaisante. Cependant, avec l'augmentation de la complexité de l'architecture des SoCs et de leurs capacités de traitement, il devient important de pouvoir concevoir des stratégies efficaces et rapides qui permettent au matériels de mieux utiliser ces capacités disponibles.

La figure ci dessous présente un exemple de SoC, intégrant quelques modules et sous-systèmes ou blocs matériels, également appelés Intellectual Property (IP), que l'on peut

FIGURE 3 – Exemple de systèmes sur puce complexes.



retrouver aujourd'hui dans de tels systèmes. Elle donne aussi un aperçu de la complexité atteinte par ces produits.

Un Système sur puce présent dans les systèmes embarqués est un ensemble de composants matériels et de logiciels conçus et intégrés dans une seule puce électronique pour réaliser les fonctionnalités demandées. Typiquement, un SoC est composé de périphériques d'entrée/sortie, d'éléments de calcul et de traitement (microprocesseur,DSP), d'unités de sauvegarde des données (mémoire) communiquant via une structure d'interconnexion. Les systèmes embarqués alimentent les innovations faites dans divers domaines : la robotique, l'aviation, les télécommunications, l'automobile, la sécurité, la médecine, etc. et leur criticité dépend du champ d'application. Depuis la création des premiers circuits intégrés en 1958, la complexité de ces circuits n'a pas cessé de s'accroître.

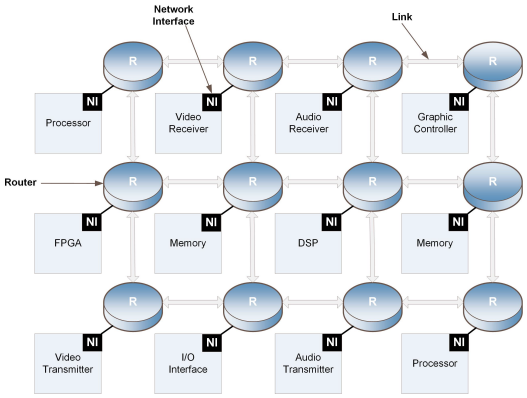
Gabriel Lezmi, vice président de Synopsys en Europe a écrit dans le magazine électronique "ElectronicsWeekly.com" que la vérification reste encore le plus grand goulot d'étranglement dans le flot de conception des SoCs actuels. Elle représente à elle seule plus que 50% de l'activité et engendre la limitation de la productivité et l'augmentation des coûts. En effet, plus que 100.000 tests sont nécessaires pour la validation d'une IP[71].La sûreté de fonctionnement, un temps réservé au monde du logiciel, devient une contrainte importante pour les systèmes matériels dès lors qu'une défaillance peut avoir un effet très néfaste sur la réalisation de la mission. C'est notamment le cas pour les systèmes embarqués lorsque la maintenance est impossible ou très coûteuse et pour les systèmes où une défaillance peut mettre en danger des vies humaines.

L'amélioration continue des procédés de fabrication permet d'envisager des architectures MPSoC (Multi Processors System on Chip) comportant plusieurs centaines de processeurs, l'intégration de ce système complet se fait sur une seule et unique puce. Ce système peut comprendre divers composants (appelés IP : Intellectual Property) : processeurs, mémoires, réseaux d'interconnexions, circuits spécifiques...etc. La communication entre ces différents composants se faisait autrefois via des bus de connexion sous différentes topologies. Par la suite, et afin d'accélérer le temps de traitement un nouveau concept a rejoint le monde des SoC à savoir : Les NoCs (ou Network on Chip).

Depuis les années 2000, les concepteurs des SoCs proposent un nouveau paradigme d'interconnexion afin de résoudre les problèmes rencontrés au niveau des structures classiques. Avec le nombre croissant des noyaux intégrés sur une seule puce, la recherche en réseau sur puce (NoC = Network on Chip) devient de plus en plus importante (figure.4). Cette structure est inspirée des réseaux informatiques et adaptée pour les systèmes sur puce. Elle offre une très grande bande passante par rapport aux autres architectures (car elle n'est pas partagée), une diversité des chemins, le parallélisme des communications, la flexibilité, l'extensibilité et la réutilisation facile des IPs. Les NoCs reposent sur une paquetisation de données pour les transporter d'une source vers une cible à travers un réseau d'aiguilleurs (routeurs) et de canaux d'interconnexions (liens). Un exemple d'une structure couramment utilisée est illustré Figure 4.

Faire communiquer ces composants au sein d'une puce est aujourd'hui assuré par un réseau de commutation de paquets appelé réseau sur puce (Network-on-Chip-NoC). Les NoCs ont pour fonction de transmettre des données entre différents nœuds de calcul eux-mêmes, composés d'un ou plusieurs cœurs, sous forme de paquets. Les routeurs sont disposés plus ou moins régulièrement, afin de diriger chaque paquet vers sa destination (i.e. routage). Chaque routeur embarque plusieurs ports de sortie et ports d'entrée, eux-mêmes connectés à un lien, afin de transférer physiquement les messages.

FIGURE 4 – Structure d'interconnexion NoC.



La défaillance du composant dans les puces réseau (NoC) a été un facteur critique pour la fiabilité du système. Afin de réduire l'impact des pannes, une étude de la tolérance aux pannes a été réalisée ces dernières années afin d'améliorer la robustesse de NoC. En raison du vaste choix de mécanismes de tolérance aux pannes et de contraintes de conception critiques, la sélection et la configuration d'un mécanisme approprié pour satisfaire les exigences de tolérance aux pannes constituent de nouveaux défis pour les concepteurs.

La principale limitation est qu'en regard de la complexité, il n'est plus possible de garantir que 100% des composants matériels présents sur la puce seront sans défauts. Les défauts peuvent être détectés soient au moment de la fabrication (en usine), soit après mise en service (opérationnel), suite à des événements interne ou externe.Pour éviter un comportement inattendu (voire l'arrêt) du système suite à l'apparition d'une faute, il est donc nécessaire d'intégrer des mécanismes de tolérance aux fautes.

Notre thèse se focalise sur l'aspect Routage dans laquelle on cherche à concevoir et implémenter un routage tolérant aux fautes statiques ou dynamiques ainsi que les objectifs qu'ils tentent d'atteindre. De ce fait, le routage constitue une partie importante dans l'architecture des réseaux sur puce, où il permet de procéder à la recherche d'une route entre une source (composant émetteur) et une destination (composant récepteur) de la manière la plus optimale possible. Son but principal est d'essayer de trouver le chemin qui permet de maximiser les performances du réseau tout en minimisant la quantité d'énergie consommée et d'éviter un maximum d'obstacles possibles.

Plusieurs algorithmes de routage ont été développés pour améliorer la transmission de données sur un réseau sur puce. Certains d'entre eux tentent d'améliorer la consommation d'énergie, d'autres visent à améliorer la latence ou encore le taux de données transmises. Pour cela divers techniques sont employées selon que l'algorithme suive une approche déterministe ou une approche adaptative. Cependant, dans chacune des situations, des problèmes ralentissant le routage peuvent subvenir à savoir : la congestion, le deadlock, le livelock et la présence de pannes. Pour les éviter, plusieurs techniques sont employées : utilisation des canaux virtuels, utilisation des tables de routage, etc. Néanmoins, ces solutions sont coûteuses en termes de ressources consommées, et n'arrivent à satisfaire qu'un seul objectif à la fois au détriment des autres. C'est donc dans ce contexte que s'insère l'objectif de notre thèse, où il est demandé de développer une approche de routage multi-objectifs (qui serait capable d'éviter le deadlock, le livelock, les zones congestionnées et tolérante aux fautes) permettant de maximiser la quantité de données transmises tout en assurant des performances raisonnables sur le réseau.

Dans cette thèse, nous proposons un nouvel algorithme de routage tolérant aux fautes intitulé DINRA-FTNoC (Design and Implementation of a New Routing Algorithm Fault Tolerance in Network on Chip), avec une architecture de routage fiable. La solution proposée permet de gérer un grand nombre d'erreurs qui peuvent survenues dans différents

endroits à savoir : buffer, Crossbar et les liaisons (qui sont les composants les plus sensibles aux défauts). Contrairement aux travaux antérieurs, l'architecture proposée tolère de multiples fautes sans dégradation considérable des performances en termes de latence et fiabilité.

Problématique

Tous les réseaux sur puces ont des caractéristiques propres à eux qui les rendent plus attrayants dans certains domaines, notamment le nombre de ressources supportées, la rapidité des communications, l'énergie nécessaire, le nombre de ressources utilisées, etc. Au niveau des communications, bien que le débit soit une des métriques les plus importantes, il ne faut pas non plus oublier la fiabilité du transit des données. Les interconnexions des systèmes sur puce ne cessent de diminuer de taille également et sont maintenant rendues à l'échelle du nanomètre.

Bien que la réduction de la taille des transistors et des connexions n'a cessé d'augmenter les performances des processeurs et de réduire leur coût, elle a également affecté la sûreté de fonctionnement.Quand un transistor est exposé à des rayonnements ionisants de haute énergie, des paires électron-trou sont créées. La finesse de gravure réduite ces dernières années à quelques nanomètres menace de manière significative les prochaines générations des circuits. Concernant les fautes transitoires, des circuits plus petits ont tendance à mettre en œuvre de plus faibles charges pour maintenir les états des registres, et les rendent de ce fait plus sensibles aux bruits. Lorsque la marge de bruit diminue, la probabilité que l'impact d'une particule de haute énergie puisse perturber la charge sur les circuits augmente, qui elle-même augmente à son tour la probabilité de fautes transitoires. Plus un circuit comporte de transistors, plus de fils sont nécessaires pour les connecter, résultant en une plus forte probabilité de fautes, tant lors de la fabrication que du fonctionnement de ces dispositifs.

Les réseaux sur puces sont plus sujets aux fautes en raison d'un plus grand nombre de transistors. Par ailleurs, la température est un autre facteur de fautes transitoires ou permanentes. Plus la puce comporte de dispositifs, plus de puissance sera drainée de l'alimentation, ce qui – à surface égale – va augmenter les pertes par dissipation avec comme conséquence une augmentation de la température et de la probabilité d'erreurs. Aujourd'hui, les architectures des circuit sont de plus en plus compliquées, ce qui augmente la probabilité d'erreurs de conception. D'autre part, cela rend également difficile le débogage des erreurs. Dans un proche avenir, changer les paramètres comme les dimensions et la taille va augmenter l'apparition de défauts dans le circuit. Pour assurer leur intégrité, la tolérance aux fautes doit être une considération importante dans la conception des circuits modernes. Un système fiable doit intégrer un mécanisme de tolérance contre les possibles erreurs.

Cette diminution de taille amène les interconnexions à devenir de plus en plus sensibles et vulnérables aux fautes. Ces fautes peuvent avoir un effet permanent aussi bien que transitoire. Ces fautes peuvent être internes au circuit lui-même par exemple les variations de tension ou la diaphonie (crosstalk), comme elles peuvent également provenir de l'extérieur

telles les interférences magnétiques et les effets des particules alpha. C'est pourquoi de plus en plus les réseaux sur puce sont développés dans une optique de tolérance aux fautes. Il est devenu essentiel pour rencontrer un certain niveau de fiabilité d'intégrer des techniques pour surmonter l'effet des fautes dans ces systèmes.

La tolérance aux fautes n'est pas nouvelle en soi dans le domaine des réseaux sur puce et plusieurs recherches ont été menées sur ce sujet. Le but ultime est qu'aucune donnée ne soit interrompue, autrement dit, qu'aucun message lors de son transfert ne soit modifié ou, tout simplement, n'arrive pas à destination. Pour ce faire, dans la majorité des cas, l'erreur en premier lieu doit être d'abord détectée et, par la suite, isolée. Dans la littérature, les méthodes de détection d'erreur les plus utilisées sont le contrôle par redondance cyclique (Cyclic Redundancy Check, CRC) [51], le code de Hamming [51] et le bit de parité. Une fois l'erreur détectée, il faut définir de quelle manière elle sera traitée. Il est préférable d'opter seulement pour l'isolation des composants fautifs et ne pas s'intéresser à la correction d'erreur.

Contributions de la thèse

La contribution de notre thèse est de proposer et d'évaluer un nouvel algorithme de routage tolérant aux fautes avec les propriétés suivantes :

1. L'algorithme de routage proposé est totalement adaptatif et tolérant aux pannes (au niveau des liens ou routeurs) tout en évitant le phénomène de congestion (qui peut aussi être considéré comme une faute dans un réseau).
2. L'algorithme proposé assure **une couverture totale des pannes** : En cas de défaillance dans un NoC, l'algorithme garantit que tout paquet de données est acheminé vers sa destination s'il y'a au moins un chemin entre la source et la destination. Cela contraste avec des travaux antérieurs qui ne prennent en charge qu'un nombre limités de fautes. Certains autres travaux nécessitent de désactiver des nœuds sains.
3. L'algorithme proposé est 1) **entièrement distribué** ; c'est à dire que le mécanisme de tolérance aux pannes ne s'appuie sur aucun composant centralisé, 2) le mécanisme est exempt du phénomène d'interblocage.
4. L'approche proposée ne requiert aucun canal virtuel.
5. Lorsque plusieurs chemins entre les noeuds source et destination sont possibles, la méthode de routage permet généralement aux noeuds de transfert de choisir dans la sélection du saut suivant le chemin le moins encombré. La combinaison de cette liberté avec des informations locales sur l'encombrement peut permettre aux routeurs d'adapter l'itinéraire afin de viser un taux d'encombrement inférieur.
6. Certaines pannes de composants (nœuds ou liaisons) ne nécessitent pas de reconfigurer l'algorithme de routage. Cette propriété peut être exploitée pour ne pas nécessiter de temps de reconfiguration lors de la mise hors tension du Noc afin d'économiser de l'énergie à l'aide de la technique de Power Gatting.
7. Détection de l'inaccessibilité : Notre algorithme peut également détecter l'absence d'un chemin, c'est-à-dire lorsqu'un paquet ne peut pas atteindre sa destination en raison de partitions déconnectées du réseau.

Dans cette thèse, nous nous concentrons principalement sur le développement d'un algorithme de routage totalement adaptatif et tolérant aux fautes transitoires et permanentes pour les réseaux sur puce avec contrôle de flux par trous de ver (WormHole). Notre approche tente d'identifier les composants défaillants et les désactiver afin d'améliorer l'adaptativité de l'architecture en termes de fonction de routage. L'efficacité de l'algorithme proposé est montré en les évaluant pour des profils de trafic synthétiques. En outre, les frais généraux de puissance et de surface ne sont pas analysés.

Organisation du manuscrit

Les structures d'interconnexion classiques ne répondent pas aux exigences des futurs systèmes qui intègrent un très grand nombre d'IPs. Les réseaux sur puce (NoC) est un nouveau paradigme d'interconnexion pour les systèmes sur puce (SoC). Nous avons donc proposé une nouvelle architecture avec un algorithme de routage tolérant aux fautes permanentes et transitoires. Cet architecture est basée sur le principe de Power Gatting inspirée par Catnap.

Le reste du document est organisé selon le plan suivant :

Chapitre 1 : Dans ce chapitre, nous allons introduire le paradigme des réseaux sur puce ainsi que les notions qui lui sont associées. A cet effet, nous détaillerons les différents éléments de base, qui constituent cette architecture d'interconnexions, puis nous décrirons les points de caractérisation des réseaux sur puce, à savoir les métriques de performances, les différentes topologies, les algorithmes de routage, les protocoles de communication, le format de paquets, le contrôle de flux et de congestion.

Chapitre 2 : Puisque la tolérance aux fautes est un aspect central de notre travail, une explication de la provenance et de la classification des fautes qui peuvent survenir à l'intérieur des réseaux sur puce est également détaillée. Ensuite nous présenterons un état de l'art sur les algorithmes de routage tolérant aux fautes sur les réseaux sur puce (structures à 2 D), et puis nous montrerons leurs points fort et leurs limites.

Chapitre 3 : Dans cette partie, nous détaillerons l'architecture proposée afin de palier au problème de la tolérance aux fautes dans les réseaux sur puce. Nous montrerons comment un NoC , après l'occurence de pannes, peut continuer à fonctionner en mode dégradé. A cet effet, nous détaillerons la topologie proposée, l'algorithme de routage utilisé, le contrôle de flux, la structure du réseau et son comportement en face des erreurs ainsi que la gestion de la congestion.

Le dernier chapitre sera consacré aux résultats expérimentaux de la nouvelle architecture et a la comparaison avec les travaux cités dans l'état de l'art. Nous terminerons ce document par une conclusion qui synthétise notre contribution au domaine ainsi que quelques perspectives envisageables au moyen et au court terme.

Résumé

La révolution technologique de ces dernières années a rendu possible la confection de systèmes embarqués de plus en plus performants. Parmi ces équipements, Les systèmes multiprocesseurs sur puce (MPSoC)sont capables d'intégrer un grand nombre de transistors et de composants hétérogènes ensemble dans une seule puce. Faire communiquer ces composants au sein d'une puce est assuré par un réseau de commutation de paquets appelé réseau sur puce (Network-on-Chip-NoC). Les NoCs sont constitués de ressources de traitement telles que des processeurs, des DSP ou des mémoires et des routeurs. Ces organes sont connectés par des liens leur permettant de communiquer les uns avec les autres en s'envoyant des messages. Cependant, le passage à des architectures matérielles de plus en plus réduites rend les circuits plus vulnérables aux pannes (fautes ou dysfonctionnements).

Un réseau sur puce peut, en effet, se retrouver avec des routeurs ou des liens non opérationnels,qui ne peuvent plus être utilisés pour acheminer les messages. Une solution potentielle à ce problème est l'introduction de mécanismes de tolérance aux pannes permettant au système de fonctionner en mode dégradé malgré la présence de composants défectueux. Le système devrait fonctionner correctement lorsque certaines erreurs se produisent dans les routeurs ou les liens. Cependant, la communication a un impact énorme sur les performances du Network on Chip, et la conception d'un algorithme de routage efficace est plus que nécessaire. L'objectif assigné à l'algorithme de routage est de trouver un nouveau chemin pour diriger les paquets de la source vers la destination dans un réseau défectueux.

De nombreux algorithmes de routage tolérants aux pannes sont utilisés pour surmonter les défauts dans les Networks on chip. Cependant, ces algorithmes de routage souffrent de plusieurs problèmes, dont celui de la congestion. Dans ce travail, une approche originale inspirée de Catnap est proposée pour les NoC utilisant des mécanismes de détection de congestion locale et globale avec une architecture de sous-réseaux hiérarchiques. Avec l'aide de ces deux techniques, le NoC devient tolérant aux pannes et est capable d'utiliser efficacement le débit.

Mots-clés : Congestion, NoC, Fautes, Tolérance, sous réseaux, routage

Abstract

Thanks to the technological advances in recent years, new embedded systems have arisen. They are able to integrate billions of transistors and heterogeneous components together into a chip called multiprocessor on chip (MPSoC) system. To provide communication between these components within a chip, packet-switching networks called Network-on-Chip (Noc) are now available. However, the transition to increasingly smaller technologies makes circuits more vulnerable to faults.

NoCs can integrate several resources such as : processors, DSPs or memories and routers, which are connected by links so that they are able to communicate with each other by sending messages. The network-on-a-chip can therefore end up with non-operational routers/ links, which can no longer be used to route messages. A potential solution to this problem is the introduction of fault tolerance mechanisms that allows the system to operate in degraded mode despite the presence of defective components. The system should work properly when certain errors occur in the routers / links. However, communication has a huge impact on Network on Chip performance, and designing an efficient routing algorithm is required.

To handle communication requirements, the routing algorithm must find a new path to direct packets from the source to the destination in a faulty network. Many fault tolerant routing algorithms are used to overcome faults in Network on chip. However, these algorithms suffer from other problems such as congestion. In this work, an original Catnap-inspired approach is proposed for NoCs using a local and global congestion detection mechanisms with a hierarchical subnetworks architecture. With the help of these two techniques, the NoC becomes fault-tolerant and is able to use flow efficiently.

Keywords : NoC, Congestion, Fault, Tolerance, Sub Network, Routing.

الملخص

بفضل التقدم التكنولوجي في السنوات الأخيرة ، نشأت أنظمة مدمجة جديدة. فهي قادرة على دمج مليارات الترانزستورات والمكونات غير المتجانسة معًا في شريحة تسمى نظام المعالجات المتعددة على الرقاقة . لتوفير اتصال و تبديل حزم بين هذه المكونات داخل شريحة تسمى الشبكة على الرقاقة. ومع ذلك، فإن الانتقال إلى التقنيات الأصغر حجما يجعل الدوائر أكثر عرضة للأعطال.

يمكن أن تدمج الشبكات على الرقاقة العديد من الموارد مثل: المعالجات أو DSPs أو الذاكرات وأجهزة التوجيه، والتي ترتبط بواسطة روابط حتى يتمكنوا من التواصل مع بعضهم البعض عن طريق إرسال الرسائل. وبالتالي ، يمكن أن ينتهي الأمر بالشبكة على الشريحة بأجهزة توجيه / روابط معطلة ، و بذلك لم يعد من الممكن استخدامها لتوجيه الرسائل. أحد الحلول المحتملة لهذه المشكلة هو إدخال آليات تحمل الأخطاء التي تسمح للنظام بالعمل في الوضع المتدهور على الرغم من وجود مكونات معطلة . ومع ذلك يجب أن يعمل النظام بشكل صحيح عند حدوث أخطاء معينة في أجهزة التوجيه / الروابط. ، فإن الاتصالات لها تأثير كبير على أداء الشبكة على الرقاقة ، كما أن تصميم خوارزمية توجيه فعالة أمر مطلوب.

لمعالجة متطلبات الاتصال، يجب أن تجد خوارزمية التوجيه مسارًا جديدًا لتوجيه الحزم من المصدر إلى الوجهة في شبكة معيبة. حيث يتم استخدام العديد من خوارزميات توجيه للتغلب على الأخطاء في الشبكة على الشريحة. ومع ذلك ، فإن هذه الخوارزميات تعاني من مشاكل أخرى مثل الازدحام. في هذا العمل، يُقترح نهج أصلي مستوحى من Catnap لشبكات باستخدام آليات اكتشاف الازدحام المحلية والعالمية مع بنية شبكة فرعية هرمية. و بمساعدة هاتين الطريقتين تصبح الشبكة على الرقاقة متسامحة مع الأخطاء وقادرة على استخدام التدفق بكفاءة.

الكلمات المفتاحية : الشبكة على الرقاقة ، الازدحام ، الخطأ ، التسامح ، الشبكة الفرعية ، التوجيه.

Chapitre 1

Réseaux sur puce - Concepts de base

Sommaire

Introduction

Dès que le réseau sur puce fut proposé comme la solution la plus prometteuse pour remplacer les interconnexions classique plusieurs travaux ont essayé de développer des réseaux spécifiques, ces travaux ont construit une bibliographie riche permettant ainsi de caractériser le réseau sur puce par rapport aux autres supports de communication. Le but de ce premier chapitre est d'introduire les notions de bases associées au paradigme des réseaux sur puce. A cet effet, nous détaillons les différents éléments de base puis nous décrivons les points de caractérisation des réseaux sur puce, à savoir : la topologie, le protocole de communication, le format de paquet, le contrôle de flux, la classification des algorithmes de routage ainsi que les différents problèmes liés au routage et les métriques de performances.

Les systèmes sur puce (SoC)[16] sont des systèmes embarqués composés de plusieurs modules sur une même puce (processeurs, mémoires, périphériques d'entrée / sortie). Avec les SoC, il est maintenant possible de traiter des informations et d'exécuter des tâches critiques à une vitesse plus élevée et de réduire la puissance sur une puce minuscule. Ceci est dû au nombre croissant de transistors qui peuvent être intégrés sur une seule puce qui continue à doubler tous les 18 mois comme le prédisait Gordon Moore[56]. Cela a fait rétrécir la taille de la puce tout en conservant de hautes performances. Cette évolutivité de la technologie a permis aux SoC de se développer de manière continue en termes de nombre de composants et de complexité et d'évoluer vers des systèmes dotés de nombreux processeurs intégrés sur un seul SoC. A titre d'exemple, le processeur Intel Xeon comprend 2,3 milliards de transistors. Avec un niveau d'intégration aussi élevé, le développement de nombreux noyaux sur un seul dé est devenu possible. Ces systèmes sont appelés systèmes multiprocesseurs sur puce (MPSoC). Par exemple, Tilera Tile64 et Intel Polaris contiennent respectivement 64 et 80 cœurs.

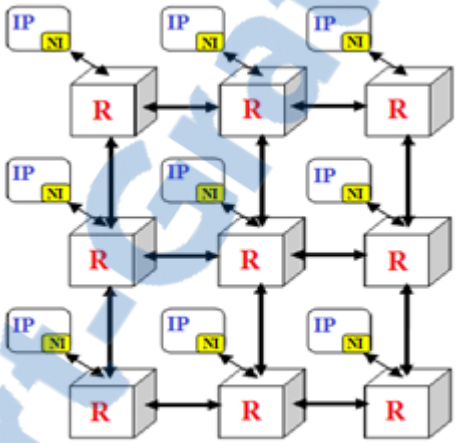
Il existe plusieurs types de systèmes d'interconnexion utilisables dans un SoC dont le point-à-point, le bus partagé ou hièrarchique, le crossbar et le réseau sur puce. En Génaral chaque type d'interconnexions à ces avantages et ces inconvénients. Avec des technologies d'intégration croissantes, il est a présent envisageable implémenter dans un SoC des systèmes d'interconnexion plus avancés comme un réseau sur une seule puce.

Dans cette section nous montrerons les différents éléments de base du NoC : les routeurs, les liens et les interfaces réseau.

1.1 Les réseaux sur puce (Network on Chip)

Un NoC typique se compose de trois éléments principaux. Les routeurs qui permettent d'acheminer les données à partir d'un nœud source vers un nœud destination, des liens et leurs rôles acheminer les données entre chaque pair de nœud, le troisième élément c'est les interface réseau ou (Network Interface-NI en Anglais) voir (figure 1.1). Les IPs (Intellectual Properties) pourraient être n'importe quel composant tel : un microprocesseur, un circuit intégré spécifique à une application (ASIC), une mémoire, ou une combinaison de composants reliés entre eux. Ces IPs sont connectées aux routeurs de réseau par l'intermédiaire d'une interface de réseau.

FIGURE 1.1 – Eléments de base d'un NoC.

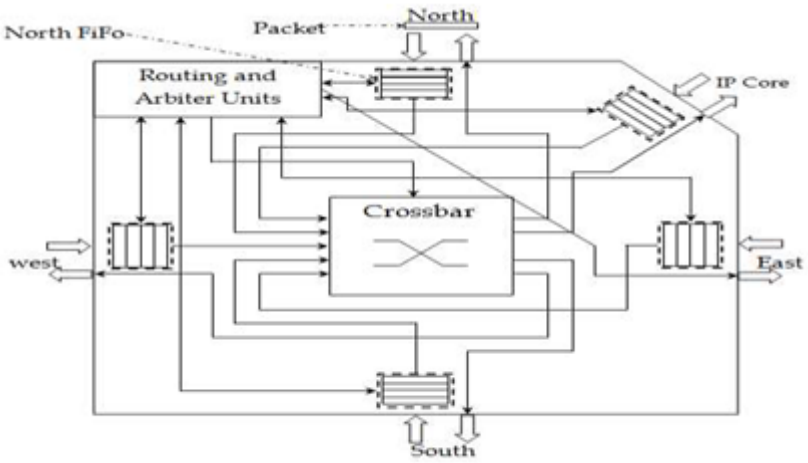


1.1.1 Les routeurs

Les nœuds ou routeurs[16], sont typiquement constitués de ports d'entrée/sortie contenant (ou pas) des files d'attente pour stocker les données qui ne pourront pas être transmises tout de suite, d'une matrice de commutation pour relier physiquement les ports d'entrée aux ports de sorties, des blocs de contrôle et d'arbitrage des paquets et enfin, d'une table de routage ou d'un bloc de calcul de la fonction de routage si ce-dernier est distribué.

Le routeur est l'entité responsable de l'acheminement du paquet à travers les différents liens d'interconnections du réseau, en effet c'est grâce à cette entité qu'on peut appliquer physiquement les différents algorithmes de routage, alors il y a une dépendance majeure entre l'algorithme de routage à adopter par le réseau et la structure du routeur. La figure 1.2 représente une architecture typique d'un routeur à savoir : **les ports d'entrées sortie,**

FIGURE 1.2 – Architecture typique d'un routeur.



des mémoires tampons, une unité de routage et une unité de multiplexage dite Crossbar.

1. **Buffer :** les tampons sont utilisés pour contenir le paquet en transit. Chaque canal physique d'entrée et / ou de sortie est associé à un tampon. Le paquet en transit peut être un paquet entrant ou sortant dans un routeur. Selon les besoins, différents routeurs NoC implémentent des mémoires tampon FIFO (First-In First-Out), soit au niveau des ports d'entrée, soit au niveau des ports de sortie, soit aux deux ports. Cette implémentation réduit les coûts de mise en mémoire tampon des données. Si un paquet (disons p1) occupe un tampon pour un canal, l'autre paquet (disons p2) ne peut pas accéder au canal physique, même si p1 est bloqué. Dans une approche alternative, la mémoire tampon peut être multiplexée dans des canaux virtuels afin de fournir une qualité de service et éviter l'inter-blocage. Ces canaux logiques sont multiplexés sur le canal physique, comme indiqué à la Figure 1.2. Comme la mémoire tampon ajoute un coût important (puissance et surface) à un routeur, certaines implémentations récentes de routeur n'ont pas besoin de canaux virtuels.
2. **Unité de Routage :** cet unité consiste à calculer le canal de sortie approprié pour un paquet entrant. Cet unité implémente le schéma de routage. En règle générale, les routeurs NoC utilisent deux implémentations différentes afin de calculer l'itinéraire : **routage basé sur table et machine à états finis.**
3. **Arbitre :** Plusieurs paquets peuvent demander le même canal de sortie simultanément. L'unité d'arbitrage a pour but de résoudre ces demandes simultanées pour le même canal de sortie. Il fournit l'arbitrage entre eux. Lorsqu'un paquet demande un canal de sortie particulier et si ce canal est occupé par l'autre paquet, le premier paquet attend dans le canal virtuel d'entrée (tampon). Une fois que le canal

occupé est libéré par l'autre paquet, le premier paquet participe à nouveau à l'arbitrage et est acheminé le long du canal demandé, s'il gagne en arbitrage. Ainsi, le rôle d'un arbitre est similaire à celui d'un arbitre qui résout les conflits entre plusieurs paquets demandant le même canal de sortie dans un routeur. Les routeurs NoC implémentent diverses règles d'arbitrage telles que rotation par rotation, prise en compte des conflits de contention, priorité, round-robin, premier arrivé, premier servi, etc.

4. **Crossbar :** L'unité Crossbar est responsable de l'interconnexion des ports d'entrée aux ports de sortie du routeur. L'unité d'arbitrage contrôle les données de sortie de la barre transversale. Toutes les lignes de données d'entrée possibles sont liées aux ports d'entrés de la barre transversale.
5. **Contrôleurs de liaison :** Les contrôleurs de liaison sont responsables d'acheminement des paquets via le canal physique qui inter-connecte les ports d'entrée et de sortie des routeurs adjacents. Ce contrôle de transmission de données est nécessaire pour éviter les réplications incorrectes et le débordement de données. Les contrôleurs de liaison coordonnent les unités de transmission et contrôle de flux d'une part et de la liaison physique d'autre part. Les techniques de contrôle existantes telles que le stop-and-go et le crédit sont généralement mises en œuvre pour cette unité. Cette unité est également responsable de la mise en œuvre d'interfaces de synchronisation de données pour synchroniser le transfert de données correct d'un routeur à un autre.

Nous allons suivre le parcours d'un paquet à l'intérieur d'un routeur afin de comprendre clairement le rôle de chacun de ses composants. Pour commencer, les paquets qui arrivent et sortent du routeur passent par les ports d'entrée et de sortie.

Ce flux est géré par un protocole du contrôle du flux du réseau. Ce mécanisme asservit le débit de l'émetteur selon les capacités de réception du destinataire. Le flux de données étant sous contrôle, les données peuvent être stockées dans les buffers, l'espace mémoire du routeur. Comme nous pouvons le voir sur la figure 1.2, il y a des buffers à chaque entrée du routeur pour récupérer les données entrantes. L'algorithme de routage va ensuite déterminer la destination du nouveau paquet arrivant. Le paquet contient les données que l'émetteur souhaite transférer au récepteur mais aussi des informations nécessaires pour le routage (dans le flit d'en-tête).

Avec un algorithme de routage simple, l'information de position du noeud source et de la destination ainsi que la taille du paquet suffisent à aiguiller un paquet. Lorsque la direction est sélectionnée, l'arbitre décide de laisser passer ou non le paquet selon une politique d'arbitrage. C'est l'arbitre qui donne la décision finale sur la direction du paquet et est également en charge de la gestion du contrôle du flux du routeur. Si le paquet obtient la priorité alors le crossbar est configuré pour faire passer le paquet de l'entrée à la sortie

désirée, sinon il reste dans le buffer un cycle supplémentaire en attendant son tour.

Le crossbar est une matrice de commutateurs qui permet de relier n'importe quelle entrée à n'importe quelle sortie. Enfin, lorsque le paquet a traversé le crossbar, ce paquet passe par le port de sortie puis par les liens inter-routeurs pour se retrouver dans le buffer du routeur suivant. Ce procédé est répété jusqu'à ce que le paquet arrive à une interface réseau où il sortira du NoC.

	Buffer	Link	Crossbar	Allocators	Routing
Area	50%	39%	5%	3%	3%

TABLE 1.1 – Surface des composants d'un routeur [73]

Le tableau 1.1 montre une répartition typique de la surface des différents composants. Bien que la taille de la surface d'un composant ait une influence sur la probabilité de défaillance permanente, les composants plus petits doivent également être pris en compte dans le modèle de panne car une seule défaillance de l'un d'entre eux pourrait compromettre le bon fonctionnement du routeur entier ou même le rendre défectueux. Outre les défauts permanents pouvant survenir au sein du routeur, les liens qui les unissent sont également une source de défauts. Des liaisons défectueuses peuvent provoquer la transmission erronée de bits entre routeurs, voire interrompre la connexion entre eux au cas où le fonctionnement correct ne serait plus garanti.

1.1.2 Les liens

Les liens [16],[41] sont des connexions logiques entre deux (ou plusieurs) éléments communicants. Ils peuvent être composés d'un ou plusieurs canaux physiques. Ces derniers servent à transporter les données entre deux routeurs ou encore entre un routeur et une interface réseau. Enfin, il existe trois types de liens qui sont définis par le sens des transmissions. Ils peuvent être unidirectionnels en entrée, unidirectionnels en sortie ou bidirectionnels (entrée/sortie).

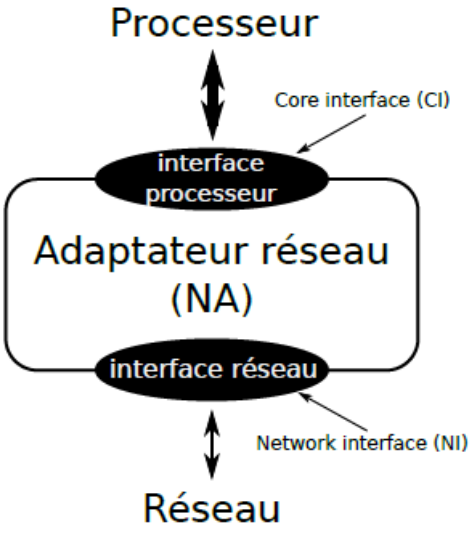
1.1.3 Les interfaces réseaux

Le troisième élément dans les réseaux sur puce est l'interface réseau (NI). Cet élément est l'intermédiaire entre l'IP et le réseau. L'interface réseau est très importante car elle sépare les calculs (au niveau des IPs) et les communications (au niveau du réseau). Elle permet aussi la réutilisation des coeurs et des infrastructures de communication indépendamment l'une de l'autre. Les interfaces réseaux servent à adapter les éléments de calculs à

la structure de communication. Pour cela, l'interface réseau fait l'adaptation de protocole entre la ressource et le réseau, donc elle convertit le protocole d'entrée/sortie des noeuds de calcul vers le protocole de communication utilisé dans le NoC[5].

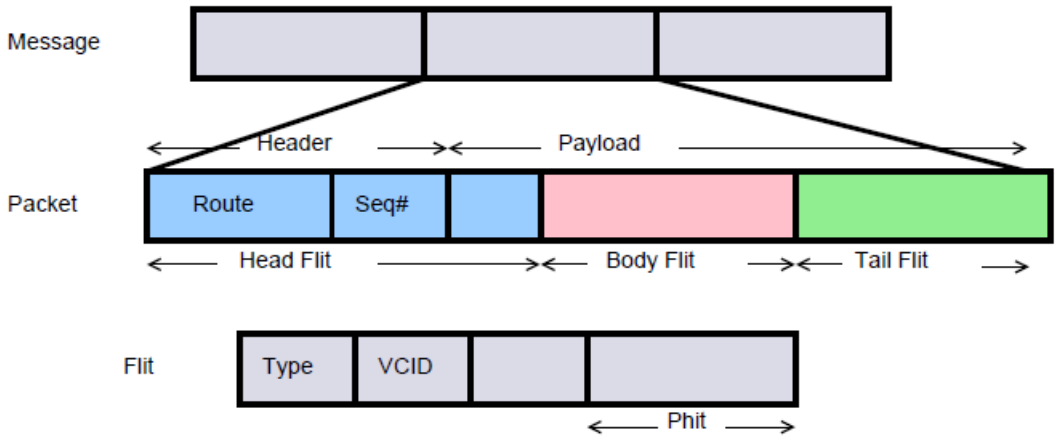
Le bloc (NI) est composé de deux parties : la première est la partie back-end qui est liée au routeur ; elle est indépendante de l'unité de traitement. La deuxième est la partie front-end connectée à la ressource de traitement qui sert d'adaptateur entre le protocole de l'unité de traitement et celui de la structure de communication(fig.1.3).

FIGURE 1.3 – L'adaptateur réseau avec ses 2 interfaces pour le réseau et le processeur.



Afin d'être échangés efficacement à travers le réseau, La NI divise les messages en un ou plusieurs paquets. Chaque paquet représente un segment du message auquel est rajouté un entête de paquet. Cet entête contient les informations de routage qui sont nécessaires à la propagation du paquet dans le réseau. Les flits associés à un paquet de données sont constitués d'un flit entête (head), d'un flit de queue et d'un certain nombre de flits du corps entre les deux (fig.1.4). L'entête comporte les informations de routage et de séquençage. La charge utile représente les données réelles qui doivent être transmises. Enfin, le paquet possède également une partie qui indique la fin du paquet, c'est la queue ; elle contient généralement le code de vérification d'erreur.

FIGURE 1.4 – Compositionn d'un message.



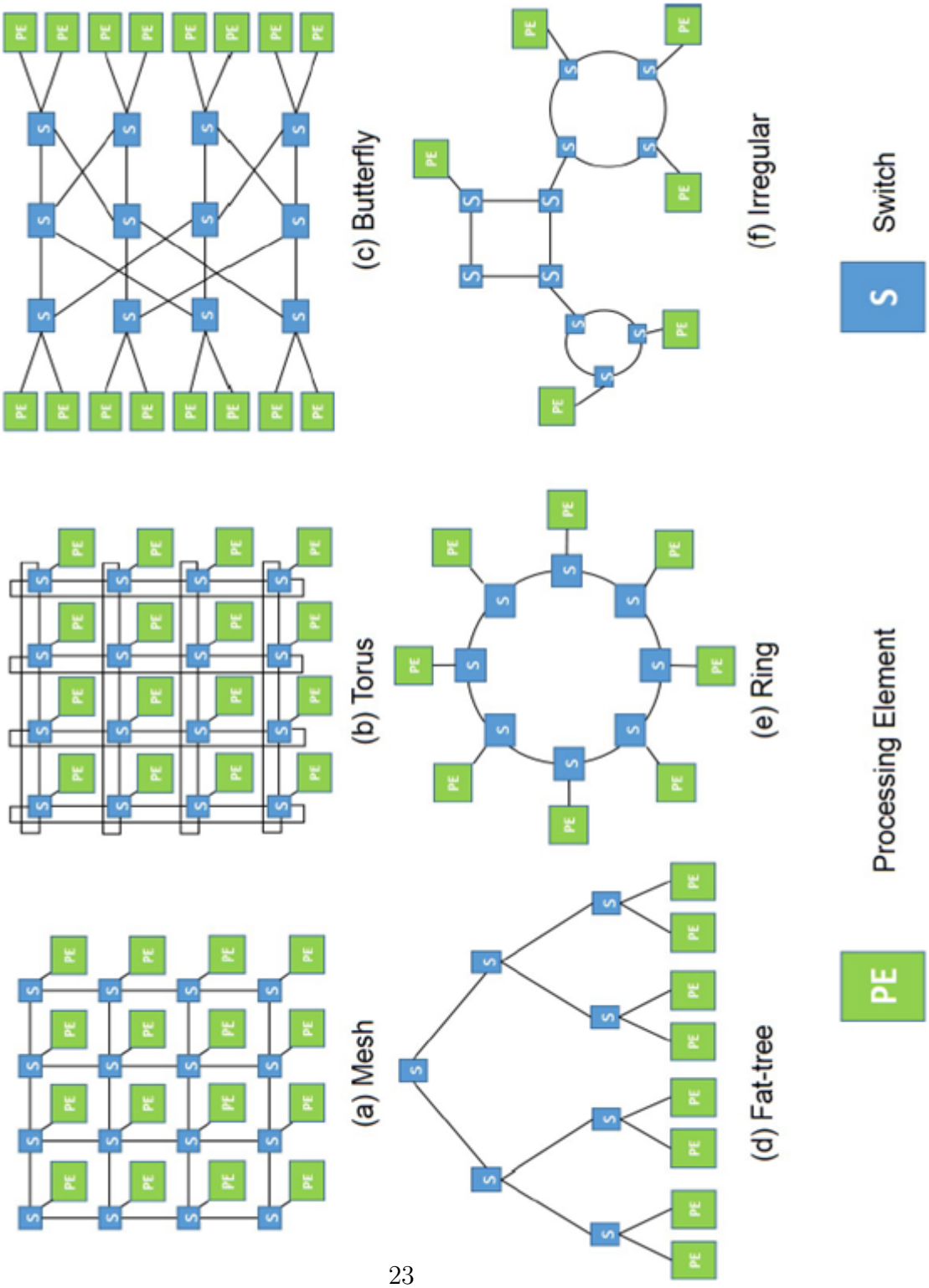
1.2 La topologie

La topologie du réseau fait référence à l'arrangement statique des nœuds et des liens du réseau. Or, la première étape dans la conception d'un réseau sur puce est la sélection de la topologie car cette-dernière va conditionner la stratégie de routage employée ainsi que la méthode de contrôle de flux[56] .

La topologie est un point important de la performance des NoCs. Elle est représentée par un graphe G (N, C), où N est l'ensemble des routeurs qui composent le réseau et C l'ensemble des canaux de communication. Le NoC peut être organisé en différentes topologies. Plusieurs topologies ont été proposées et mises en œuvre pour les systèmes multi-coeurs, y compris maille, tore, anneau, papillon et des réseaux d'interconnexion irrégulière[55],[61](fig.1.5). La complexité d'implémentation de la topologie dépend de deux facteurs : le nombre de liens à chaque noeud et la facilité de poser la topologie sur une puce. Comme les routeurs et les liens traversés par un paquet consomment de l'énergie, l'effet d'une topologie sur le nombre de sauts affecte ainsi directement la consommation d'énergie du réseau.

Une topologie de réseau peut être classée comme étant directe, indirecte ou hybride. Pour les topologies directes (Mesh, Tore, Anneau), chaque élément de calcul (IP) est associé à un routeur ; donc tous les routeurs sont des sources et des destinations. Les paquets sont transmis directement entre les nœuds terminaux. Dans une topologie indirecte (Papillon, Arbre élargi), les routeurs sont différents des éléments de calcul ; seuls les éléments de calcul sont les sources et les destinations du trafic ; les nœuds intermédiaires changent simplement le trafic vers et à partir des éléments de calcul[26]. Pour les réseaux hybrides, cette topologie est basée sur une combinaison entre les deux précédentes topologies (directe et indirecte).

FIGURE 1.5 – Différents topologie dans les NoCs.



On peut aussi classer les topologies selon leur régularité ; il y a des topologies régulières et d'autres irrégulières. On dit qu'un réseau ou une topologie est régulière lorsque tous les nœuds ont le même degré (tous les noeuds ont le même nombre de voisins). Dans le cas contraire, on dit que la topologie est irrégulière. Le routage dans les réseaux irréguliers est très compliqué et difficile. Les topologies irrégulières permettent plus de liberté et ainsi de dimensionner précisément le réseau requis. Elle peut être issue d'une topologie régulière qui a été retaillée au plus juste de façon à enlever les éléments non utilisés[41]. Une topologie irrégulière nécessite en revanche une plus grande attention pour le routage car les règles à appliquer ne sont plus triviales.

Dans la littérature les topologies régulières sont les plus utilisées. Parmi eux on cite celle en grille 2D (2D mesh). Cette topologie offre plus d'avantage car elle est facile à implémenter sur des circuits reconfigurables (FPGA) avec une grande scalabilité (capacité d'accroître facilement la structure matérielle pour répondre à une exigence de performances). Comme elle permet d'utiliser des stratégies de routage simple et donc peu coûteuses en termes de ressources. La topologie du réseau définit certains critères physiques à savoir :

- **Le degré du routeur :** Désigne le nombre de liens à chaque nœud.
- **Le diamètre :** C'est la distance maximale entre une paire de nœuds dans le réseau.
- **La distance moyenne :** C'est le nombre moyen de liens entre deux ressources.
- **La diversité de chemins :** Chaque paire source/destination a plusieurs chemins en son sein. Cette métrique permet au réseau d'être tolérant aux fautes.
- **La régularité :** On dit qu'un réseau est régulier lorsque tous les routeurs ont le même degré.
- **La connectivité :** Présente le nombre de voisins directs pour chaque nœud dans le réseau.
- **La largeur de bissection :** C'est le nombre minimal de liens qu'il faut couper pour séparer le réseau en deux parties égales.

Ces critères physiques ont un impact sur les performances globales du réseau, en termes de latence moyenne et de bande passante du réseau[70]. A part ces topologies montrées précédemment il peut avoir d'autres topologies, on peut les considérer comme des versions améliorés ou inspirés des réseaux présentés précédemment.

Quelle que soit la topologie choisie, un réseau sur puce est dit homogène ou hétérogène. Un réseau est dit homogène lorsque tous les unités de calcul du réseau sont identiques. Dans le cas contraire, le réseau est dit hétérogène. L'approche homogène est la plus simple à mettre en oeuvre. L'approche hétérogène est la plus optimisée en terme de performance mais plus complexe à mettre en oeuvre du fait du mélange de types d'unité. Dans le domaine de l'embarqué, cette seconde approche est la plus pertinente. En effet, la surface étant limitée, il est souvent nécessaire de faire cohabiter des unités de calcul hétérogènes afin d'assurer la complémentarité en terme d'efficacité de calcul.

La survenue de défauts dans les liaisons ou les routeurs affectent d'une manière directe

les caractéristiques topologiques et, par conséquent, les performances de l'intégralité du NoC. Compte tenu des taux d'erreur croissants prévus pour les futurs SoC, la sélection de la topologie doit prendre en compte la robustesse[75].

1.2.1 Mécanismes de commutation

La commutation du paquet est le mécanisme avec lequel les données circulent au niveau du réseau sur puce. Nous distinguons deux types de commutation données : 1) commutation de circuits et 2) commutation de paquets[45]. Dans la première méthode, le chemin entre chaque paire source/destination doit être établi d'abord est réservé avant de commencer d'envoyer les données. Cela peu garantir certaines performances, car le message est sûr d'être transféré à sa destination sans qu'il soit nécessaire de le mettre en buffer, de le répéter ou de le régénérer une autre fois. De plus, si pendant l'établissement du chemin, un problème est rencontré (tel que une panne ou une congestion élevée), le nœud source peut recalculer un autre chemin plus sûr. Cependant, la configuration d'un nouveau chemin peu augmenter la latence, et générer de l'encombrement.

La deuxième méthode consiste à acheminer les messages en les découpant sous forme de paquets/flits. Chaque paquet comporte un entête contenant les adresses nécessaires pour son routage dans les nœuds du réseau. À l'arrivée les messages sont reconstitués à partir des paquets reçus. Ce type de commutation offre une utilisation plus rationnelle des ressources mais demande un contrôle de flux et de congestion.

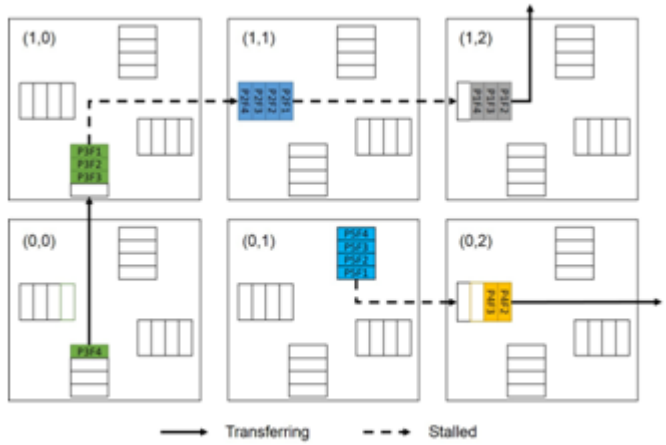
La commutation par paquets est plus fréquente et est utilisée dans environ 80% des NoCs [27]. Dans la commutation par paquets, les routeurs communiquent entre eux en transmettant des paquets/ flits à travers le réseau. Afin que la transmission d'un paquet donné ne bloque plus la communication des autres dans le réseau.

La commutation de paquet nécessite que les routeurs puissent stocker les flits en transit dans le NoC. Des buffers sont donc présents dans les routeurs pour remplir ce rôle. La dimension des buffers est définie selon la politique de mémorisation du réseau. Les méthodes de commutation ont un impact important sur les performances du NoC et chacune d'elles a ses avantages et ses inconvénients. Il existe plusieurs mécanismes de commutation basés sur le principe de commutation des paquets, dont les plus connus sont : **Store and Forward (SAF), trou de ver (Wormhole) et Virtual Cut Through (VCT) [62].**

Différé ou Store-And-Forward (SAF) : La politique SAF correspond à un protocole de commutation de paquet dans lequel les routeurs vont stocker totalement les paquets avant de les renvoyer. Ainsi, il est possible que le paquet doive attendre si le routeur suivant n'a pas suffisamment de place pour le stocker[34]. Bien que le SAF (Figure 1.6) permette d'utiliser des algorithmes de routages plus élaborés, il introduit une latence à

chaque routeur traversé. En outre, le SAF requiert une quantité d'espace de stockage importante car les routeurs ont besoin de mémoriser plusieurs paquets complets en même temps.

FIGURE 1.6 – Commutation de paquet Store And Forward.



Passage à travers (VCT) : La technique de commutation VCT [5] (Figure 1.7) a été introduite dans le but de réduire la latence du modèle SAF. En effet, dans le cas du VCT, la granularité des éléments commutés est plus petite qu'un paquet. Ce dernier est fractionné en flits. Avec ce mode, un nœud peut commencer à envoyer un paquet si le nœud suivant lui garantit qu'il peut stocker le paquet dans sa totalité, dans le cas contraire, le nœud doit pouvoir garder le paquet. La capacité de mémorisation du nœud est donc la même que pour le mode SAF mais la latence est diminuée, en absence de contention, puisqu'il n'est plus nécessaire d'attendre la réception complète du paquet pour qu'il passe d'un nœud à l'autre.

Trou de ver ou WormHole (WH) : La politique WH (Figure 1.8) est basée sur le principe que les flits d'un paquet peuvent être transmis d'un routeur à l'autre dés qu'il y a de la place pour un flit et plus nécessairement pour un paquet complet[72]. Ainsi, la taille des files d'attente en entrée des routeurs est plus petite, de l'ordre de quelque flits. Cette technique de commutation offre une faible latence pour la traversée des routeurs car ces derniers n'ont plus à stocker les paquets en entier. En outre, le coût en surface est réduit car les files d'attente sont beaucoup plus petites. Cependant, il peut arriver qu'un paquet occupe plusieurs routeurs en même temps, et dans ce cas, il est possible qu'il bloque la transmission d'autres paquets, aboutissant à une contention en cascade, voire même à une situation de blocage cyclique appelée inter-blocage (cf. section Inter blocage)[3].

Circuit virtuel commuté ou Switched virtual circuit (SVC) La technique SVC [72] consiste à définir des circuits virtuels entre une source et une destination et ce, au sein

FIGURE 1.7 – Commutation de paquet Virtual Cut Through.

d'un réseau à commutation de paquets. Dans ce modèle, un paquet spécial est d'abord envoyé dans le réseau afin de réserver les ressources nécessaires pour le chemin entre la source et la destination, correspondant au circuit virtuel entre les deux. Lorsque le circuit a été bien établi, l'émission des données peut commencer. Cette technique n'est efficace que si le temps de transmission des données est beaucoup plus long que le temps d'établissement du circuit virtuel.

Ces trois modes de commutation partagent le principe de diviser le message à transmettre en un ensemble de paquets où chaque paquet lui-même est formé par un ensemble de FLIT, un FLIT lui-même est composé par un certain nombre de phits (habituellement unitaires), la taille de ce dernier, dans la plupart de temps, est adaptée à la taille des liens d'interconnexions du réseau (ceci est défini par le nombre de bits de données pouvant être transmis en parallèle entre deux routeurs voisins), un autre point commun entre ces trois modes de commutation est le fait que les différents paquets formant un message peuvent avoir des différents chemins de routage.

Certains termes du réseau de communication couramment utilisés comme message, paquet, flit et phit sont abordées dans ce document. La figure 1.9 montre la structure d'un paquet.

1. **Message :** il représente les données qui doivent être transférées à partir de la source vers la destination. Le message est défini dans la couche application. La taille du message peut être fixe ou variable. Un message peut être divisé en plusieurs paquets.
2. **Paquet :** un paquet représente une unité d'un ensemble de données. Il peut être transféré dans le réseau indépendant des autres paquets du même message. Il contient suffisamment d'informations, formatées pour une structure précise, qui traversent

FIGURE 1.8 – Commutation de paquet WormHole.

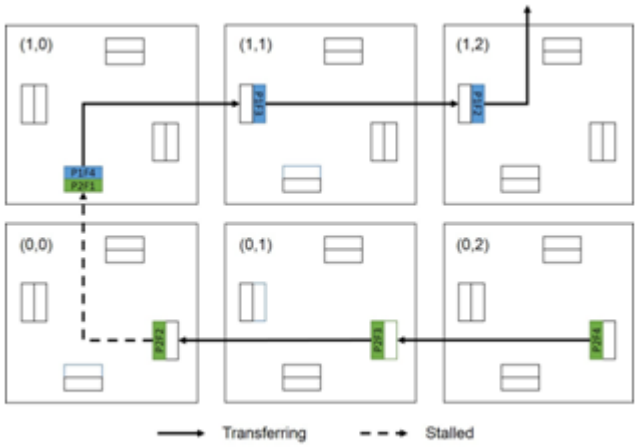
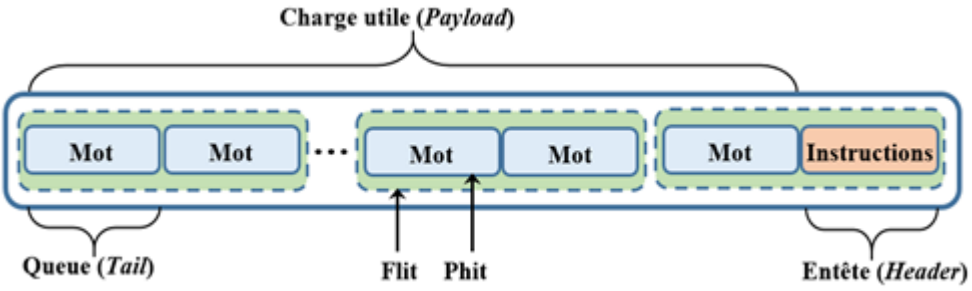


FIGURE 1.9 – La structure d'un paquet NoC.



le réseau pour atteindre leur destination. Un paquet comporte principalement deux parties, la première partie sert à contenir des informations de contrôle et de routage et est appelée entête (header) et l'autre partie, appelée charge-utile (payload), contient des données. Parfois, les paquets contiennent également une queue (tail) indiquant la fin du paquet. Un paquet peut être de taille fixe ou variable et il peut être décomposé en unités de données plus petits appelés flits.

3. **Unités de contrôle de flux, ou Flow Control Unites (Flit) :** un flit est la plus petite quantité d'information (le plus petit morceau du message) pour lequel on peut définir un contrôle de flux. Il se compose d'un nombre constant de bits. Un flit peut être formés d'un ou plusieurs phits (sur l'exemple de la figure 1.9 un flit contient deux phits).
4. **Unités de transfert physique, ou Physical Transfer Unites (Phit) :** en fonction de la taille du flit et de la largeur des liens, plusieurs cycles peuvent être nécessaires pour transmettre un flit. Un phit représente donc la quantité d'information que l'on peut transmettre en un cycle (typiquement un flit a le même nombre de bits que de fils dans un lien).

Exemple : *Considérant qu'un paquet d'une taille fixe de 512 bits est utilisé dans le NoC. La taille du flit est fixée à 32 bits. Un paquet peut donc être décomposé en 16 flits. Si 32 fils sont disponibles entre tous les routeurs voisins du réseau, alors la taille du phit sera également de 32 bits. Mais si seulement 16 fils sont disponibles, la taille du phit sera alors de 16 bits et chaque flit sera ensuite transféré entre les routeurs en deux étapes de communication (chaque flit sera composé de deux phits).*

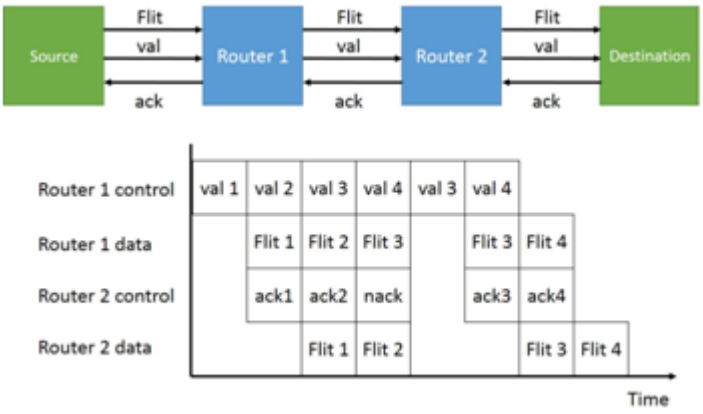
1.2.2 Le contrôle de flux

Le contrôle de flux détermine comment les ressources, telles que les buffers et la bande passante sont alloués, et comment les collisions de paquets sont résolues[72]. Chaque fois que le paquet est tamponné, bloqué, abandonné ou mal acheminé, cela dépend de la stratégie de contrôle de flux. Une bonne stratégie de contrôle doit être en mesure d'éviter la congestion des canaux tout en réduisant la latence. **Stop & Go, Credit-based, et ACK / NACK** ce sont les stratégies le plus utilisés dans NoC [72] et sont expliqués dans cette sous-section.

ACK/NACK : Dans la technique ACK / NACK (Figure 1.10), lorsque les flits sont envoyés sur un lien, une copie est conservée dans un buffer par l'émetteur. Quand un flit arrive à un buffer du routeur qui suit le routeur émetteur, si le buffer de ce dernier a l'espace disponible, le flit est accepté et un signal d'accusé de réception (ACK) est renvoyé à l'émetteur. Au lieu de cela, s'il n'y a pas d'espace disponible, le flit est supprimé et un acquittement négatif (NACK) est envoyé. Le flit doit être conservé à son origine jusqu'à

ce qu'il reçoive un accusé de réception positif. Quand un ACK est reçu par l'émetteur, il supprime sa copie du flit de son buffer. Quand un NACK est reçu, l'émetteur rembobine sa file d'attente de sortie et commence à renvoyer des flits à partir de celle qui est interrompue. ACK / NACK peut être mis en œuvre soit de bout en bout entre l'émetteur et le récepteur, soit de routeur en routeur.

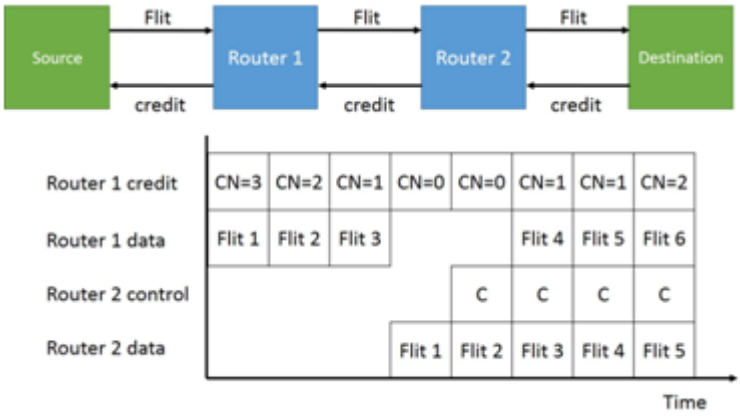
FIGURE 1.10 – Contrôle de flux ACK/NACK.



Stop & Go : Le protocole STOP & GO utilise deux liens entre chaque paire d'émetteur et de récepteur. Quand le tampon mémoire du récepteur a suffisamment de place pour recevoir une donnée, il fait passer le signal GO à 1, et dès que le tampon mémoire du récepteur n'a plus assez de place, ce dernier fait passer le signal STALL à 1. C'est à dire lorsque l'espace occupé dans le buffer récepteur atteint le seuil d'arrêt (Stop), un signal est envoyé à l'émetteur afin d'arrêter la transmission, en prenant en compte ce qui reste encore de la taille de la mémoire du buffer si elle est suffisante pour les flits qui sont toujours transmis par l'émetteur. Lorsque l'occupation du buffer diminue jusqu'à devenir inférieure ou égale au deuxième seuil, go, un autre signal est envoyé pour réactiver le flux de flits.

Credit-based :(Figure 1.11) Dans ce protocole l'envoi des données à partir de l'émetteur a lieu instantanément sans devoir attendre un acquittement de la part du récepteur. La disponibilité de ce dernier est détectée par un signal dit (credit), aussi la communication dans ce cas de protocole est asynchrone, en effet l'émission de données est synchronisée avec l'horloge de récepteur alors que l'écriture dans la mémoire tampon du routeur récepteur est synchronisée avec l'horloge de l'émetteur. Sachant que le compteur est initialisé à la capacité de la mémoire buffer du récepteur et il décrémente chaque fois qu'un paquet est envoyé. Lorsque le récepteur supprime un paquet de ses buffers d'interface, il envoie un crédit à l'expéditeur, ce qui ajoute alors à son compteur de crédit. Un compteur de crédit est installé au niveau de chaque routeur qui suit l'espace libre disponible dans le buffer du récepteur.

FIGURE 1.11 – Contrôle de flux Crédit-based.



Illustrant un exemple de ce contrôle de flux. Les nœuds amont ont des informations sur le nombre d'emplacements vides dans les buffers aval. Nous appelons cette information CN (numéro de crédit). Chaque fois qu'un nœud en amont envoie à des tampons en aval, le nombre est décrémenté de un. Lorsque les serveurs en aval envoient des signaux à d'autres nœuds, ils envoient également un signal de contrôle de crédit aux routeurs en amont, et lorsque le routeur en amont reçoit le signal, le CN associé au chemin est incrémenté de manière appropriée. La Figure 1.11 illustre le flux de données et un exemple de transmission.

Le protocole de communication (credit-based) montre deux avantages ; le premier avantage pouvoir de transmettre les données au bout d'un seul cycle, le deuxième avantage c'est l'aspect asynchrone de la communication, ce qui facilite la mise en place d'un mode globalement asynchrone localement synchrone (GALS) au niveau du fonctionnement du réseau, le seul inconvénient présenté par le protocole credit-based est la difficulté de l'implémentation au niveau du routeur ceci est expliqué par le nombre supplémentaire des signaux mis en jeu.

Ces protocoles permettent de gérer l'ensemble des règles qui contrôlent la communication entre les différentes entités au sein de réseau, il permettent aussi de valider et de synchroniser la circulation de données entre ces derniers.

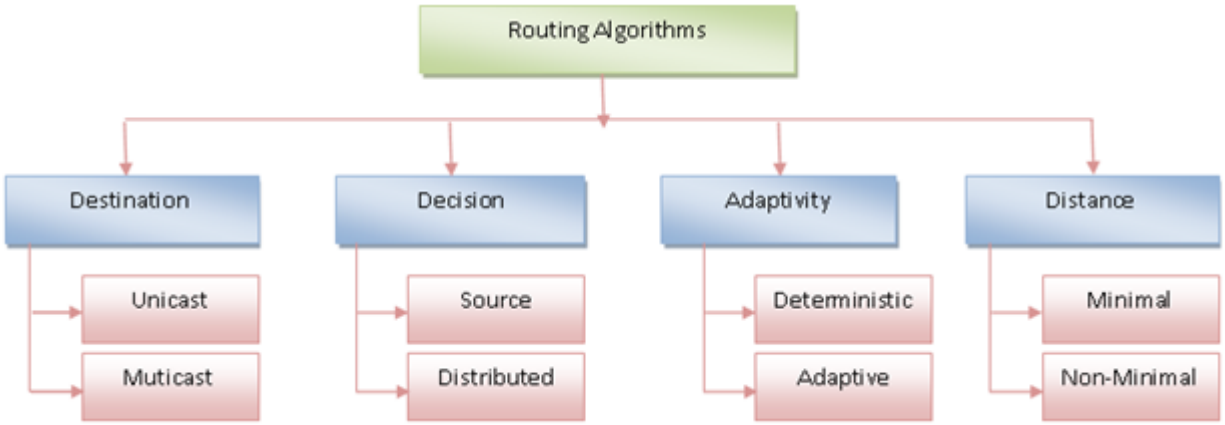
1.3 Classification des techniques de routage

Comme nous l'avons définit précédemment, la topologie est l'organisation spatiale du réseau composé par les routeurs. L'algorithme de routage est l'étape qui suit la détermination de la topologie du réseau. En fait, une topologie donnée définit les chemins disponibles entre l'ensemble des nœuds. L'algorithme de routage décide quel chemin le message doit

prendre afin d'être acheminé efficacement de sa source à sa destination. Le choix de l'algorithme de routage revêt une très grande importance dans le but d'optimiser la performance du réseau. Le but de l'algorithme de routage est de répartir le trafic de la même manière entre les chemins fournis par la topologie du réseau, améliorant ainsi la latence du réseau et le débit. Les algorithmes de routage peuvent être classés selon plusieurs critères[36].

Dans la littérature on trouve une diversité d'algorithmes de routage appliqués dans le développement des réseaux sur puce qu'on peut classer selon différents critères (Voir Figure 1.12), La figure suivante présentée ci-dessous montre ces différents critères :

FIGURE 1.12 – Classification des algorithmes de routage dans les NoCs.



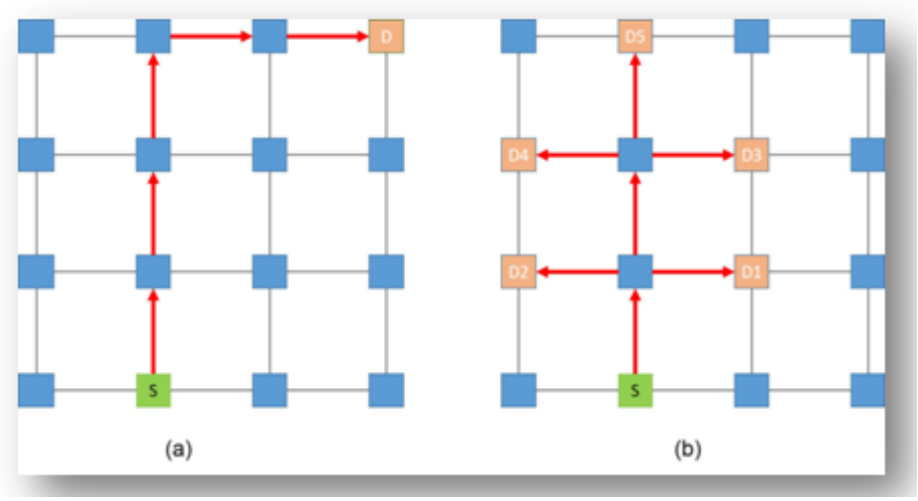
1.3.1 Routage unicast vs multicast

Selon le nombre de destination, vers lesquels les paquets seront routés, les algorithmes de routage peuvent être classés en routage unicast ou en routage multicast, comme indiqué dans la figure en 1.13. Dans le routage unicast les paquets sont envoyés d'un seul nœud source vers un seul nœud de destination. En revanche, le routage multicast envoie les paquets d'un seul nœud vers plusieurs nœuds de destination.

1.3.2 Routage source vs distribué

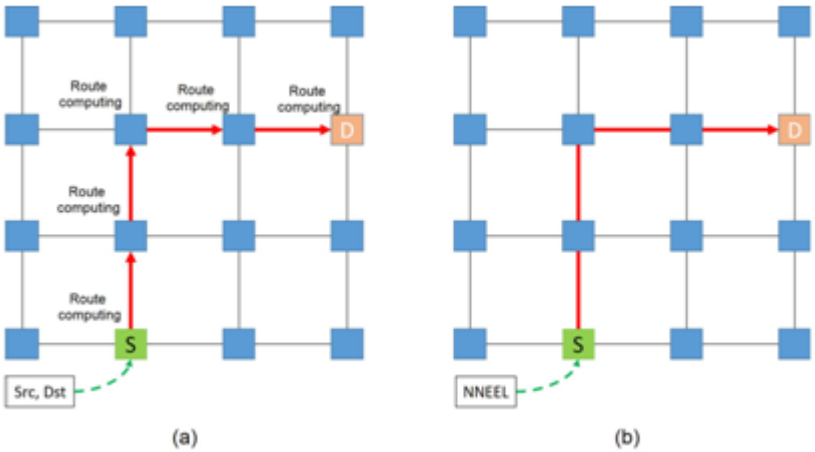
Le routage dans le NoC peut être source ou distribué (Figure 1.14). Dans le routage source, l'itinéraire de l'ensemble du trajet depuis la source vers la destination est pré-calculé et fourni dans l'entête du paquet contrairement au routage distribué, où l'entête du paquet ne contient que l'adresse de destination et que la trajectoire est calculée dynamiquement par la participation des routeurs intermédiaires du chemin. Dans le routage distribué, de

FIGURE 1.13 – Routage Unicast vs Multicast.



multiples chemins de la source vers la destination sont possibles. Quand un paquet entre dans un routeur, l'adresse de destination est lue à partir de l'entête et en conséquence, la fonction de routage calcule tous les ports de sortie possibles où ce paquet peut être transmis. Ensuite, une fonction de calcul d'itinéraire sélectionne un des ports de sortie admissible pour transmettre le paquet. La sélection du port de sortie dépend des conditions dynamiques du réseau tel que la congestion et les défauts dans les liens.

FIGURE 1.14 – Routage Source vs Distribué.



Avec le routage source, toutes les décisions de routage sont prises à l'intérieur du nœud source avant l'injection de tout paquet dans le réseau. A cet effet, chaque source contient des listes ou des tableaux qui contiennent des informations sur l'itinéraire complet pour atteindre toutes les autres ressources dans le réseau.

Afin d'acheminer un paquet à travers le réseau en utilisant le routage source, la ressource expéditrice consulte sa table de routage pour obtenir le chemin complet pour l'accès à la destination souhaitée. Ce chemin est alors écrit dans le champ dédié dans l'entête du paquet.

Le paquet doit suivre le chemin en traversant le réseau vers sa destination. Chaque routeur qui reçoit ce paquet lit le champ de chemin dans l'entête du paquet et le transmet au port de sortie destiné. Contrairement à un routeur utilisé dans le routage distribué, ce routeur ne nécessite aucun calcul supplémentaire n'est nécessaire pour prendre les décisions de routage du fait que les paquets contiennent déjà des décisions pré-calculées.

Le routage source n'est pas beaucoup utilisé jusqu'à présent pour les NoC, en raison de son apparente surcharge (overhead) pour stocker les informations du chemin dans l'entête. Cependant, les chemins dans le routage source sont pré-calculés hors ligne (offline), **donc le routage source ne peut fournir qu'une adaptabilité, très limitée, au chemin, en cas de panne ou de congestion de trafic.**

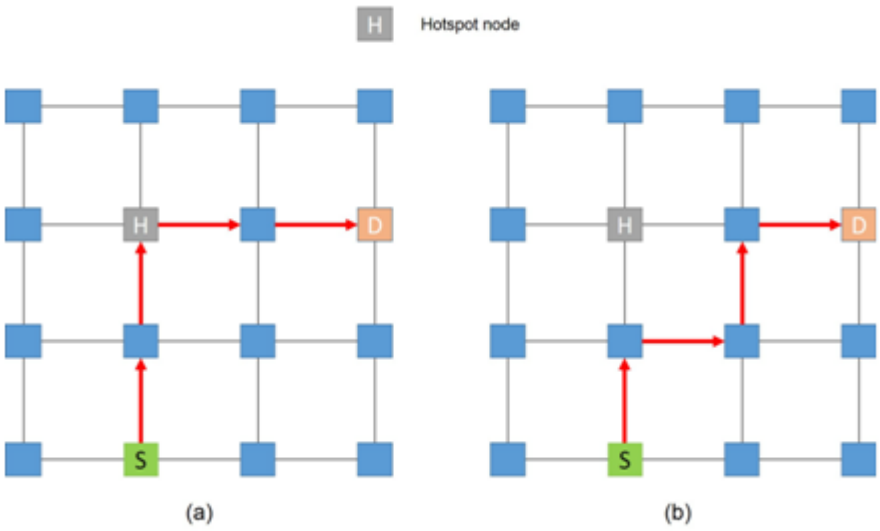
1.3.3 Routage déterministe vs Adaptatif

Du point de vue adaptabilité, on peut classer les algorithmes de routage en deux catégories ; les algorithmes de routage déterministe et les algorithmes de routage adaptatifs. Les algorithmes de routage déterministes choisissent toujours le même chemin pour transférer les paquets. Ces algorithmes sont utilisés dans les réseaux réguliers et irréguliers. Dans le cas le plus simple, chaque routeur dispose d'une table de routage qui indique les routes à tous les autres routeurs du réseau. Lorsque la structure du réseau change, chaque routeur doit actualiser sa table. Ce type de routage ne tient pas compte de la diversité de chemins du réseau et n'est pas sensible à l'état du réseau. Cela peut entraîner des déséquilibres de charge dans le réseau en cas de défaillance. L'avantage qu'il est simple et peu coûteux pour le mettre en œuvre. Un autre point sa capacité à éviter le problème d'inter-blocage (Deadlock).

Parmi les algorithmes les plus connus, il y a le routage en XY. Dans l'algorithme de routage XY, le paquet est toujours acheminé en premier dans la direction des « X » jusqu'à ce qu'il atteigne le nœud qui a la même coordonnée « X » du nœud de destination. Ensuite, il est acheminé dans la direction des « Y » jusqu'à atteindre le nœud de destination. En cas de défaillance au niveau des liens et/ou des routeurs, le routage en XY ne peut pas acheminer les paquets.

Le routage adaptatif est l'algorithme de routage le plus sophistiqué, dans lequel le chemin qu'un message doit prendre dépend de la situation du trafic et de l'état de réseau. Par exemple, dans le cas ou un message trouve un état de congestion et/ou une défaillance

FIGURE 1.15 – Routage Deterministe vs Adaptatif.



au niveau du nœud et/ou du lien ; le message doit choisir un nouveau chemin à emprunter vers la destination.

Il y a deux types de routage adaptatif : le routage adaptatif minimal et le routage adaptatif non minimal (Figure 1.16). Dans la première catégorie la route empruntée par les paquets est toujours la plus courte. L'algorithme est efficace lorsqu'il existe plusieurs chemins courts entre chaque paire source/destination. Dans ce cas l'algorithme utilise toujours le chemin le moins encombré. En revanche, la deuxième catégorie utilise toujours une route non encombrée. Cet algorithme ne prend pas en compte la longueur du chemin entre l'émetteur et le récepteur. Typiquement, un algorithme de routage adaptatif choisit toujours l'état du réseau c.à.d. les chemins libres de la congestion, le chemin le plus court est le meilleur (sain).

L'inconvénient de ces algorithmes de routage c'est qu'ils sélectionnent les chemins qui peuvent augmenter le nombre de sauts entre la source et la destination ce qui augmente la latence.

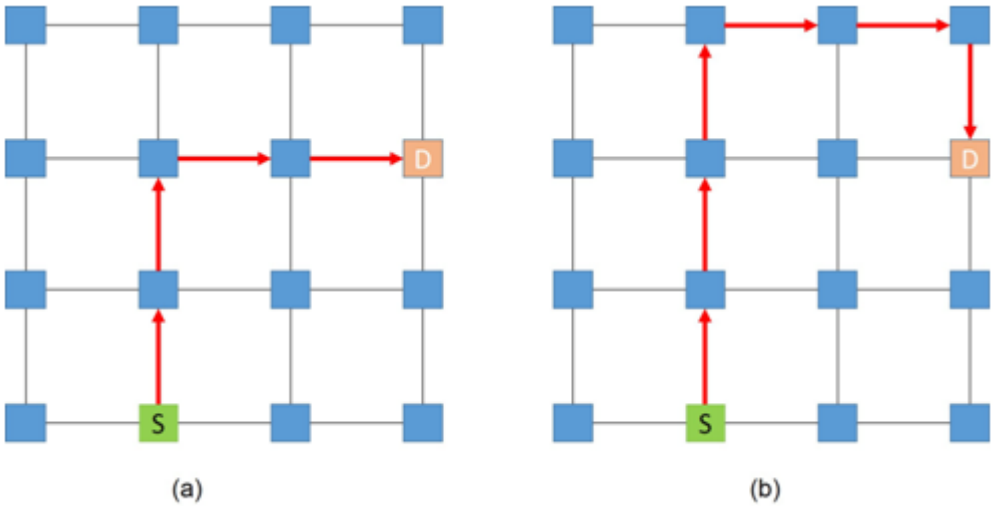
1.3.4 Routage statique vs dynamique

Dans le routage statique, le chemin ne peut pas être changé après qu'un paquet ait quitté la source. Dans le routage dynamique, un chemin peut être modifié à tout moment au cours du routage en fonction des conditions de fonctionnement du réseau.

1.3.5 Routage Minimal vs non-minimal

Un routage qui utilise les plus courts chemins possibles pour la communication est appelé routage minimal. Il est également possible d'utiliser des chemins plus longs pour le transfert de données à partir de la source vers la destination. Cette possibilité résulte de l'adaptabilité offerte par l'algorithme de routage. Le type de routage qui utilise des chemins plus longs pour la communication bien que les plus courts chemins existent est connu comme routage non-minimal. **Le routage non-minimal a certains avantages par rapport au routage minimal, comme la possibilité d'équilibrer la charge du réseau et la tolérance aux pannes.**

FIGURE 1.16 – Routage Minimal vs Non Minimal.



En outre, le routage dépend fortement de la topologie du réseau qui traduit la manière dont les liens de communication connectent les différents nœuds du système. Plusieurs topologies plus ou moins complexes ont été proposées pour différents systèmes. La complexité de la topologie est liée au nombre de liens physiques qu'elle engendre, reflétant son coût, et la distance maximale entre deux éléments distants du système (diamètre du réseau) reflétant sa latence. Le choix de la topologie peut avoir un impact sur les performances du système notamment si le nombre d'éléments connectés est important et la latence est critique. Ces topologies convergent vers des formes régulières et structurées (maille, tore) dans le but de faciliter l'acheminement des paquets ainsi que le passage à l'échelle.

1.4 Les problèmes de routage

Il y a quelques propriétés des algorithmes de routage qui sont essentiellement requis pour les réseaux d'interconnexion à savoir la connectivité, l'adaptabilité, la liberté face aux inter-blocages (deadlock) et livelock, ainsi que la tolérance aux pannes[74]. La connectivité est la capacité d'acheminer des paquets provenant de n'importe quel noeud source à n'importe quel nœud destination. L'adaptabilité est la possibilité d'acheminer des paquets à travers des chemins alternatifs dans la présence des contentions. La liberté de Deadlock et de livelock est la capacité de garantir que les paquets ne seront pas bloqués ou de circuler à travers le réseau pour toujours sans la possibilité d'atteindre la destination. La tolérance aux pannes est la capacité d'acheminer les paquets en présence de commutateurs défectueux.

1.4.1 Deadlock (Inter-blocage)

Un blocage se produit quand un ou plusieurs paquets ne peuvent pas avancer vers leurs destinations parce que le buffer demandé par le message est saturé, bloqué par un autre message qui attend aussi (figure 1.17). En effet, cela se produit lorsqu'un routeur est en attente d'envoyer un paquet vers un autre routeur qui est lui aussi en attente et ainsi de suite, jusqu'au routeur initial.

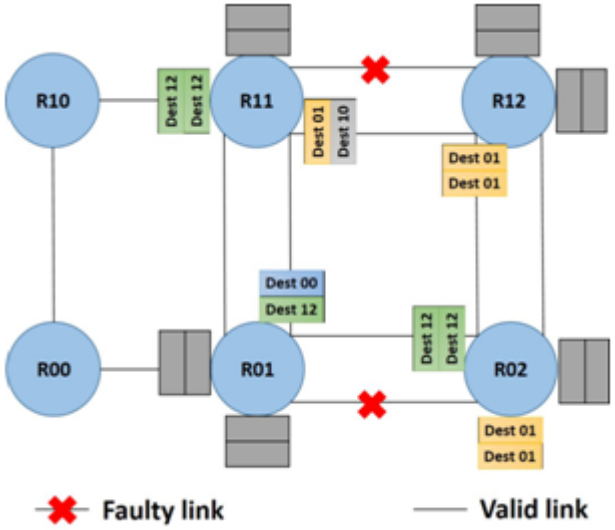
La technique de commutation de paquet Wormhole est la plus encline à l'inter-blocage (deadlock). On peut éviter ce genre de problème par l'application d'un algorithme de routage approprié par exemple le fait d'interdire certains virages dans une topologie Mesh[51], ou encore grâce à l'application des canaux virtuels.

La Figure 1.17 illustre un exemple de blocage dans un NoC. la dépendance est provoquée par l'échange de flits entre R02 et R01. En raison de la présence de défauts, les choix pour un routage minimal sont limités et les deux communications sont dépendantes les unes des autres ; ainsi, aucun d'entre eux ne peut progresser sur le réseau. Sur la même figure, nous pouvons voir que les flits Dest10 et Dest00, stockés respectivement dans les ports d'entrée de R11 et R01, sont victimes de cette impasse ; c'est-à-dire que même leurs canaux de sortie sont libres, ils doivent attendre dans le tampon jusqu'à ce que le blocage soit résolu.

La présence de défauts de liaison ou de routeur peut entraîner des blocages qui ne se produisent pas en fonctionnement normal. La plupart des mécanismes permettant d'éviter les inter-blocages tentent d'éviter la formation de cycles, généralement en restreignant les tournants ou en utilisant des canaux virtuels (VC) :

—**Turn Model** interdit aux paquets de prendre des virages particuliers pour éviter les

FIGURE 1.17 – Un exemple de Deadlock dans un NoC adaptatif.



cycles[32]. Dans l'exemple précédent, si un paquet n'avait pas tourné dans le sens contraire des aiguilles d'une montre, le blocage aurait été évité. Dans le modèle de tournants, tous les tournants possibles que les paquets peuvent prendre dans un réseau sont analysés avec les cycles que ces tournants pourraient causer. Un graphique de dépendance acyclique est généré, ce qui crée un réseau sans blocage.

—**Les canaux virtuels** peuvent être utilisés dans le routage adaptatif pour éviter les inter-blocages. Des VC peuvent être ajoutés pour reconnecter les VC du réseau afin de fournir essentiellement des tampons supplémentaires pour la propagation de paquets[26]. Dans l'exemple précédent, si l'un des paquets avait été placé dans un VC différent, le cycle aurait disparu et donc l'impasse. Le nombre de VC est généralement réduit au minimum car ils imposent une complexité, et donc des frais généraux de surface et de puissance, à la micro-architecture du routeur.

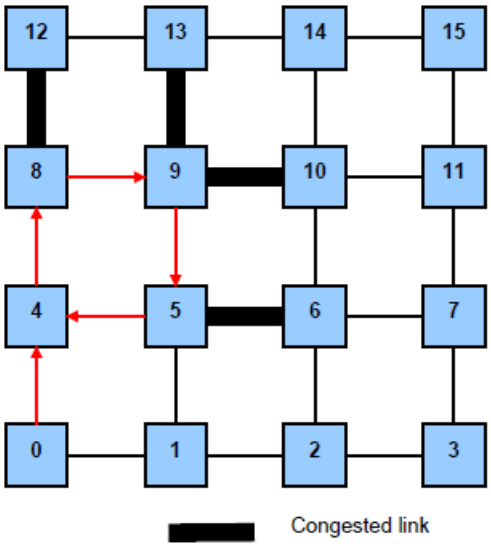
1.4.2 Livelock (Inter-blocage dynamique)

Ce genre de problème survient lorsque les paquets circulent autour d'une destination sans l'atteindre. Donc, les données ne peuvent pas être acheminées vers la destination. Ce genre de problème se produit dans le cas des algorithmes de routage non minimaux où on cherche à délivrer le paquet sans prendre en considération le plus court chemin. Ce genre de problème doit être évité.

Cependant, il est possible que les paquets ne soient pas dans une configuration de blocage, et ils peuvent être dans livelock. Un paquet peut se déplacer autour de sa destination

et ne jamais l'atteindre. Le livelock se produit car à chaque fois, les canaux demandés par le paquet pour atteindre la destination sont conservés par d'autres paquets. Pour un routage minimal, livelock ne se produit jamais car, à chaque nœud, un paquet avance d'un pas vers sa destination et enfin l'atteint. Cependant, dans le cas d'un routage non minimal, des paquets peuvent suivre des chemins non minimaux et un livelock peut se produire si aucune mesure n'est prise pour garantir la progression.

FIGURE 1.18 – Un exemple de Livelock.



La figure 1.18 montre un exemple de livelock. Le noeud source est 0 et le noeud de destination est 12. Un paquet de 0 à 12 trouve un encombrement à 8 et est mal acheminé à 9. Au noeud 9, il trouve plus d'encombrement et est de nouveau égaré à 5. Cela provoque un cycle dans lequel le paquet fait deux pas en avant de 5 à 8, puis deux pas en arrière de 8 à 5.

1.4.3 Situation de starvation

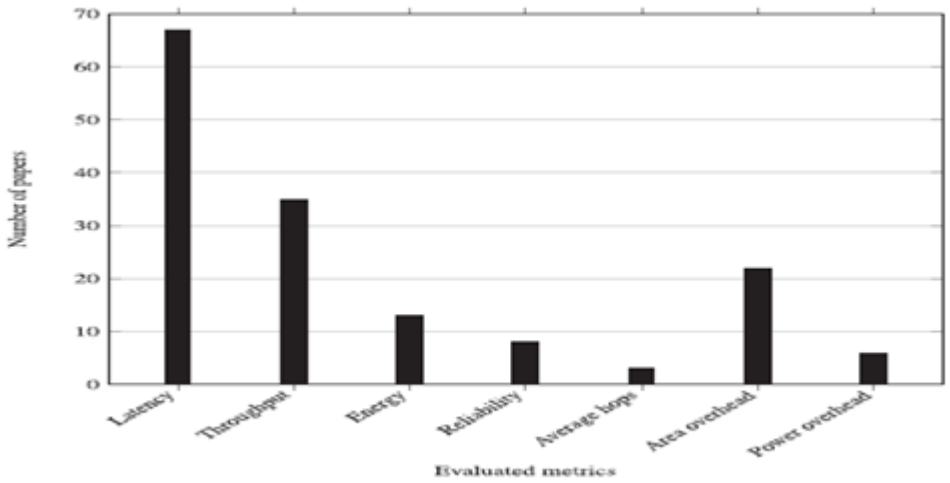
L'attribution des priorités différentes pour les paquets de données dans le NoC peut causer ce genre de problème. Donc les paquets de priorité élevée peuvent facilement atteindre leur destination, alors que les paquets à faible priorité ne seront jamais délivrés à leur destination. Ce genre de situation se produit parce que les paquets de haute priorité réservent les ressources et les paquets de faible priorité restent en attente de la disponibilité de ces derniers. Ce type de problème peut être évité en utilisant un algorithme de routage approprié qui réserve une partie de la bande passante de réseau au service des paquets à faible priorité

1.5 Paramètres de performance du NoC

Une métrique de routage est une donnée quantitative ou qualitative utilisée pour évaluer le coût d'un chemin. Le « meilleur chemin » du point de vue du routage est celui qui satisfait toutes les contraintes (lorsqu'il en existe) et possède le coût le plus faible au regard de la métrique utilisée. Il faut noter que la métrique de routage et contrainte ne sont pas exclusives, l'une par rapport à l'autre. Par exemple, on pourrait construire des routes offrant le nombre de sauts minimum, tout en éliminant les liens dont le délai excède une valeur donnée. La valeur de la métrique est influencée par différents facteurs. Elles peuvent être liés à l'environnement ou émaner du réseau lui-même. Les facteurs environnementaux sont ceux qui ne sont pas sujets aux conditions du trafic. On peut citer par exemple, la position du nœud destinataire. Les facteurs émanant du réseau sont ceux qui dépendent directement ou indirectement du trafic généré par le réseau. Parmi ceux-ci on retrouve de façon non exhaustive, la congestion.

Comme d'autres réseaux, la performance d'un NoC est évaluée par de nombreux critères tels que la latence, le débit, la consommation d'énergie, la probabilité de perte de paquets, la tolérance aux pannes, la superficie du routeur, etc(Figure 1.19). Ces performances sont fortement liés à l'architecture du réseau NoC lui-même, il est donc primordial d'avoir une bonne compréhension de ces métriques et de la façon de les évaluer, afin d'être en mesure d'optimiser l'architecture du réseau. Certains de ces paramètres sont présentés ci-dessous.

FIGURE 1.19 – Mesures de performance des NoCs. [79]



1.5.1 La latence

La latence est sans doute l'un des critères les plus importants servant à la caractérisation des NoCs. La latence est définie par le délai moyen (en nombre de cycles) nécessaire pour transférer une unité de données (message, paquet ou flit) de la source vers la destination. Plus précisément, pour un seul paquet, la latence est le nombre de cycles écoulés depuis la première tentative d'injection du flit d'entête (header) jusqu'à la réception du dernier flit (tail) par la destination[26]. La valeur de la latence va dépendre de deux choses : la position relative de l'émetteur par rapport au récepteur et la congestion dans le réseau. En effet, cette dernière aura des conséquences quant aux choix d'arbitrages et de routages, qui pourront faire augmenter la latence des paquets.

1.5.2 Le débit

C'est le nombre total de paquets qui atteignent leur destination par unité de temps.

1.5.3 La consommation d'énergie

La consommation d'énergie est un autre paramètre qui joue un rôle important dans la performance de tous types de réseaux. Les algorithmes de routage du NoC sont également évalués sur la base de l'énergie consommée par les routeurs correspondants.

1.5.4 La diversité de chemin

La diversité de chemin[72] est une métrique qui va caractériser la capacité du réseau à être tolérant aux fautes. En effet, un réseau offre de la diversité de chemin si pour tous (ou presque tous) les couples (source/destination), il existe plusieurs chemins entre eux. C'est pourquoi, lorsque la diversité de chemin est faible ou inexistante, cela peut entrainer des problèmes de contentions à travers tout le réseau.

1.5.5 La perte de paquet

La perte de paquets est également utilisée pour évaluer la performance des algorithmes de routage dans le NoC. Elle est définie comme la probabilité de perdre de paquets dans le NoC.

1.5.6 La tolérance aux pannes

La tolérance aux pannes est la capacité d'un algorithme de routage d'acheminer les paquets sans la présence de défauts dans les liens/routeurs. Il s'agit d'une mesure du nombre et de type de fautes tolérées par l'algorithme.

Avec l'évolution des technologies, l'intégration des systèmes sur puce ne cesse d'augmenter et avec elle, le nombre de composants à interconnecter. En conséquence, il est important que les réseaux sur puce soit plus fiable. Cette fiabilité du réseau représente son aptitude à fonctionner même dans la présence des défauts afin de maximiser les performances du système. A cet effet, on va focaliser sur ces trois dernières mesures dans ce manuscrit.

Voiçi les principaux NoCs rencontrés dans les domaines de la recherche et de l'industrie, nous pouvons citer :

— Le réseau Æthereal[39] développé par *Philips*
— Le réseau MANGO[19] (Message-passing Asynchronous Network-on-chip providing Guaranteed services over OCP interfaces) de la Technical University of Denmark
— le réseau Proteo[73] qui est le fruit d'un consortium entre les universités finlandaises (Tampere University of Technology et University of Turku) et l'institut suédois (Stockholm Royal Institute of Technology)
— Le réseau QNoC[24] développé par le Technion-Israel Institute of Technology
— Le réseau SPIN[2] (Scalable Programmable Integrated Network) créé au sein de l'Université Pierre et Marie Curie
— Le réseau STNoC[1] ou Spidergon conçu par STMicroelectronics
— Le réseau XPIPES[18] élaboré par l'Université de Bologne
— Le réseau uSpider II[25] élaboré à l'Université de Bretagne Sud

Le tableau 1.2 récapiitule les principales caractéristiques de ces NoCs. Nous remarquons une homogénéité dans certains choix de protocoles de communication. Le type de commutation et la politique de mémorisation sont similaires pour tous les NoCs excepté le NoC MANGO. Concernant l'algorithme de routage utilisé, il est principalement déterministe excepté pour le STNoC et le SPIN. La topologie la plus présente est le mesh-2D (grille 2D).

A partir de ces informations, il est possible de spécifier les caractéristiques communes des NoCs à savoir :

1. Topologie : grille 2D
2. Gestion de flux : crédit d'émission
3. Type de commutation : paquet
4. Mémorisation : wormhole
5. Algorithme de routage : déterministe

Réseau	Topologie	Gestion de flux	Type de Commutation	Politique de Mémorisation	Algorithme de routage
Æthereal	Grille 2D	Crédit	paquet	WormHole	Déterministe
MANGO	Grille 2D	/	Circuit/paquet	Circuit Virtuel	Déterministe
Proteo	Tore	Non spécifie	paquet	WormHole	Déterministe
QNOC	Grille 2D	Crédit	paquet	WormHole	Déterministe
SPIN	Arbre éargi	Crédit	paquet	WormHole	Déterministe
STNOC	spécifique	Non spécifie	paquet	WormHole	Non spécifie
XPIPES	Grille 2D-Tore	ACK/NACK	paquet	WormHole	Déterministe
uSpider	Irrégulière	Non spécifie	paquet	WormHole	Déterministe

TABLE 1.2 – Comparaison de quelques caractéristiques de plusieurs NoCs.

1.6 Conclusion

Dans ce premier chapitre, nous avons présenté les principales caractéristiques des réseaux sur puce. Même si on présente une solution plus optimisée que celle du but et qu'elle est aussi acceptée par l'industrie, elle est toutefois encore en cours de développement et fait face à certains défis de conception, parmi eux on cite (ces paramètres doivent être justifiés par le concepteur), la topologie du réseau, le type de commutation choisi, l'algorithme de routage à appliquer, le protocole de communication (contrôle de flux), la structure du routeur, ainsi nous avons soulevé un certain nombre de problématiques concernant les algorithmes de routage qui s'articule autour d'un réseau sur puce dans le contexte des systèmes sur puce.

Les performances globales de NoC dépendent de plusieurs paramètres de réseau, tels que la topologie, la technique de commutation, le contrôle de flux et les stratégies de routage. Cette thèse est axée sur les algorithmes de routage. Un algorithme de routage affecte plusieurs exigences non fonctionnelles d'un système basé sur le NoC. Performance, fiabilité, consommation d'énergie, la dissipation de puissance, les aspects thermiques et la tolérance aux pannes font partie des principaux paramètres affectés par l'algorithme de routage. Des recherches importantes ont été publiées sur l'amélioration des algorithmes de routage dans le domaine des réseaux sur puce. Les principaux problèmes abordés incluent le développement de micro-architectures de routeur hautes performances, tolérantes aux pannes et à faible coût (puissance et surface), l'élaboration de politiques de sélection tenant compte de la bande passante et des conflits, et la conception de fonctions de routage hautement adaptatives et sans blocage. Dans tous les cas, la fonction de routage (une phase de l'algorithme de routage) a un effet significatif sur les performances globales de tout algorithme de routage.

La tolérance aux fautes pourrait être appliquée à plusieurs niveaux : niveau circuit,

niveau système, niveau routage. Dans ce travail nous nous concentrons uniquement sur la tolérance aux fautes au niveau routage. Dans le chapitre suivant, nous allons présenter un état de l'art sur les algorithmes de routage tolérant aux pannes dans les réseaux sur puce.

Chapitre 2

Etat de l'art sur les Algorithmes de Routage Tolérant aux Fautes dans les NoCs.

Sommaire

Dans le premier chapitre, nous avons introduit les notions de base utilisées dans cette thèse, afin de donner au lecteur le vocabulaire et les éléments nécessaires à la manipulation de la thématique globale étudiée. La première partie définit la terminologie de base liée à la sûreté de fonctionnement. La terminologie est principalement extraite de [36]. Ensuite, nous identifions les principales méthodes et leurs caractéristiques pour rendre un système tolérant aux fautes. Enfin la dernière section de ce chapitre s'intéresse à l'étude bibliographique à la base de cette recherche. Les domaines pertinents liés à cette étude comprennent les travaux les plus récents sur l'optimisation des performances en termes de fiabilité dans les réseaux NoC.

2.1 Introduction

La constante miniaturisation des transistors a mené les réseaux sur puce, et même tous les systèmes, à devenir de plus en plus vulnérables aux fautes. Il est devenu essentiel pour rencontrer un certain niveau de fiabilité d'intégrer des méthodes de tolérance aux fautes dans les systèmes. Les fautes peuvent affecter les circuits de différentes façons. Elles peuvent avoir des effets permanents comme des défauts de fabrication par exemple. Elles peuvent également engendrer de mauvais fonctionnements ayant un comportement transitoire comme dans le cas d'interférences magnétiques.

L'objectif est que les systèmes remplissent leurs fonctionnalités même en présence de fautes. Dans le cas des réseaux sur puce, la fonctionnalité principale est de transmettre des messages entre les différents ssources. La quête de la tolérance aux fautes n'est pas nouvelle en soi dans le domaine des réseaux sur puce.Plusieurs recherches ont été menées sur ce sujet. Le but ultime est qu'aucune donnée ne soit altéré, autrement dit, qu'aucun paquet lors de la transmission ne soit modifié ou, tout simplement, n'arrive pas à destination.

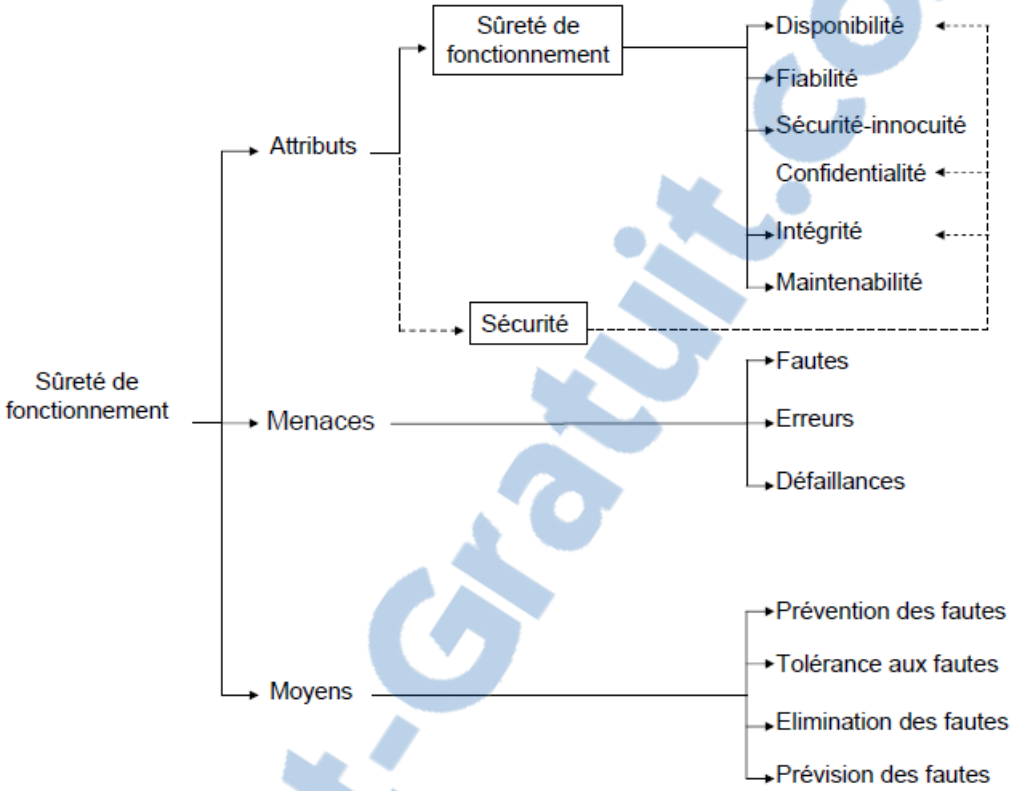
Les réseaux sur puce sont plus sensibles à différents facteurs qui ont un impact considérable sur leur rendement et fiabilité. Les sources de ces défaillances sont : les variations de processus, les variations pendant la durée de vie et le bruit intrinsèque. Des fautes non détectées ou non traitées peuvent se propager aux niveaux supérieurs de la pile protocolaire de communication et peuvent influer sur le bon fonctionnement des différents composants du système (routeurs et liens), ou peuvent conduire à la défaillance totale du système. Donc, il est nécessaire d'ajouter des mécanismes de tolérance aux pannes durant la conception, en particulier pour les applications critiques, comme dans les secteurs de l'automobile et l'aviontion. A cet effet, la tolérance aux fautes est un besoin important – et croissant – des nouvelles générations de systèmes.La section suivante sera consacrée à présenter les concepts de base de la sureté de fonctionnement.

La tolérance aux fautes est l'aptitude d'un système informatique à accomplir sa fonction malgré la présence ou l'occurrence de fautes[37], elle apparait comme un moyen de garantir une sureté de fonctionnement.

2.2 La sureté de fonctionnement d'un système

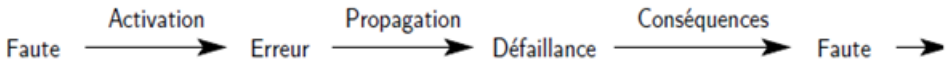
La sureté de fonctionnement d'un système informatique est sa capacité à fournir un service en lequel on peut avoir une confiance justifiée. La sureté de fonctionnement peut-être articulée autour de trois notions décrites à la figure 2.1 : **ses attributs, ses entraves et ses moyens.**

FIGURE 2.1 – L'arbre de la sureté de fonctionnement [37].



Les attributs de la sûreté de fonctionnement sont les suivants : la disponibilité, la fiabilité, la maintenabilité, la sûreté et la sécurité. Les fautes, les erreurs et les défaillances représentent les menaces de la sûreté de fonctionnement. Fautes, erreurs et défaillances sont liées par des relations de causalité illustrées sur la figure 2.2. Le mécanisme de base peut se résumer comme ceci : une faute peut entraîner une erreur qui, à son tour, peut conduire à une défaillance. Etant donné que la sortie des données d'un service peut être l'entrée d'un autre, une défaillance peut se propager d'un service à l'autre service comme une faute ; ainsi une chaîne peut être formée de la forme : faute –> erreur –> défaillance –> erreur etc.

FIGURE 2.2 – La chaîne faute-erreur-défaillance.



Une défaillance (ou panne) est l'évènement qui survient lorsque le comportement du

système dévie de sa fonction. L'erreur est la partie de l'état du système qui est susceptible d'entrainer une défaillance. La défaillance survient lorsque l'erreur affecte le service délivré à l'utilisateur. La faute est définie comme la cause adjugée ou supposée de l'erreur. Erreur et défaillance forment donc une chaîne de causalité. Pour empêcher l'occurrence de défaillances, la sûreté de fonctionnement s'intéresse à leurs causes.

Un système défaille lorsque le service qu'il délivre diffère du service attendu. La tolérance aux fautes cherche à éviter ces défaillances, ou au moins, à éviter l'arrêt total du système. Les moyens de sûreté de fonctionnement sont destinés à réduire le nombre global de défaillances d'un système. Considérant le mécanisme de la chaîne faute-erreur -défaillance, il est possible de proposer des moyens pour briser ces chaînes et améliorer ainsi la sûreté de fonctionnement du système.

La sûreté de fonctionnement met à disposition quatre moyens pour garantir la fiabilité :

— La prévention des fautes vise à empêcher l'apparition ou l'introduction des fautes dans le système. Elle repose sur des règles de développement (modularisation, utilisation de langage fortement typé, preuve formelle, etc.).
— L'élimination des fautes s'attache à réduire la présence (nombre, sévérité) des fautes. Cette méthode opère à la fois lors du développement (vérification des conditions, test de régressions, injection de fautes, etc.) ou lors de l'utilisation (maintenance).
— La prévision des fautes cherche à estimer (qualitativement et quantitativement) l'occurrence et les conséquences des fautes. Elle est réalisée par la modélisation et l'évaluation de systèmes.
— La tolérance aux fautes essaie de masquer l'occurrence des fautes et de continuer à fournir le service demandé malgré leur apparition.

Par ailleurs à ajouter des mécanismes dans le système permet la prestation des services requis, même en présence de fautes (parfois à une performance dégradée). Du fait que les fautes soient diversifiées par leur type, nature, taux et lieu d'apparition, leurs effets sont difficiles à prédire et à prévenir. Ainsi, dans les systèmes complexes à base de NoC, la tolérance aux fautes est une nécessité.

Pour mieux comprendre la notion de la tolérance aux pannes, nous introduisons la terminologie suivante :

1. Une faute est toute faiblesse (défaut) inhérente à un système qui mène à une erreur ;
2. Une erreur est un état système incorrect ou non déini que peut mener à un échec ;
3. Un échec (ou panne) est une déviation du bon fonctionnement du système ;
4. La tolérance aux pannes est la capacité d'un système fonctionnel de continuer à exécuter son fonctionnement escompté en présence de fautes ou d'erreurs.
5. La détection des pannes consiste à la détection de la fonctionnalité défectueuse dans un système par l'auto-diagnostic ou le diagnostic coopératif.

6. Le recouvrement des pannes est la récupération du fonctionnement correct, après la détection de la faute, en réparant ou en remplaçant le composant responsable de la panne.

2.2.1 Modèles de fautes

Le modèle de fautes d'un système définit la nature des fautes présentes dans le système. Il s'agit d'une étape préliminaire à la définition des mécanismes de tolérance aux fautes à mettre en place dans le système. En effet, la tolérance aux fautes devra permettre de tolérer les fautes définies dans ce système.

2.3 Les caractéristiques des fautes

Les fautes sont des événements non désirés et imprévisibles que l'on distingue par des deux caractéristiques.

1. La première caractéristique concerne l'aspect temporel des fautes désigné par deux valeurs distinctes : permanente et temporaire. Une faute permanente est présente indépendamment de conditions internes ou externes, tandis que la présence d'une faute temporaire est limitée.
2. La seconde caractéristiques se rapporte au domaine des fautes, à savoir à l'aspect spatial traduisant respectivement la localité des fautes. La caractéristique de localité dévoile des fautes internes qui appartiennent à des composants du système et qui sont considérées comme "dormantes" tant qu'elles ne sont pas activées par une action. Par opposition, les fautes externes sont dues à l'environnement physique du système

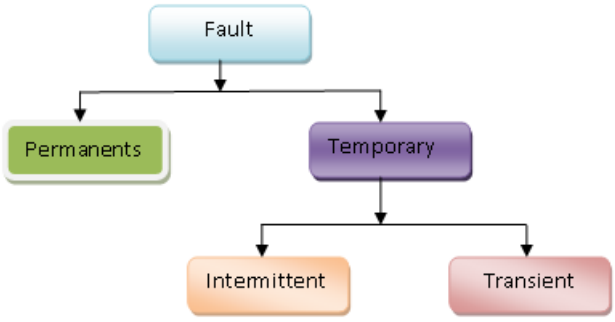
Les fautes peuvent être caractérisées par cinq attributs à savoir :la cause, la nature, la valeur, l'étendue, et la durée. La durée d'une faute spécifie la longueur de temps pendant laquelle la faute est active. Les fautes ont été classées en trois types en fonction de leur durée : **transitoire, intermittente, et permanente**.

2.3.1 Types de fautes

— Transitoire : Une faute transitoire apparaît une fois, et ne continue pas. Une erreur provoquée par une telle faute est souvent désignée comme une erreur soft. De telles erreurs peuvent conduire à des données corrompues, à l'exécution incorrecte du programme, ou à une perturbation complète d'un programme en cours [56].

— Intermittent : C'est une faute qui apparaît, disparaît, et réapparaît à plusieurs reprises dans un délai très court. Une défaillance intermittente peut se produire à plusieurs reprises, mais pas de façon continue pendant une longue période dans un dispositif.
— Permanente : Il s'agit d'une faute qui continue d'exister indéfiniment si aucune mesure corrective n'est prise. Une faute permanente, lorsqu'elle survient une fois, persiste jusqu'à la fin de l'exécution. Même une seule faute permanente peut créer de multiples erreurs jusqu'à ce qu'elle soit réparée. Ces erreurs sont appelées erreurs matérielles.

FIGURE 2.3 – Classification des fautes.



Principalement ; il existe deux grandes catégories de fautes : celles qui présentent un effet permanent et celles qui ne durent que de façon transitoire. Un routeur peut faire une erreur de routage sans pour autant corrompre la donnée, ce qui résulte à une erreur lors de la transmission du paquet dans un cas comme dans l'autre.

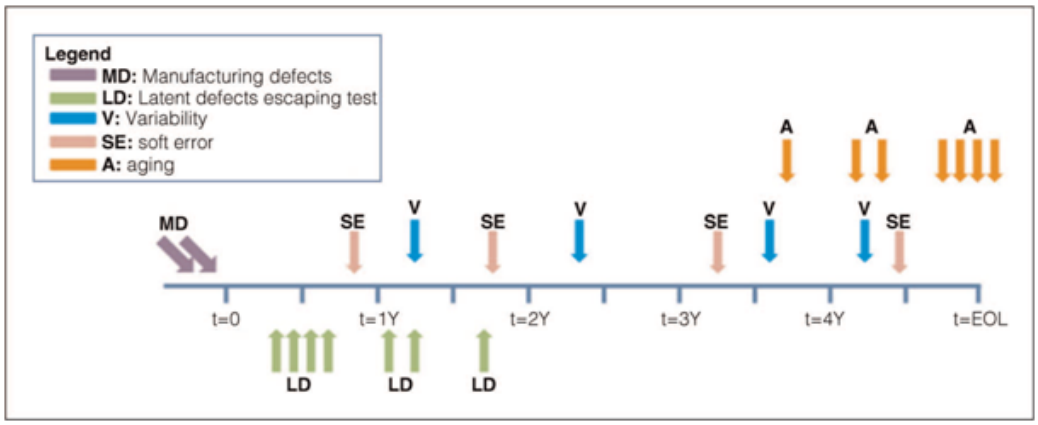
Parmi les fautes permanentes, nous pouvons noter deux types, statique et dynamique. Les fautes statiques se produisent généralement lors de la fabrication du réseau, qui nécessite des techniques de plus en plus minutieuses, et leur effet est irréversible excepté par des modifications physiques du réseau. Les fautes dynamiques, étant de nature permanente également, ont lieu de façon aléatoire durant l'exécution, contrairement aux fautes statiques. Elles peuvent être allouées aux effets de vieillissement, à l'électromigration, etc.

Les fautes transitoires sont, quant à elles, de nature non permanente. Tout comme les fautes dynamiques, elles apparaissent de façon aléatoire sur le réseau, mais se résorbent d'elles-mêmes. Elles sont souvent associées au bruit présent sur le réseau. Le fait que la tension dans le réseau soit diminuée, cela peut rendre les connexions plus susceptibles à la diaphonie (crosstalk), au bruit de couplage et aux erreurs logiques (soft errors)[56]. Elles peuvent également être dues à des radiations, des sauts et des interférences électromagnétiques. Les erreurs logiques peuvent être divisées en deux catégories : erreurs sur les liens et erreurs intra-routeurs.

Comme il a été mentionné précédemment, les fautes peuvent être dues à différentes causes autant internes, ou externes comme l'effet de particules alpha et l'électromigration. De ce fait, la probabilité des fautes est différente pour chaque circuit et dépend de plusieurs facteurs, notamment les facteurs environnementaux, les paramètres de design du circuit et l'usage du circuit à proprement dit [56].

La figure 2.4 montre une chronologie indiquant les moments de la vie d'une puce où chaque menace apparaît. Des défauts de fabrication (DM) se produisent pendant la production (c'est-à-dire lorsque t<0), des erreurs transitoires dues au rayonnement (SE) ou à la variabilité de la tension et de la température (V) sont attendues pendant tout le cycle de vie et les phénomènes de vieillissement (A) à la fin de la vie de la puce[8].

FIGURE 2.4 – Occurrence de différents types de défauts pendant la vie d'un appareil.



Du point de vue de la localité, il est important d'analyser le comportement des fautes dans les différentes composantes du système pour trouver celles où les fautes sont plus fréquentes. Selon[8], les buffers et la barre transversale (Crossbar) occupent la plus grande surface du routeur et peuvent atteindre les 80% et 10%, respectivement. Bien que chacun des composants restants ne dépasse pas les 5% de la surface totale du routeur. Consommant la plus grande partie de la zone du routeur, la probabilité d'apparition de pannes est très élevée dans ces parties si nous supposons que la distribution des défauts est proportionnelle à la distribution de la surface. Par conséquent, l'adoption de la tolérance de panne pour les liaisons inter-routeurs (comme dans la plupart des systèmes) ne suffit pas pour construire un système fiable, et la prise en compte des pannes doit également inclure les différents composants.

2.3.2 Moyens

La tolérance aux fautes est mise en œuvre par la détection d'erreur et le rétablissement du système. La détection d'erreur peut être réalisée lors d'une suspension de service. On dit alors qu'elle est préemptive. À l'opposé, on dit qu'elle est concomitante lorsqu'elle est réalisée lors de l'exécution normale du service. Le rétablissement du système vise à transformer l'état erroné en un état exempt d'erreur et de faute. Le traitement de la faute se fait en identifiant le composant défaillant et en l'excluant.

Dans la suite de ce chapitre, nous nous intéresserons à la tolérance aux fautes pour les réseaux sur puce, et en particulier au niveau routage. Les performances des réseaux sur puce dépendent largement des algorithmes de routage qui ont un impact significatif sur le débit et la latence. Par conséquent, l'optimisation des algorithmes de routage pour le NoC est une préoccupation majeure pour améliorer les performances et minimiser la consommation d'énergie dans les NoCs[51]. Donc la sélection du chemin le plus optimal pour acheminer les paquets dans le NoC peut minimiser le trafic et éviter les zones congestionnées. Les défauts dans les liaisons de communication ou les routeurs du NoC peuvent dégrader les performances du système.

La probabilité accrue des défauts permanents ou transitoires dûs aux effets internes ou externes amène le problème de la fiabilité au premier plan[67]. Les techniques de tolérance aux pannes sont appliquées pour assurer le fonctionnement sûr dans les réseaux sur puce. Par conséquent, les techniques de tolérance aux pannes développées et implémentées dans le NoC à deux objectifs : (a) maximiser les performances en terme de latence et débit et (b) augmenter la fiabilité de la communication. Les schémas à tolérance de pannes englobant la congestion et le routage vont être présentés dans la section suivante de cette étude. Le nombre de défauts dans le NoC tend à augmenter avec une augmentation du nombre de composants intégrés.

Les défauts permanents peuvent survenir pendant la fabrication ou après la mise en œuvre, rendant les routeurs/liens inutilisables. Par contre, les défauts transitoires sont introduits lors de la transmission des données et affectent principalement les données en transit[8]. Ainsi, tout lien ou routeur défectueux peut isoler les composants fonctionnels du système. Une telle défaillance peut entraîner l'interruption de la communication ou créer une situation de blocage[51]. De la même façon, des routeurs défectueux peuvent créer des zones isolées sur la puce. Cependant, plusieurs algorithmes de routage à tolérance aux fautes ont été développés pour maintenir la connectivité et la sureté de transfert des données, en présence de défauts permanents et /ou transitoires.

De plus, le routage à tolérance de pannes augmente les performances du NoC en protégeant la transmission des messages au sein du réseau intégré en éliminant l'inter-blocage. Dans cette section, notre discussion se limite aux stratégies de routage à tolérance de

pannes.

2.3.3 Tolérance aux pannes

La tolérance aux fautes regroupe les mécanismes de gestion des défaillances temporaires ou permanentes d'un composant du système. Elle est essentielle à la garantie du bon fonctionnement du système dans certaines conditions. On identifie deux problématiques principales dans le domaine de la tolérance aux fautes. La première est de permettre à des applications ou communications qui souffrent d'une défaillance de continuer à fonctionner. Dans ce domaine, on peut citer les solutions de redondance ou de détection et correction des erreurs, ainsi que les mécanismes de retransmission en cas d'erreur de communication. La deuxième problématique est d'isoler les défaillances de sorte qu'elles n'entraînent pas d'autres défaillances dans le système. Des solutions de supervision du système et d'isolation des dysfonctionnements sont nécessaires pour répondre à cette problématique.

Un certain nombre de nœuds ou de liens peuvent subir des défaillances matérielles ou avoir des problèmes de communication dus à la congestion. Cette défaillance ne doit pas compromettre toute le système. Le routage doit s'adapter pour trouver de nouveaux liens et chemins d'acheminement de données de la source à la destination. Ceci peut nécessiter par exemple, une recherche d'un chemin alternatif. Une redirection des paquets sur des routes alternatives dont les nœuds ou les liens sont encore saints est requis pour qu'un NoC soit ainsi tolérant aux pannes.

La tolérance aux défaillances est généralement divisée en deux thèmes de recherche :

— Tolérance aux défaillances transitoires : elle consiste à gérer les éventuelles erreurs àe transmission sporadique. Un paquet peut être corrompu à cause d'une erreur de bit, il faut alors le retransmettre. Puisque l'erreur de transmission est sporadique, il n'y a pas de problématique de routage ou de contournement de la panne ;

— Tolérance aux défaillances permanentes : Une panne peut affecter un nœud ou un lien. Cette panne doit être gérée ou anticipée de façon efficace. Notons que la perte d'un nœud est équivalente à la perte des liens auxquels il est connecté, d'un point de vue du réseau. Les principes de base concernant la prise en charge des fautes permanentes et que celles-ci peuvent être évitées en reconfigurant le réseau.

Cependant, la période transitoire de reconfiguration, autrement appelée période de recouvrement, est un intervalle d'incertitude pendant lequel les flux impactés par la panne peuvent ne pas tolérer une rupture temporaire de leurs communications.Cette période transitoire est relativement peu étudiée dans la littérature. Cependant, cette méthode a l'inconvénient d'une sur-utilisation des ressources réseau.

Les défaillances d'éléments du NoC (routeurs et liens) sont traitées par des algorithmes de routage tolérant aux fautes, en ce qui concerne les fautes permanentes, certaines techniques au niveau réseau ont été étudiées. Parmi celles-ci, nous comptons notamment l'envoi de plusieurs paquets similaires (inondation) et la possibilité de différents chemins à emprunter. Cette dernière permette une excellente tolérance aux fautes car, si un chemin existe, le message est assuré d'arriver à destination. Par contre dans la première, beaucoup de paquets sont transmis sans être nécessaires. Alors, une diminution de transmission des paquets inutiles résulte en une augmentation du débit. Les liens de données étant l'un des composants les plus vulnérables aux fautes. Les erreurs sur le chemin de données intra-routeur peuvent être détectées comme des erreurs sur les liens à la validation au prochain routeur.

2.4 Congestion dans les réseaux 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 buffers, les canaux virtuels, etc. Les effets résultant de ces problèmes incluent : retard de livraison de messages (latence), réduction du débit (bande passante), voire paralysie de toutes les communications.

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.

1. Dans la première étape, la congestion doit être détectée.
2. Deuxièmement, la congestion doit être notifiée aux nœuds du réseau.
3. Finalement des actions doivent être appliquées pour résoudre le problème de congestion.

Pour notifier l'état de congestion aux nœuds du réseaux, nous pouvons classer ces techniques en deux catégories en termes de champ de notification : **local/régional ou global.**

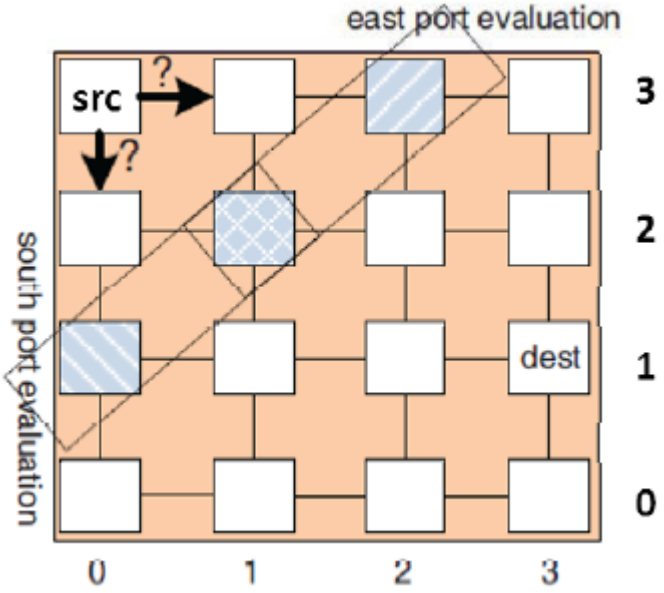
2.4.1 Congestion locale LCA

Localement[26], 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 adaptatif.

Dans un réseau utilisant un contrôle de flux basé sur le crédit, le routeur de chaque noeud 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 en aval). Cette connaissance est appelée conscience de la congestion locale (en anglais Local Congestion Awareness, abrégé LCA). Afin d'éviter la congestion, NoP (en anglais Neighbors on Path, abrégé NoP) évalue la congestion de noeuds distants de deux rangs (hops) par rapport au noeud courant en utilisant la LCA de ses voisins au lieu de ne prendre en compte que sa propre LCA [59]. La figure 2.5 présente un exemple de NoC utilisant l'approche NoP.

FIGURE 2.5 – Technique NoP : src : noeud source (0,0), dest : noeud destinataire (3,2) [60]



Lorsqu'un paquet est injecté dans le noeud 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 et 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 le nombre de tampons libres des noeuds directement voisins pour chaque cas, ici (0,1) et (1,2) pour le premier cas et (1,2) et (2,3) pour le second.

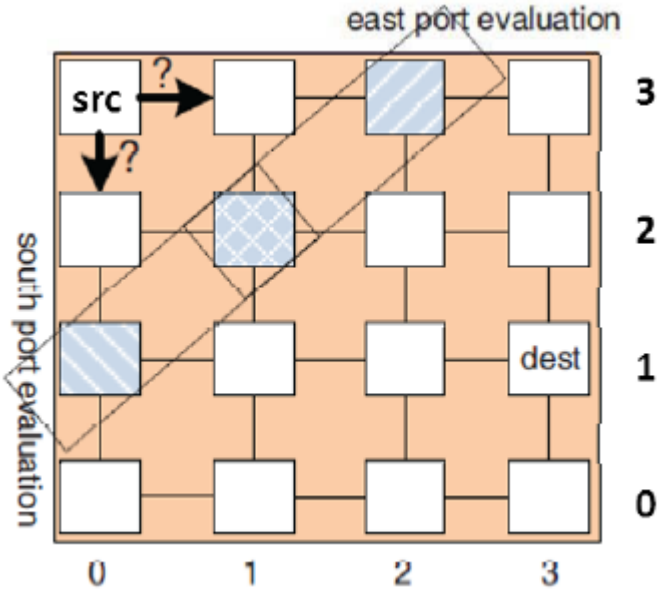
2, Propagation de la métrique de congestion : les mesures de congestion pour les deux voisins (0,2) et (1,3) sont envoyées au noeud (0,3).

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 noeuds (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 2.6 – Exemple de LCA : src : nœud source (0,0), dest : noeud destinataire (3,2)[60]



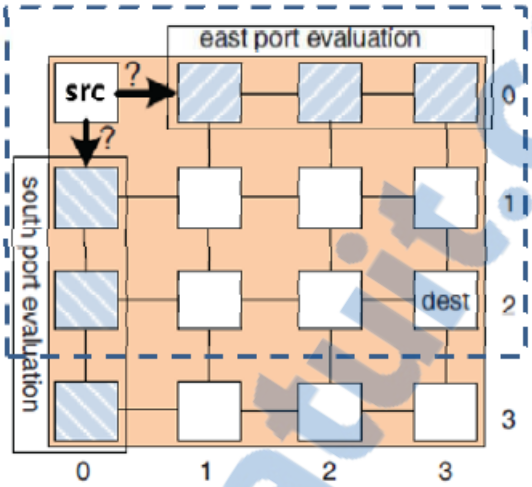
2.4.2 Congestion régionale RCA

NoP prend donc en compte seulement les informations de congestion à une distance de 2 hops. RCA (en anglais Regional Congestion Awareness, abrégé RCA[58] résoud ce problème par la propagation de l'état de congestion le long des axes (figure 2.6).

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.)

FIGURE 2.7 – Exemple de RCA : src : noeud source (0,0), dest : nœud destinataire (3,2)[60].



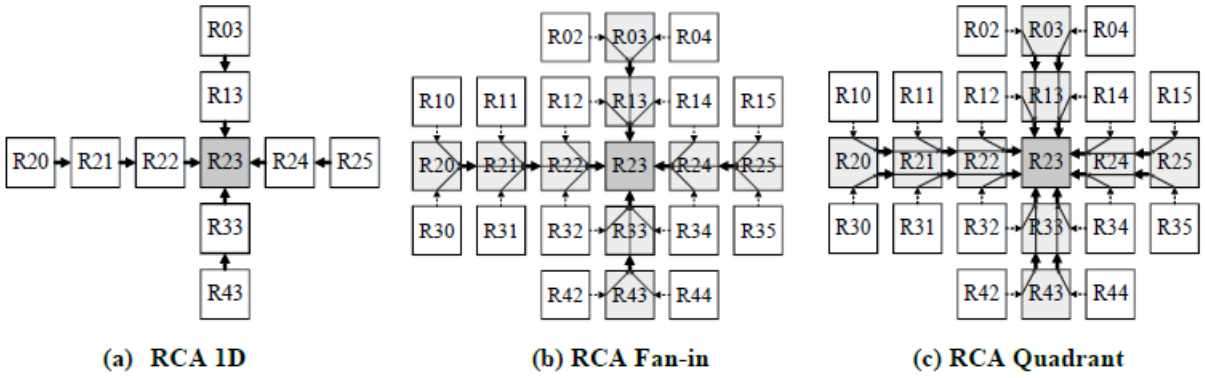
3, Agrégation de la métrique de congestion : la métrique pour le nœud (0,2) est la somme des métriques de tous les nœuds verticaux du nœud (0,3) jusqu'au nœud (0,0) ; et la métrique pour le nœud (1,3) est la somme des métriques de tous les nœuds horizontaux du nœud (3,0) jusqu'au nœud (0,0).

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 2.7, 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 2.7. Dans ce cas la mesure de congestion va entrainer un routage du paquet vers la mauvaise direction.

Le congestion au niveau régional repose sur les informations de congestion dans les routeurs voisins ainsi que sur les informations de la région intermédiaire fournies par ces routeurs. Gratz et al.[58] ont démontré que la sensibilisation à la congestion régionale (RCA) produisait de meilleurs résultats par rapport LCA. RCA a été mis en œuvre en utilisant un réseau à bande latérale pour propager les informations d'encombrement. Ces informations de congestion sont agrégées avec les informations de congestion locales (crédits disponibles), puis propagées en combinant les informations d'une à trois directions en fonction de la mise en œuvre de RCA. Des poids sont attribués aux informations de congestion locales et non locales dans le processus d'agrégation pour contrôler quel en-

semble de données a un effet plus important sur le mécanisme de routage adaptatif. RCA est implémenté selon trois méthodes différentes : RCA 1D, RCA Fanin et RCA Quadrant. RCA 1D collecte uniquement les informations, RCA Fanin collecte les informations de la ligne ou de la colonne, ainsi que les routeurs voisins de la ligne ou de la colonne. Dans le quadrant RCA, les données de congestion ne sont agrégées que dans deux directions, ce qui donne un résultat légèrement meilleur, mais en utilisant deux fois plus de temps de câblage (voir figure 2.8).

FIGURE 2.8 – Exemple de RCA (Regional Congestion Awareness.). [59]



La figure 2.9 illustre l'architecture interne de routeur RCA avec le composant de détection de la congestion ombré en gris. Il faut noter que des liens supplémentaires sont nécessaires pour la communication dans RCA.

Cette technique s'est avérée très utile pour améliorer les performances, comme l'a démontré Gratz et al.[58], mais elle souffre des interférences dues aux processus d'agrégation et de propagation de données. Car les informations de congestion reçues ne sont tout simplement pas des données brutes, mais elles peuvent contenir des informations inutiles. Le routage adaptatif basé sur la destination utilise des informations plus précises sur la connaissance de la congestion en utilisant les informations de congestion du routeur source au destination. Cette technique est utilisée dans DAR[42]et DBAR[38], ce qui permet de prendre des décisions de routage plus précises et d'améliorer les performances par rapport aux données régionales de détection de la congestion utilisées en[58].

Le routage adaptatif basé sur la destination (Destination-based Adaptive Routing-DAR) utilise la latence comme mesure du congestion et il diffuse les données pures aux routeurs voisins[42].

La figure 2.10 illustre comment les données de congestion circulent dans la région pour se propager à partir du routeur source, par exemple R30. Les routeurs ombrés en noir ont déjà les données d'encombrement du R30. Les routeurs en gris traitent actuellement les

FIGURE 2.9 – Architecture interne du routeur dans RCA . [59]

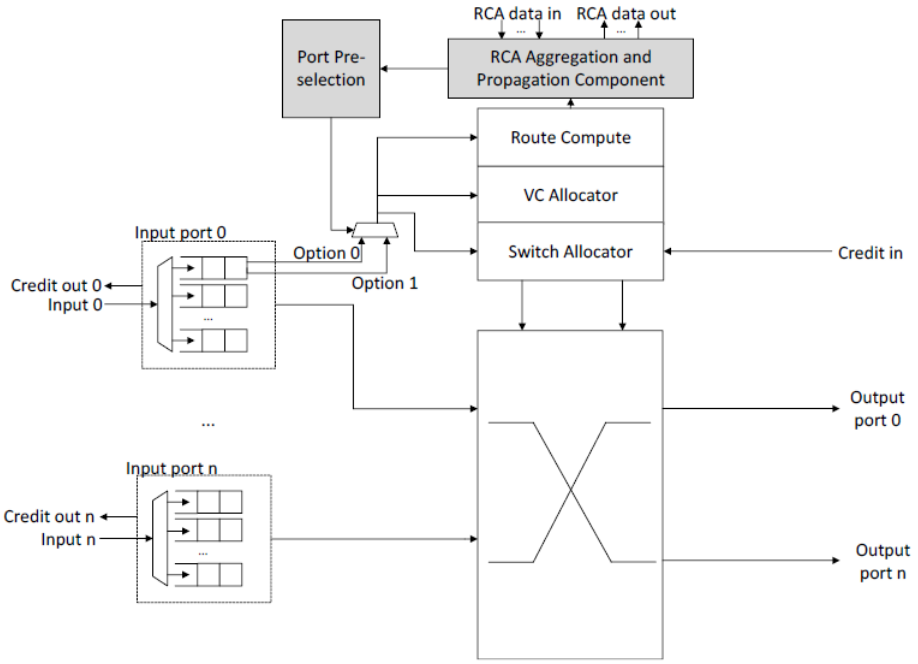
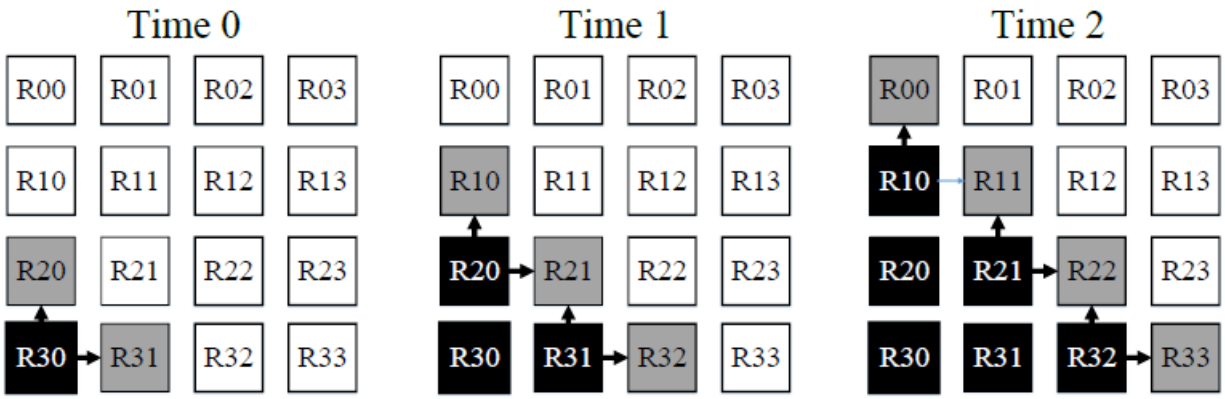


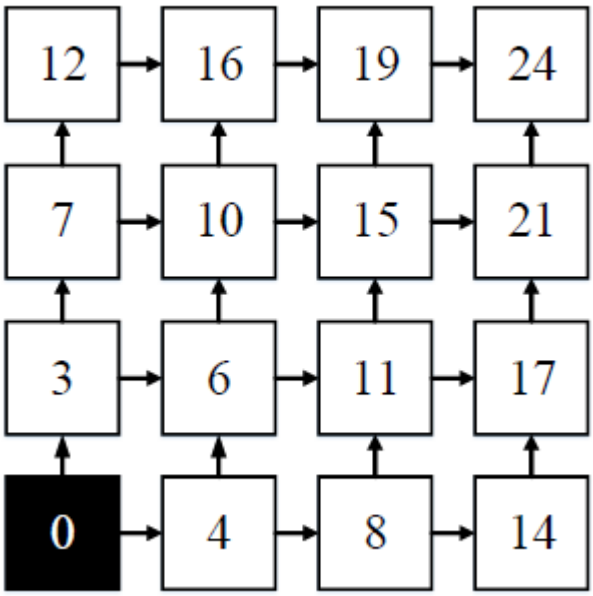
FIGURE 2.10 – Propagation de la congestion dans DAR.



données du routeur R30, tandis que les routeurs blancs n'ont toujours pas reçu les données du routeur R30. A l'instant zéro, R30 propage des données aux routeurs adjacents R20 et R31. À l'instant 1, R20 se propage aux routeurs R10 et R21 tandis que R31 se propage aux R21 et R30. Le routeur R21 recevrait des valeurs de latence des directions X (à partir de R20) et Y (R31). R21 utiliserait les informations de latence des deux directions pour déterminer un rapport de division favorisant la direction la moins encombrée. A l'instant 2, les données continuent à se propager aux routeurs R00, R11, R22 et R33. Pour la précision et la suppression des données obsolètes, les valeurs de latence ne se propageraient que dans une trame 7x7 du NoC. Pour les routeurs qui recevraient des valeurs de latence de deux routeurs, la valeur la plus faible est propagée.

La figure 2.11 montre un exemple d'encombrement (latence pour chaque nœud), un registre pour stocker les valeurs de la latence stockée pour tous les routeurs. Le routeur en bas à gauche est le routeur actuel dans l'exemple. Pour un petit exemple NoC 4x4, 8 bits sont suffisants pour stocker la latence. Pour un NoC importants, tels que les NoC 16x16, le routeur doit utiliser des registres de 10 à 12 bits pour stocker les valeurs de la latence. DAR et DBAR acheminent le paquet par le port de sortie préféré en fonction

FIGURE 2.11 – Congestion DAR (latence) vers tous les nœuds.



de la destination. DAR et DBAR estiment tous les deux le chemin le plus court vers la destination, ce qui n'est pas pris en compte dans RCA. Dans le cas de DBAR, un registre stockera les directions préférées pour la destination. Pendant le processus de sélection de la direction, le registre de directions préférées serait utilisée. Chaque registre n'aura besoin

que de 3 bits pour stocker les cinq directions. En termes de surface, DBAR est beaucoup plus simple que DAR pour calculer la latence. Les deux utilisent des liens externes menant à la surcharge du câblage. En comparaison avec RCA(8bits), le DBAR nécessite des liaisons plus larges (9bits) pour mettre en œuvre le contrôle de la congestion de réseau. DBAR calcule la latence en trois cycles, ce qui permet de réduire les liaisons de 15 bits à 5 bits. RCA 1D, RCA Fanin, DAR et DBAR ont tous besoin de 8 liaisons pour envoyer et recevoir les données de congestion.

Yuan et al. ont suggéré que la conscience de la congestion régionale pouvait être améliorée pour des régions plus encombrées en utilisant le routage source et un agent de congestion envoyant des informations à d'autres routeurs lorsque la congestion était détectée dans un point chaud[15]. Cette méthode a montré une amélioration par rapport au RCA dans le cas du trafic Hotspot intégrant un plus grand nombre de données de chaque saut dans le Flit d'en-tête pour le routage source. De plus, cette méthode ne montrerait pas d'amélioration significative par rapport au trafic synthétique standard.

2.4.2.1 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.

2.4.2.2 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.

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. 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.

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. 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.

2.4.3 Congestion globale GCA

S'il existe un moyen de connaître l'ensemble du réseau, le routage deviendra beaucoup plus facile. Au lieu de s'appuyer sur des algorithmes gloutons avec des stratégies de routage adaptatives aux niveaux local et régional, un routage globalement adaptatif peut éviter l'erreur de routage vers un point d'accès chaud et contourner complètement une région congestionnée. Ramakrishna et al. suggèrent que, en superposant les informations de congestion dans le Flit d'en-tête, chaque nœud disposera éventuellement des informations de congestion de l'ensemble du réseau[59]. Le routeur globalement adaptatif commence par calculer le chemin le plus court vers la destination en fonction des informations d'encombrement disponibles sur ce routeur particulier, puis achemine le paquet vers le saut suivant. Le routeur en aval recalculera le chemin le plus court et acheminera le paquet en utilisant ses données d'encombrement. Puisque chaque routeur a une information de congestion différente et lorsque le nœud actuel est plus proche de la destination, le routeur disposera d'informations d'encombrement plus précises. Le chemin sera pré-sélectionné avec les informations de congestion disponibles et le routeur bénéficiera également de la présélection de port.

Les techniques de tolérance aux pannes sont appliquées pour assurer le fonctionnement sûr de la puce. Les techniques de tolérance aux pannes nécessitent traitement supplémentaire et / ou redondance dans le matériel qui à son tour augmente la consommation d'énergie. Augmentation de la consommation d'énergie peut conduire à une augmentation de la température des dispositifs (sur puce) qui intensifie la probabilité des défauts et diminue la fiabilité[31] La probabilité accrue de défaillances permanentes et transitoires dues aux effets du vieillissement rapide et aux défis de développement ont amène le problème

de la fiabilité au premier plan[67]. Le nombre de défauts dans le NoC tend à augmenter avec une augmentation du nombre de composants intégrés et une réduction de la taille des fonctionnalités [30].Ces défauts peuvent réduire considérablement les performances et augmenter l'énergie consommer par la puce. De plus, tout lien erroné peut isoler les composants fonctionnels de la puce. Une défaillance de lien unique peut faire en sorte que la puce entière cesse de communiquer ou crée une situation de blocage [51]. De même, les composants défectueux peuvent créer des zones isolées sur la puce en rendant les composants / liens inutilisables. Les défauts transitoires ou les erreurs douces sont introduits lors de la transmission du données et affectent principalement les données en transit [54]. Un grand nombre de schémas à tolérance de pannes ont été développé pour maintenir la connectivité et l'exactitude des données, en présence de défauts permanents et/ ou transitoires. L'un des principaux objectifs de l'algorithme de routage est de maintenir l'équilibre du trafic et la topologie NoC. La recherche en conception tolérante aux pannes dans les NoCs se concentre principalement sur les algorithmes de routage et l'architecture des routeurs. Les algorithmes de routage doivent améliorer les performances et réduire la congestion.

Comme les systèmes sur puce, les réseaux sur puce (NoC) doivent être testés pour détecter d'éventuels défauts de fabrication ou après la mise en service. La stratégie de test pour ces systèmes concerne : (i) le test des unités de traitement embarquées (IP) et de leurs interfaces réseaux correspondantes ; (ii) le test de l'infrastructure d'interconnexion se composant des routeurs et des liens entre des routeurs (i.e. le réseau de communication). Dans notre cas, pour tester une architecture réseau, nous devons adresser deux problèmes : le test des routeurs du réseau et le test des liens entre les routeurs. Les routeurs se composent de buffers de type FIFO et de parties logiques pour le routage[67].

Cependant, celles-ci rencontrent beaucoup de problèmes car le test des systèmes sur puce est beaucoup plus complexe que le test des systèmes sur carte. Il se compose de tests individuels tels que le test de chaque IP, le test de la logique, le test des interconnexions. De plus, il est très difficile de programmer ces tests individuels pour s'adapter aux contraintes telles que le temps total de test, la consommation d'énergie, et le coût de test en termes de la surface, etc. Il faut également répondre aux exigences de la couverture de faute... etc. Ceux-ci sont les défis majeurs de l'implémentation du test pour un système sur puce[67].

En contre-partie, l'utilisation du test intégré conduira obligatoirement à un surcoût en matériel s'accompagnant d'une perte de performance. Cependant les méthodes de test doivent garantir une qualité de test suffisante et un temps d'application de test acceptable pour tester le circuit. En ce qui concerne la détection des erreurs dans les réseaux sur puce, la solution BIST intégrée, solution traditionnelle dans les circuits intégrés, peut être empruntée en tant que mécanisme de détection des erreurs permettant de gérer les erreurs dans les routeurs, les liaisons et les interfaces réseau. Presque tous les composants de NoC peuvent être vérifiés par BIST et désactivés si nécessaire.

On rappelle que dans le cadre de cette thèse, nous nous intéressons non pas aux test mais seulement à la détection de fautes.

2.5 Algorithme de Routage Tolérant aux Fautes

Les routeurs dans les NoCs acheminent les données en fonction de l'algorithme de routage implémentés. Les algorithmes de routage sont divisés en deux catégories ; déterministes et adaptatifs. Les problèmes de routage se produisent lorsque les paquets ne peuvent pas être acheminés de l'expéditeur vers le destinataire. Les performances des NoCs dépendent largement des algorithmes de routage qui ont un impact significatif sur le débit et latence. Par conséquent, l'optimisation des algorithmes de routage est une préoccupation majeure dans l'amélioration des performances NoC[29].

Chaque algorithme de routage est destiné à résoudre un ou plusieurs problèmes, expliqués précédemment dans ce chapitre, afin d'améliorer les performances du réseau. Les objectifs du routage dans les NoCs peuvent être exprimés comme suit :

1. Pas de Blocage : l'algorithme du routage doit garantir que le NoC ne soit pas atteint de blocage ;
2. Free-Livelock : Chaque paquet transmis sur le réseau ne devra pas courir indéfiniment pour trouver un chemin d'accès à sa destination ;
3. Pas de Congestion : L'algorithme de routage doit éviter la congestion ;
4. La tolérance aux pannes : Le protocole doit être en mesure de surmonter les défaillances du réseau afin de poursuivre son service normal.

2.5.1 Algorithme de Routage conscient de la congestion vs inconscient

Les défauts dans les liaisons de communication ou les routeurs du NoC peuvent dégrader les performances du système. Les performances des réseaux sur puce dépendent d'un certains nombre de facteurs, dont le protocole de routage utilisé. La fonctionnalité d'un algorithme de routage est modélisée à l'aide de deux fonctions : le calcul de la route et la sélection [54]. La fonction de calcul d'itinéraire produit un vecteur de canaux de sortie éligible pour un paquet. La fonction de sélection est consiste à sélectionner un canal de sortie de ce vecteur. La sélection peut se faire sur la base de l'état actuel des canaux de sortie.

Selon la fonction de sélection utilisée, les algorithmes de routage peuvent être classés en deux catégories : inconscient ou conscient de la congestion. Dans la première catégorie, les

décisions de routage sont indépendantes de l'état de congestion du réseau. Cette politique peut perturber l'équilibre de la charge car l'état du réseau n'est pas pris en compte. Au contraire, les algorithmes de routage sensibles à la congestion prennent en compte l'état de congestion actuel du réseau dans leurs décisions de routage. Le niveau de congestion actuel est déterminé à l'aide d'informations locales ou globales. Dans les approches prenant en compte les conditions de trafic locales, la décision de routage est prise uniquement en fonction de l'état d'encombrement des voisins adjacents. Ces méthodes fournissent une vision limitée, donc moins précise, de l'état du réseau. Les algorithmes de routage basés sur des informations globales fournissent une meilleure répartition de la charge de trafic. Cependant, la collecte d'informations globales, puis l'utilisation de ces informations dans la sélection des canaux de sortie rendent les méthodes de routage globales sensibles à la congestion plus complexes. Les algorithmes compatibles avec les encombrements peuvent tirer parti de différentes mesures, telles que le nombre d'emplacements de mémoire tampon disponibles, les canaux virtuels disponibles et la demande des combinaisons de ces facteurs.

On peut se référer au document d'étude mené par[54] sur les méthodes tolérantes aux fautes dans les NoCs, qui traitent en détail les modèles d'erreur et les mécanismes de détection des fautes transitoires / intermittentes. Les travaux exposés dans[75] donne un aperçu sur les techniques de routage à tolérance de pannes dans le contexte des réseaux sur puce. Dans le cadre de cette thèse, nous fournissons un bref aperçu des principales techniques offrant la tolérance aux pannes avec ou sans prises de la congestion, c'est-à-dire des techniques capables de traiter les défaillances survenant après la mise en service de la puce.

Dans la section suivante, nous discuterons quelques algorithmes de routage et de la façon dont ils ont été utilisés pour fournir une tolérance aux pannes dans les réseaux sur puce.

2.5.1.1 Algorithme de routage tolérant aux fautes inconscient de la congestion

Plusieurs articles sur les algorithmes de routage tolérants aux fautes ont été récemment publiés. Des architectures reconfigurables ont été utilisées pour remédier aux défauts.

Zhang et al.[81] propose un algorithme de routage reconfigurable, déterministe, tolérant aux pannes et distribué. L'idée principale de l'algorithme est de créer un chemin pour acheminer les paquets à travers un contour sans cycle qui entoure un routeur défectueux. Par conséquent, il réalise la restauration de tous les chemins interrompus causés par le routeur défectueux. L'algorithme utilise 9 contours canoniques d'un réseau de maillage pour établir des chemins au moment de l'exécution. Lorsqu'un paquet ne peut pas trouver une route utilisant l'algorithme normal (à cause de défauts dans le système), le chemin de routage est modifié en connectant les différents contours afin d'établir un chemin. Pour

éviter les blocages, il faut interdire certains virages sur le chemin.

L'architecture Vicis[10] proposée peut tolérer la perte de routeurs ou de liaisons due à des défauts permanentes. La reconfiguration au niveau du réseau est implémentée par mise à jour des tables de routage en fonction des informations provenant des unités BIST (Self-Test) intégrées dans chaque routeur.

Chaix et al.[30] proposent EPR (Explicit Path Routing), un algorithme de routage adaptatif sans blocage. L'objectif de l'EPR est de limiter la dégradation du temps de latence des paquets dans le NoC, même en cas de défaillance. Pour éviter les inter-blocages une variante du modèle de restriction de virage est implémentée. Certains retournements sont essentiellement interdits en fonction du chemin du paquet afin de briser les dépendances de cycle. EPR fournit également des performances élevées dans des conditions erronées en étudiant l'état du réseau au moment de l'exécution. Chaque fois que le système est en train de reconfigurer, un paquet d'écho est envoyé, lequel détermine le chemin optimal d'une paire source-destination. Une fois que le chemin d'une paire source-destination défectueuse est déterminé, ce chemin est utilisé indéfiniment.

Tsai et al.[78] proposent BFT-NoC(Bidirectional Fault-Tolerant NoC). Cette technique capable est de traiter les défaillances de canaux statiques et dynamiques. En utilisant des canaux bidirectionnels, un routeur défectueux peut être atteint tant qu'une de ses liaisons est toujours fonctionnelle. En revanche, les schémas utilisant des liens unidirectionnels se traduiront par des nœuds inaccessibles au moment où l'un de ses liens devient défaillant. Dans des conditions parfaites, BFT-NoC fonctionne de la même manière qu'un routeur NoC traditionnel. Cependant, une fois le défaut détecté, les routeurs BFT-NoC passent par une phase de reconfiguration au cours de laquelle le mode de canal (transmission ou réception) est déterminé en fonction de la demande du réseau. Cette approche permet une dégradation progressive des NoCs.

Pour Vitkovskiy et al.[76], les liaisons défectueuses pourraient être contournées en réacheminant les paquets dans un chemin en forme de U autour des régions défectueuses. Cette idée est illustrée par le protocole de routage «FTLR», qui traite l'inter-blocage et le livelock en utilisant deux canaux virtuels.

Wachter et al[28] ont introduit un algorithme de routage source tolérant aux pannes avec une exploration de chemin qui ne dépend pas de la topologie et qui peut donc être appliqué aux topologies régulières ou irrégulières. Trois étapes sont nécessaires : rechercher, revenir en arrière et effacer & calculer. Leur algorithme lance le mécanisme de recherche lorsque le dernier paquet envoyé n'a pas atteint sa destination. Seek essaiera de rechercher un nouveau chemin et cela ne sera fait qu'une fois pour chaque paquet manqué. Seek est basé sur un compteur de sauts et chaque routeur stocke les requêtes dans ses tables internes. Les tables ne dépendent pas de la taille du maillage, mais les entrées de table sont

uniquement liées au nombre maximal de recherches simultanées. Lorsque le routeur cible est trouvé, le processus de retour en arrière démarre. Un paquet de retour sera envoyé au routeur source en fonction des informations utiles. Lorsque le paquet de retour n'est pas atteint, le nœud cible est inaccessible ou le nombre maximal de recherches simultanées est dépassé. La dernière étape consiste à vérifier les tours non valides (ce qui peut provoquer un blocage) et, le cas échéant, l'algorithme force le paquet à utiliser un ou plusieurs canaux virtuels. Cette approche utilise des tables de routage et n'est pas évolutive.De plus, le temps total de recherche de chemin et de calcul est faible comparé à la littérature et présente une surcharge de 50%, mais il garantit également une accessibilité optimale, même dans des scénarios de panne graves.

Algorithme de gradient[66] propose un algorithme de routage adaptatif permettant de tolérer un seul nœud défaillant. L'algorithme divise l'ensemble du réseau en huit zones par ligne de Gradient. Chaque zone est dotée d'un chemin principal et de deux chemins alternatifs pour la livraison de paquets lorsque le chemin de routage principal est défectueux. L'algorithme de routage Gradient sélectionne plus d'itinéraires alternatifs avec des sauts minimaux par rapport aux autres algorithmes de routage adaptatifs existants (DyXY).

Behrouz et al.[69] proposent le protocole de routage «FaulToleReR» basé sur deux phases. La première consiste à rechercher l'ensemble des chemins disponibles dans le réseau, y compris les défauts. La seconde sélectionne ceux à utiliser entre chaque nœud source/destination, en tenant compte des informations stockées dans les tables de routage. Avec «FaulToleReR», le réseau est toujours connecté. Le problème du deadlock et livelock est ainsi traité.

Dans[15], les auteurs se concentrent sur une classe particulière de schémas de routage qui n'utilisent pas de canaux virtuels pour fournir une tolérance aux pannes. Dans la conception de tout système de routage tolérant aux pannes, le défi majeur est d'obtenir un état sans blocage en présence de pannes. L'algorithme proposé nommé ZoneDefense qui permet non seulement d'inclure les défauts dans des blocs convexes, mais étend également les informations de position des blocs défectueux dans les colonnes correspondantes. Les nœuds ayant des informations sur les nœuds défectueux créent des zones de défense. Ainsi, les paquets connaissent à l'avance les nœuds défectueux et peuvent se router dans une autre direction.

Les auteurs[71] ont présenté une méthode de routage à tolérance de pannes nommé (FTR) basée sur l'algorithme de routage XY. L'algorithme FTR est capable de gérer une seule défaillance de liaison. En cas d'erreur, FTR reconfigure les chemins pour dévier les paquets sur d'autres chemins disponibles.

Dans[33], les auteurs ont proposé un routage tolérant aux pannes inspiré par le comportent des colonies de fourmis nommé (ACO-FAR)et qui équilibre la charge de trafic sur

le réseau. L'approche proposé utilise trois étapes pour l'arrivée du paquet à sa destination à savoir : 1) rencontre du défaut ; 2) recherche ; et 3) sélectionnez. Par conséquent, il met en œuvre trois méthodes correspondantes : 1) la notification d'informations de défaillance ; 2) méthode de recherche d'itinéraire ; et finalement sélection d'itinéraire. Le routeur peut vérifier les chemins disponibles et orienter les paquets vers un chemin moins encombré et sans défaillance. l'approche proposée ne traite que les fautes permanentes.

Dans[79], Zhang et al. présenté un nouveau mécanisme appelé "DMT" qui est capable de délivrer des paquets sur différents canaux de sortie en utilisant des chemins non minimaux. Grâce à cela, "DMT" ignore les défauts détectés et évite l'interblocages et livelock.

Letian Huang et al. dans[40], ont proposé une architecture de routeur reconfigurable avec des canaux de contournement fournissant la connectivité des cœurs du réseau lorsque les routeurs sont en cours de test. L'algorithme de routage est totalement adaptatif nommé TARRA. Une autre propriété est que l'architecture proposée permet de prendre en considération plusieurs routeurs en cours de test. TARRA utilise également un et deux canaux le long des dimensions X et Y, respectivement. Pour prouver l'algorithme d'acheminement TARRA est sans inter-blocage, les auteurs ont partitionné le réseau en deux sous-réseaux disjoints. Chaque sous-réseau a des ensembles de canaux disjoints.

Discussions

1. Certains chercheurs ([14], [63],[13], par exemple) proposent des méthodes de couverture complète/élevée ou modérée, dans lesquelles, lors de nouvelles défaillances, un contrôleur central collecte les informations de panne, calcule la nouvelle configuration et la distribue aux routeurs. . Ces méthodes centralisées présentent deux défis principaux. (i)le contrôleur central peut facilement devenir le seul point de défaillance puisqu'il se trouve sur le chemin critique du processus de reconfiguration[12]. (ii) il doit exister un moyen extrêmement fiable de collecte d'informations sur les défauts et de distribution de routage, généralement pris en charge à l'aide de méthodes basées sur TMR3[63]. Ainsi, les approches centralisées peuvent devenir impraticables et/ou coûteuses pour tolérer les erreurs d'exécution.

2. **Couverture totale des fautes**. Comme mentionné précédemment, un grand nombre de défaillances aléatoires peuvent se produire dans les composants NoC, ce qui peut conduire à des destinations déconnectées (inaccessibles). En tant que tel, ce travail a pour objectif premier d'obtenir une couverture complète (en cas de défaillance), c'est-à-dire de garantir la livraison d'un paquet à sa destination lorsqu'un chemin entre la source et la destination existe ou encore d'indiquer que la destination est inaccessible, quelle que soit la destination choisie. le nombre et l'emplacement des défauts. Cependant, la plupart des travaux passés ne prennent en charge qu'un nombre de fautes limités[81]. Par exemple, l'algorithme de routage dans[81] ne traite qu'un routeur / région défectueux.

3. **Frais généraux** réduits. La surcharge de la surface imposée par un algorithme de routage est une préoccupation majeure. Les frais généraux imposés s'ajoutent non seulement aux coûts de mise en œuvre et à la dissipation de puissance, mais également à la vulnérabilité de l'algorithme aux erreurs d'exécution. Cependant, la plupart des travaux qui résolvent un grand nombre de défauts utilisent une ou plusieurs tables de routage [[4],[12],[9],[10],[11],[63],[28]]. Par exemple, les tables de routage de chaque nœud d'un réseau 8x8 peuvent nécessiter 256 bits [4], 500 bits [11], voire 8K bits[28] pour stocker des chemins sains vers des destinations. La surcharge de leur surface imposée se traduit directement par une probabilité de défaillance accrue dans l'ensemble du routeur. Ainsi l'utilisation d'une table de routage pour la reconfiguration des composants défaillante
4. **Estimation du surcoût de reconfiguration**. Enfin, mais surtout, un algorithme pratique de tolérance aux pannes devrait permettre un fonctionnement ininterrompu du réseau une fois que certains composants sont désactivés. Une reconfiguration rapide devient une nécessité, en particulier avec l'utilisation de tests en ligne agressifs, dans lesquels les composants du réseau (lien réseau, par exemple) deviennent indisponibles non seulement en raison de défaillances détectées, mais également périodiquement et fréquemment au cours de leurs tests en ligne[11]. Cependant, la plupart des travaux existants avec une couverture de défauts modérée/élevée ou totale souffrent de longues phases de reconfiguration [[4],[63],[28]], au cours desquelles ils interrompent le fonctionnement normal du réseau et mettent en pause la livraison des paquets jusqu'à la fin du processus. reconfiguration. Par exemple, dans un réseau 10x10, lors d'une nouvelle occurrence de défaillance, il faut jusqu'à 10 000 cycles[4], 100 000 cycles[28], voire 100 ms [63] pour reconfigurer le réseau. Contrairement aux travaux avec une reconfiguration rapide, les travaux avec une reconfiguration à la volée, y compris ce travail, n'ont pas de phase de reconfiguration distincte : un nouveau chemin vers la destination est calculé dynamiquement par paquet au fur et à mesure que le paquet voyage sur le réseau.

   Un autre inconvénient de ces approches est qu'elles ne prévoient pas un mécanisme efficace pour contrôler le problème de congestion, afin de prendre une bonne décision au moment du routage. La table 2.1 donne une synthèse de quelques algorithmes de routage tolérant aux fautes inconscient de la congestion.

2.5.1.2 Algorithme de routage tolérant aux fautes avec prise de congestion

Ebrahimi et al.[48], introduisent une méthode appelée MD , algorithme de routage minimal et résilient aux fautes. Premièrement, la méthode permet de tolérer une seule défaillance au niveau de liaison en utilisant le chemin le plus court entre chaque paire de noeuds source/destination, si un chemin existe. Deuxièmement, pour éviter les encombrements, les canaux de sortie peuvent être choisis de manière adaptative chaque fois que la

	Réf	Année	Couverture	Reconfiguration	O(Area)
Zhang et al.	[81]	2008	few	fully dist	low
ARIADNE [3].	[4]	2011	full	distributed	high
uDIREC [32].	[63]	2013	full	central	high
FTDR-H [18].	[9]	2013	high	fully distr	high
Wachter et al.	[28]	2013	high	distributed	high
BLINC [25].	[11]	2014	moderate	distributed	high
d2 LDBR.	[14]	2015	moderate	central	low
TOSR [5].	[12]	2015	moderate	distributed	high

TABLE 2.1 – Comparaison de quelques algorithmes de routages tolérant aux fautes inconscient de la congestion.

distance entre le noeud actuel et de destination est supérieure à un saut dans les deux dimensions. MD tire parti des avantages d'un et de deux canaux virtuels le long des dimensions X et Y. Cette idée de MD est développée dans la deuxième approche, appelée algorithme MAFA[47] (Minimal and Adaptive Fault-Tolerant Algorithm), pour rendre un réseau résilient contre deux liaisons défaillantes. Une fiabilité accrue entraîne l'utilisation d'un canal virtuel supplémentaire le long de la dimension Y dans MAFA plutôt que MD. Au total, MAFA utilise deux canaux virtuels sur les deux dimensions. MD et MAFA traitent les liaisons défectueuses et HiPFaR[49] et MiCoF[50] traitent les routeurs défectueux (max 06). L'objectif de ces méthodes est de tolérer les erreurs en utilisant les chemins le plus court du réseau tant que ce chemin existe. À l'aide de ces algorithmes, les paquets sont éventuellement acheminés selon des chemins minimes et non défectueux, ce qui évite de faire face à des composants défectueux et de rendre inutile tout réacheminement autour d'eux.

CAFTA[46]ont proposé un algorithme qui permet de tolérer n'importe quel nombre de défaut au niveau de liens ou routeurs. En même temps, il est capable d'équilibrer la charge pour éviter les points chauds sans utiliser de tables de routage. Il combine un routage adaptatif Nord-Dernier et Sud-Dernier qui utilise des restrictions de virage pour éviter les blocages lors de l'utilisation de quatre VC par port d'entrée. Lorsqu'un lien ou un nœud échoue, tous les voisins sont informés. Les nœuds sont ainsi informés des défaillances adjacentes leur permettant de router les messages autour des liens défaillants. Pour équilibrer le trafic, un signal «Flits-stay» est utilisé pour compter le nombre de flits restants qui doivent encore passer par le routeur, ce qui augmente / diminue en conséquence. CAFTA fournit 97,68 % des paquets transmis avec succès dans un maillage 16*16 lorsque 384 liaisons sont défectueuses simultanément et 93,40 % pour 103 nœuds défectueux.

Par conséquent, plusieurs chercheurs travaillent à améliorer les algorithmes de routage. De nouveaux algorithmes de routage adaptatifs sont introduits pour améliorer la capacité

de tolérance aux pannes dans les NoC[43]. Leur nouveauté permet aux routeurs de réduire la congestion dans les NoC. L'algorithme de routage présenté sélectionne le chemin pondéré pour préserver les performances de NoC

Les auteurs de [33] utilisent une technique d'optimisation des colonies de fourmis pour proposer un algorithme de routage sensible aux pannes pour NoC. Cette approche permet au routeur de trouver le chemin d'encombrement minimal qui n'est pas défectueux. Après avoir trouvé l'obstacle dans le NoC, l'algorithme d'optimisation tente de rechercher un nouveau chemin sur la base des informations de défaut et sélectionne finalement le chemin optimal. Ils améliorent le débit en comparant les recherches associées. Le coût de la mise en œuvre dans l'algorithme d'optimisation des colonies de fourmis est élevé. Par conséquent, la technique de colonie de fourmis modifiée proposée dans[20].

Un autre algorithme de routage adaptatif est présenté dans[44]. Les auteurs améliorent le débit et la dissipation de puissance en supprimant les fils supplémentaires et utilisent des bits libres pour propager les informations d'encombrement. Dans[21], une sélection de chemin développée dans un algorithme de routage adaptatif a été suggérée. Ils prennent en compte les informations de canal et de commutation pour trouver une situation de congestion. Les blocages dus à des topologies irrégulières dans les réseaux de stockage et à l'état d'équilibrage de la charge sont deux problèmes des réseaux sur puce.

Pour accroître la fiabilité et les performances des réseaux NoC, les auteurs dans[7] ont proposé un nouvel algorithme de routage appelé MUGEN. Ce travail comprend une méthode optimisée pour échanger des messages entre différentes classes de canaux virtuels. La fonction de sélection utilise les informations de liaison de routeur distantes pour éviter les blocages et une nouvelle métrique de congestion utilisée pour orienter les décisions de routage vers des zones moins encombrées. Les auteurs affirment que l'algorithme proposé présente des résultats prometteurs en termes de tolérance aux pannes et de performances.

Un travail récent propose un algorithme appelé PDA-FTR[80].L'algorithme de routage tolérant aux pannes prenant en compte la diversité de chemins et sélectionne des chemins minimaux pour la livraison des paquets. Cependant, si les chemins de routage minimaux sont bloqués par des erreurs, alors PDA-FTR utilisera des chemins non minimaux pour éviter l'encombrement des paquets dans la région défectueuse. S'il y a deux autres candidats de chemin d'acheminement, PDA-FTR fait référence aux informations de longueur de tampon effective (EBL) de chaque canal de sortie candidat afin de sélectionner le meilleur canal de sortie. Le canal avec le FPD le plus élevé est considéré comme le chemin minimal et utilise un routage adaptatif minimal. D'autre part, lorsque le chemin minimal n'est plus disponible, un algorithme de routage adaptatif non minimal sera adopté pour éviter le blocage des paquets. De plus, les informations FPD et BO sont combinées et définies comme étant la longueur effective du tampon (EBL). Cette métrique est la tranche libre pondérée déterminée par le FPD, qui est exprimée en terme de produit de la FPD normalisée et

des espaces tampons non occupés. Par conséquent, lorsque la fonction de routage retourne deux directions de sortie candidates ou plus pour un paquet, nous choisirions la direction avec une EBL plus élevée.

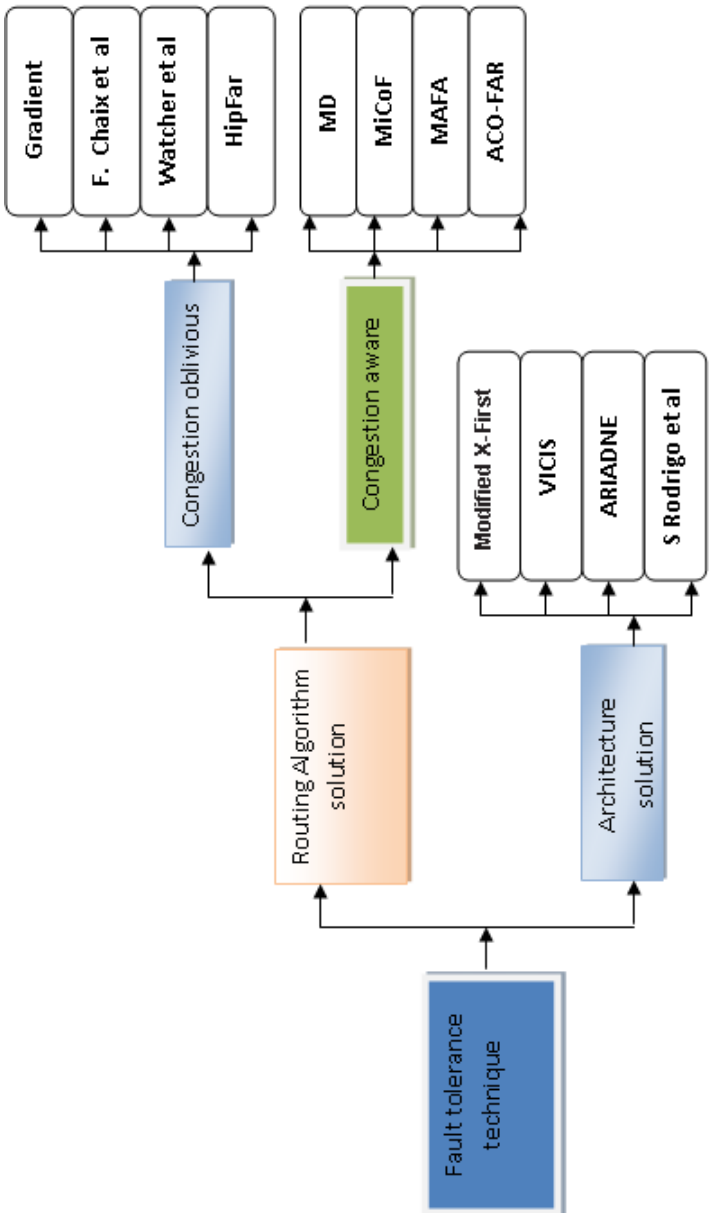


FIGURE 2.12 – Some related works on fault-tolerant techniques in NoC.

Table 2.2 – Most known Fault-Tolerant Routing Algorithms. C : Congestion

FT Routing Algorithm	Years	Components	Type	C
Modified X-First[81]	2008	Up One switch failure	Permanent	No
F. Chaix et al.[30]	2010	Many faulty switches	Permanent	No
ARIADNE[4]	2011	Links and switches	Permanent	No
Vicis[10]	2012	Links and switches failures	Permanent	No
S. Xiangming[73]	2012	Links and switches failures	Transient faults	No
MAFA[49]	2013	Up to 6 faulty links	Permanent	Yes
Gradient[66]	2013	Switch failures	Permanent	No
MD [50]	2013	Up to 2 faults Link	Permanent	Yes
A Vitskovkiy et al.[76]	2013	Links	Permanent	No
HipFar[49]	2013	Up to 6 faulty nodes	Permanent	Yes
C. Chen et al.[22]	2013	Any number of faults links	Permanent	No
Wachter et al. [28]	2013	Switch	Permanent	No
ACO-FAR[33]	2014	Up to 4 faulty switches	Permanent	Yes
Zone Defence[15]	2014	Many faulty switches	Permanent	No
M. Dimopoulos et al.[46]	2014	Links and switches	Permanent and transient	Yes
R.J. Behrouz et al. [69]	2014	All faulty links/nodes	Permanent	No
MUGEN[7]	2016	Links and switches	Permanent	Yes
PDA-FTR[80]	2017	Switches	Permanent	Yes

2.6 Conclusion

L'utilisation des technologies submicroniques profondes dans un système embarqué cause une augmentation de la susceptibilité au "Single Event Upset" (SEU) qui peuvent diminuer la fiabilité des circuits. Un faute transitoire se produit lorsque un rayonnement provoque le changement de valeur d'un bit dans certaines bascules du circuit. Cette modification indésirable peut causer le dysfonctionnement de l'architecture. Un autre problème qui mène à l'apparition de défauts permanents dans le circuit est le " vieillissement ". Dans le cas d'un défaut permanent l'élément fautif doit être éliminé de la communication. Une autre solution est de ré-acheminer le paquet en évitant le composant défectueux. Ainsi un routage tolérant aux pannes est nécessaire. Dans ce chapitre consacré à l'état de l'art, de nombreux travaux de recherche dans le domaine de la tolérance aux fautes ont été présentés. Les différentes approches présentées dans ce chapitre ont pour but de remplacer ou de pousser à l'extrême la technologie standard, en offrant la possibilité d'avoir une bonne tolérance aux fautes avec de faibles coûts d'implémentation. Ainsi le développement de nouveaux algorithmes tolérant aux fautes transitoires et aux défauts de fabrication comme ceux présentés dans ce chapitre, va permettre de réaliser d'une façon économique des circuits plus fiables. Dans ce chapitre, nous avons mis l'accent sur les principales approches et les solutions proposées dans la littérature pour assurer ou améliorer la tolérance aux

pannes dans les NoCs. De plus nous avons classifier les mécanismes appliqués en fonction de la congestion.

Toutes les solutions de tolérance aux pannes étudiées ne prennent pas en compte ce facteur clé qui influence d'une manière significative les performances du NoC à savoir la latence et le débit. En présence des fautes, les performances peuvent se dégrader compte tenu de l'état des liens ou routeurs ce qui peut augmenter le taux de perte des paquets dans le réseau. L'une des solutions pour pallier à ce problème est de considérer la congestion comme une faute. Pour remédier à ce problème nous avons ainsi développé un algorithme de routage tolérant aux fautes permanentes et transitoires pour des réseaux sur puce. L'algorithme prend en compte des chemins alternatifs pour les paquets lorsque la voie d'acheminement principale tombe en panne. L'algorithme proposé permet d'éviter plus de fautes et tolère des défaillances multiples dans le pire état du trafic. L'adaptativité permet d'alterner des chemins entre la même paire de nœuds source /destination, cette propriété permet de supporter la tolérance aux pannes, ou de l'optimisation de performance en évitant des zones du réseau qui seraient surchargées ou congestionnées. Les performances du réseau NoC sont estimé selon sa latence, son débit et sa fiabilité. Ainsi, le concepteur d'un NoC doit prendre en compte l'impact de chaque paramètre sur les résultats. C'est ce qui sera détaillé dans le chapitre suivant et lors de la descriptions de nos expérimentations.

Chapitre 3

Conception et Implémentation d'un Nouvel Algorithme de Routage -DINRA

Sommaire

Les performances et la tolérance aux pannes sont les deux principaux problèmes rencontrés dans la conception des réseaux d'interconnexion sur puce pour les architectures multiprocesseurs à grande échelle. La tolérance aux pannes est la capacité du réseau à fonctionner en présence de composants en pannes. Cependant, les techniques utilisées pour réaliser la tolérance aux pannes entraînent souvent une dégradation considérable des performances.

L'algorithme de routage proposé est appelé DINRA. C'est un algorithme de routage unicast puisqu'il envoie un paquet d'une source à une destination. La deuxième propriété qu'il est distribué càd il prend la décision de routage pendant le temps de déplacement d'un paquet d'un nœud à l'autre. En outre, il s'agit d'un algorithme de routage totalement adaptatif, car il prend en compte les conditions de réseau pour le routage d'un paquet. L'algorithme de routage DINRA achemine un paquet en fonction de deux facteurs principaux. Le premier facteur est l'état des noeuds/liens. Le deuxième facteur concerne les conditions du réseau (congestion). L'algorithme de routage proposé améliore le débit du réseau et réduit la perte de paquets sur le réseau par rapport à PDA-FTR. En effet, ce

dernier offre une adaptabilité partielle. Cependant, DINRA permet au paquet de changer de chemin en fonction de l'état actuel du réseau pendant le processus du routage. Pour cette raison, le routage évite les régions encombrées lors du routage du paquet, ce qui garantit que ce dernier ne sera ni bloqué ni perdu avant d'atteindre la destination. En conséquence, le nombre de paquets bloqués sera réduit dans le routage DINRA ainsi la latence et augmentera le débit.

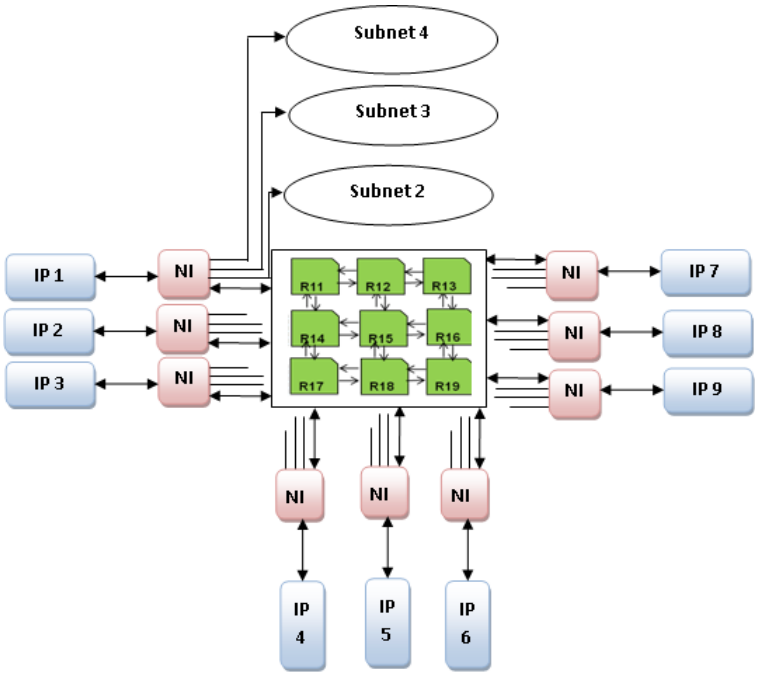
3.1 Architecture Globale du Réseau

Les réseaux sur puce constituent un domaine de recherche très actif depuis leur apparition au début des années 2000. Depuis lors, de nombreuses architectures ont été proposées dans la littérature. Les routeurs, les interfaces réseau (NI), les IPs et les liens constituent les principaux éléments d'un NoC. Dans cette section, nous présentons l'architecture proposée et ses principaux composants. Avant de donner les détails, il est nécessaire de préciser que nous avons adopté deux hypothèses importantes : premièrement, les liens reliant la propriété intellectuelle (IP) aux interfaces réseau d'entrée et de sortie (NI) sont toujours non défectueux. Deuxièmement, nous supposons qu'il existe au moins un chemin entre une paire (source, destination). Ces hypothèses sont nécessaires pour livrer n'importe quel paquet de la source à la destination.

La Fig.3.1 montre l'architecture globale. Le réseau sur puce est divisé en sous-réseaux. Chaque IP est connectée à quatre routeurs, chacun appartenant à un sous-réseau disjoint. Ainsi, chaque routeur n'appartient qu'à un seul sous-réseau. Par exemple, nous construisons 4 sous-réseaux avec u réseau 6 * 6, chaque routeur ne peut communiquer qu'avec les routeurs du même sous-réseau. Cette méthode représente une alternative innovante pour obtenir une fiabilité efficace. Lorsqu'un sous-réseau est encombré ou défectueux, nous devons désactiver l'ensemble du sous-réseau et l'isoler. En se concentrant sur les sous-réseaux, nous pouvons voir qu'ils ont tous homogènes (le même modèle de connectivité).

Les trois IP situées à l'est du NoC sont connectées aux trois routeurs du même sous-réseau. La même topologie est appliquée pour les IP ouest et sud. Nous avons implémenté un algorithme (Section 3.2.4) pour sélectionner le sous-réseau. Nous avons ajouté 2 bits au flit d'en-tête dans NI pour distinguer chaque sous-réseau des autres. Le principal avantage de cette architecture est que le modèle de connectivité a plusieurs chemins alternatifs qui peuvent être utilisés pour augmenter la tolérance de panne, comme cela sera montré dans la section 3.4. Lorsque le premier sous-réseau est inutilisable, nous activons le deuxième sous-réseau et lorsque le deuxième sous-réseau est défectueux, nous activons le troisième sous-réseau, ainsi de suite. Le pire scénario se produit lorsque le sous-réseau supérieur est défectueux ou congestionné. Pour offrir une tolérance aux pannes plus élevée, la nou-

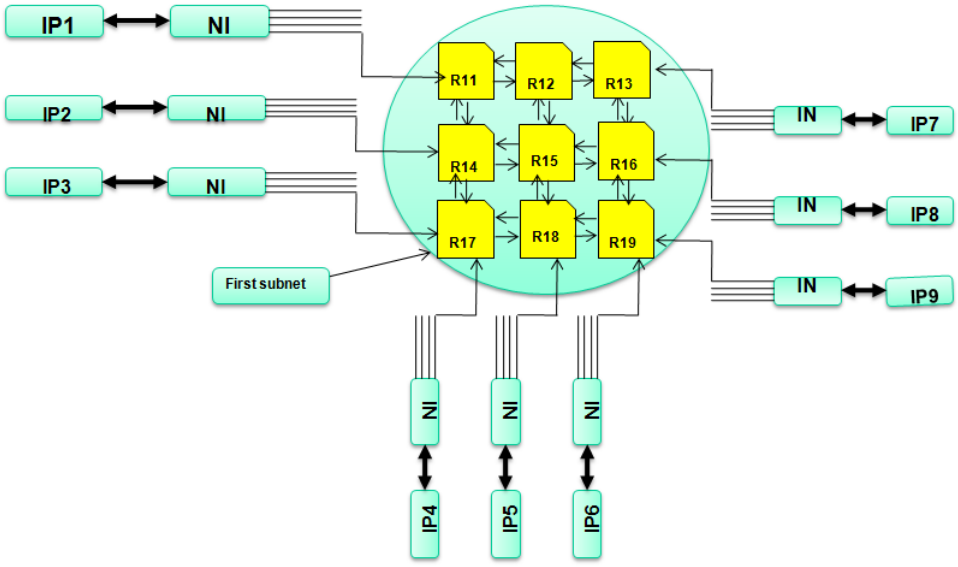
FIGURE 3.1 – Architecture Globale Proposée.



NI : Network Interface, R : Router, IP : Intellectual Property

velle architecture nécessite des liens supplémentaires. Nous résolvons cela par une courte connexion entre les nœuds d'extrémité et les cœurs IP.

FIGURE 3.2 – Un sous réseau (Subnet).



3.1.1 Architecture du Routeur

L'implementation d'un nouvel algorithme de routage implique souvent des changements considerables au niveau de la microarchitecture des routeurs. Un routeur est une entité qui facilite la communication entre les cœurs IP dans les réseaux sur puce. Nous présentons dans cette partie l'architecture de routeur proposée et ses composants principaux. Le routeur est la pièce maîtresse dans la conception DINRA-NoC. Les deux composants ajoutés à l'architecture de routeur de base sont le registre de défauts (FR) et le module de test (TM), sont illustrés dans la Figure 3.8. Chaque nœud d'extrémité a un nombre maximal de 4 ports d'entrée et de 4 ports de sortie. Un port d'entrée / sortie est utilisé pour connecter le commutateur au NI. Le nombre de ports d'entrée dépend de la position du routeur dans le sous-réseau et de la perspective d'éliminer les ports supplémentaires et de réduire la surcharge de la zone et la consommation d'énergie. Chaque routeur peut être connecté à un maximum de quatre routeurs adjacents, ainsi qu'à la propriété intellectuelle locale (IP) et NI.

La figure 3.1 illustre l'architecture détaillée d'un NoC de taille 6x6 utilisant une politique de commutation de trou de ver (WormHole) et le contrôle de flux basé sur les crédits. Pour localiser et distinguer les routeurs du sous-réseau, nous attribuons à chaque routeur une adresse unique définie en coordonnées XY. Pour identifier chaque routeur, nous définissons deux paramètres :

— SN (ID) : Identification du sous-réseau. C'est un numéro indiquant le sous-réseau,
— (X, Y) : Identifier les coordonnées du routeur dans son Subnet.

L'unité de routage considère ces trois adresses (X, Y, SN ID) pour transmettre un paquet. En effet, le SN ID (Tableau 3.1) est un nombre binaire défini par 2 bits et chaque sous-réseau a son code d'identification unique.

Table 3.1 – Codification des différents Sous Réseaux.

ID SN	Value
00	Subnet 0
01	Subnet 1
10	Subnet 2
11	Subnet 3

3.2 Algorithme de Routage Proposé

La politique de routage, mise en œuvre par l'algorithme de routage, doit éviter plusieurs problèmes :

1. le contournement des éléments en panne, un routeur ou une liaison peuvent cesser de fonctionner. Il faut informer les autres routeurs que cette voie est indisponible et proposer des routes alternatives. Ces adaptations aux modifications dynamiques doivent être opérées d'une manière rapide pour être efficaces.
2. la congestion, phénomène qui se produit lorsqu'un trafic trop important converge vers un routeur ou une liaison qui ne peut l'écouler. Cette congestion locale risque alors de se propager vers les routeurs adjacents et de conduire ainsi à un blocage du réseau.

Cette section se concentre sur l'algorithme de routage proposé dans cette thèse. Pour répondre à toutes les exigences de la tolérance aux pannes, la solution proposée peut, lors de la phase initiale du processus, fournir une détection en ligne des pannes permanentes et transitoires. Une fois la détection effectuée, nous pouvons isoler les composants défectueux. Dès que ces étapes ont été effectuées, l'algorithme de routage assure la livraison des paquets à leur destination lorsqu'un chemin existe. A cette étape, nous pouvons indiquer si la destination est inaccessible ou non. De plus, les routeurs ne nécessitent aucun canal virtuel. Ils fonctionnent de manière totalement distribuée pour transmettre les paquets en cas de nœuds défectueux.

Dans l'algorithme de routage proposé, le chemin des paquets dépend de l'état d'encombrement et / ou des défauts du réseau. Lorsque le sous-réseau est encombré ou défectueux, un autre sous-réseau est choisi.

3.2.1 Détection de la Congestion

3.2.1.1 Power gating

Le power gating est une technique d'optimisation visant à réduire la consommation d'énergie statique d'un circuit en coupant le courant de tout ou partie d'un circuit qui n'est pas utilisé. L'activation et la désactivation des parties du circuit est gérée par des transistors PMOS à faibles courants de fuite. Ils sont contrôlés par une logique de mise en veille spécifique au design. L'activation et la désactivation des sous-réseaux se fait en fonction de l'état de congestion local du NoC. Quand un routeur du premier sous-réseau devient congestionné, il alerte ses voisins et l'information est propagée pour demander l'activation du sous-réseau suivant. Cette méthode permet de réduire la puissance jusqu'à 44% comparée au multi-NoC sans power gating. Cependant, aucun gain n'est observé avec le power gating sur NoC unique à cause du faible nombre d'opportunités à endormir les routeurs.

Les applications réelles ont tendance à avoir des besoins en bande-passante variables. Lorsque ces besoins sont faibles, il est judicieux d'utiliser le power gating sur les routeurs

ne transférant pas de données. C'est pourquoi les auteurs proposent de diviser la bande passante du NoC en 4 sous-réseaux de 128 bits au lieu d'un seul à 512 bits. Quand les besoins en bande-passante sont faibles, alors un seul des quatre sous-réseaux est alimenté en courant et les autres sont de nouveau alimentés à mesure que les besoins en bande-passante augmentent.

Il existe des métriques utilisées dans des contributions sur les NoCs pour adapter leurs décisions en fonction de l'état courant de celui-ci. Par exemple, dans[68], la congestion est mesurée afin d'activer ou désactiver des sous-réseaux de leur NoC. En nous inspirant de cette méthode, nous pouvons utiliser des métriques permettant au routeur de prendre la décision de transmettre les données. Il est important que le coût matériel des stratégies mises en place soit minimal. *Ainsi, les stratégies de transmissions locales sont préférées aux stratégies globales car elles nécessitent un coût matériel moins important.*

3.2.1.2 Contention et congestion

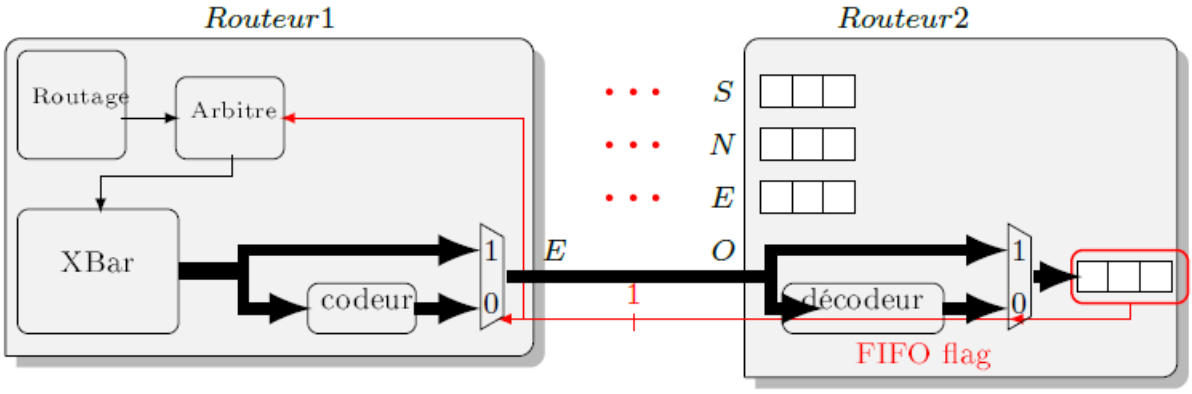
La contention dans un routeur apparaît lorsque les données d'au moins deux entrées d'un routeur doivent être envoyées à la même sortie. Quand cela arrive, un paquet est mis en attente le temps du transfert du premier. Par conséquent, dans cette situation, transmettre plus lentement va augmenter les risques de congestion du NoC. La congestion dans un réseau se produit lorsqu'il y a une augmentation de trafic trop importante provoquant un ralentissement de celui-ci. Dans cette situation, il est important d'éviter de transmettre les flits afin de ne pas faire empirer la situation dans le NoC.

Il existe une méthode peu coûteuse en matériel permettant de connaître l'état de congestion local d'un NoC. Il suffit de vérifier le taux de remplissage des buffers des routeurs. Lorsqu'un buffer en entrée d'un routeur est plein, cela signifie que la direction des flits stockés est bloquée, et plus d'autres buffers se remplissent, plus la congestion est importante. Dans le cas où les buffers sont vides ou seulement occupés par un flit, le trafic dans le réseau est fluide et dans ce cas, une transmission peut être envisagée.

La figure 3.3 montre la mise en place de la stratégie de transmission qui analyse l'occupation des buffers nommée OCC. Les buffers étant des FIFOs (voir chapitre 1), un signal d'alerte (flag) indiquant que le remplissage dépasse un flit est implanté sur chaque buffer du Routeur2 de la figure 3.3. Ainsi, nous implantons dans chaque FIFO un signal qui indique au routeur envoyeur (ici Routeur1) quand le buffer contient plus d'un flit, afin de contrôler le mode de transmission. La stratégie OCC nécessite seulement une interconnexion interrouteur pour fonctionner, ceci est appliqué pour chaque liaison inter-routeur.

Le routage des trous de ver (WormHole) nécessite moins de mémoire de stockage pour le transfert des paquets par rapport aux autres stratégies . Cela peut provoquer un état d'en-

FIGURE 3.3 – Stratégie occupation des buffers (OCC). La sortie Est du Routeur1 est contrôlée selon l'occupation des buffers de l'entrée Ouest du Routeur2. Les signaux de FIFO des entrées restantes sont connectées aux autres routeurs voisins du Routeur2



combrement car les exigences en matière de mémoire tampon peuvent varier en fonction de l'application (Trafic). Le nombre croissant de cœurs contribuera à un trafic supplémentaire qui augmentera également la congestion. Il est donc nécessaire de réduire l'encombrement et d'améliorer les performances en même temps pour les NoC de plus grande taille. Lorsque le trafic augmente ou dépasse un niveau déterminé, la latence augmente et, par conséquent, le débit diminue. Inspiré de Catnap[68], notre sélection de routage est effectuée en fonction de la congestion du sous-réseau. Cette information de congestion est obtenue par le signal d'arrêt émis par le RCS (statut régional de congestion) utilisé dans notre système NoC. Pour la détection d'encombrement dans chaque sous-réseau, nous n'avons utilisé qu'un seul bit supplémentaire dans la mémoire tampon de chaque nœud, ce qui est le minimum supplémentaire requis pour la détection.

3.2.1.3 Détection Locale de la Congestion (Local Congestion Detection)

Le NI d'un nœud peut mesurer le taux d'injection, qui est défini par le nombre de flits injectés dans le réseau sur une période de temps (4 cycles). Ainsi, si le BFM (occupation maximale de la mémoire tampon) du routeur est supérieur à un certain seuil, ce sous-réseau est considéré comme encombré et un bit d'état d'encombrement local (LCS) est défini true. Le mécanisme de détection d'encombrement local est défini par BFM par rapport à un nœud de réseau, selon Catnap. Le seuil le plus performants pour diverses politiques de détection de congestion est égale à 9 flits. Cette information est stockée dans l'interface réseau. Le tableau 3.2 montre les codes associés aux différents états de Sous Réseau. Ainsi, pour mesurer le BFM, le NI connecté à chaque routeur garde une trace de l'occupation de la mémoire tampon d'entrée de chaque routeur.

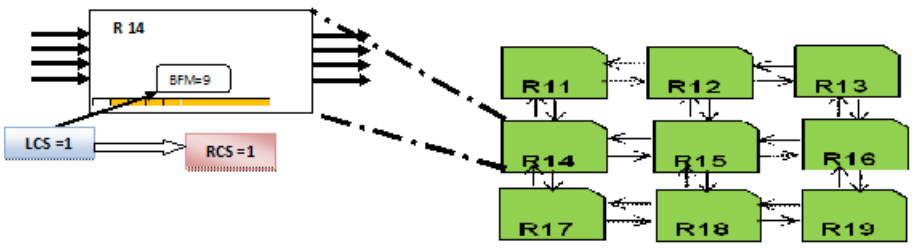


FIGURE 3.4 – Détection Locale de la Congestion.

3.2.1.4 Détection Régionale de la Congestion (Regional Congestion Detection)

Nous utilisons un réseau OU simple avec un bit qui est défini sur true, lorsque l'état d'un des routeurs est vrai (LCS). Cette valeur de bit, que nous appelons l'état régional de congestion (RCS), peut être lue et communiquée à tous les nœuds d'un même sous-réseau. Le réseau OU est conçu comme un réseau H-Tree. Le NI d'un nœud définit son RCS si l'état de congestion local (LCS) est vrai. Le service LCS est déterminé et basé sur le BFM du routeur local. Un nœud d'un sous-réseau détecte l'encombrement local (LCS) si l'état d'encombrement est vrai (basé sur le mode BFM du routeur local) ou si l'état d'encombrement régional (RCS) est vrai.

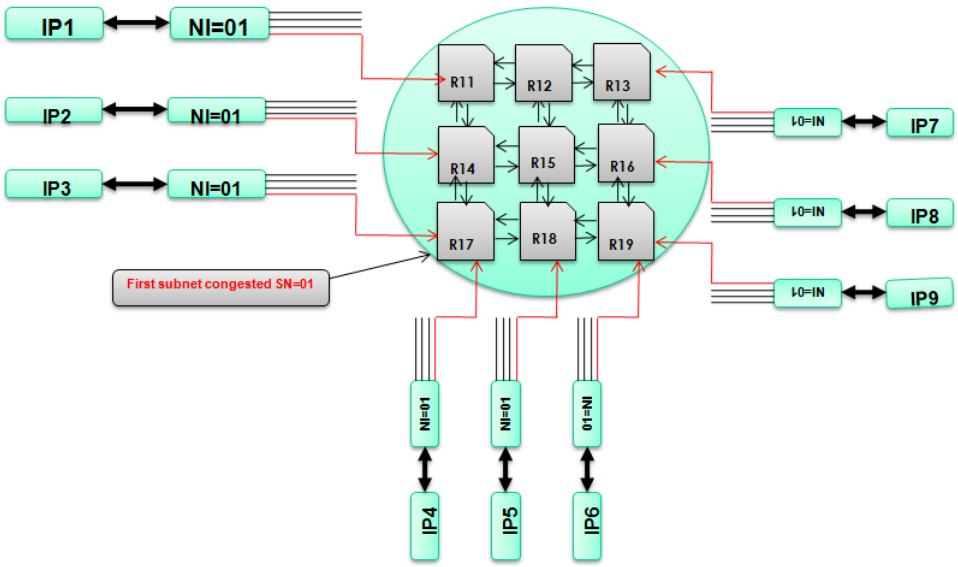


FIGURE 3.5 – Détection Régionale de la Congestion.

Par exemple : avec un NoC composé de quatre sous-réseaux, un routeur du premier sous-réseau est considéré encombré lorsque le BFM est supérieur à un seuil (9 flits, voir sur la Figure.3.4, le routeur R14 est encombré LCS = 1). Dans ce cas, les routeurs du même sous-réseau sont informés et cette information est propagée pour demander l'activation du sous-réseau suivant (RCS = 1, voir Fig.3.4). Le sous-réseau est considéré comme encombré. Simultanément, l'état du sous-réseau est mis à jour (State Subnet = 01, voir tableau 3.2) dans l'interface réseau NI afin d'éviter d'injecter de nouveaux paquets dans ce sous-réseau.

3.2.1.5 Etat d'un Sous Réseau

Lorsque la tête d'un paquet atteint NI, l'un des sous-réseaux est sélectionné (en fonction de l'état du sous-réseau) et le paquet est injecté dans ce sous-réseau, voir Tableau 3.2 (Premier sous-réseau par défaut). Tous les flits d'un paquet voyagent dans le même sous-réseau jusqu'à ce qu'ils atteignent la destination. Le NI est connecté à plusieurs routeurs, comme illustré à la Fig.3.1, chaque routeur appartenant à un seul sous-réseau.

TABLE 3.2 – Codification des différents états d'un sous réseau.

State SN	Code
Normal	00
Congested	01
Out of order	10
Disable	11

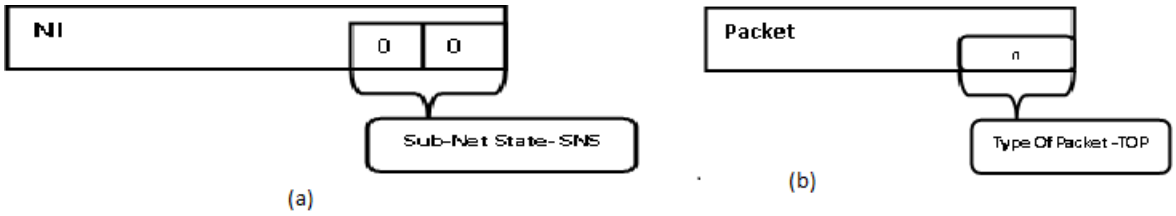


FIGURE 3.6 – (a) Etat d'un sous réseau,(b) Type d'un Paquet(TOP)

Nous distinguons deux types de paquets : les paquets critiques et les paquets non critiques. Nous utilisons 1 bit pour distinguer TOP (type de paquet), comme indiqué sur la figure 3.6 (b). Cette information est ajoutée à Flit d'en-tête Flit de chaque paquet. Donc, TOP est égal à 0 si le paquet est non critique. Sinon, sa valeur est 1.

3.2.2 Les Mécanismes de Détection d'erreurs

Le but des techniques de tolérance aux fautes est de limiter les effets d'une faute, ce qui signifie augmenter la probabilité qu'une erreur soit acceptée ou tolérée par le système. Une caractéristique commune de toutes les techniques des tolérances aux fautes est l'utilisation de la redondance. La redondance est tout simplement l'ajout de ressources matérielles ou temporelles au-delà de ce qui est nécessaire pour un fonctionnement normal du système[64]. Il peut s'agir de matériel : certains modules matériels sont dupliqués, ou tripliqués comme dans la TMR, de temps : des parties d'un programme sont exécutées plusieurs fois, de l'information : le circuit ou le programme utilise une information redondante, ou un mélange de ces trois solutions.

Les solutions traditionnelles impliquant une redondance excessive sont pénalisées en surface, en puissance et en performance[23], et d'autres approches moins coûteuses ne fournissent pas la détection des fautes et les capacités de correction nécessaires. Les systèmes embarqués tolérants aux fautes doivent être optimisés afin de répondre aux contraintes de temps et de surface[65], c'est pourquoi une attention particulière est requise lors du choix des techniques de redondance pour les applications critiques.

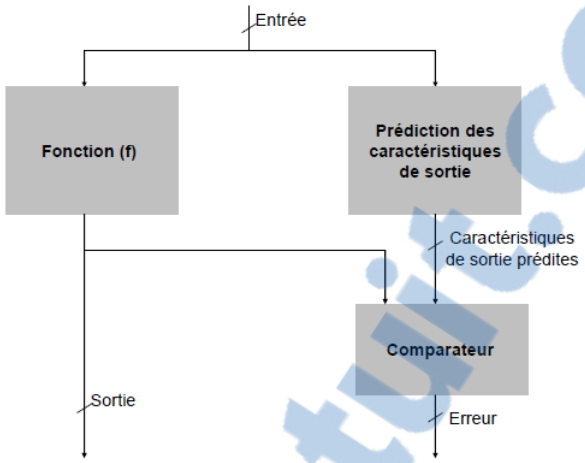
En conséquence, dans cette section on présentera les différentes techniques de tolérance aux fautes existantes en termes de capacité de détection d'erreur, de performances et de surcoût matériel. Ensuite, nous identifierons les techniques qui peuvent pleinement atteindre nos objectifs de conception.

La détection d'erreur génère un signal d'erreur ou un message dans le système.Elle peut être basée sur la détection préventive ou sur une vérification concurrente. La détection préventive est essentiellement une technique **hors-ligne** qui a lieu alors que la prestation de service normale est suspendue, et contrôle le système pour détecter les erreurs latentes et les fautes dormantes, alors que la vérification concurrente est une technique **en ligne** qui a lieu pendant la prestation de service normale [23].De même, Bickham définit la détection d'erreurs concurrente (*Concurrent Error Detection, CED*) comme un processus de détection et de rapport d'erreurs tout en réalisant, en même temps, les opérations normales du système [23].

Les techniques de CED sont largement utilisées pour améliorer la fiabilité des systèmes [76]. L'architecture globale d'un système CED est montrée dans la figure ci-dessous .

Plusieurs techniques de CED basées sur la redondance ont été proposées et utilisées commercialement pour la conception de systèmes de calcul fiables [76]. Elles ont été répertoriées en trois classes : redondance matérielle, redondance temporelle et redondance d'informations.

FIGURE 3.7 – Architecture générale d'un système de détection d'erreurs simultanée [23]



Un système tolérant aux fautes utilise l'une ou plusieurs d'entre elles. Ces techniques se distinguent principalement par leurs capacités de détection d'erreurs et les contraintes qu'elles imposent lors de la conception du système.

1. **Redondance matérielle** est l'approche couramment utilisée [76]. Elle fait référence à l'ajout de ressources matérielles supplémentaires, tel que le doublement du système, en utilisant un comparateur en sortie pour détecter les erreurs. Il est tenu compte ici de la structure du circuit et non pas de la fonctionnalité. Il est tout aussi efficace pour les fautes transitoires que pour les fautes intermittentes ou permanentes. *Cependant, les exigences en surface et en puissance sont très élevées.*

2. **Redondance temporelle** Elle désigne une technique de redondance qui nécessite une seule unité effectuant une même opération deux fois de suite. Si une différence est constatée entre les deux calculs successifs, cela signifie qu'une faute transitoire ou intermittente est apparue lors de l'un ou l'autre des calculs [23]. *Dans cette approche, il y a une pénalité en termes de temps supplémentaire,* cependant la pénalité matérielle est moindre. Il s'agit d'une technique de réplication temporelle, sans considération de la fonctionnalité du circuit.Dans cette technique, les fautes intermittentes et transitoires sont détectées (comme indiqué dans la Figure 3.7) *mais les fautes permanentes ne le sont pas.*

3. **Redondance d'information** L'idée sous-jacente d'un schéma de redondance d'information est d'ajouter de l'information redondante aux données transmises, stockées ou traitées, afin de déterminer si des erreurs ont été introduites [23]. C'est une façon de protéger les données grâce à un codage mathématique qui peut être réutilisé ensuite pour décoder les données originales (comme montré dans la Figure 3.7).

   Typiquement, la redondance d'information est utilisée pour protéger des éléments de

stockage (comme la mémoire, les caches, les piles de registres, etc.) [76] comme par exemple dans les processeurs Power 6 et 7 [76]. Ces codes sont classés en fonction de leur capacité de détection et de l'efficacité du code de correction, et de leur complexité.Les codes de détection d'erreurs (Error Detecting Codes, EDC) ont un surcoût matériel moindre que les codes correcteurs d'erreurs. Nous ne rentrerons pas dans les détails, mais ce qui nous intéressent ces leurs caractéristiques.

La stratégie de codage de parité est la plus simple et offre le plus faible surcoût matériel [76]. Elle est basée sur le calcul des parités paires ou impaires pour les données d'un mot de longueur N. La parité peut être calculée avec l'opération XOR entre les bits de données. Un code de parité a une distance de 2 et peut détecter toutes les erreurs impaires.

Les codes à redondance cyclique (Cyclic Redundancy Check, CRC) constituent une autre classe d'EDC. Ils sont communément employés pour détecter les erreurs dans les systèmes numériques [23]. Les CRC sont des codes de contrôle de parité avec la propriété supplémentaire que le décalage cyclique d'un mot code est également un mot code.

L'idée est d'ajouter une somme de contrôle à la fin de la trame de données de telle manière que le polynôme représenté par la trame résultante est divisible par le polynôme générateur G(x) convenu par l'expéditeur et le récepteur. Lorsque le destinataire reçoit la trame contenant la somme de contrôle, il la divise par G(x) et si le reste n'est pas nul, il y a eu une erreur de transmission. Il est alors évident que les meilleurs polynômes générateurs sont ceux qui sont le moins susceptibles de diviser régulièrement en une trame qui contient des erreurs. Les CRC se distinguent par les polynômes générateurs qu'ils utilisent. Ils ne peuvent pas déterminer directement la position du bit d'erreur pendant le processus de décodage. Par conséquent, ils sont limités à la détection d'erreur uniquement. En résumé, en augmentant la complexité des codes, leur efficacité augmente également. Choisir le bon code dépend des besoins de l'application. Dans ce travail,le CRC peuvent être utilisés pour une meilleure couverture des erreurs.

3.2.3 Détection des Fautes

BIST [77] est la solution traditionnelle permettant aux réseaux sur puce de gérer les défaillances des routeurs, des liaisons et des interfaces réseau. Cette méthode est un schéma de détection d'erreur. Mais cette technique nécessite des circuits externes pour détecter les erreurs et du temps afin de les déterminées, qui peut réduire les performances du NoC en termes de temps et de latence. Pour développer notre approche, nous avons opté pour le cycle de redondance cyclique (CRC), une solution permettant de détecter les erreurs en comparant les entrées et les sorties de chaque routeur afin de détecter les paquets erronés et les composants défectueux dans un NoC (Fig.3.8). Cette approche est inspirée de [77], avec quelques modifications. Chaque routeur est équipé d'un module de test (Test Module,

TM). L'objectif de ce mécanisme est d'activer la détection de panne en ligne. Le polynôme CRC utilisé est g = 1 + x + x4, nous ajoutons donc 4 bits à l'en-tête de chaque paquet. Lorsqu'une erreur est détectée n'importe où dans le routeur / lien, nous savons que l'erreur existe, mais nous ne pouvons pas la corriger avec le décodage CRC.

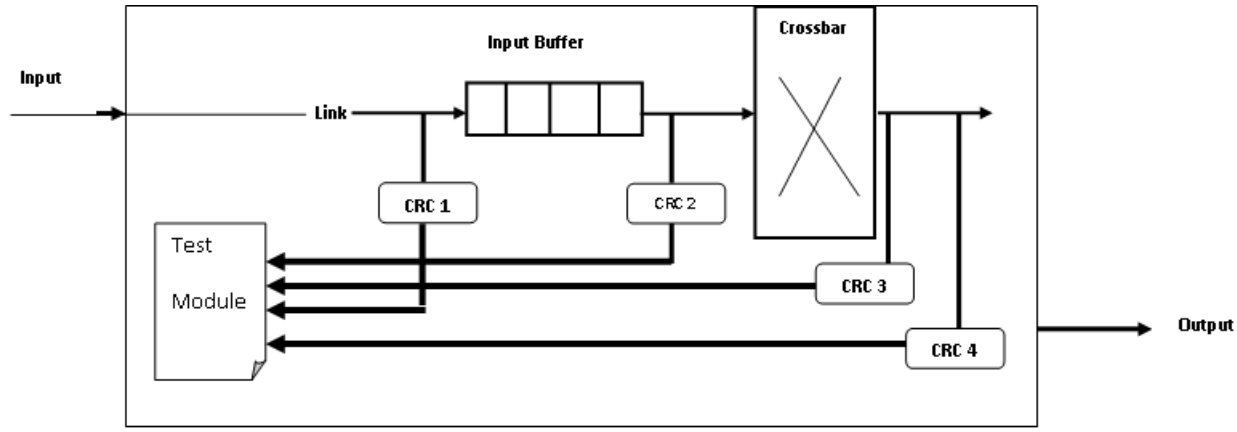


FIGURE 3.8 – Architecture interne du routeur.

Pour tester et diagnostiquer l'infrastructure de communication dans le NoC, il faut tout d'abord détecter le défaut. Ensuite, il est nécessaire d'identifier les composants défectueux : routeurs, liens ou cœurs. Enfin, nous analyseront comment utiliser efficacement les composants restants sans défaut. Dans le routeur, le défaut peut se produire dans tous les composants (barre transversale, tampons et autres). À cette densité élevée dans les réseaux sur puce, la prise en compte des défauts uniquement dans les routeurs ou les liaisons ne fournit pas une sécurité optimale. Les autres composants tels que les tampons d'entrée et la barre transversale doivent faire l'objet d'une attention particulière pour assurer la tolérance aux pannes et améliorer la fiabilité du système.

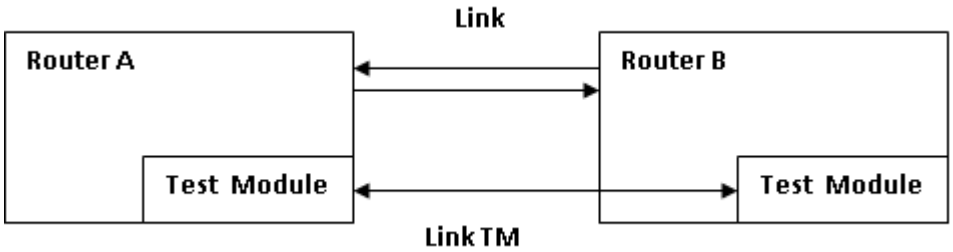


FIGURE 3.9 – Communication entre deux routeurs voisins.

Pour atteindre cet objectif, nous présenterons dans cette section le module de test (TM) ajouté à notre routeur (voir la figure 3.8). La TM est l'un des composants principaux de

notre système, car elle gère le diagnostic de toutes sortes de pannes dans quatre composants principaux : liaisons inter-routeur, tampons d'entrée, barre transversale et entête. Nous commençons dans cette sous-section pour décrire les principales fonctionnalités du Test–Module (TM) et son rôle important dans l'orchestration des différents processus à l'intérieur du routeur. Le module de test détecte et localise les routeurs ou les liaisons défectueux. Par conséquent, dans le même sous-réseau, seuls les routeurs adjacents peuvent communiquer cet information.

Ces informations sont toujours envoyées au FR (registre des pannes) afin de conserver l'occurrence de pannes dans les tampons d'entrée, la barre transversale et les flits d'en-tête. L'objectif est d'utiliser ces informations lors de la sélection du prochain port. Nous avons utilisé un mécanisme simple de détection de pannes basé sur un contrôle de redondance cyclique (CRC) dans chaque composant de sortie du routeur qui lit le flit entrant afin de détecter toute erreur. En fonction de cette vérification, le Test-Module (TM) envoie au nœud en amont un signal à un bit pouvant être égal à 0 ou 1, respectivement pour valide ou défectueux, comme indiqué sur la Fig 3.8. Chaque routeur envoie les informations collectées à tous les nœuds voisins. Cette information est représentée dans un signal à 3 bits (voir Tableau 3.3) représentant l'état du routeur / lien dans chaque direction (Nord, Est, Sud et Ouest). En fonction de ces états, notre algorithme (DINRA) lit l'état de défaut des nœuds / liaisons suivants reçus du TM et vérifie le nombre de consignes de sécurité possibles. Par exemple, lorsqu'un défaut est détecté dans la mémoire tampon, un signal est envoyé pour informer le module TM de la présence du défaut. Le même cas est appliqué pour les autres composants.

TABLE 3.3 – Codification de différents états de liens et de routeurs.

State	Value	Description
1	00	Port East unsafe
1	01	Port North unsafe
1	10	Port South unsafe
1	11	Fail Router
0	00	Port East safe
0	01	Port North safe
0	10	Port South safe
0	11	Safe Router

Lors de la réception de ce signal, la TM désactive le port de sortie entier et le routeur défaillant pour économiser l'énergie dynamique. Simultanément, la TM met à jour le tableau d'état FR en signalant le lien connecté au routeur défaillant. Ces informations sont envoyées en permanence à tous les nœuds voisins.

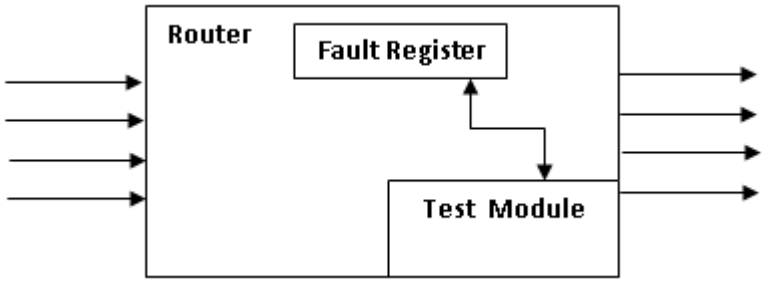


FIGURE 3.10 – Communication entre le registre de défauts et le module de test.

Enfin, pour conserver les informations sur les défauts dans les routeurs ou les liaisons, le TM interagit avec l'unité FR pour échanger des informations sur les défauts et des signaux de commande, comme illustré à la Fig. 3.8.

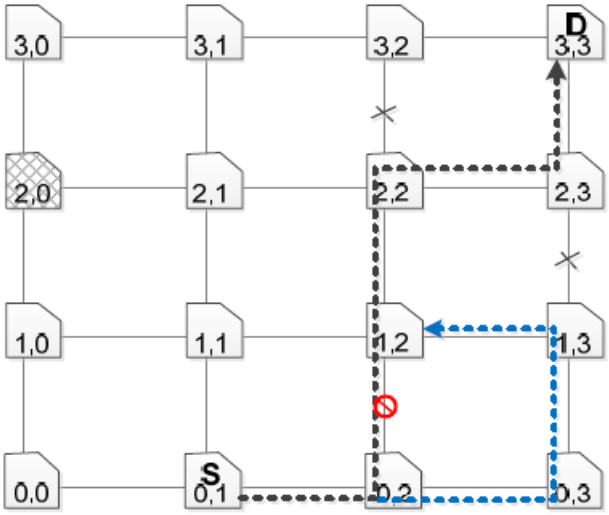
3.2.3.1 Détection des Fautes Permanentes

Les erreurs permanentes ne peuvent être éliminées par une simple réinitialisation du circuit. Les routeurs adjacents à un routeur présentant une défaillance permanente sont informés de son état. Ceci est fait pour empêcher tout trafic vers ce routeur défectueux (par exemple, en désactivant les ports de sortie menant à ce routeur). Le même cas est appliqué pour les liens défectueux.

3.2.3.2 Détection des Fautes Transitoires

Pour identifier le problème à résoudre, considérons le scénario décrit par la figure 3.11 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)et(1,2) et divise le paquet. Le sous-paquet 1 va arriver au noeud destination par les ressources réservées tandis que le sous-paquet 2 est coincé dans le noeud (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. un pseudo flit est stocké dans le routeur (0,1) et un temporisateur est initialisé à kl cycles. Si le contournement réussit, le pseudos flits d'en-tête est donc supprime. Sinon , après kl tentatives de retransmission, si le lien de sortie est encore défectueux et donc pas disponible. Le processus de retransmission est lancé.

FIGURE 3.11 – Faute dynamique arrive sur le lien (0,2)et(1,2).



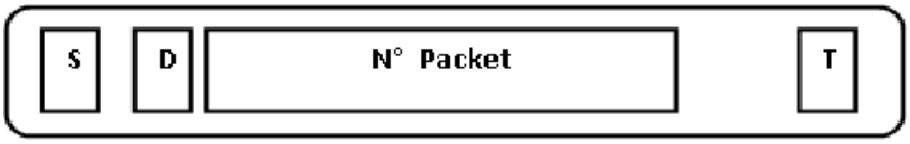


FIGURE 3.12 – Le format du message de notification.

S : Source, D : Destination, T : Type of Packets

Retransmission des Paquets

Il existe un cas où certains des paquets donnés sont stockés dans la mémoire tampon et que le lien de sortie ou le routeur deviennent soudainement défectueux pendant le transit des paquets. Cela modifie la validité du chemin (faute dynamique), qui sera ultérieurement prouvé comme étant défectueux. Cette corruption crée des sous-paquets qui ne peuvent pas arriver à leur destination. Une copie de l'en-tête de ce paquet sera stockée dans une mémoire tampon spéciale. Un message de notification est envoyé au nœud source pour transmettre à nouveau tous les paquets. Nous devons également supprimer les flits qui ont été transmis dans le cas d'une destination inaccessible ou uniquement pour transmettre les paquets qui n'ont pas encore été transmis dans le cas d'une erreur temporaire. Voir Fig.3.12, le message de notification dans les deux cas.

Selon l'état du réseau, il existe trois cas de figure : Cas I, lorsque le paquet est envoyé d'un IP à un autre et que le sous-réseau actuel est congestionné. Cas II et III, lorsque le paquet est envoyé du routeur actuel au saut suivant, dans ce cas, l'IP de destination peut être accessible ou non. Il existe deux cas où un paquet est considéré comme perdu et doit être retransmis une autre fois par la source (cas II et III) :

— Un fragment de paquet arrive dans un sous-réseau défectueux : par exemple, un routeur sans sortie saine ou un paquet est bloqué par des restrictions de routage vers sa destination.
— Le deuxième cas : la destination n'est pas atteignable lorsqu'un fragment de paquet ne peut pas atteindre le nœud de destination avant un «Timeout» T (une valeur empirique qui varie en fonction de la taille de NoC).

Afin de gérer et de tolérer les erreurs temporaires qui peuvent survenir lors du transfert de paquets (par exemple, lorsqu'un routeur / lien devient soudain défectueux), un message de notification est envoyé au nœud source pour retransmettre à nouveau tous les paquets n'ayant pas atteint leur destination. Nous désignerons un sous-ensemble de paquets (15 %) et les marquerons comme des paquets critiques.

Les erreurs transitoires surviennent généralement pendant une très courte période et ne sont pas destructives. Ainsi, après k tentatives, le module de test peut considérer les défauts dynamiques (temporaires) comme permanents. Nous voulons donc tolérer plusieurs défauts en ligne sans réinitialiser le circuit.

3.2.4 Algorithme de Routage

Les données sont transférées entre des IP en paquets le long de chemins calculés par l'algorithme de routage. Il existe deux types de stratégies de routage : le routage déterministe et le routage adaptatif. Dans le routage déterministe, le chemin de routage est

complètement déterminé par les adresses source et de destination. Ses avantages incluent la simplicité de l'architecture du routeur et la propriété sans blocage. En raison de la logique matérielle simplifiée, le routage déterministe offre une latence plus faible lorsque le NoC n'est pas encombré. Cependant, à mesure que le taux d'injection de paquets augmente et que certaines liaisons et routeurs deviennent encombrés, le routage déterministe risque de se dégrader car il ne peut pas répondre dynamiquement à la congestion du réseau[52]. En outre, des pannes de liaison permanentes peuvent rendre le NoC inutilisable.

En revanche, le routage adaptatif prend en compte les variations de trafic dans le réseau et calcule dynamiquement des chemins alternatifs pour acheminer les données via des régions moins encombrées. De plus, si l'architecture NoC est équipée de mécanismes de détection de défaillance de liaison, le routage adaptatif peut résoudre ces défaillances et faciliter ainsi la tolérance aux pannes.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 utilisée de canaux virtuels.

Le routage de paquets est l'un des facteurs importants dans la conception des NoCs. Pour atteindre la tolérance de panne, l'algorithme de routage a besoin de plusieurs chemins pour acheliner les paquets vers chaque paire de destination source. Premièrement, l'algorithme de routage proposé doit être adaptatif pour pouvoir choisir entre eux. Deuxièmement, la topologie doit fournir un autre itinéraire. Les algorithmes de routage complexes peuvent introduire une surface supplémentaire et une surcharge d'énergie [48]. Dans notre approche, l'objectif principal est d'optimiser les performances en termes de latence et de fiabilité en obtenant un taux élevé de remise des paquets réussi.

Voici quelques explications sur notre algorithme : NI connecté à un routeur injecte d'abord un paquet dans le sous-réseau-0 (ligne 1) et tous les flits d'un paquet voyagent dans le même sous-réseau jusqu'à leur destination. Si le sous-réseau-0 est encombré ou défectueux, un sous-réseau plus élevé (s + 1) peut être activé plus rapidement afin d'éviter une perte de performances (ligne 11) et le même cas est appliqué aux autres sous-réseaux. Pour acheminer des paquets dans le même sous-réseau, nous utilisons l'algorithme de routage Last North et, dans le cas contraire, nous utilisons le South Last pour éviter les composants défectueux (Links ou Routers-line 7). Et, si nous ne pouvons pas acheminer le paquet dans le second cas, nous activons le sous-réseau le plus élevé (s + 1) et définissons le sous-réseau d'état sur défectueux. Le même processus est appliqué si le sous-réseau actuel passe à congestionnée sur la ligne 12 .

3.2.4.1 Tolérance aux Fautes - Fiabilité

DINRA a toutes les informations sur l'état de défaut des prochains liens et nœuds. Il y a trois directions possibles pour acheminer les flits. Le statut de congestion est la

Algorithm 1 path computation

1. State_sub_s0 = Active & RCS =0 ;
2. For (s=0 , s = 3 , s++) /*– Source S(x,y,z), Destination D(x',y',z')
3. If state_subnet_s = 00 /*– — normal state
4. If RCS = 0 then
5. For each S, D
6. North Last /*———-Routing Algorithm by default
7. if (link_state && router) == unsafe
8. South Last /*——-Routing Algorithm
9. else /*—— Activate s+1
10. state_sub = Faulty ; s=s+1 ;
11. Else s+1 /* ———— Next Subnet
12. state_sub = Congested ; s=s+1 ;
13. state_sub_s= Active /*———active next subnet
14. End loop
15. End loop

priorité la plus haute. Nous avons adopté cette condition pour assurer la tolérance aux pannes et une meilleure performance, que ce soit en présence ou en l'absence de pannes. Lorsqu'il n'y a pas de route valide disponible, DINRA choisit une autre route dans les priorités du deuxième sous-réseau (diversité de chemin et encombrement), comme illustré dans l'algorithme.

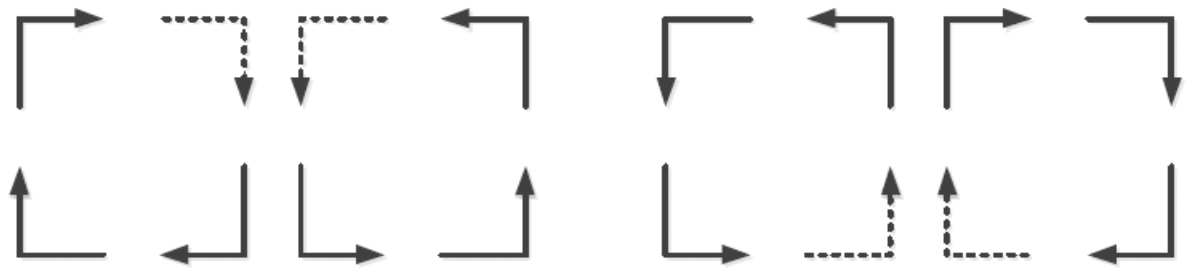
DINRA -FTNoC adopte la commutation Wormhole. De cette manière, le transfert de paquets peut être exécuté de manière efficace tout en maintenant une petite taille de la mémoire tampon.

3.2.4.2 Algorithme de Routage DINRA et Deadlock

Les algorithmes de routage peuvent être distingués selon la façon avec laquelle ils sont implémentés, on effet on trouve deux types d'implémentation ; ceux basés sur les tables de routage (look up table) et ceux basés sur les machines à état finis (FSM) : pour le premier type un routeur recevant un message qui ne lui est pas destiné consulte sa table de routage afin de déterminer le lien de sortie choisi pour l'acheminement du paquet. Pour le deuxième type d'implémentation la décision du routage est définie suivant les états des ports des sorties qui lui sont associés.

Cet algorithme combine deux algorithmes de routage adaptatif North-Last et South-Last. Les deux utilisent des restrictions pour éviter les blocages [32]. Le problème d'inter-blocage peut survenir avec le routage adaptatif. La plupart d'entre eux utilisent des canaux virtuels (VC) ou Turn Modèle afin d'éviter le blocage.

FIGURE 3.13 – Routage South-Last (à gauche) / North-Last (à droite).

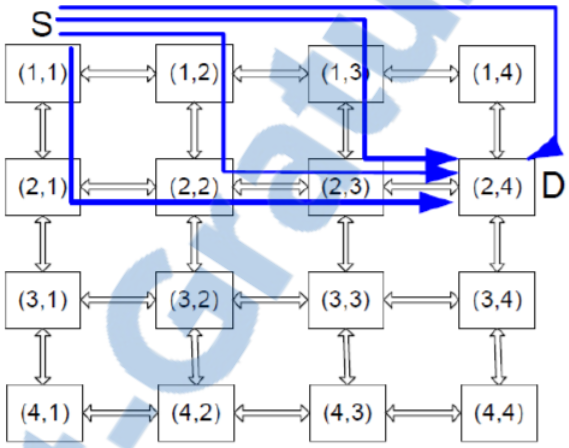


Algorithmes de routage adaptatif North-Last

C'est un algorithme de routage partiellement adaptatif. Dans un réseau maillé, il applique deux restrictions de routage à n'importe quel nœud, à savoir le virage nord-ouest et le virage nord-est. Les paquets ne peuvent pas changer de port NORTH à port WEST ou

NORTH au port EAST d'un nœud, voir la Figure 3.13. Selon cet algorithme, le message sera acheminé dans la direction NORD uniquement s'il s'agit de la dernière direction à suivre. Une fois qu'un message est tourné vers le nord, plus aucun virage n'est autorisé ; par conséquent, le virage nord doit être fait en dernier. Dans cet algorithme, le message est d'abord acheminé de manière adaptative dans les directions ouest, Sud, Est et enfin dans la direction Nord. Si, dans une communication, il est nécessaire de déplacer un paquet dans la direction NORD avec d'autres directions, ce paquet doit traverser les autres direction et finalement dans la direction du NORD.

FIGURE 3.14 – Exemple pour l'algorithme North Last.



Supposons que dans un réseau, le nœud SOURCE est (1,1) et le nœud de destination est (2,4). La figure 3.14 illustre les chemins de routage possibles dans l'algorithme de routage North Last.

Discussion : De nombreux algorithmes de routage utilisent turn modèle [32] pour éviter les blocages et offrir une flexibilité élevé pour éviter les zones défectueuses ou encombrées dans un NoC. Cette approche nécessite que certains tournants soient interdits. Pour assurer la fonctionnalité sans interblocage de notre algorithme, notre approche est basée sur la même que celle utilisée par [32]. Dans DINRA, nous avons combiné des protocoles de routage à deux tours : Nord Last et South Last (cf. ligne 6-8.) Dans le modèle North-Last turn, la direction nord est prise en dernier lieu. L'idée est d'exploiter ce concept lors de la sélection de la sortie appropriée, Output = West ; Sud ; Est ; Nord le dernier élément est la dernière direction, qui dans ce cas est Nord, la même chose pour Sud - elle peut être écrite dans le dernier comme suit : Sortie = Ouest ; Nord ; Sud est. Donc, en cas de lien défectueux, nous utilisons la seconde alternative pour router les paquets vers la destination. Cela prouve que les paquets n'ont plus besoin de reprendre les mêmes directions. Ainsi, nous nous assurons qu'il n'y a pas de risque d'impasse. Pour que l'algorithme soit sans verrouillage en temps réel, tous les paquets s'épuisent en cas de retransmission.

3.2.4.3 Adaptativité

Du point de vue adaptabilité, on peut classer les algorithmes de routage en deux catégories ; les algorithmes de routage déterministe et les algorithmes de routage adaptatif. Pour le routage adaptatif, plusieurs chemins peuvent être considérés lors de la transmission du paquet et le chemin sélectionné est celui qui montre le minimum possible de congestions, contrairement aux algorithmes déterministes le paquet dans ce cas pourrait réaliser plusieurs tours avant d'atteindre sa destination.

L'une des propriétés principales de l'architecture DINRA est sa capacité à être adaptée pour fournir plusieurs itinéraires permettant de transférer des paquets pour chaque paire de nœud source-destination. Cela augmente considérablement la fiabilité de l'ensemble du NoC. DINRA a prouvé son rendement en garantissant à la fois la tolérance aux pannes et l'encombrement.

3.3 Conclusion

Dans ce chapitre une nouvelle architecture de routeur pour les NoCs a été présentée . Le routeur proposé est équipé de deux composants principaux pour améliorer encore la fiabilité et les performances. Tout d'abord, nous avons présenté Test Module TM, un mécanisme intelligent permettant de gérer les défaillances dans les liens, tampons, crossbar et flit d'entête. deuxièmement Fault Register (FR) afin de fournir des chemins d'échappement aux cas où des défaillances sont détectées dans les liens ou routeurs en sauvegardant l'état des liens et routeurs voisins.

Les deux composants présentés ont montré une grande capacité à absorber la dégradation des performances dans le cas d'un taux de pannes élevé combinés avec l'algorithmes de routage. Architecture proposé offre une fiabilité élevée. Ceci est analysé plus en détail dans le chapitre suivant.

Algorithm 2 North-Last

INPUTS : Current router address - Xcurrent is (X-coordinate of Current router) and Ycurrent is (Y-coordinate of Current router).
Destination router address - Xdestination is (X-coordinate of Destination router) and Ydestination is (Y-coordinate of Destination router).
Procedure :

1. Begin
2. Ex = Xdestination - Xcurrent
3. Ey = Ydestination - Ycurrent
4. If (Ey > 0 and Ex < 0) then
5. Select Output Channel between (NORTH, EAST) ;
6. End if ;
7. If (Ex >= 0 and Ey > 0) then
8. Select Output Channel = EAST ;
9. End if ;
10. If (Ex < 0 and Ey < 0) then
11. Select Output Channel between (NORTH, WEST) ;
12. End if ;
13. If (Ey < 0 and Ex >= 0) then
14. Select Output Channel = WEST ;
15. End if ;
16. If (Ey = 0 and Ex > 0) then
17. Select output Channel = SOUTH ;
18. End if ;
19. If (Ex < 0 and Ey = 0) then
20. Select Output Channel = NORTH ;

Chapitre 4

Expérimentations et résultats

Sommaire

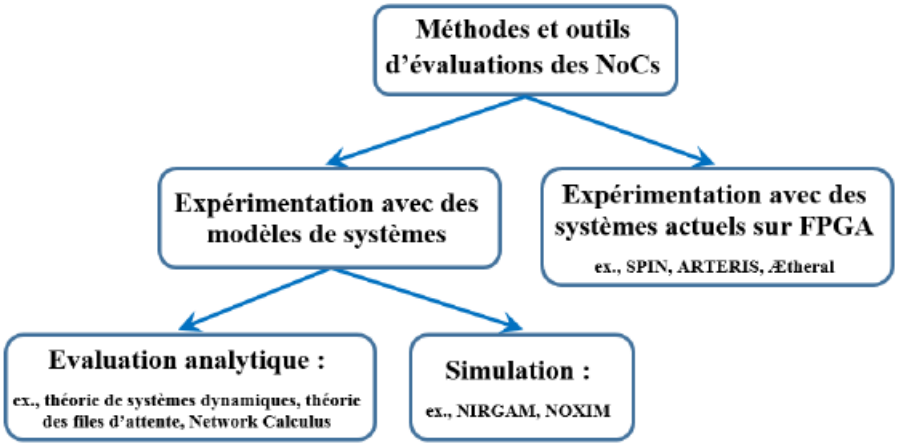
Dans le chapitre précédent nous avons présenté notre architecture. Ce chapitre a pour but d'évaluer notre algorithme que nous avons proposé. Nous allons implémenter l'algorithme de routage tolérant aux fautes (DINRA) dans une plateforme de simulation pour réseau sur puce. Des simulations intensives nous permettent d'évaluer l'efficacité de l'architecture proposée par rapport à celles existantes. Au même temps nous comparerons la capacité à tolérer des fautes de DINRA avec l'algorithme PDA FTR .

4.1 Plateforme d'expérimentation

Les systèmes embarqués sur puce (SoC : Systems-on-Chip) sont devenus de plus en plus complexes grâce à l'évolution de la technologie des circuits intégrés. Des études récentes ont montré que pour améliorer les performances du réseau sur puce (NoC : Network-on-Chip), l'architecture de celui-ci pouvait être personnalisée, soit au moment de la conception, soit au moment de l'exécution. L'objectif principal de cette thèse est d'implémenter de nouvelles approches pour améliorer les performances des NoCs, notamment la latence, le débit, la consommation d'énergie, et la simplicité de mise en œuvre.

Les architectures d'interconnexion sur puce adoptées pour les systèmes sur puce sont caractérisées par un compromis de qualité de service de la communication entre la latence, le débit, la charge des liens, la consommation d'énergie et les exigences de la surface de silicium. Les architectures NoC peuvent être évaluées soit par expérimentation avec les systèmes existants, soit par expérimentation avec des modèles de systèmes (modèles de simulation ou d'évaluation analytique), comme le montre la figure 4.1. Cependant, **l'expérimentation des performances des NoCs avec des vrais systèmes sur FPGA est très coûteuse**. Par conséquent, la modélisation est la solution la plus efficace pour l'évaluation et l'optimisation de l'architecture du NoC avant sa mise en œuvre sur la puce.

FIGURE 4.1 – Classification des méthodes et des outils d'évaluations des NoCs.[69]



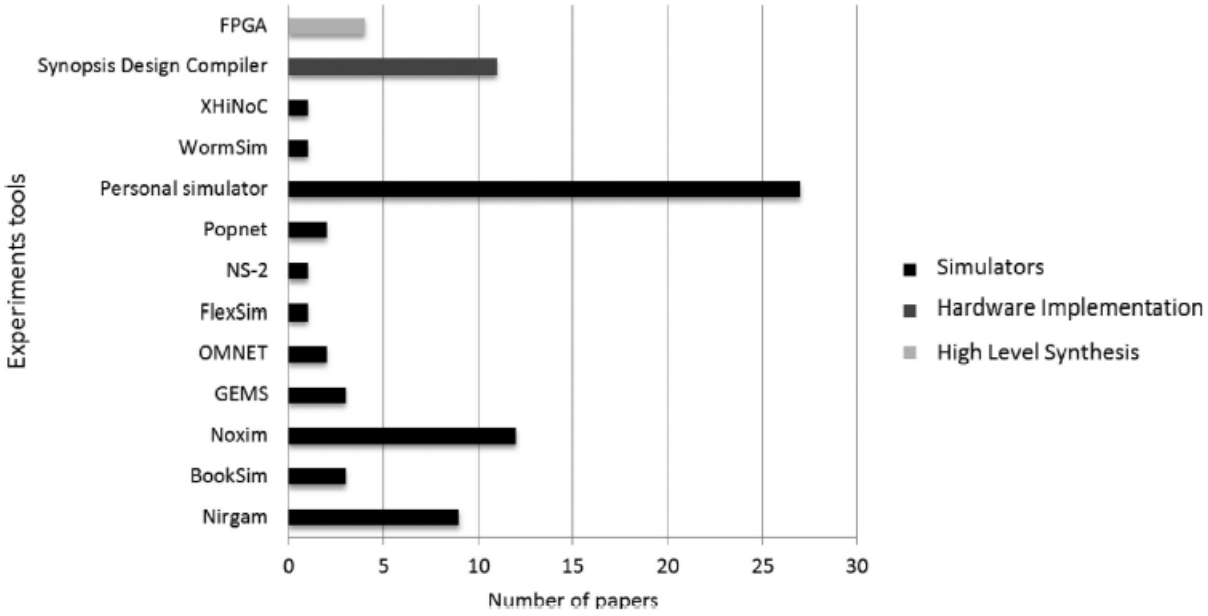
4.1.1 Evaluation par simulation

Cette section a pour but de présenter quelques simulateurs de NoC existantes, afin de choisir l'un d'entre eux qui offre un bon rapport de simplicité, rapidité, possibilité de personnalisation de l'architecture NoC (topologie, espace tampon, etc.), possibilité d'évaluation des performances (latence et consommation d'énergie) du NoC et prise en charge des topologies, des algorithmes de routage et des applications les plus couramment utilisés pour le NoC. Ces contraintes sont considérées pour le choix du simulateur utilisé dans la suite pour la simulation de notre approches pour le NoC.

Il existe de nombreux simulateurs disponibles dans la littérature qui peuvent être utilisés pour simuler un réseau NoC. On peut citer Atlas, Spider, NoCGen, Noxim, Sicosys, Nostrum, Nirgam, gpNoCSIM, Garnet, NNSE, BookSim, WormSim, HNoCS, Chaos, etc. Deux d'entre elles sont populaires et sont plus utilisées dans de nombreux ouvrages : Noxim et

Nirgam . Cependant, nous observons que dans la plupart des cas, les auteurs optent pour leurs propres simulateurs développés dans différentes langues telles que C++ et VHDL (Voir figure 4.2).

FIGURE 4.2 – Les outils d'expérimentation utilisés dans la la littérature pour les NoCs. [73]



L'étude de l'existant nous permet de voir les différentes approches pour simuler une plate-forme NoC. Il apparait qu'au moment du choix aucune solution ne correspondait exactement à nos besoins de simulations de réseau d'interconnexion. Cette étude met en évidence que dans la majorité des cas, la modélisation du réseau d'interconnexions est réalisée en décrivant chaque élément de cette interconnexion. Ceci peut poser un problème si on envisage de simuler un réseau de grande taille. Par exemple en SystemC, il en résulte un grand nombre de threads SystemC ce qui risque de ralentir la simulation.

Pour cette raison nous avons voulais voir plus en détail les simulateurs de réseau d'interconnexion les plus proches de nos besoins de simulation, leur description est présentée dans cette section, afin de pouvoir choisir un simulateur qui convient le plus à la simulation de nos travaux sur le NoC.

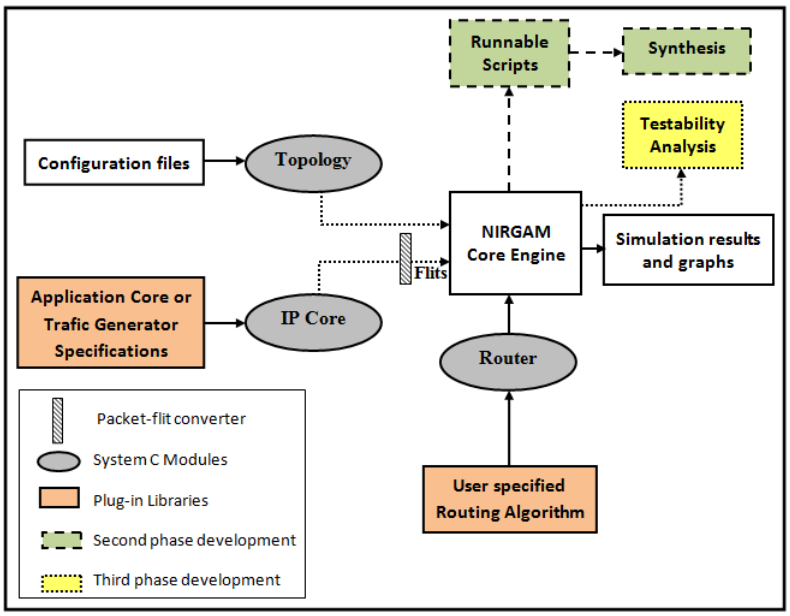
NIRGAM [1] est simulateur SystemC à base d'événements discrets et de cycle précis pour la conception de NoC en termes d'algorithmes de routage et d'applications sur diverses

1. NIRGAM A Simulator for NoC Interconnect Routing and Application Modeling. http ://nirgam.ecs.soton.ac.uk Avaiblle [O nline]

topologies (2D Mesh et 2D Torus actuellement mises en oeuvre). NIRGAM est le fruit d'une recherche collaborative entre le groupe d'électronique de conception des systèmes à l'École d'électronique et d'informatique de l'Université de Southampton au Royaume-Uni et le département d'informatique et de génie informatique de l'Institut national de technologie à Malaviya, à Jaipur en Inde. Ce projet est financé par l'EPSRC (UK). NIRGAM permet la génération de NoC, de la simulation jusqu'à l'évaluation du NoC (cf. figure 4.3). Il est écrit en SystemC et peut être exécuté en toute plate-forme UNIX avec unstandard compilateur C++. Il support diverses distributions et scénarios de trafic. Il est disponible gratuitement sur son site Internet (http ://nirgam.ecs.soton.ac.uk/), et il est opensources, ce qui permet aux concepteurs de pouvoir l'adapter à leurs besoins de simulation, comme par exemple implémenter d'autres topologies, ou d'autres patterns trafic, etc.

Cependant, l'outil en sa version originale ne considère que les topologies 2D Mesh et 2D Torus et ne supporte pas la personnalisation de l'architecture NoC. D'importantes applications ne sont pas également prises en compte dans cet outil.

FIGURE 4.3 – Architecture Nirgam



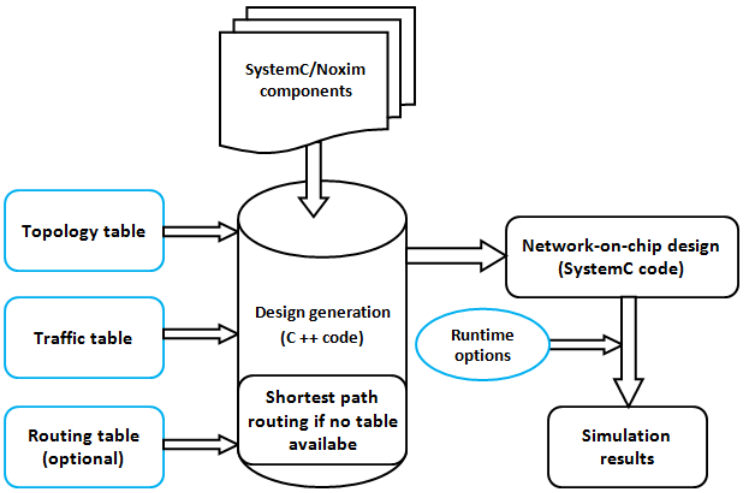
Noxim [2] est un outil de simulation MSoC/NoC développé à l'Université de Catane (Italie). L'outil est disponible gratuitement sur le site Noxim via SourceForge sous conditions de la licence GPL. Il est compilé pour un environnement Linux. Noxim est écrit en SystemC, il utilise le moteur de simulation SystemC ainsi que la bibliothèque de composants de réseau de niveau comportemental pour la simulation rapide. Il a une interface de ligne de commande pour définir plusieurs paramètres du NoC. En particulier, l'utilisateur peut

2. SourceForge, Noxim : Network-on-Chip Simulator, 2008. [Online].

personnaliser la taille du réseau, la taille de la mémoire tampon, la distribution de la taille des paquets, l'algorithme de routage, la stratégie de sélection, la vitesse de l'injection de paquets, la distribution de temps de trafic, le modèle de trafic, la distribution de trafic point-chaud (hotspot), etc.

L'outil Noxim permet la conception, la simulation et l'évaluation d'un NoC (cf. figure 4.4). Le simulateur permet d'évaluer le NoC en termes de débit, de latence et de consommation d'énergie. Ces informations sont fournies à l'utilisateur à la fois en termes de résultats moyens globaux et de résultats pour chaque communication. Plus précisément, l'utilisateur est autorisé à recueillir différents paramètres d'évaluation, y compris le nombre total de paquets/flits reçus, le débit moyen global, la latence max/min globale, la consommation totale d'énergie, et latence/débit/énergie pour chaque communication, etc.

FIGURE 4.4 – Architecture Noxim



4.1.2 Simulateur utilisé :

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.

Notre choix de simulateur est porté sur le simulateur Noxim.L'intérêt principal du choix de Noxim est qu'il est un simulateur open-source. Ce logiciel libre est écrit en System C et peut être exécuté en toute plate-forme UNIX avec un standard compilateur C++, nous utiliserons le compilateur g++ dans le cadre de notre projet. Ce qui nous permis d'apporter des modifications à ce simulateur afin de l'adapter à nos besoins de simulation. En plus,

il est totalement configurable (type et taille de la topologie, taille des tampons, taille des flits, les canaux virtuels, pattern de trafic, générateurs de trafic, etc.), il est capable d'évaluer plusieurs paramètres de performance tels que la latence, le débit, la consommation d'énergie. Noxim fournit aussi en sortie des résultats. Afin de les traduire sous forme de graphes en utilisant des outils comme Gnuplot ou MatLab. La version que nous utilisons dans le cadre de cette thèse a été modifiée par moi-même.

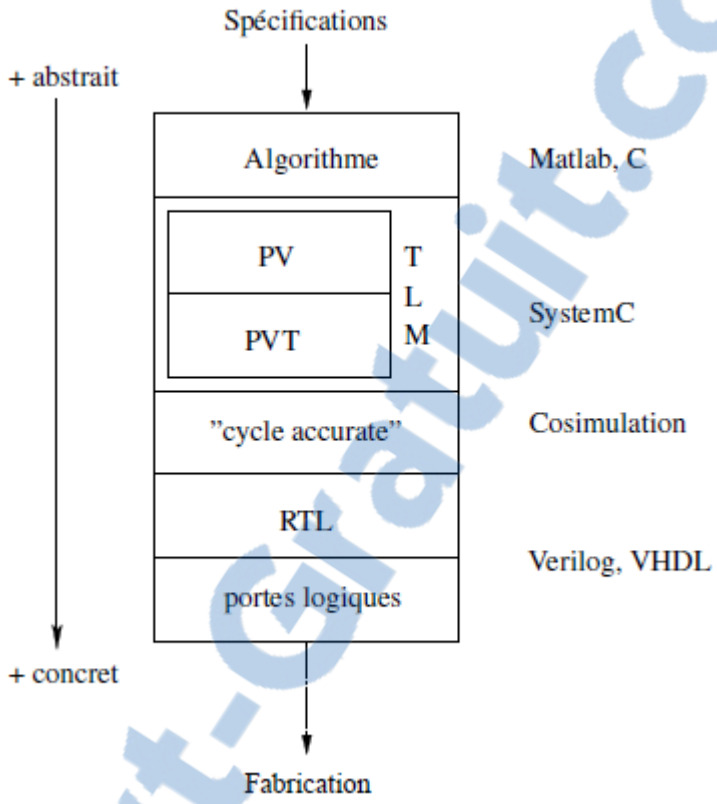
Le coeur du flot de conception des SoCs est le niveau "Register Transfer Level)" (RTL). Les descriptions RTL sont très précises : la valeur de chaque bit d'information est connue à chaque top d'horloge. Cependant, la simulation des descriptions RTL est trop lente pour permettre le développement et la validation du logiciel embarqué. Pour cette raison, les descriptions RTL ne conviennent ni pour le développement du logiciel embarqué, ni pour l'évaluation des choix d'architecture qui doivent être effectués tôt. La solution, actuellement en plein essor, consiste à développer des modèles complémentaires, qui abstraient toutes les informations non primordiales. Ce nouveau niveau d'abstraction est appelé niveau de modélisation transactionnel (TLM Transaction Level Model). Il n'existe actuellement aucun outil capable de construire automatiquement une description RTL à partir d'un modèle transactionnel (les modèles TLM ne contiennent pas assez d'information pour cela).

Aucun des langages prévus pour le matériel (VHDL, Verilog, ...), ne permet en l'état un développement efficace des modèles de SoCs. Une collaboration entre plusieurs industriels, l'OSCI pour Open SystemC Initiative, a mis au point un nouveau langage : SystemC, conçu comme une extension de C++. Celui-ci vise et parvient à rassembler les avantages de la programmation logicielle et de la description matériel. SystemC permet de décrire l'architecture du SoC : les composants matériels sont représentés par des modules qui sont reliés via des ports et des canaux de communication. Le comportement du matériel et du logiciel embarqué est décrit grâce à des processus exécutant du code C++ général. Tous les composants peuvent ainsi se modéliser en SystemC, quelle que soit leur nature, et à tous les niveaux d'abstraction du flot de conception des SoCs. Plusieurs industries ont ensuite développer chacune de son côté des librairies pour les communications au niveau d'abstraction transactionnel[35].

La figure 4.5 donne un aperçu du flot de conception. Le point de départ est toujours des spécifications écrites dans un langage informel. Ensuite, une première implantation très abstraite permet de préciser et fixer les fonctionnalités voulues. Puis, une première architecture est proposée et des modèles temporisés permettent de la corriger et de la raffiner. Progressivement, on se rapproche du comportement détaillé de la puce finale. Chaque niveau d'abstraction a ses technologies et ses outils propres ; les principaux noms sont donnés à droite de la figure.

SystemC[34] est une extension du langage C++ sous forme d'une bibliothèque de classes et de macros qui enrichit le langage source par les structures nécessaires pour modéliser la

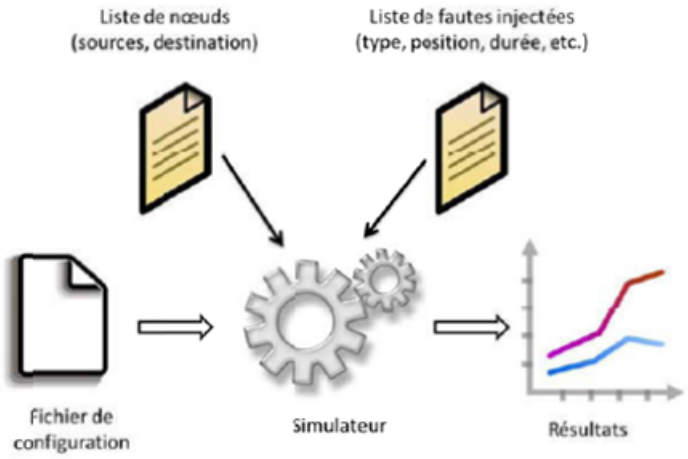
FIGURE 4.5 – Les différents niveaux d'abstraction permettant d'aller des spécifications à une description synthétisable.[35]



concurrence, les communications et le temps. SystemC est né de l'initiative de OSCI10. Le SystemC s'est imposé comme étant un langage de conception de haut niveau incontournable. En effet, les langages C/C++ jouaient déjà un rôle majeur dans le développement des logiciels embarqués et l'adoption d'une solution similaire pour la conception du matériel s'avère un choix judicieux. Cette bibliothèque est bâtie sur le principe de la séparation des aspects de communication et de calcul afin de permettre plus de modularité et de flexibilité au modèle. Ce principe commun montre que le style de représentation de SystemC correspond parfaitement à celui de TLM.La méthodologie TLM est indépendante des languages de conception du matériel (SystemC, SystemVerilog, SpecC, etc.). Ainsi, SystemC n'est pas indispensable pour TLM.Actuellement, la vitesse de simulation d'un modèle SystemC-TLM est 10,000 fois plus rapide que celle au niveau RTL. La simplicité du modèle SystemC ajoutée à la rapidité d'exécution engendrent systématiquement une amélioration de l'efficacité de la simulation et du déboggage.

Nos expérimentations consistent à effectuer des campagnes d'injections de fautes et de comparer notre approche avec les algorithmes de référence et cela pour différents types

FIGURE 4.6 – Flot de simulation



de trafics synthétisés. Les critères de performance pertinents que nous étudierons dans ce chapitre sont la latence moyenne, débit et le ratio de paquets livrés avec succès. (voir figure 4.6).

4.1.3 Génération de trafics synthétisés

Les modèles de trafic les plus largement utilisées pour l'analyse de la latence dans les réseaux d'interconnexion sont : Aléatoire (Uniforme), Brouillé (Shuffle), Transposé de type 1 et 2, Bit- Complément (Bit-Complement), , Bit-Inverse (Bit-Reversal), Point-chaud (Hotspot), auto-similaire (Self-Similar)[8].

Trois différents trafics sont utilisés pour l'évaluation et l'analyse de l'algorithme que nous avons proposé : « Uniform Random », « Shuffle » et « Bit complément ». Pour décrire ces trafics synthétisés, nous définissons si (di) comme le i-ème bit de l'adresse du noeud source (noeud destination) en binaire. La longueur en nombre de bits d'une adresse est a=log2N, ou N est le nombre de noeuds 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.Le modèle de trafic Uniforme est une référence standard utilisée dans les études de routage réseau.
— Shuffle : Chaque noeud envoie seulement au noeud dont l'adresse de l'expéditeur est décalé avec rotation à gauche par un bit, c'est à dire, pour l'adresse source
— Bit complement : chaque noeud 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=0111=7 lorsque s=1000=8)

Il existe d'autres types de trafics synthétisables comme « Hotspot », « Tornade » [57], etc. Nous nous limiterons dans ce qui suit à l'utilisation des trois types de trafic Uniform Random , Shuffle 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.

Pour chaque simulation, nous définissons un certain nombre de noeuds sources qui injectent les paquets et un certain nombre de noeuds destination qui consomment / éjectent les paquets reçus. Le nombre de noeuds 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 noeuds source / noeuds 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.

4.1.4 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).

4.1.5 Injection de fautes

L'injection de fautes est une technique de validation pour les systèmes tolérants aux fautes qui consiste en la réalisation d'expériences contrôlées où les observations du comportement du système en présence de fautes sont induites explicitement par l'introduction volontaire de fautes dans le système[17].L'objectif est de comparer le comportement nominal du circuit (sans injection de faute), avec son comportement en présence de fautes injectées pendant l'exécution d'une application.

Ces techniques d'injection de fautes sont devenues trés populaires pour évaluer et améliorer la fiabilité des systèmes embarqués. Elles peuvent être réalisées soit au niveau physique, ou simulées. Dans la première les fautes sont directement injectées dans le matériel et perturbent l'environnement (comme le bombardement d'ions lourds, les interférences électromagnétiques, le laser, etc.). En revanche, dans la seconde les campagnes d'injection de fautes peuvent être effectuées en utilisant plusieurs méthodes, en particulier les approches de simulation haut niveau. Cette méthode a été largement utilisée pour sa simplicité. L'avantage dans cette méthode est que la simulation peut être plus coûteuse en temps, cependant elle peut permettre une analyse plus complète et fournir des résultats plus précis et revenir moins cher que l'injection de fautes physique. Pour concevoir un système tolérant aux fautes, il est important que le système soit conscient des failles susceptibles d'apparaître après sa mise en service.

Pour injecter des fautes en générale en a besoin d'au moins trois d'informations :

1. Quand : quand l'injection de fautes sera-t-elle faite pendant la simulation ?
2. Où la faute sera-t-elle injectée ? Dans quelle partie du circuit ?
3. Finalement quel sera le type de faute injectée ?

quand Lorsque l'on étudie le comportement d'un circuit en présence de fautes, on peut vouloir faire l'injection de fautes de manière aléatoire . L'injection de faute lors d'une simulation, après un certain laps de temps. Dans ce cas, nous ne savons pas dans quelle partie exacte du système la faute est injectée.

Localisation de faute (Où) Dans notre travail, nous avons affaire à l'injection de fautes dans un NoC tolérant aux fautes. Une faute se produisant dans un lien ou un routeur. La localisation d'une faute est souvent décrite de façon aléatoire,càd choisir au hasard dans quel partie du système.

Tous ces informations de base sont nécessaires pour le simulateur pour l'injection de fautes.

Dans cette thèse nous nous intéressons aux fautes de liens et aux fautes de nœuds (routeurs). Ces composants peuvent donc être le siège de fautes ayant des effets permanents

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 deux types de fautes en même temps. Le ratio de fautes différentes injectées en même temps est celui communément admis[17], à savoir 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 : Dans nos simulations, nous allons mesurer :

1. **La latence** : 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 noeud destination. La latence mesurée correspond à la latence moyenne des paquets délivrés avec succès parmi les 5000 paquets injectés.
2. **Fiabilité :** 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 Evaluation

Dans l'industrie des semi-conducteurs, les frais de test augmentent le coût de la conception de circuits intégrés et de leur fabrication. Généralement, les essais industriels sont destinés à trouver les **fautes permanentes** qui peuvent être produites au moment de la fabrication. Cependant, les fautes les plus fréquemment rencontrées dans les systèmes informatiques sont des effets temporaires comme les fautes transitoires ou intermittentes. Ils sont la principale cause de défaillance des systèmes numériques[6]. En raison de l'augmentation de la probabilité des fautes transitoires dans les toutes dernières technologies, les concepteurs doivent de plus en plus analyser l'impact potentiel de ces fautes sur le comportement des circuits.

Le modèle d'erreur utilisé pour évaluer les fautes dépend de leur durée. Les fautes permanentes peuvent être tolérées en remplaçant le composant défectueux alors qu'une faute transitoire peut s'auto-réparer. Les fautes intermittentes sont traitées comme des fautes permanentes ou transitoires en fonction de leur fréquence. Certaines techniques courantes d'évaluation des systèmes tolérants aux fautes sont traitées dans[60]. Entre autres techniques, l'injection de fautes est largement utilisée comme une approche efficace pour évaluer la tolérance aux fautes[53].

Nous consacrons cette section à l'évaluation et à l'analyse des performances du système proposé. Nous avons sélectionné trois modèles de trafic : Uniform, Shuffle et Bit Complément pour évaluer les performances de l'algorithme proposé. La première série d'analyses étudiées est la latence et la fiabilité pour chacun des modèles de trafic susmentionnés. Nous avons observé la variation de performance de DINRA sous différents taux de défaillance des liaisons et des routeurs (0 %, 5 %, 10 %, 20 % et 40 %). Au cours de l'évaluation, nous avons divisé les pannes en deux parties : les parties les plus importantes sont attribuées aux pannes transitoires et la partie restante est considérée comme permanente. Pour évaluer notre algorithme, nous utilisons un simulateur disponible au public : Noxim.

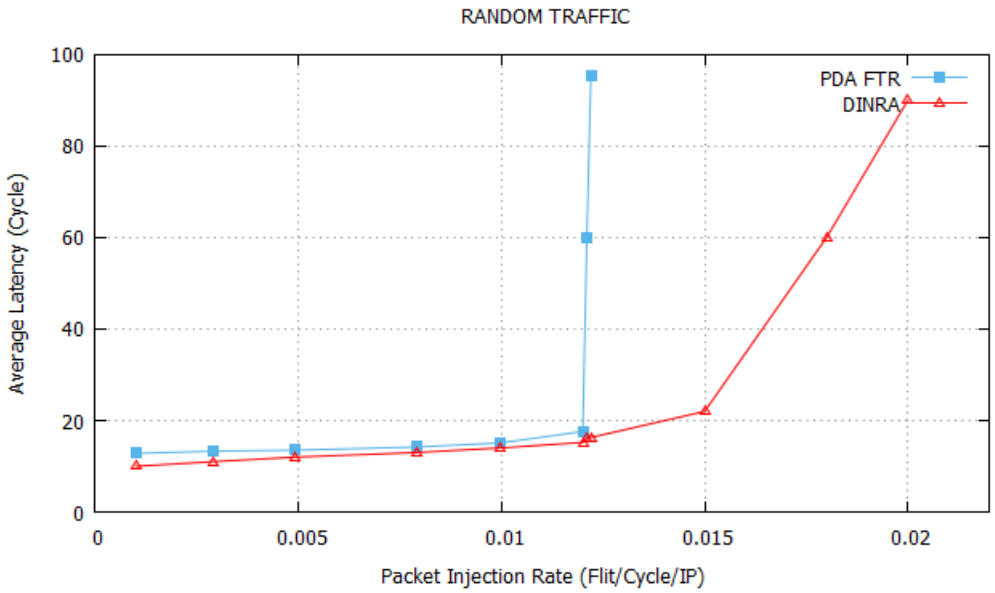


FIGURE 4.7 – La Latence moyen de DINRA et PDA-FTR dans un NoC 8 * 8 avec trafic aléatoire, avec quatre routeurs défaillants.

Dans toutes nos expériences, les simulations sont effectuées sur 100 000 cycles à différents taux d'injection et 10000 cycles sont destinés à la mise en température(warm up). À la fin des 100 000 cycles, les statistiques sont collectées. A la fin de la simulation, nous nous assurons qu'aucun paquet n'est dans le réseau. Ainsi, lors de ce drain, aucun nouveau paquet n'est injecté et le réseau ne contient aucun paquet, ce qui nous aide à confirmer l'absence de blocages. Une centaine d'itérations sont exécutées pour chaque simulation et les résultats moyens sont présentés. Pour injecter des défauts, le nombre sélectionné de liens et de routeurs est désactivé de manière permanente au début de la simulation et est sélectionné de manière aléatoire. La durée d'une itération de simulation varie entre 30 secondes et 3 minutes en fonction du taux de trafic et des défauts injectés.

4.2.1 Analyse des performances

Afin de connaître les performances en termes de latence associée à cette nouvelle architecture, nous avons mesuré la latence moyenne du réseau en fonction des différents taux d'injections et avec des tailles différentes. Aussi, On a adopté trois types de trafic pour la simulation des performances.

4.2.1.1 Evaluation de la Latence

Dans cette partie pour évaluer notre approche, nous avons varié le taux d'injection de défauts et nous avons mesuré le temps de latence moyen. Premièrement, nous avons évalué le latence de communication de DINRA en calculant la latence moyenne. La figure 4.7 (resp. Fig.4.8 et 4.9) illustre les résultats de latence / flit sous un modèle de trafic aléatoire (resp. Shuffle et Bit Complement). Lorsqu'aucun défaut n'est détecté (0 %), ces tests ont révélé que nos solutions réduisent la latence avec une moyenne de 19,4 % par rapport à l'algorithme de routage PDA-FTR. Comme nous l'avons indiqué précédemment, DINRA tire parti du routage adaptatif pour transmettre les flits au prochain nœud voisin, ce qui est plus rapide que l'algorithme de routage PDA-FTR. De plus, DINRA prend en compte l'état de congestion et l'équilibre du trafic (voir ligne 8 algorithm1). Comme prévu, pour les valeurs de taux de défaut élevées, DINRA fonctionne mieux que PDA-FTR. En conséquence, la latence / flit le long du réseau est réduite. Lorsque vous observez la variation de latence sur différents taux de défaillance, DINRA fonctionne mieux que PDA-FTR même avec un taux de défaillance inférieur à 10 %. Lorsque ce taux atteint 25% et 40%, le temps de latence augmente de 12% seulement et 30 % respectivement.

Cette amélioration des performances est obtenue grâce à l'emploi de routages locaux et prospectifs qui donnent à notre système l'occasion de recalculer le routage s'il détecte que celui déjà calculé conduira à un chemin de blocage coûtant plusieurs cycles d'horloge pour un routage non minimal.

Comme prévu, les défauts injectés affectent la latence moyenne du réseau. Celui-là tend à augmenter avec l'augmentation de la charge du réseau et également avec l'augmentation du nombre de défauts de routeur / liaison. On peut voir sur les Fig.4.9 ou Fig.4.10 que la latence moyenne de DINRA augmente si nous injectons plus de trafic et / ou plus de défauts.

Cette performance est fortement liée à la nouvelle architecture selon laquelle le choix du chemin est plus diversifié. De cette manière, lorsqu'un lien défectueux est détecté dans un chemin, il en existe toujours d'autres avec des liens valides dans le même sous-réseau, sinon dans d'autres. DINRA profite de cette propriété pour router les flits. Ainsi, la congestion est plus importante, car plus de flits seront bloqués dans les tampons. Cela explique pourquoi

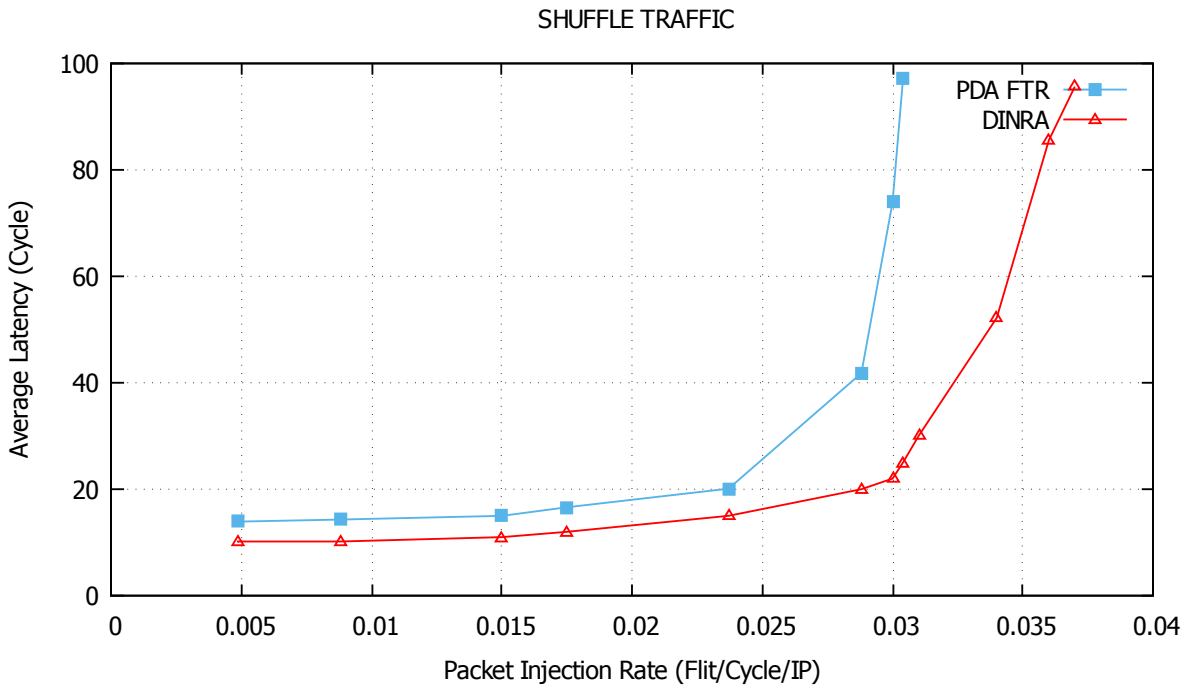


FIGURE 4.8 – Temps moyen de latence de DINRA et PDA-FTR dans un NoC 8 * 8 sous trafic Shuffle, avec quatre routeurs défaillants.

les performances chutent à des taux de défaillance plus élevés. Ceci est principalement causé par la congestion du trafic. Pour mieux voir les effets du routage DINRA, nous observons les résultats de latence du modèle de trafic aléatoire illustré à la Fig.4.11. Dans le cas où aucun défaut n'est détecté. observé, DINRA réduit encore la latence de 16,29 % par rapport à PDA-FTR. La latence atteint sa valeur la plus élevée(limites) avec un taux de défaillance de 45% avec un taux d'injection plus élevé (PIR) (100 000 flits).

Outre l'encombrement dû au nombre croissant de liaisons et de routeurs défaillants expliqués ci-dessus, nous avons également un impact important sur la latence.

$$\text{Success ratio} = \frac{\text{Total. arrived packets}}{\text{Total injected packets}} \times 100 \tag{4.1}$$

Nous avons évalué la latence dans un réseau défectueux en augmentant le taux d'injection de paquets. Les résultats montrent que la latence est la performances la plus sensible et un très bon indicateur de la qualité de service du réseau.

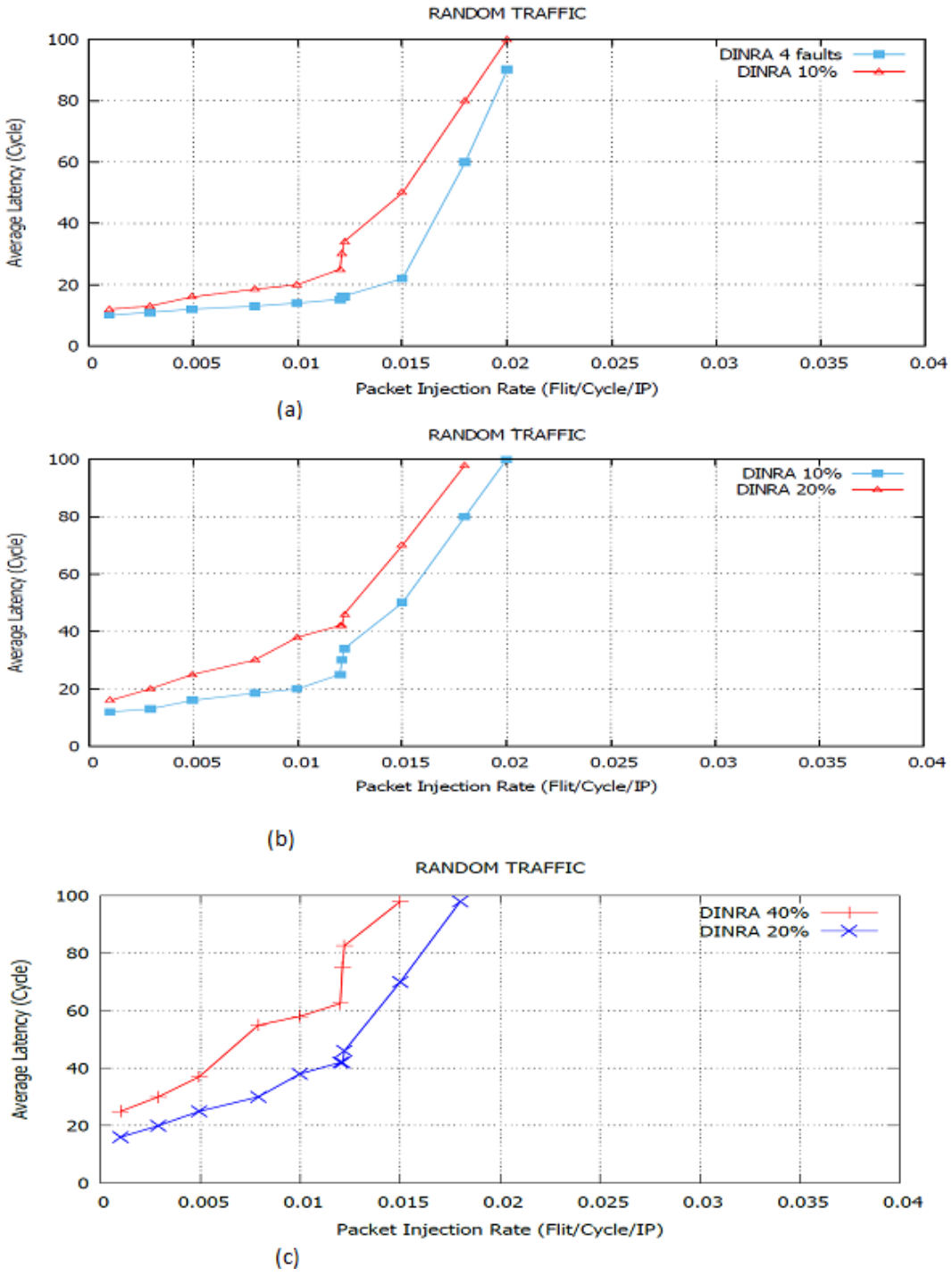


FIGURE 4.9 – Latence moyenne de DINRA avec différents taux d'injection de fautes (faute de routeurs/liens) sous le trafic « Uniform Random », (a) 10% ,(b) 20% et c 40% pour un NoC 8x8

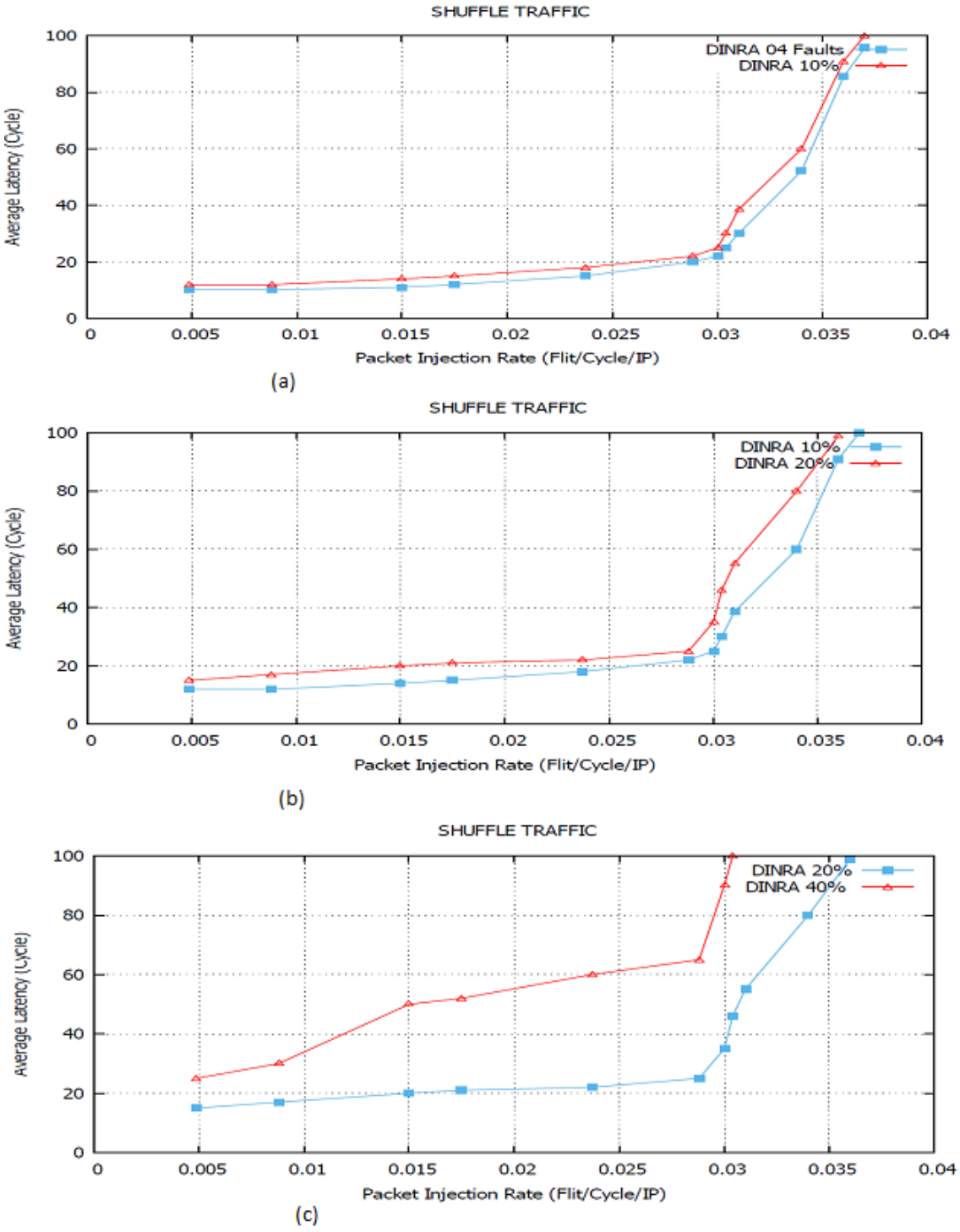


FIGURE 4.10 – Latence moyenne de DINRA avec différents taux d'injection de fautes (faute de routeurs/liens) sous le trafic « SHUFFLE », (a) 10% ,(b) 20% et c 40% pour un NoC 8x8

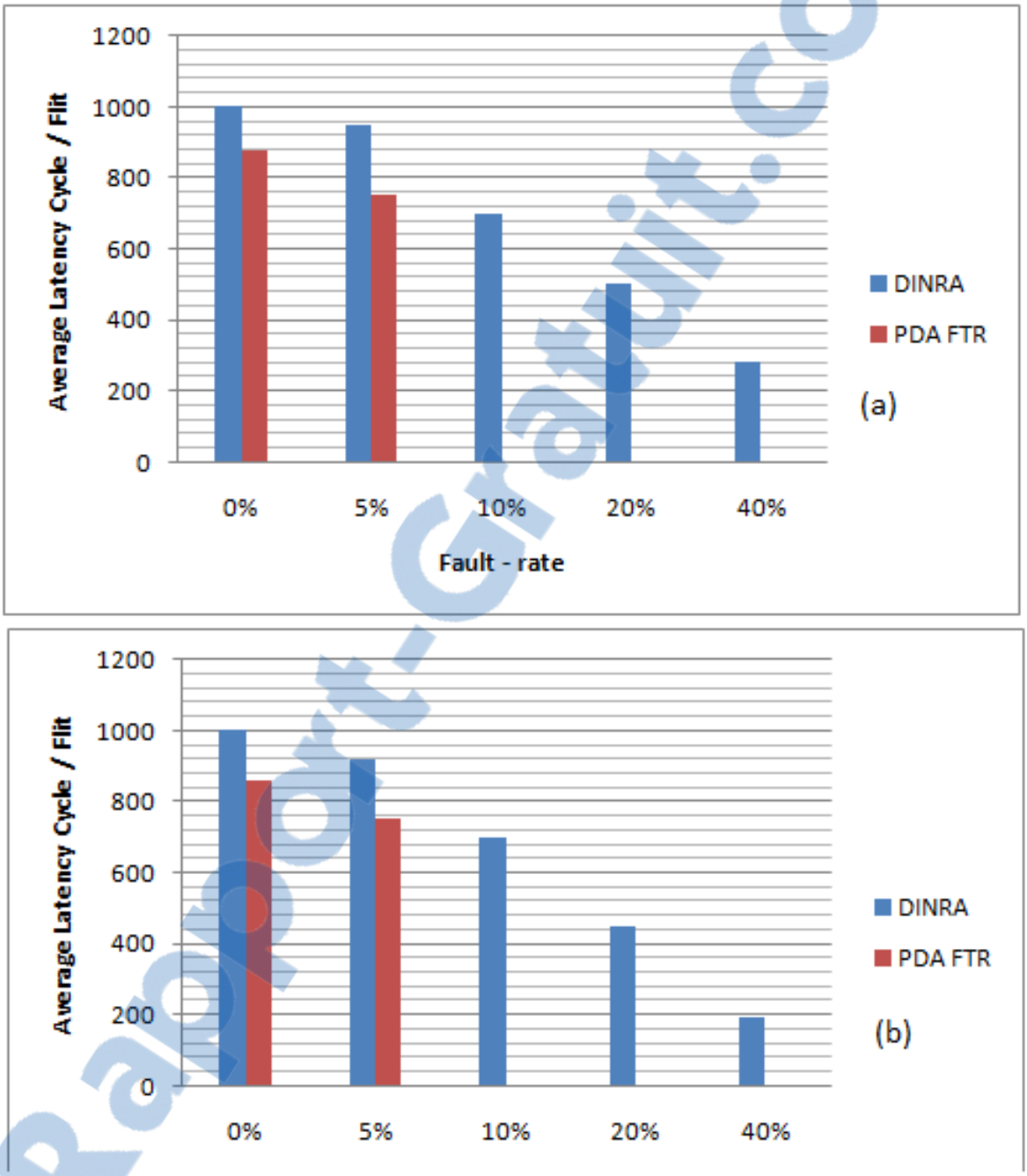


FIGURE 4.11 – Évaluation de la latence moyenne sous différent taux de fautes (Routeur) avec : (a) Uniform b) Shuffle dans un NoC 8x8 .

TABLE 4.1 – Comparison of Success ratio % with PDA-FAR Routing Algorithm, faulty routers.

No. of Faults	Gradient	PDA-FTR	DINRA
2	2.8%	0.7%	1.00%
4	3.2%	0.5%	1.00%
20%	-	-	0.5%
35%	-	-	0.15%

4.2.1.2 Evaluation de la Fiabilité

Nous avons voulu ensuite déterminer l'impact du taux de défaillance sur le taux de réussite de livraison de paquets. Pour cela nous avons mesuré le nombre de paquets n'ayant pas atteint leur destination pour un NoC 8*8 et pour différentes situations de défaillance. 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.

Dans cette sous-section, nous discuterons de la fiabilité de DINRA. Nous rappelons que la fiabilité est la capacité d'un système à livrer tous les paquets à leurs destinations. Pour la deuxième évaluation, nous avons calculé les taux de réussite de chaque algorithme en utilisant les deux modèles de trafic (équation 1). Nous pouvons expliquer cette fiabilité par le fait que DINRA trouve toujours le chemin entre une source et une destination, peu importe l'emplacement de la panne et l'importance du trafic. Le seul problème possible pouvant affecter l'arrivée d'un paquet donné à sa destination est la présence d'un blocage. Cependant, grâce au modèle de virage hybride adopté, le blocage est évité. Par conséquent, le résultat le plus frappant qui ressort des données est que tous les paquets peuvent atteindre leurs destinations, ce qui offre une grande fiabilité pour différents taux de défaillance et injections.

On voit que DINRA permet de garantir un niveau de fiabilité élevé. Dans le cas d'un NoC 12x12, DINRA permet de délivrer plus de 95,8% des messages lorsque 10% des liens sont défectueux. Avec 40% des liens défectueux, le taux de livraison avec succès varie selon le type de trafic de 50,5 à 60%.

Nous avons fait la même série d'expériences mais en prenant cette fois le modèle des défauts de routeur (Fig. 13). Là encore, nous avons constaté une augmentation du temps de latence moyen en fonction du nombre de défauts et du taux d'injection de paquets. Il convient de noter que la latence moyenne augmente ici plus rapidement que dans le cas précédent ou que seules les liaisons étaient considérées comme défectueuses (Fig.12). Une faute de routeur est un routeur avec toutes ses liaisons défectueuses. Pour l'algorithme

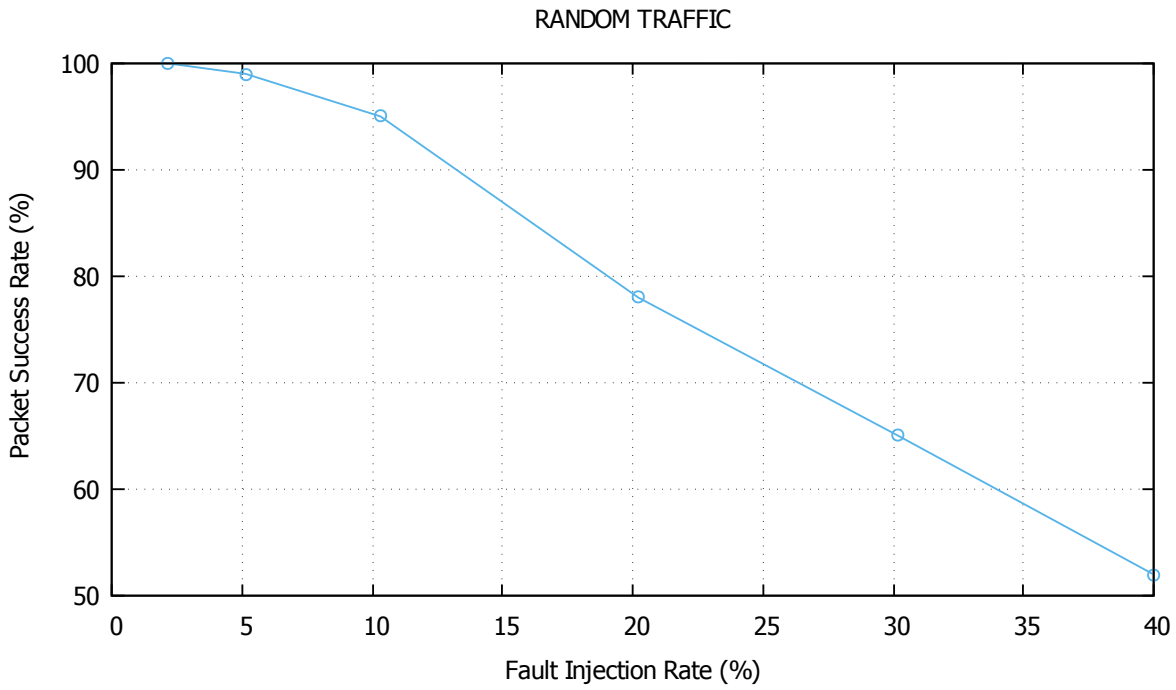


FIGURE 4.12 – Taux de paquets livrés avec succès avec différents rapports d'injection de défaut (défaut de liaison) sous trafic "Uniform" pour un NoC 12x12.

DINRA, plus les routeurs sont défectueux, plus il est difficile pour un paquet de trouver un chemin vers sa destination. Néanmoins, avec plus de 30 % des routeurs défaillants, le service est toujours assuré mais avec un taux d'injection de paquets plus faible en fonction du type de trafic, par rapport à une situation sans défaillance (voir Tableau 4.1), "Nombre de défaillances "représente le nombre de composants défectueux sur le réseau.

Nous constatons que DINRA permet de garantir un haut niveau de fiabilité. Dans le cas d'une taille NoC 12x12, il peut transmettre plus de 95 % des messages lorsque 10 % des liens sont défectueux. Quand nous avons 40% des liens défectueux, le taux de livraison varie avec succès de 50% à 60% pour le "trafic aléatoire". Lors de la prise en compte des défaillances de nœud (routeur), le débit de paquets remis avec succès diminue, comme on pouvait s'y attendre étant donné l'impact plus important d'une défaillance du routeur. Néanmoins, même dans le scénario où 40% des routeurs sont défectueux, le taux de livraison varie de 0 à 10% pour un trafic "Aléatoire uniforme" (Fig.12 et 13).

Avec le rajout des techniques de transfert de paquets et de gestion de la congestion, DINRA améliore nettement les performances et la fiabilité par rapport aux PDA FTR. L'injection de défaut affecte la latence et le débit moyens du réseau, ce qui augmente avec le nombre de défauts de liaison, de routeurs et le nombre de paquets injectés dans le réseau.

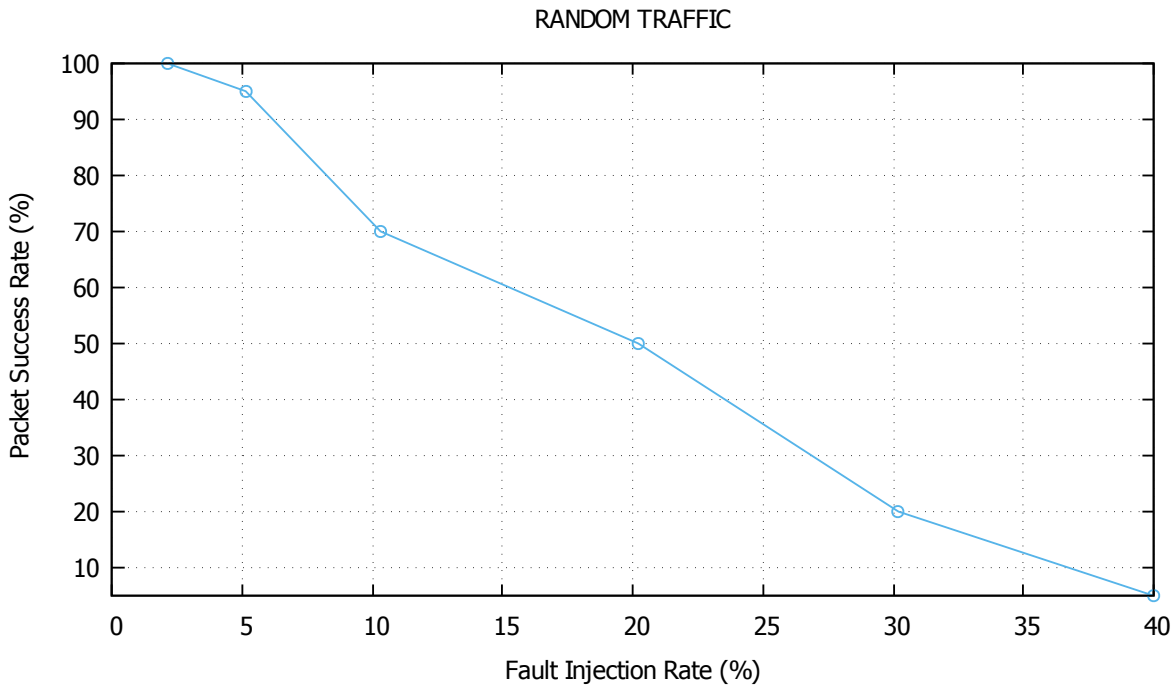


FIGURE 4.13 – Taux de paquets livrés avec succès avec différents rapports d'injection de défaut (défaut de routeurs) sous trafic "Uniform" pour un NoC 12x12.

4.2.1.3 Débit

Nous avons évalué le débit, qui est défini comme le trafic accepté du serveur réseau à une latence donnée. Le débit du réseau est présenté par la formule ci-dessous :

$$Network\ Throughput = Saturation\ Throughput \times No.of nodes \tag{4.2}$$

Nous voulions ensuite déterminer l'impact du débit et la taille du réseau. Pour cela, nous avons mesuré le débit pour quatre tailles de NoC et pour une seule défaillance. (Fig.14) résume les simulations effectuées pour différentes tailles. Nous pouvons constater, comme prévu, que nos expériences confirment que DINRA est adapté aux systèmes NoC de petite et grande taille.

4.3 Conclusion

Dans cet article, nous avons présenté un algorithme de routage innovant à tolérance de pannes, appelé DINRA, pour les réseaux sur puce (NoC). L'algorithme fonctionne avec

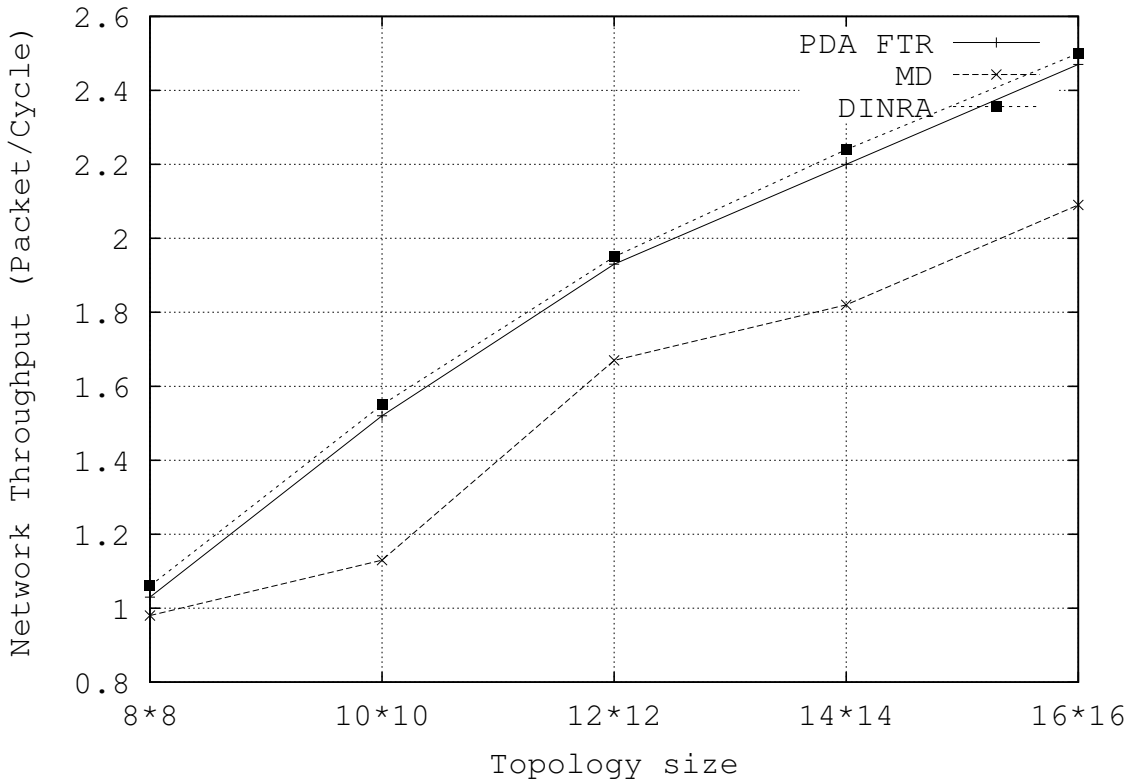


FIGURE 4.14 – Débit de DINRA sous différentes tailles et dans le cas d'un seul défaut. (routeur).

une architecture proposée qui vise à éviter une défaillance grave du système lorsque certains composants défectueux (routeurs et liens) sont observés dans un NoC. La solution proposée préserve les performances du réseau. L'évaluation effectuée prouve que DINRA fonctionne mieux que l'algorithme PDA-FTR, même à un taux de défaillance de 40% dans les liaisons et les routeurs. Malgré les bons résultats obtenus, notre travail présente clairement certaines limites qui devraient être corrigées pour améliorer ses performances et sa fiabilité. La première est que nous devons analyser les meilleures techniques pouvant être mises en œuvre pour la détection des pannes. Deuxièmement, il est également intéressant d'étudier si le temps requis pour gérer une détection de panne peut être amélioré. Nous étudions actuellement la consommation d'énergie et la complexité matérielle de l'architecture proposée. Dans les travaux futurs, nous prévoyons de nous concentrer sur ces paramètres.

Conclusion et perspectives

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

Les réseaux sur puce sont actuellement les architectures de communications les plus adaptées pour les systèmes embarqués multi-coeurs. Ces réseaux doivent être capables de supporter différents flux de données en temps réel. Leurs performances dépendent directement de la stratégie de routage adoptée. Dans cette thèse, nous avons présenté un nouvel algorithme et architecture tolérante aux fautes appelée DINRA. L'architecture proposée parvient à éviter la défaillance du système au présence d'un grand nombre de défauts tout en garantissant une dégradation progressive des performances. Pour résoudre le problème de défaillance de liaison et routeur, Le NoC doit ainsi être capable de s'affranchir de la présence de panne. Lorsqu'un ou plusieurs nœuds ou liaisons (zone) sont déclarés fautifs, une solution consiste à mettre en œuvre un algorithme adaptatif de routage des paquets de données, tolérant à la fois les fautes transitoires et permanentes toute en réduisant la latence de communication et en améliorant les performances du NoC. L'objectif est le contournement de la zone fautive afin de maintenir la qualité de service du réseau tout en fiabilisant le réseau.

L'architecture hiérarchique proposée tire avantage de Power Gating inspirée de Catnap. Afin de gérer l'occurrence de fautes dans les éléments du routeur un mécanisme intelligent appelé Test Module TM a été introduit dans chaque noeud. Le TM est également utilisé comme solution légère pour identifier les fautes statique et dynamiques. Si un routeur a ses indications d'état fautif activées à ses routeurs voisins, sa position est contournée selon des règles de routage à élaborer. Nous avons approuvé le mécanisme TM avec une unité de stockage de fautes- Fault Register(FR) afin d'éviter les composants défectueux et réduire davantage la latence causée par la présence de pannes. Nous avons montré que notre algorithme et sans inter-blocage (Deadlock-Free) et aux situations de bouclages (livelock) induit par la combinaison de deux algorithme de routage adaptatif « North Last et South Last ».

D'après l'évaluation des performances, le système proposé fonctionne toujours mieux que le système proposé par PDA FTR, même avec un taux de défaillance de 40% des routeurs et liaisons dans le cas de trafic synthétique. De plus, l'algorithme présente un autre avantage : il réduit l'encombrement qui est considéré comme un défaut temporaire. Les simulations montrent que notre algorithme proposé réduit la latence de plus de 15% et que le débit est amélioré de 20% par rapport au routage PDA-FTR. Un autre avantage de cette thèse est que nous avons inclus dans cet algorithme un mécanisme de détection de congestion local et régional. 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 plus résilient.

Comme perspectives, ce travail a pour vocation à être complété, en effet en nombreux points.

Malgré les bons résultats obtenus avec l'architecture de sub-net proposée, certains points devraient être optimisés pour améliorer ses performances et sa fiabilité. Le premier est qu'il manque le mécanisme de détection des pannes et on suppose simplement la présence d'un tel module. Par conséquent, une étude plus complète devrait être menée pour analyser les meilleures techniques pouvant être mises en œuvre. La solution peut être basée sur les modules de test ou sur les codes de détection d'erreur qui peuvent être intégrés dans le Flit.

Dans cette thèse, nous n'avons pas étudié le problème de la puissance thermique ni leurs effets sur le système proposé. Sachant que c'est aussi un paramètre important. L'importance vient du fait que la puissance thermique est l'une des principales raisons de l'augmentation des taux de défaillances, en particulier des défaillances permanentes, et de la diminution du temps moyen jusqu'à la défaillance, qui représente la durée de vie d'un système donné. La mise en œuvre de t'elle techniques sensibles à la puissance thermique visent à améliorer la fiabilité du NoC et à préserver les bonnes performances obtenues.

Les travaux d'estimation 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 énergétique induits par l'ajout de nos fonctionnalités de tolérance aux fautes et de gestion de congestion.

Il est envisagé d'inclure ce NoC sur carte FPGA comme *Nios II embedded Kit d'Altera* assurant une sûreté de fonctionnement à un système de communication sur puce.

Enfin, les simulations ont été réalisées pour trois types de trafic. Pour vérifier dans un contexte plus réel la validité de notre algorithme et l'architecture proposée, il serait intéressant de réaliser des simulations avec des trafics issus d'applications réelles, de type « benchmark ».

Bibliographie

[1] A. Greiner L. Mortiez A. Adriahantenaina H. Charlery et C. A. Zeferino. « Design of cost-efficient interconnect processing units : Spidergon STNoC. » In : *CRC press* (2008) (cf. p. 42).

[2] A. Greiner L. Mortiez A. Adriahantenaina H. Charlery et C. A. Zeferino. « Spin : a scalable, packet switched, on-chip micro-network ». In : *Design, Automation and Test in Europe Conference and Exhibition* (2003), p. 70–73 (cf. p. 42).

[3] A. Greiner et al. A. Andriahantenaina H. Charlery. « SPIN : a scalable, packet switched, on-chip micro–network ». In : *Design automation and test in Europe conference and exhibition (DATE)* (2003), p. 70–73 (cf. p. 26).

[4] L.-S. Peh A. DeOrio et V. Bertacco. « ARIADNE :Agnostic reconfiguration in a disconnected network environment ». In : *International Conference on Parallel Architectures and Compilation Techniques (PACT),* (2011), p. 298–309 (cf. p. 69, 70, 74).

[5] N. Calazans A. Mello L. Tedesco et F. Moraes. « Virtual Channels in Networks on Chip : Implementation and Evaluation on Hermes NoC ». In : *in Proc. of the Symposium on Integrated Circuits and Systems Design* (2005), p. 178–183 (cf. p. 21, 26).

[6] M. Fazeli A. Vahdatpour et S. Miremadi. « Transient error detection in embedded systems using reconfigurable components. » In : *International Symposium on Industrial Embedded Systems, 2006 (IES'06)*. 2006, p. 1–6 (cf. p. 111).

[7] N.E.Zergainoh A.Charif et M.Nicolaidis. « MUGEN : A high-performance fault-tolerant routing algorithm for unreliable Networks-on-Chip ». In : *IOLTS* (2015), p. 71–76 (cf. p. 71, 74).

[8] Akram Ben Ahmed. « High-throughput Architecture and Routing Algorithms Towards the Design of Reliable Mesh-based Many-Core Network-on-Chip Systems ». Thèse de doct. 2015 (cf. p. 51, 52, 108).

[9] C. Feng et al. « Addressing Transient and Permanent Faults in NoC With Efficient Fault-Tolerant Deflection Router ». In : *IEEE TVLSI,* (2011) (cf. p. 69, 70).

[10] D. Fick et al. « A highly resilient routing algorithm for fault-tolerant NoCs ». In : *DATE,* (2009) (cf. p. 66, 69, 74).

[11] D. Lee et al. « Brisk and Limited-Impact NoC Routing Reconfiguration ». In : *DATE,* (2014) (cf. p. 69, 70).

[12] M. Balboni et al. « Synergistic Use of Multiple On-Chip Networks for Ultra-Low Latency and Scalable Distributed Routing Reconfiguration ». In : *DATE,* (2015) (cf. p. 68–70).

[13] P. Ren et al. « Fault-tolerant routing for on-chip network without using virtual channels ». In : *DATE,* (2014) (cf. p. 68).

[14] R. Bishnoi et al. « Distance-Driven Routing to Handle Permanent Failures in 2D Mesh NoCs ». In : *DATE,* (2015) (cf. p. 68, 70).

[15] H. Li B. Fu Y. Han et X. Li. « Zonedefense : A fault-tolerant routing for 2- d meshes without virtual channels ». In : *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 22.1 (2014), p. 113–126 (cf. p. 61, 67, 74).

[16] L. Benini et G. De Michel. « Networks on chips : a new soc paradigm ». In : *Computer* (2002) (cf. p. 16, 17, 20).

[17] A. Benso et P. Prinetto. « Fault Injection Techniques and Tools for Embedded Systems Reliability Evaluation ». In : *1st ed. Boston : Springer- Verlag* (2003) (cf. p. 110, 111).

[18] D. Bertozzi et L. Benini. « Xpipes : A network-on-chip architecture for gigascale systems-on-chip ». In : *IEEE circuits and systems magazine* 4.2 (2004), p. 18–31 (cf. p. 42).

[19] T. Bjerregaard. « The MANGO clockless network-on-chip : Concepts and implementation ». Thèse de doct. 2005 (cf. p. 42).

[20] Chao C. H. Lin S. Y. Chang E. J. Hsin H. K. et Wu A. « Regional ACO-based cascaded adaptive routing for traffic balancing in mesh-based network-on-chip systems ». In : *IEEE Transactions on Computers* 64.3 (2015), p. 868–875 (cf. p. 71).

[21] Lin S. Y. Chang E. J. Hsin H. K. et Wu A. Y. « Path-congestion-aware adaptive routing with a contention prediction scheme for network-on-chip systems ». In : *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 33.1 (2014), p. 113–126 (cf. p. 71).

[22] C. Chen et S.D. Cotofana. « An effective routing algorithm to avoid unnecessary link abandon in 2D mesh NoCs ». In : *IEEE* (2013), p. 311–318 (cf. p. 74).

[23] P. D Vigna D. Garcia R. Jardine J. Klecka D. Bernick B. Bruckert et J. Smullen. « NonStop advanced architecture ». In : *Dependable Systems and Networks, 2005 (DSN'05)* (2005), p. 12–21 (cf. p. 86).

[24] E. Friedman D. Rostislav V. Vishnyakov et R. Ginosar. « An asynchronous router for multiple service levels networks on chip ». In : *11th IEEE International Symposium on Asynchronous Circuits and Systems* (2005), p. 44–53 (cf. p. 42).

[25] R. Dafali. « Design of dynamic reconfigurable Network-on-Chip ». Thèse de doct. 2011 (cf. p. 42).

[26] W. J. Dally et B. Towles. *"Principles and Practices of Interconnection Networks.* Sous la dir. de Morgan Kaufmann Edition. 2004 (cf. p. 22, 38, 41, 54).

[27] A. Kulmala E. Salminen et T. Hamalainen. « On network-on-chip comparison ». In : *Euromicro DSD* (2012), p. 503–510 (cf. p. 25).

[28] A. Amory E. Wachter A. Erichsen et F. Moraes. « Topology-agnostic fault-tolerant noc routing method ». In : *Proceedings of the Conference on Design, Automation and Test in Europe.* (2013), p. 1595–1600 (cf. p. 66, 69, 70, 74).

[29] Liljeberg P Plosila J Ebrahimi M Daneshtalab M et Tenhunen H. « LEAR : a low-weight and highly adaptive routing method for distributing congestions in on-chip networks ». In : *In : 20th Euromicro international conference on parallel, distributed and network-based processing (PDP '12)* (2012), p. 520–524 (cf. p. 64).

[30] N.-E. Zergainoh F. Chaix D. Avresky et M. Nicolaidis. « fault-tolerant deadlock-free adaptive routing for on chip interconnects ». In : *IEEE/ACM Design Automation Test in Europe* (2011), p. 909–912 (cf. p. 66).

[31] Pande PP Ganguly A et Belzer B. « Crosstalk-aware channel coding schemes for energy efficient and reliable NOC interconnects ». In : *IEEE Trans Very Large Scale Integr VLSI Syst* 17.11 (2009), p. 1625–1639 (cf. p. 62).

[32] C. Glass et L. Ni. « The turn model for adaptive routing ». In : *f the 19th annual international symposium on Computer architecture (ISCA 92), New York, NY, USA.* 1992, p. 278–287 (cf. p. 38, 96, 97).

[33] C. A. Lin H. K. Hsin E. J. Chang et A. Y. .Wu. « Ant colony optimizationbased fault-aware routing in mesh-based network-on-chip systems ». In : *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 33.11 (2014), p. 1693–1705 (cf. p. 67, 71, 74).

[34] S. Heath. *Embedded Systems Designs.* Sous la dir. d'Editeur du livre. Elsevier Science Publisher second edition, 2003 (cf. p. 25).

[35] Claude Helmstetter. « Validation de modèles de systèmes sur puce en présence d'ordonnancements indéterministes et de temps imprécis ». Thèse de doct. 2009 (cf. p. 106, 107).

[36] S. YALAMANCHILI J. DUATO et L. NI. *Interconnection Networks : An Engineering Approach.* Revised Printing. Murgan Kaufmann, 2003 (cf. p. 32).

[37] Laprie J.C et R. Brian. « Origins and integration of the concepts ». In : (2007) (cf. p. 46, 47).

[38] N. Enright Jerger et Z. Wang S. Ma. « DBAR : An Efficient Routing Algorithm to Support Multiple Concurrent Applications in Network-on-Chip ». In : *of Computer Architecture (ISCA) 38th Annual International Symposium on.* 2011, p. 413–424 (cf. p. 58).

[39] J. Dielissen K. Goossens et A. Radulescu. « Æthereal network on chip : concepts, architectures, and implementations ». In : *IEEE Design Test of Computers* 22.5 (2005), p. 414–421 (cf. p. 42).

[40] M. Ebrahimi M. Daneshtalab X. Zhang G. Li L. Huang J. Wang et A. Jantsch. « Non-blocking testing for network-on-chip ». In : *IEEE Transactions on Computers* 65.3 (2016), p. 679–692 (cf. p. 68).

[41] A. Leroy. « Optimizing the on-chip communication architecture of low power systems-onchip in deep sub-micron technology ». Thèse de doct. 2006 (cf. p. 20, 24).

[42] B. Lin et R.S. Ramanujam. « Destination-Based Adaptive Routing on 2D Mesh Networks ». In : *of Architectures for Networking and Communications Systems (ANCS)), ACM/IEEE Symposium on*. 2010, p. 1–12 (cf. p. 58).

[43] Li Y Liu J Harkin j et Maguire L. « Fault-tolerant networks-on-chip routing with coarse and fine-grained look-ahead ». In : *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 35.2 (2016), p. 260–273 (cf. p. 71).

[44] Chen T Liu S et Chen Y. « FreeRider : Non-local adaptive network-on-chip routing with packet-carried propagation of congestion information ». In : *IEEE Transactions on Parallel and Distributed Systems* 26.8 (2015), p. 2272–2285 (cf. p. 71).

[45] M. Welzl M. Ali et S. Hellebrand. « A Dynamic Routing Mechanism for Network on Chip ». In : *23rd NORCHIP Conference* (2005) (cf. p. 25).

[46] L. Anghel M. Benabdenbi N. Zergainoh M. Dimopoulos Y. Gang et M. Nicolaidis. « Fault-tolerant adaptive routing under an unconstrained set of node and link failures for manycore systems-on-chip ». In : *Microprocessors Microsystems* 38.6 (2014), p. 620–635 (cf. p. 70, 74).

[47] J. Plosila M. Ebrahimi M. Daneshtalab et H. Tenhunen. « Mafa : adaptive fault-tolerant routing algorithm for networks-on-chip ». In : *In Digital System Design (DSD), 2012 15th Euromicro Conference on IEEE* (2014), p. 201–207 (cf. p. 70).

[48] M. Daneshtalab M. Ebrahimi et J. Plosila. « High performance fault-tolerant routing algorithm for NoC-based many-core systems ». In : *IEEE*. 2013, p. 462–469 (cf. p. 69, 94).

[49] M. Daneshtalab M. Ebrahimi et J. Plosila. « High performance fault-tolerant routing algorithm for NoC-based many-core systems, » in : *IEEE* (2013), p. 462–469 (cf. p. 70, 74).

[50] M. Daneshtalab M. Ebrahimi et J. Plosila. « Minimal-path faulttolerant approach using connection-retaining structure in networks-on-chip, » in : *Networks on Chip (NoCS), Seventh IEEE/ACM International Symposium on, IEEE* (2013), p. 1–8 (cf. p. 70, 74).

[51] P. Liljeberg M. Ebrahimi M. Daneshtalab et H. Tenhunen. « HAMUM - A NovelRouting Protocol for Unicast and Multicast Traffic in MPSoCs ». In : *of PDP* (2010), p. 525–532 (cf. p. 37, 52, 63).

[52] Q.A. Zeng M. Li et W.B. Jone. « DyXY : a proximity congestion aware deadlock-free dynamic routing method for network on chip ». In : *ACM/IEEE Design Automation Conference (DAC)* (2006) (cf. p. 94).

[53] V. Pasca M. Nicolaidis et L. Angheli. « I-BIRAS : Interconnect Built-In Self-Repair and Adaptive Serialization for Inter-Die Communication in 3D Integrated Systems ». In : *of European Test Symposium, ETS, Trondheim, Norway*. 2011 (cf. p. 111).

[54] X. Zhao M. Radetzki C. Feng et A. Jantsch. « Methods for fault tolerance in networks-on-chip ». In : *ACM Computing Surveys* 46.1 (2013), p. 8–38 (cf. p. 63–65).

[55] H. Moussa. « Architectures de reseaux sur puce pour decodeurs canal multiprocesseurs ». Thèse de doct. 2009 (cf. p. 22).

[56] S. Murali et G. De Michel. « a tool for automatic topology selection and generation for NoCs ». In : *41st Design Automation Conference*. 2004, p. 914–919 (cf. p. 16, 22).

[57] G. Michelogiannakis J. Balfour B. Towles J. Kim N. Jiang D. U. Becker et W. J. Dally. « A Detailed and Flexible Cycle-Accurate Networkon- Chip Simulator ». In : *of the 2013 IEEE International Symposium on Performance Analysis of Systems and Software*. 2013 (cf. p. 109).

[58] A. Sprintson P. Gratz et M. Ramakrishna. « GCA : Global Congestion Awareness for Load Balance in Network-on-Chip ». In : *of the 7th IEEE/ACM International Symposium on Networks on Chip (NoCS),* 2008, p. 1–8 (cf. p. 56–58).

[59] A. Sprintson P. Gratz et M. Ramakrishna. « GCA : Global Congestion Awareness for Load Balance in Network-on-Chip, » in : *of the 7th IEEE/ACM International Symposium on Networks on Chip (NoCS)*. 2013, p. 1–8 (cf. p. 62).

[60] K. Chakraborty P. M Wells et G. S Sohi. « Adapting to intermittent faults in multicore systems. » In : *ACM SIGPLAN Notices*. T. 43. 3. 2008, p. 255–264 (cf. p. 111).

[61] A Ivanov P. P. Pande C Grecu et R Saleh. « Performance evaluation and design trade-offs for network-on-chip interconnect architectures ». In : *IEEE Trans Computer* (2005) (cf. p. 22).

[62] K. G. Rauwerda et T. L. Smitl P. T. Wolkotte G. Smit. « An energy-efficient reconfigurable circuit switched network-on-chip ». In : *of the 19th IEEE international parallel and distributed processing symposium*. 2005 (cf. p. 25).

[63] R. Parikh et V. Bertacco. « uDIREC : unified diagnosis and reconfiguration for frugal bypass of NoC faults ». In : *MICRO,* (2013) (cf. p. 68–70).

[64] Christian Poellabauer. « Real-Time systems ». In : (2005) (cf. p. 86).

[65] Eles.P Pop. P Izosimov. V et Peng.Z. « Design optimization of time-and costconstrained fault-tolerant embedded systems with checkpointing and replication ». In : *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 17.3 (2009), p. 389–402 (cf. p. 86).

[66] I. Pratomo et S. Pillement. « Gradient – an adaptive fault-tolerant routing algorithm for 2D mesh network-on-chips ». In : *Des. Archit. Signal Image Process.* (2013), p. 1–8 (cf. p. 67, 74).

[67] Yu Q et Ampadu P. « Dual-layer adaptive error control for network-on-chip links ». In : *IEEE Trans Very Large Scale Integr VLSI Syst* 20.7 (2012), p. 1304–1317 (cf. p. 52, 62, 63).

[68] S. K. Satpathy R. Das S. Narayanasamy et R. G. Dreslinski. « Catnap : energy proportional multiple network-on-chip ». In : *ACM of the 40th annual international symposium on Computer architecture*. 2013, p. 320–331 (cf. p. 82, 83).

[69] M. Modarressi R.J. Behrouz et H. Sarbazi-Azad. « Fault-tolerant routing algorithms in networks on-chip, » in : *Routing Algorithms in Networks-on- Chip, Springer* (2014), p. 193–210 (cf. p. 67, 74).

[70] Yehia S. « Conception des réseaux sur puce (NoCs) et génération des modèles ». Thèse de doct. 2012 (cf. p. 24).

[71] Y. Liu S. S. Jiang S. Y. Jiang G. Luo et X. T. Li. « Fault-tolerant routing algorithm simulation and hardware verification of noc ». In : *IEEE Transactions on Applied Superconductivity* 24.5 (2014), p. 1–5 (cf. p. 67).

[72] E. Salminen. « On design and comparison of On-Chip Networks ». Thèse de doct. 2010 (cf. p. 26, 29, 41).

[73] D. Siguenza-Tortosa et J. Nurmil. « Proteo : a new approach to networkon-chip ». In : *IASTED-Communication Systems and Networks (CSN 2002)*. 2002 (cf. p. 42).

[74] Jayant Kumar Singh. « ,Performance Evaluation of DifferentRouting Algorithms in Network on Chip ». Thèse de doct. 2014 (cf. p. 37).

[75] J. Navaridas S.Werner et M.Lujan. « A Survey on Design Approaches to Circumvent Permanent Faults in Networks-on-Chip ». In : *ACM Computing Surveys* 48.4 (2016), p. 1–38 (cf. p. 25, 65).

[76] C. Nicopoulos V. Soteriou et A. Vitkovskiy. « Dynamic fault-tolerant routing algorithm for networks-on-chip based on localised detouring paths ». In : *IET Comput. Digit. Tech* 7.2 (2013), p. 93–103 (cf. p. 66, 74).

[77] A.Erichsen A.M.Amory V.Fochi E.Wachter et F.G.Moraes. « An Integrated Method for Implementing Online Fault Detection in NoCBased MPSoCs ». In : *In : ISCAS* (2015), p. 1562–1565 (cf. p. 88).

[78] S.-J. Chen W.-C. Tsai D.-Y. Zheng et Y. H. Hu. « A fault-tolerant noc scheme using bidirectional channe ». In : *IEEE/ACM Design Automation Conference* (2011), p. 918–923 (cf. p. 66).

[79] L. Huang G. Li X. Zhang M. Ebrahimi et A. Jantsch. « Routing-Level Solution for Fault Detection, Masking, and Tolerance in NoCs ». In : *2015 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing,* (2015), p. 365–369 (cf. p. 68).

[80] E.J.Chang Y.Y.Chen et H.K.Hsin. « Path-Diversity-Aware Fault-Tolerant Routing Algo- rithm for Network-on-Chip Systems ». In : *IEEE Transactions on parallel and distributed systems* 28.3 (2017), p. 838–849 (cf. p. 71, 74).

[81] A. Greiner Z. Zhang et S. Taktak. « reconfigurable routing algorithm for a fault-tolerant 2d-mesh network-on-chip ». In : *IEEE/ACM Design Automation Conference* (2008), p. 441–446 (cf. p. 65, 68, 70, 74).

Résume

Un réseau sur puce peut, en effet, se retrouver avec des routeurs ou des liens non opérationnels, qui ne peuvent plus être utilisés pour acheminer les messages. Une solution potentielle à ce problème est l'introduction de mécanismes de tolérance aux pannes permettant au système de fonctionner en mode dégradé malgré la présence de composants défectueux. Le système devrait fonctionner correctement lorsque certaines erreurs se produisent dans les routeurs ou les liens. Cependant, la communication a un impact énorme sur les performances du Network on Chip, et la conception d'un algorithme de routage efficace est plus que nécessaire. L'objectif assigné à l'algorithme de routage est de trouver un nouveau chemin pour diriger les paquets de la source vers la destination dans un réseau défectueux.De nombreux algorithmes de routage tolérants aux pannes sont utilisés pour surmonter les défauts dans les Networks on chip. Cependant, ces algorithmes de routage souffrent de plusieurs problèmes, dont celui de la congestion. Dans ce travail, une approche originale inspirée de Catnap est proposée pour les NoC utilisant des mécanismes de détection de congestion locale et globale avec une architecture de sous-réseaux hiérarchiques. Avec l'aide de ces deux techniques, le NoC devient tolérant aux pannes et est capable d'utiliser efficacement le débit.

Mots-clés :

Congestion; NoC; Fautes; Tolérance; sous réseaux; Routage; Erreurs; Fiabilité; Equilibrage de charge; Inter-blockage.