Optimisation de la consommation d'énergie et de la latence dans les réseaux sur puces

Erwan Moréac

UNIVERSITE BRETAGNE LOIRE

THÈSE / UNIVERSITÉ DE BRETAGNE SUD

UFR Sciences et Sciences de l'Ingénieur

sous le sceau de l'Université de Bretagne Loire

Pour obtenir le grade de :

DOCTEUR DE L'UNIVERSITÉ DE BRETAGNE SUD

Mention : STICC

École Doctorale SICMA

présentée par

Erwan Moréac

Lab-STICC, Laboratoire des Sciences et Techniques de l'Information, de la Communication et de la Connaissance

Optimisation de la consommation d'énergie et de la latence dans les réseaux sur puces

Thèse soutenue le 25-10-2017,
devant la commission d'examen composée de :

M. Patrick Girard
Directeur de recherche CNRS, LIRMM Montpellier / Président, Rapporteur

M. Yannick Le Moullec
Professeur, Tallinn University of Technology / Rapporteur

Mme. Cécile Belleudy
Maître de conférences HDR, Université de Sophia Antipolis Nice / Examinatrice

Mme. Lilia Zaourar
Ingénieure de recherche, CEA List / Examinatrice

M. Pierre Bomel
Ingénieur de recherche, Université de Bretagne-Sud / Invité

M. André Rossi
Professeur, Université d'Angers / Directeur de thèse

M. Johann Laurent
Maître de conférences HDR, Université de Bretagne-Sud / Co-directeur de thèse

Remerciements

Je remercie Patrick Girard, Directeur de recherche CNRS, pour avoir accepté d'être rapporteur de ce travail.

Je remercie Yannick Le Moullec, Professeur à Tallinn University of Technology, pour avoir accepté d'être rapporteur de ce travail.

Je tiens également à remercier Cécile Belleudy et Lilia Zaourar, respectivement Maître de conférences à l'Université de Sophia Antipolis et Ingénieur de recherche au CEA List pour leur participation à ce jury.

Je remercie tout particulièrement André Rossi et Johann Laurent, mes directeurs de thèse pour leur encadrement, tous les conseils qu'ils m'ont donné ainsi que pour leur disponibilité durant ces trois années.

Ma reconnaissance va à Pierre Bomel pour tout le temps qu'il a investi à participer activement à nos recherches, que je considère comme un encadrant à part entière.

Je remercie Antoine Courtay, pour ses travaux qui ont été le point de départ de ma thèse.

Je remercie Jean-Philippe Diguet et Emmanuel Boutillon pour toutes les discussions et débats très constructifs qui m'ont permis de progresser et d'avoir une vision différente de mon sujet de thèse.

Je remercie Guy Gogniat et Marc Sevaux, Professeurs des Universités et Directeur du laboratoire Directeur du laboratoire *Lab-STICC* à mon entrée et à ma sortie de thèse pour m'avoir accueilli dans leurs équipes de recherche respectives. Je remercie également tous les membres du *Lab-STICC*, ces trois années de travail ont vraiment été une excellente expérience grâce à la bonne ambiance qui règne au labo et dans la salle de pause.

Je ne peux pas faire une section remerciements sans remercier tous les doctorants du *Lab-STICC*, de l'IRDL et du LBCM avec lesquels j'ai passé plusieurs années à discuter, rigoler, débattre, passer de bonnes soirées etc... Merci à Charly et Hugues pour mes débuts de thèse, à Yohann, Paul, Mathilde, Antoine[2], Alexandre, Fanny, Mourad, Vincent, Gaby, Max, Nico, Mikael, Pierre, Delphine, Tatiana, Albane, Guillaume et j'oublie encore des personnes à l'instant où j'écris ces lignes.

Je remercie mes amis du Bagad du Faouët, ma deuxième famille, où je passe toujours d'agréables moments avec Nico, Pierrick, Matthieu, Max, Yoyo, Rolupe et bien sûr tous les autres membres que je ne cite pas ici. Je vais aussi remercier mes amis d'une manière générale, avec qui j'ai passé les trois dernières années, vous revoir tous était une bouffée d'air frais même si les lendemains de soirées ne paraissaient pas aussi vivifiants !

Je remercie Audrey pour sa patience malgré mes nombreuses indisponibilités et son écoute. Enfin, je remercie ma famille dont le soutien a été d'une très grande aide tout au long de ma vie.

Table des matières

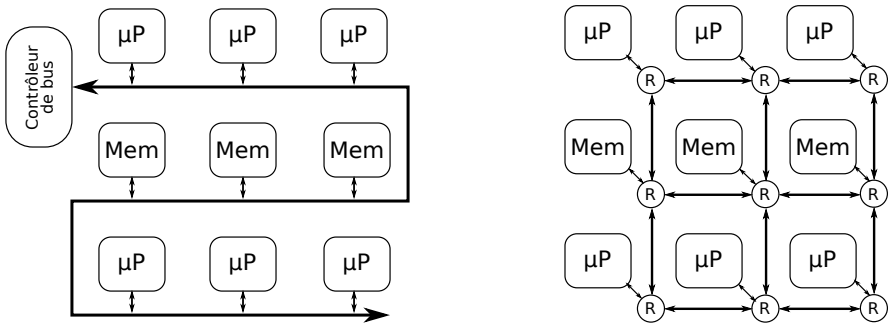
Contexte et problématique

L'avancée technologique nous permet aujourd'hui de réaliser des applications de plus en plus complexes. Cette évolution a été possible grâce à la miniaturisation des transistors qui permet d'en intégrer plusieurs milliards sur une seule puce [1]. La conception de systèmes sur puce (SoC), a également nécessité le développement de méthodes de conception à même de maîtriser la complexité croissante des applications, comme la synthèse de haut niveau [2]. Un système sur puce comprend généralement au moins un cœur de processeur et d'autres éléments tels que de la mémoire et des accélérateurs matériels. Afin de s'adapter aux exigences croissantes des applications, l'architecture des SoCs est aussi devenue plus complexe et est passée d'un système sur puce mono-processeur à un système multi-processeur (MPSoC). Un MPSoC peut être hétérogène en contenant des processeurs et accélérateurs matériels différents ou homogène avec par exemple une grille de plusieurs processeurs identiques fonctionnant en parallèle. Par conséquent, cet accroissement du nombre d'éléments communiquant induit également une augmentation des besoins en bande-passante à l'intérieur d'une même puce.

Il existe deux types d'architecture de communication pour véhiculer des données entre composants d'un même SoC. La première architecture illustrée à la figure 1a est le bus partagé, une grande ligne de communication autorisant une seule communication à la fois entre deux composants. Cela a pour conséquence de générer de la latence puisque les autres processeurs attendent leur tour pour communiquer.

La figure 1b montre la seconde architecture possible, le réseau sur puce (NoC). De la même manière qu'un réseau d'ordinateurs à l'échelle macroscopique, ce réseau a l'avantage d'établir des communications entre composants en parallèle contrairement au bus partagé. Le transfert de données en parallèle est possible grâce aux routeurs aiguillant les données en fonction de leur destination. Cela améliore considérablement le débit des données à l'intérieur de la puce et évite que les communications ne deviennent le goulot d'étranglement des performances des SoCs. Cependant, cette architecture de communication apporte de nouvelles problématiques.

Le coût matériel et les communications supplémentaires qu'engendre l'architecture du NoC provoquent une hausse significative de la consommation d'énergie du système. L'impact énergétique du réseau sur puce (ou NoC) devient une contrainte majeure dans la conception d'une puce et peut représenter jusqu'à 40% de la consommation de celle-ci [3]. Depuis plusieurs années, le critère de consommation est devenu un paramètre critique au même titre que les performances et la surface dans la conception d'un système. En effet, de



(a) Schéma d'un bus partagé. (b) Schéma d'un NoC.

FIGURE 1 – Deux architectures de communications possibles dans un même SoC. Dans ces exemples, les SoCs sont constitués de μP (processeur) et de blocs mémoire (Mém). Les cercles avec un R représentent les routeurs.

manière générale, une hausse de la consommation de puissance génère une augmentation de la température de la puce. Cela est dû à l'effet thermique qui échauffe un conducteur traversé par un courant et plus le nombre de composants est important, plus cet effet est sensible. Cette hausse de la température entraine plusieurs problèmes, la durée de vie des composants d'un système peut être réduite à cause du phénomène d'électromigration qui est facilité plus la température augmente. L'électromigration est le déplacement d'un groupe d'atomes induit par un flux d'électrons dans un conducteur et si ces déplacements sont trop importants, cela fragilise le conducteur. De même, l'augmentation de la puissance induit un système de refroidissement plus important et donc un coût et un poids plus important, ce qui est doublement pénalisant pour un système embarqué. En outre, toute augmentation de la consommation d'énergie se fait au détriment de l'autonomie d'un système embarqué dans la mesure où les progrès de la technologie des accumulateurs ont été beaucoup moins spectaculaires que ceux des circuits à tous points de vue. La figure 2 montre l'évolution des différentes caractéristiques des systèmes mobiles de 1990 à 2010 et nous pouvons voir que la capacité des batteries ne s'est accrue que d'un facteur 3 tandis que la vitesse des CPU (Central Processing Unit) a explosé en étant multipliée par environ 6000, augmentation illustrant les besoins en énergie beaucoup plus importants. De plus, une hausse de l'énergie consommée par le circuit impose un sur-dimensionnement de la batterie provoquant une hausse du poids et du coût du système. Ainsi, bien que le NoC permette d'atteindre un débit supérieur à celui qu'un bus peut offrir, son utilisation se traduit généralement par une augmentation considérable de la consommation. Il devient donc nécessaire de concevoir des techniques pour réduire la consommation d'énergie des NoCs aussi bien pour limiter la température que pour préserver l'autonomie du système.

L'objectif de cette thèse est donc d'apporter des techniques d'optimisation visant à réduire la consommation d'énergie du NoC tout en minimisant leur im-

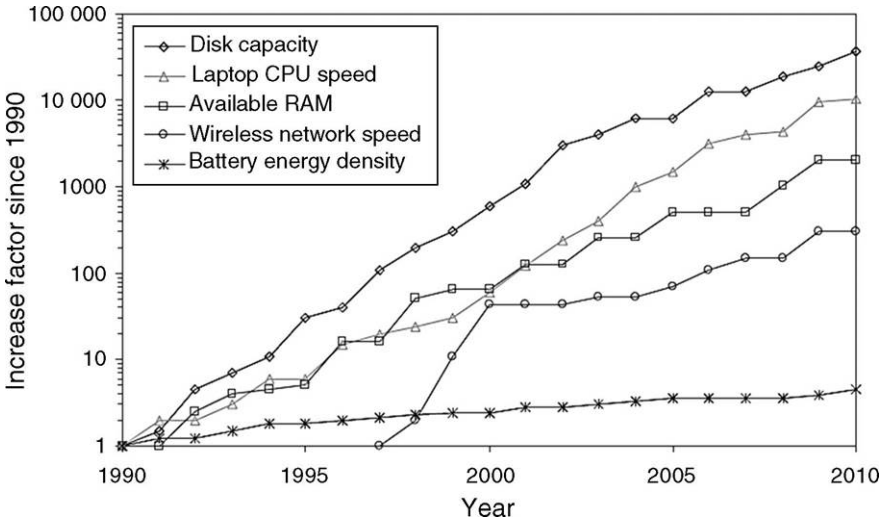


FIGURE 2 – Évolution des différentes caractéristiques des systèmes mobiles de 1990 à 2010. Ce graphique est directement issu de [4].

pact sur les performances. Comme nous souhaitons développer des techniques d'optimisation applicables pour le plus grand nombre de systèmes possibles employant un NoC, nos optimisations seront focalisées sur le NoC indépendamment de l'architecture utilisée. De plus, nous voulons développer des techniques d'optimisation indépendantes des caractéristiques du NoC telle que la topologie, afin de permettre l'utilisation de nos techniques dans le plus grand nombre possible de réseaux.

Contributions de la thèse

Les trois contributions principales de cette thèse sont les suivantes :

— L'amélioration de la modélisation du NoC : une modélisation fiable est une étape indispensable pour déterminer quels sont les éléments les plus consommateurs du NoC d'une part, puis essentielle pour évaluer nos propositions d'optimisation d'autre part. Ainsi, la première contribution de cette thèse est d'étudier les modèles de consommation des NoC, puis d'en proposer de nouveaux afin d'en améliorer la précision.

— Le développement d'un outil de simulation énergétique du NoC : une fois que nous avons un modèle de consommation fiable du réseau, il est nécessaire de le simuler dans le but de pouvoir comparer les techniques d'optimisation proposées et d'estimer leur impact énergétique.

— La proposition de techniques d'optimisation de la consommation du NoC : deux techniques sont présentées, offrant des compromis différents entre consommation et latence.

Organisation du document

Ce manuscrit est organisé en deux parties, composées chacune de deux chapitres. La première partie se concentre sur la modélisation du NoC et notre contribution sur son amélioration tandis que la deuxième partie présente les techniques d'optimisation de la consommation du NoC de la littérature puis nos contributions dans ce domaine.

Partie 1 : Modélisation. Le premier chapitre présente donc un état de l'art sur le NoC dans le but de connaître ses différents composants et son fonctionnement en détail. Cette étude bibliographique est étendue avec l'analyse de la modélisation de la consommation d'énergie du NoC dans la littérature.

A partir des observations effectuées dans le chapitre 1, nous proposons d'améliorer la modélisation de la consommation du NoC. Pour cela, nous intégrons à un simulateur de NoC (Noxim), un modèle de liens considérant le crosstalk en étant bit-près. Cette réalisation nous permet de quantifier l'erreur d'estimation commise par le modèle de liens de la littérature des NoCs. Nous mettons ainsi en évidence le rôle capital de la nature des données dans la consommation des liens. De plus, nous montrons la part importante que représentent les interconnexions dans la consommation du NoC. Enfin, nous mettons à disposition toutes les améliorations apportées au simulateur d'origine avec le simulateur nommé Noxim-XT, ce qui conclut la partie modélisation.

Partie 2 : Optimisation. Le troisième chapitre est une étude bibliographique des différentes techniques d'optimisation pouvant être réalisées dans les NoCs. Cette exploration de techniques nous permet de sélectionner les directions d'optimisation les plus prometteuses. A la fin de ce chapitre, nous décidons du type d'optimisation vers lequel nous nous orienterons.

Le quatrième chapitre présente les différentes techniques d'optimisation que nous avons proposées. La part de l'énergie du NoC consommée par les interconnexions étant importante, ces techniques visent à la réduire. Le choix de cette stratégie est justifié par l'étude présentée au chapitre 3. Nous proposons donc deux techniques, le Smart Temporal Shielding et la Communication Inspirée par le Cortex. Ces deux techniques visent à réduire au maximum les effets de diaphonie capacitive en réduisant les transitions énergivores et en diminuant l'activité des interconnexions.

Nous concluons la thèse par un bilan des contributions apportées concernant d'abord la modélisation du NoC, puis de l'optimisation de sa consommation d'énergie. Ensuite, nous discutons des extensions possibles de nos travaux. Les perspectives à court terme sont abordées avec principalement l'amélioration des techniques d'optimisation développées au cours de cette thèse. Enfin, des perspectives à moyen terme sont suggérées.

Première partie

Modélisation

1

État de l'art du NoC et sa modélisation

Sommaire

Introduction

Ce chapitre a pour objet l'étude de la littérature consacrée à la modélisation de la consommation des réseaux sur puces. Il présente donc l'état de l'art sur la modélisation de la consommation en termes de puissance et d'énergie du NoC. Ce chapitre est divisé en deux parties. La première explique ce qu'est un NoC et la deuxième expose la modélisation de ce type de réseau.

Dans la section 1.1, nous présentons les principales caractéristiques du NoC. L'architecture matérielle y est exposée dans un premier temps suivie des différents paradigmes de la communication dans un NoC. La modélisation de la consommation d'un circuit est présentée dans la section 1.2. Les acteurs de la consommation y sont décrits puis sont exposées les méthodes pour mesurer la consommation ou l'estimer. La section 1.3 présente la manière dont est réalisée la modélisation des éléments du NoC : l'adaptateur réseau, le routeur et les interconnexions. Dans la section 1.4, nous évoquons les simulateurs de NoC qui utilisent les modèles décrits dans les sections précédentes. Enfin, un bilan termine cet état de l'art en exposant les différents verrous restants dans l'estimation de la consommation d'énergie du NoC.

1.1 Réseaux sur puces

La miniaturisation des transistors permet d'intégrer dans une seule puce une multitude de blocs logiques (IP : Intellectual Property) communiquant entre elles, entraînant une saturation des communications dans les architectures à base de bus de communication. Le NoC est alors reconnu comme la solution aux contraintes de bande passante dans les MPSoCs [5]. Quelques définitions de la littérature du NoC sont présentées dans cette section ainsi que ses principales caractéristiques matérielles visant à faire mieux comprendre son fonctionnement.

1.1.1 Définition d'un NoC

Le NoC est un système composé de plusieurs éléments de routage (routeurs) connectés selon une topologie spécifique [6]. Les éléments communicants du SoC sont connectés via des adaptateurs réseaux (NA, Network Adapter) au NoC. L'architecture du NoC peut intégrer différents protocoles réseaux définissant les mécanismes d'échange des données. La figure 1.1 représente un NoC avec les utilisateurs (IPs) et éléments principaux du réseaux.

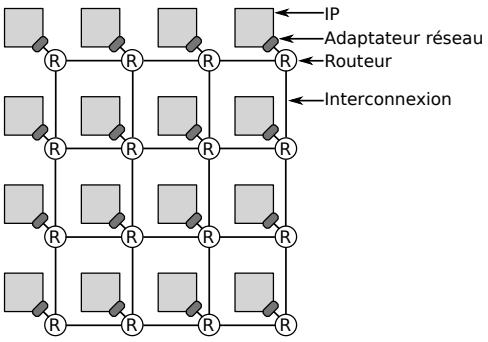


FIGURE 1.1 – Schéma classique d'un NoC mesh-2D (grille 2D).

1.1.2 Architecture matérielle du NoC

Les principaux éléments définissant l'architecture matérielle de communication d'un NoC sont présentés dans les paragraphes suivants.

1.1.2.1 Topologie

La topologie indique comment sont agencés les différents composants matériels du NoC, leur disposition spatiale et leurs connexions entre composants. Il existe de nombreuses topologies et le choix d'une topologie est réalisé selon les besoins en communication du SoC. En effet, la latence des transmissions entre 2 IPs dépend de la distance physique qui les séparent et aussi des chemins de routage disponibles. La figure 1.2 montre quelques exemples de topologies que l'on peut retrouver dans un NoC. La topologie peut être régulière ce qui induit la possibilité de schémas de routage simples comme les topologies des figures 1.2.a,b. Il existe d'autres topologies dont le comportement des routeurs diffère en fonction de leur position, comme l'illustre la figure 1.2.c. Il y a également des topologies irrégulières qui autorisent un meilleur dimensionnement du réseau pour l'application considérée mais qui requièrent la mise en œuvre d'un routage spécifique (voir figure 1.2.d).

1.1.2.2 Routeur

Le routeur est un nœud du réseau qui oriente les données arrivant à lui vers le bon port de sortie. Sa fonction se résume donc à retransmettre les données d'une direction donnée vers une direction de sortie. Un routeur est connecté à d'autres routeurs et à une IP via un adaptateur (NA). Il possède des ports d'entrée et de sortie qui sont mis en relation suivant le besoin de routage spécifié par le paquet de données à transférer. Il peut posséder des buffers stockant les données en transit. Ces buffers peuvent être de taille plus ou moins importante suivant le protocole ou le mécanisme de routage choisi.

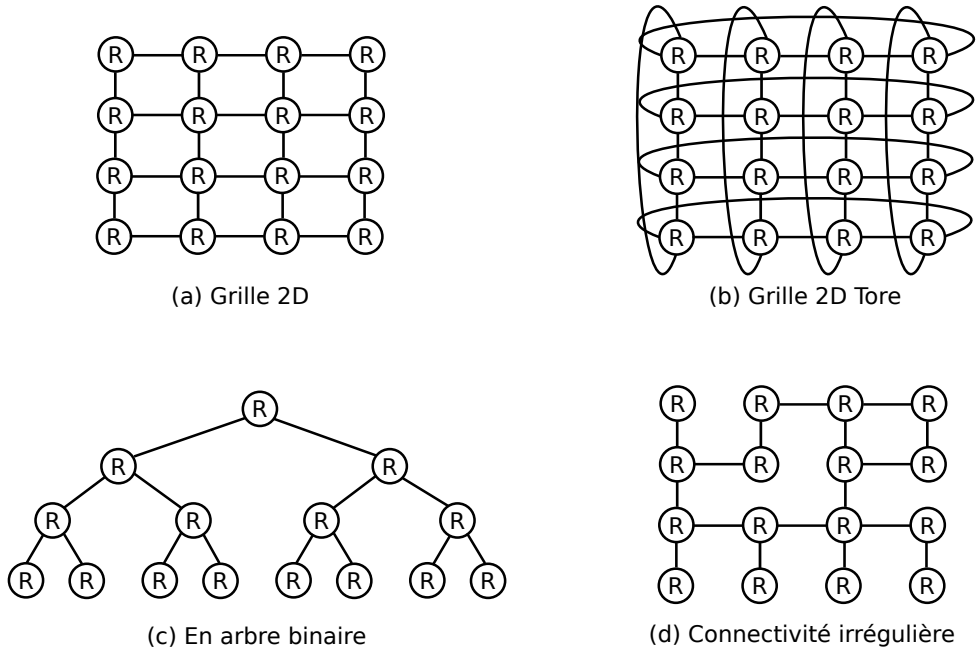


FIGURE 1.2 – Différents exemples de topologies de NoC.

Les dimensions d'un buffer s'expriment en fonction de sa largeur, définie par celle des liens du réseau (en bits) et en profondeur par rapport au nombre de mots pouvant être stocké. Les composants du routeur seront détaillés en section 1.3.

1.1.2.3 Adaptateur réseau

L'adaptateur réseau (NA) est le lien entre les utilisateurs du NoC et le réseau lui-même. Il doit rendre l'utilisation du NoC transparente pour le processeur à coût minimal en termes de calcul et de surface. Cette couche supplémentaire permet de dissocier les calculs des communications, c'est pourquoi le NA est représenté en 2 parties : le côté interface processeur (CI) et l'interface réseau (NI) comme illustré à la figure 1.3. Étant donné que le NA injecte les messages dans le réseau, il est en charge de la gestion du contrôle du flux des messages dans le réseau. Ce procédé est rendu possible par l'établissement d'une communication entre les routeurs et les NAs, les routeurs donnant un retour sur leur état pour indiquer s'ils sont en mesure de stocker des messages supplémentaires.

1.1.2.4 Interconnexions

Il s'agit du lien physique qui permet le passage des données d'un routeur à un autre. Le nombre de fils physiques utilisés pour une interconnexion détermine

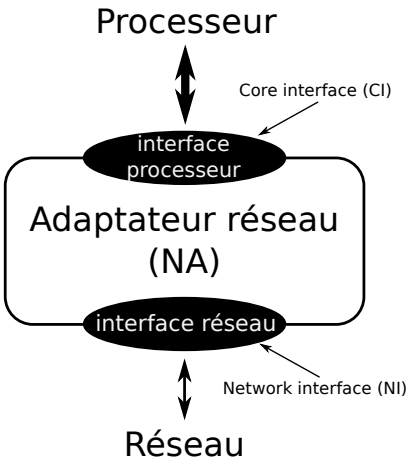


FIGURE 1.3 – L'adaptateur réseau avec ses 2 interfaces pour le réseau et le processeur.

la bande passante maximum par cycle d'horloge. Les fils sont composés d'inverseurs (répéteurs) qui amplifient le signal pour limiter l'atténuation. Les liens présents entre les routeurs peuvent être uni-directionnels ou bi-directionnels. Maintenant que les éléments de l'architecture du NoC sont présentés, le prochain paragraphe présente comment tous ces composants interagissent entre eux.

1.1.3 Communication dans un NoC

La communication entre ces composants se fait au moyen de protocoles implantés dans les routeurs et les NAs. Ces protocoles définissent les règles du transfert d'un message d'un point à un autre du réseau. Nous retrouvons parmi eux le formatage des données injectées dans le réseau, le protocole de gestion de flux, le type de commutation, la politique de mémorisation et également l'algorithme de routage utilisé par le NoC. Nous allons voir dans un premier temps comment la donnée de l'IP est transformée pour être injectée dans le NoC.

1.1.3.1 Transmission d'un message

Lorsqu'un processeur souhaite envoyer un message par le NoC, le message n'est pas envoyé selon le protocole de communication des processeurs. En effet, le message est découpé en paquets afin d'être envoyé morceau par morceau (figure 1.4a) dans le réseau à l'instar des réseaux à notre échelle tels qu'internet. La quantité de données transmises simultanément entre deux composants du réseau est limitée par la capacité du lien, donc par sa largeur, exprimée en

nombre de bits. Ainsi, un paquet doit aussi être découpé en plusieurs morceaux quel que soit le protocole de commutation adopté dans le réseau. Ce découpage est illustré à la figure 1.4b. La granularité des données envoyées sur le réseau correspond à l'unité physique *phit* (PHysical unIT), qui détermine la quantité de bits transportables sur le lien en un cycle. En revanche, le contrôle de flux sur un lien peut être réalisé avec une granularité de un ou plusieurs mots (phits). Par conséquent, un niveau de granularité plus gros est défini avec l'unité de contrôle de flux *flit* (FLow control unIT) comme illustré à la figure 1.4c,d. Le contrôle se fait donc à la fréquence des flits, alors que les données circulent à la fréquence des phits. Néanmoins, il est très fréquent de rencontrer des NoCs dont le flux est contrôlé mot par mot et par conséquent, les flits et phits sont de même taille. Voyons en détail le contenu d'un paquet circulant dans un NoC.

Un paquet est constitué d'un en-tête contenant les informations nécessaires au routage de celui-ci vers sa destination présentée à la figure 1.4. La suite du paquet est la charge utile divisée en plusieurs phits, et se termine par la queue indiquant la fin du paquet. Le type du phit est identifié par une étiquette, un groupe de bits (2 dans l'exemple) qui indique le type du phit aux éléments du NoC. Par exemple, dans la figure 1.4, l'étiquette 00 correspond à l'en-tête, l'étiquette 01 à la charge utile et la valeur 10 à la queue qui est le dernier élément du paquet.

Une fois que le paquet est arrivé à destination, il est reconstitué par l'adaptateur réseau qui reforme le message au format de l'IP récepteur.

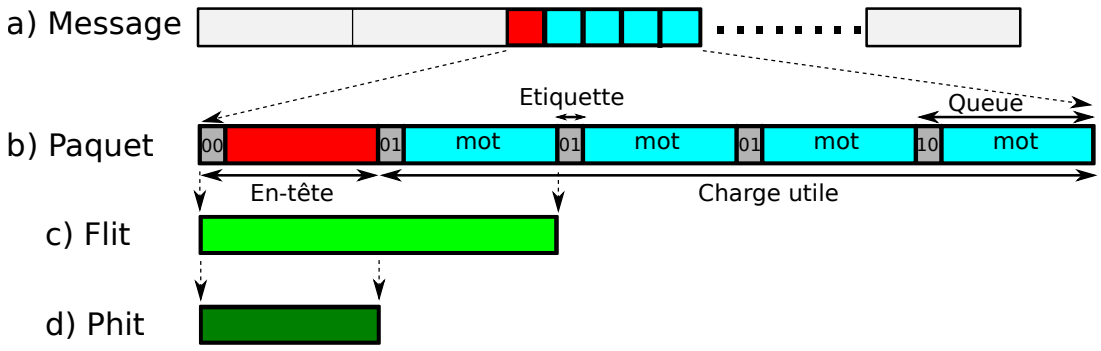


FIGURE 1.4 – Découpage d'un message en paquet et composition d'un paquet.

1.1.3.2 Gestion du flux

Le transfert des *flits* à travers les routeurs et NAs est géré avec un protocole de flux de contrôle. En effet, des règles sont établies pour indiquer si le récepteur est disponible pour recevoir une nouvelle donnée. Plusieurs méthodes existent pour gérer le flux des données :

— Le protocole *ACK/NOACK* utilisé par le NoC XPIPES [7] est un mécanisme d'accusé de réception, la donnée est conservée par l'émetteur tant que le destinataire n'a pas envoyé le signal ACK d'acquittement, signalant la bonne réception du paquet. Si la donnée n'est pas reçue, le signal NOACK passe de 0 à 1 pendant un cycle d'horloge, sinon c'est le signal ACK qui passe de 0 à 1 pendant un cycle.
— Le protocole *STALL/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 de réception n'a plus assez de place, le récepteur fait passer le signal STALL à 1.
— Le contrôle de flux par crédits d'émission est également utilisé par plusieurs NoCs tels que le SPIN [8] et AEthereal [9]. L'émetteur possède une quantité donnée de crédits, leur nombre est décrémenté lorsqu'une donnée est envoyée. A posteriori, lorsque le récepteur a consommé les données reçues, il attribue à l'émetteur un nombre de crédits correspondant à l'espace disponible de sa mémoire de réception.

Ces protocoles permettent la gestion entre les différents éléments du réseau. Nous allons désormais aborder l'exploitation des ressources du réseau.

1.1.3.3 Type de commutation du réseau

Le type de commutation précise comment sont utilisées les ressources du NoC. Nous distinguons deux types de commutation :

— La commutation de circuit (circuit switching) : elle repose sur la réservation d'un ensemble de ressources du réseau pour établir un chemin entre la source et la destination pendant le transfert de tout le message. À la fin de l'envoi, le chemin est dissout.
— La commutation de paquet (packet switching) : elle consiste à acheminer les messages en les découpant sous forme de paquets. 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 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.

1.1.3.4 Politique de mémorisation

La politique de mémorisation est choisie selon la granularité de transmission/contrôle du flux de données (flit/phits). Parmi les politiques de mémori-

sation, nous distinguons :

— *Store-And-Forward (SAF)*, chaque composant du réseau attend la réception de la totalité du paquet, le stocke entièrement avant de le transmettre au suivant
— *Virtual Cut Through (VCT)* : cette technique a été introduite pour réduire la latence du SAF. Un composant réseau fait suivre les flits d'un paquet au plus vite, mais il peut stocker l'intégralité du message s'il y a une contention
— *Wormwhole* : cette politique consiste à acheminer le paquet sous forme de flits, sans possibilité de stocker la totalité du paquet. Ainsi la taille des mémoires est réduite avec le risque d'interblocage en cas de contention, autrement nommé *deadlock*.

1.1.3.5 Algorithmes de routage

C'est la logique de routage qui va définir le parcours des données quel que soit le type de commutation du réseau. Deux types d'algorithmes de routage peuvent être distingués :

— Les algorithmes de routage déterministes établissent les chemins des paquets en fonction de l'adresse de destination utilisant toujours le même chemin entre chaque paire de nœuds. Le routage ordonné par dimensions X-Y est un exemple d'algorithme de routage déterministe utilisé fréquemment sur les topologies en grille 2D.
— Les algorithmes de routage adaptatifs s'ajustent facilement à toutes les topologies. Les paquets peuvent y circuler librement sans restriction. Un paquet peut contourner tous les conflits qu'il croise. En revanche, cette technique de routage est complexe à mettre en place et engendre des conflits. En effet, les paquets peuvent tourner dans le réseau en occupant les ressources sans trouver leur destination ou entraînant une incertitude sur la durée de transmission.

Les configurations de ces protocoles varient selon la conception des NoCs. Le prochain paragraphe présente différents NoCs et leur méthode de communication.

1.1.4 Quelques NoC

Dans les travaux sur les réseaux sur puces, différents NoCs ont été proposés. Parmi les principaux NoCs rencontrés dans les domaines de la recherche et de l'industrie, nous pouvons citer :

— Le réseau Æthereal développé par *Philips*
— Le réseau MANGO (Message-passing Asynchronous Network-on-chip providing Guaranteed services over OCP interfaces) de la *Technical University of Denmark*

— le réseau Proteo 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 développé par le *Technion-Israel Institute of Technology*
— Le réseau SPIN (Scalable Programmable Integrated Network) créé au sein de *l'Université Pierre et Marie Curie*
— Le réseau STNoC ou Spidergon conçu par *STMicroelectronics*
— Le réseau XPIPES élaboré par *l'Université de Bologne*
— Le réseau μSpider II élaboré à *l'Université de Bretagne Sud*

Le tableau 1.1 répertorie les différentes 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).

TABLEAU 1.1 – Comparaison de quelques caractéristiques de plusieurs NoCs

Réseau	**Topologie**	**Gestion du flux**	**Type de commutation**	**Politique de mémorisation**	**Algorithme de routage**
Æthereal [9]	Grille 2D	crédit d'émission	Paquet	Wormhole	Déterministe
MANGO [10]	Grille 2D	Non spécifié	Circuit / paquet	Circuit Virtuel	Déterministe
Proteo [11]	Tore	Non spécifié	Paquet	Wormhole	Déterministe
QNoC [12]	Grille 2D	crédit d'émission	Paquet	Wormhole	Déterministe
SPIN [8]	Arbre élargi	crédit d'émission	Paquet	Wormhole	Adaptatif
STNoC [13]	Spécifique	Non spécifié	Paquet	Wormhole	Non spécifié
XPIPES [7]	Grille 2D, tore, hypercube	ACK/NOACK	Paquet	Wormhole	Déterministe
μSpider II [6]	Irrégulière	Non spécifié	Paquet	Wormhole	Déterministe

A partir de ces informations, il est possible de dresser le profil moyen d'un NoC, ses caractéristiques sont les suivantes :
— Topologie : grille 2D
— Gestion de flux : crédit d'émission
— Type de commutation : paquet
— Mémorisation : wormhole
— Algorithme : déterministe

Avant de présenter spécifiquement la modélisation de la consommation du NoC, nous allons aborder la modélisation de la consommation d'un circuit en général.

1.2 Modélisation de la consommation d'un circuit

La modélisation de la puissance d'un circuit permet d'identifier quels éléments sont les plus consommateurs. Les pics de puissance provoquent une hausse de la température qui peut compromettre la fiabilité et la durabilité de la puce.

A partir d'un modèle de puissance d'une puce, il est possible de déterminer l'énergie consommée par celle-ci durant son fonctionnement. L'énergie consommée par la puce est une information importante pour estimer le coût de son alimentation électrique. Dans le cas d'un système mobile, la consommation d'énergie conditionne la dimension des batteries. La suite de cette section détaille le calcul de la puissance d'un circuit, puis les différentes méthodes qui permettent la mesure ou l'estimation de la consommation d'énergie de ce même circuit.

1.2.1 Puissance statique et dynamique

La puissance d'un circuit électrique est égale à la somme de la puissance statique et dynamique [14, 15]. La puissance du circuit est exprimée par l'équation suivante :

$$P = P_{statique} + P_{dynamique} \tag{1.1}$$

$P_{dynamique}$ équivaut à la puissance dissipée pendant les changements d'états des composants logiques et des interconnexions dans le circuit et $P_{statique}$ est la puissance consommée quand le circuit est alimenté.

1.2.1.1 Puissance statique

Dans une puce, la puissance statique est causée par les courants de fuites des transistors lorsqu'ils sont alimentés. La puissance statique est causée principalement par 3 types de courants de fuite [16] qui sont illustrés sur la figure 1.5.

- I_{seuil} circule entre le drain et la source car il y a une tension drain-source lorsque le transistor ne commute pas. Ce courant de fuite génère la plus grande consommation d'énergie [16].
- Avec la miniaturisation des transistors, la réduction de la taille de l'oxyde de grille (S_iO_2) créé un courant entre la grille et le substrat entraînant une diminution de l'impédance et l'apparition d'un courant de fuite I_{grille} entre l'oxyde de grille et le substrat. Il est présent quel que soit l'état du transistor (passant ou bloqué) [16].
- Enfin, I_{diode} est le courant de fuite qui circule aux jonctions PN du transistor. Ce courant est présent continuellement.

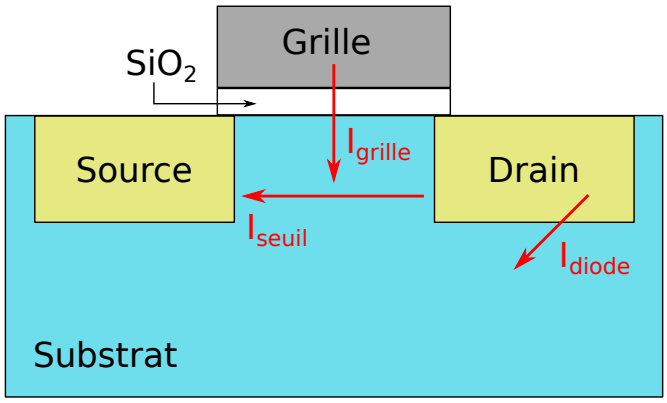


FIGURE 1.5 – Principaux courants de fuite observés sur un transistor MOSFET (Metal Oxide Semiconductor Field Effect Transistor).

Le calcul de la puissance statique est donc le produit des courants de fuites par la tension d'alimentation V_{dd} :

$$P_{statique} = (I_{seuil} + I_{grille} + I_{diode}) \times V_{dd} \tag{1.2}$$

1.2.1.2 Puissance dynamique

La puissance dynamique est la puissance nécessaire pour charger puis décharger les capacités activées du circuit. Contrairement à la puissance statique, elle n'est pas uniquement générée par les transistors. En effet, les interconnexions qui relient les transistors entre eux, et à un niveau plus haut, les interconnexions reliant les composants d'un circuit entre eux, se chargent et se déchargent comme les transistors. Cela génère donc de la puissance dynamique supplémentaire. Par conséquent, celle-ci dépend à la fois du taux de commutation des transistors et des interconnexions que contient le circuit. Le taux de commutation détermine le ratio des capacités du circuit qui commutent. De la même manière, quand nous parlons du taux de commutation des interconnexions, cela représente le taux des fils commutant sur un nombre de fils donné. La puissance d'un circuit est calculée avec l'équation suivante [14] :

$$P_{dynamique} = \alpha \times C \times V_{dd}^2 \times f \tag{1.3}$$

— α représente le taux de commutation du circuit compris entre 0 et 1
— C représente la capacité du circuit qui commute (F)
— V_{dd} représente la tension d'alimentation (V)
— f représente la fréquence du circuit (Hz)

Pour résumer, le circuit est représenté comme une grande capacité se chargeant et se déchargeant selon l'activité réelle de celui-ci. Ainsi, C est la capacité

équivalente de tous les transistors et interconnexions qui commutent dans le circuit. Nous allons maintenant mettre en évidence la relation entre la puissance et la consommation d'énergie d'une puce.

1.2.1.3 Relation entre la puissance et l'énergie

L'énergie est l'intégrale par rapport au temps de la puissance. La puissance donne l'énergie consommée à un instant t tandis que l'énergie représente ce qui est utilisé pendant une durée de fonctionnement donnée comme le montre la figure 1.6. Cette relation donne l'équation suivante :

$$E = P \times t \tag{1.4}$$

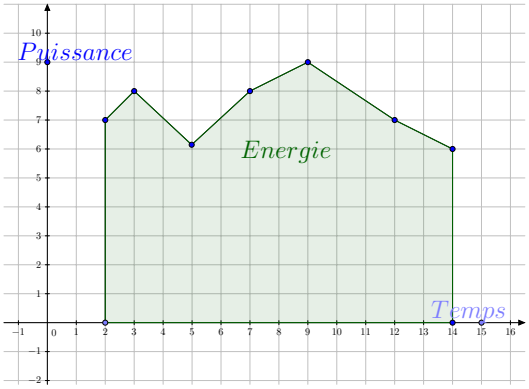


FIGURE 1.6 – Rapport entre l'énergie, la puissance et le temps.

(1.4) est une approximation valable pour des durées t très courtes, pendant lesquelles la puissance peut-être considérée comme constante. Lorsque nous parlerons de consommation dans la suite de cette thèse, nous ferons référence à l'énergie consommée. La suite de cette section introduit les différentes méthodes d'estimation de la consommation d'énergie d'un circuit.

1.2.2 Méthodes d'estimation de la consommation d'énergie

L'estimation de la consommation d'énergie d'un circuit est soit possible par la mesure physique de ce circuit en fonctionnement soit en utilisant des outils automatique s'appuyant sur des modèles de consommation. Ces modèles peuvent être situés à des niveaux d'abstractions différents et selon ce niveau, l'estimation sera plus ou moins précise. Ainsi, plus le modèle est proche du transistor et plus le modèle est précis. Cependant, cela implique des temps d'estimation de plus en plus longs.

La suite de cette section aborde brièvement les différentes méthodes pour obtenir des mesures ou des estimations de la consommation d'un circuit.

1.2.2.1 Mesures physiques

Il est possible de prendre des mesures directement sur le circuit lorsqu'il est produit dans un laboratoire de fabrication (fablab). La puissance du circuit est mesurée sous différents stimuli pour en définir le modèle. C'est la méthode la plus précise car ces mesures permettent de faire ensuite, des estimations basées sur un circuit qui a réellement fonctionné. Cependant, produire un circuit ASIC (Application-Specific Integrated Circuit) sur mesure dans une technologie actuelle demande un temps de développement et un coût important rendant cette option difficilement envisageable.

Les méthodes d'estimation de la consommation présentées ci-après se basent sur des modèles de consommation, qui sont établis à différents niveaux d'abstraction.

1.2.2.2 Niveau transistors

A ce niveau d'abstraction sont pris en compte les courants électriques circulant dans les transistors. Pour cela, un simulateur électrique est employé tel que SPICE [17]. Ce type de simulateur modélise les transistors du circuit en fonction de la technologie ainsi que de leurs conditions de fonctionnement comme la tension d'alimentation et de seuil. Ensuite, par l'utilisation des formules issues des lois physiques, le comportement du circuit est reproduit avec différents stimuli. Les estimations fournies à ce niveau d'abstraction sont proches de la réalité, avec une marge d'erreur de 5% [1] [14]. Les contraintes majeures à l'emploi d'un estimateur de ce niveau sont d'une part le modèle au niveau transistor du circuit qui est difficile à obtenir avec un circuit de grande taille et d'autre part, les temps de simulations très longs dus à l'émulation du comportement physique de chacun des transistors du circuit.

1.2.2.3 Niveau portes logiques

Ce niveau d'abstraction revient à définir précisément l'architecture par un assemblage de portes logiques en se servant d'une librairie technologique fournissant les caractéristiques de ces portes. Ceci est réalisé avec un langage de description de circuits (VHDL ou Verilog). Une synthèse est alors opérée et donne le coût matériel avec une estimation de la puissance dynamique et statique. La fréquence d'horloge et les tensions d'alimentation des portes sont des exemples de paramètres qui peuvent être explorés à ce niveau. Pour ce faire, il est possible d'utiliser un outil tel que PrimeTime PX [18] de la suite de logiciels Synopsys. L'inconvénient de cette méthode est que l'activité n'est pas reproduite précisément en stimulant les entrées. A la place de cela, un taux de

1. Cela dépend aussi de la qualité du modèle utilisé.

commutation moyen du circuit est demandé pour reproduire la stimulation du circuit.

1.2.2.4 Niveau architectural

Grâce à ce niveau d'abstraction plus élevé, le temps de simulation est réduit au détriment de la précision de l'estimation. Au lieu de considérer les états des portes logiques, ce sont des événements discrets ayant lieu pendant la simulation du circuit (transfert d'un paquet, routage d'un paquet etc...) qui identifient l'utilisation d'une partie du circuit, et donc sa consommation d'énergie. Cela nécessite de décomposer le circuit en sous-composants pour réduire la complexité d'estimation ; ainsi, il est possible d'estimer la puissance d'un sous-composant selon son activité avec des équations paramétriques basées sur les paramètres technologiques. Ce sont ces équations qui vont déterminer le compromis temps/précision du modèle. Ensuite, les modèles des sous-composants sont assemblés en tant que blocs élémentaires pour construire des modèles plus grands. Cela constitue la première étape de l'estimation de la consommation au niveau architectural.

La seconde étape consiste à réaliser des simulations du circuit en utilisant les modèles de puissance basés sur les événements discrets. Une fois que l'on connaît la puissance de l'utilisation des différentes parties du circuit, il est possible d'analyser la consommation au cycle d'horloge près en se basant sur une simulation cycle précis du circuit. Le comportement des composants est modélisé avec un langage de description comme SystemC [19]. Le simulateur doit assurer la synchronisation entre les différents composants du système. L'estimation des performances à ce niveau sera donnée par le simulateur d'architecture en nombre de cycles. Les simulations peuvent aussi être bit-près en plus d'être cycle-près. L'intérêt de simulations CABA (Cycle-Accurate Bit-Accurate) est qu'elles offrent les estimations les plus précises à ce niveau.

Pour conclure, ce procédé de modélisation donne des résultats moins précis que les méthodes précitées mais avec des temps d'estimations plus courts, si une simulation au niveau porte logique requiert des heures, les simulations au niveau architectural du même circuit se font en quelques secondes ou minutes. L'estimation de la consommation va maintenant être appliquée spécifiquement aux NoCs. Dans un premier temps, nous étudierons la question du choix du niveau d'abstraction, puis la modélisation du NoC sera présentée en détails.

1.3 Modélisation de la consommation du NoC

Dans la littérature de l'estimation de l'énergie consommée par les NoCs, il existe des modèles très utilisés faisant office de référence. ORION [20], ORION 2.0 [21] et DSENT [22] sont les modèles de consommation les plus utilisés

par la communauté scientifique des NoCs. Ces modèles estiment l'énergie du NoC au niveau architectural, ils définissent donc une architecture de chaque composant du réseau, à savoir : le routeur, le NA et les interconnexions. Ce niveau d'estimation a été choisi à cause de la complexité du réseau sur puce. En effet, cet assemblage de routeurs et d'interconnexions est complexe et définir son architecture au transistor près pour une estimation avec un simulateur électrique est très difficile. De plus, les temps de simulations ne seraient pas raisonnables. Dans le but de donner un ordre de grandeur, la simulation d'un bus de communication de 3 bits requiert déjà de nombreuses heures [23]. Ainsi, pour un circuit de la dimension d'un réseau sur puce, cela n'est pas possible.

Développé en 2002, ORION [20] est la première contribution dans la modélisation de la consommation des NoCs, les bases du modèle du NoC sont posées avec une attention particulière sur l'architecture du routeur. Ce modèle est amélioré pour donner ORION 2.0 [21], qui opère une mise à jour des technologies disponibles et propose d'estimer les courants de fuites que subissent le routeur et les interconnexions (répéteurs). Enfin, DSENT [22] prolonge ces travaux en permettant de modéliser les composants nécessaires à un NoC optique.

Nous allons voir dans la suite de cette section comment la modélisation de la consommation est réalisée pour chaque composant du réseau en commençant par l'adaptateur réseau, puis le routeur et enfin les interconnexions entre routeurs.

1.3.1 Modélisation de l'adaptateur réseau

L'adaptateur réseau est constitué de 2 interfaces, l'une pour le réseau et l'autre pour le processeur. Ainsi, l'architecture de ces 2 interfaces dépend d'une part des caractéristiques du réseau et d'autre part du protocole employé par le processeur. De ce fait, la modélisation de la puissance de l'adaptateur réseau n'est pas toujours précisée dans les modèles des NoCs. En effet, dès que le type d'application cible change, il faudrait à nouveau modifier le modèle du NA puisque les utilisateurs du réseau changent. Cela nécessite d'avoir un modèle différent pour chaque protocole d'IP utilisant le NoC. C'est pour cette raison que la modélisation du NA n'est pas étudiée en détail dans la modélisation de l'énergie du NoC. Il est quand même possible de trouver des propositions d'adaptateurs réseaux avec leurs modèles tel que l'adaptateur réseau réalisé par Bjerregaard [24] et conçu pour le NoC MANGO [10] (côté réseau) et le protocole OCP (côté processeur).

Le prochain composant présenté est le routeur. Son fonctionnement global est d'abord expliqué puis chaque sous-composant du routeur est décrit puis modélisé.

1.3.2 Modélisation du routeur

Les routeurs sont les nœuds du réseau qui aiguillent les paquets de données de la source jusqu'à la destination. De cette manière, les paquets transitent d'un NA à un autre par l'intermédiaire des routeurs. Le routeur est un composant complexe contenant plusieurs sous-composants remplissant différentes fonctions, c'est pourquoi il est modélisé en assemblant les modèles de ses sous-composants. La figure 1.7 montre les sous-composants d'un routeur configuré avec des buffers en entrée. Nous considérons comme routeur par défaut un routeur utilisé dans une topologie en grille 2D (2D-mesh) avec 4 entrées/sorties vers d'autres routeurs et une entrée/sortie locale (vers une interface réseau). Cette configuration est utilisée car nous avons vu en section 1.1.4 que la grille 2D était la topologie la plus répandue dans les NoCs de la littérature. Les différents composants sont listés ci-dessous :

— Buffer
— Crossbar
— Arbitre
— Routage

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 [2]. Ce flux est géré comme énoncé à la section 1.1 par un protocole du contrôle du flux du réseau. Ce mécanisme asservit le débit de l'émetteur selon les capacités de réception du destinataire, cela est réalisé de proche en proche entre les routeurs et les adaptateurs réseaux [25].

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 1.7, 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 nœud 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.

2. Ce ne sont pas des composants à proprement parler, ils représentent la frontière entre les routeurs.

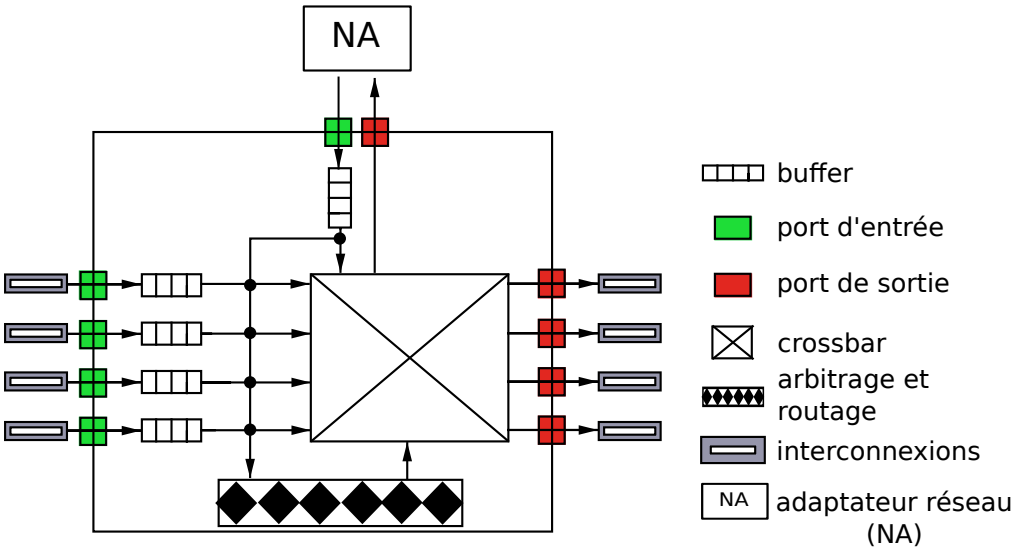


FIGURE 1.7 – Détail des composants d'un routeur avec buffers en entrée. Les interconnexions inter-routeurs et le NA sont également représentés.

Le crossbar est une matrice de commutateurs qui permet de relier n'importe quelle entrée à n'importe quelle sortie. Le crossbar peut être configuré pour faire passer plusieurs paquets venant d'entrées différentes simultanément à condition qu'ils soient orientés vers une sortie différente. 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.

Maintenant nous allons voir en détail chacun des sous-composants du routeur ainsi que la modélisation de leur consommation d'énergie.

1.3.2.1 Buffers

Les buffers sont souvent représentés par une FiFo (First in First out), rappelant le principe qui fait que les premiers paquets arrivés sont les premiers sortis. Ainsi, un buffer va être en mesure de stocker plusieurs unités de données, la granularité du stockage de données dépend de la politique de mémorisation du réseau. Il est possible d'avoir plusieurs lignes de buffers pour le même lien physique, cette configuration s'appelle des canaux virtuels (VC : Virtual Channel) [26]. Ce nom provient du fait que les différents buffers d'une même entrée (ou sortie) doivent se partager un seul lien physique ce qui nécessite de la logique supplémentaire [27]. Un exemple est montré à la figure 1.8 où 2 paquets A et B désirent prendre la même sortie. Une sélection est nécessaire pour savoir lequel de ces paquets passera le premier. Ce système est intéressant dans le cas où A

est bloqué car sa destination est occupée, le lien physique peut laisser passer le paquet B grâce aux canaux virtuels. Sans canaux virtuels, le paquet B se serait retrouvé derrière A sans pouvoir passer.

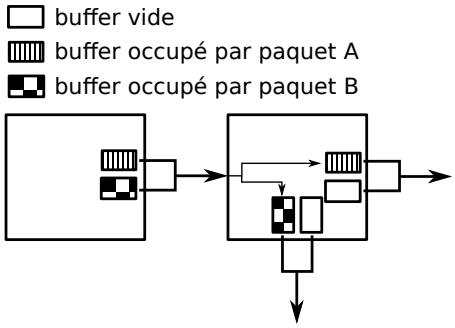


FIGURE 1.8 – Schéma de canaux virtuels illustrant le transfert de 2 paquets A et B. L'utilisation du lien physique entre les routeurs est contrôlée pour permettre le transfert paquet par paquet.

La modélisation des buffers dans ORION 1.0 [20] (et aussi dans ORION 2.0 et DSENT) est réalisée en 4 étapes :

- Sa micro-architecture est définie au plus bas niveau possible pour que le modèle du buffer soit l'union des éléments de cette micro-architecture comme illustré à la figure 1.9. L'architecture d'une cellule mémoire pour 1 bit est définie avec le câblage pour son accès en lecture ou en écriture. Ensuite, le nombre de cellules est multiplié pour avoir le stockage en mémoire désiré pour modéliser le buffer.
- Les dimensions et paramètres faisant varier la consommation énergétique sont énumérés
- Les diverses capacités des buffers sont calculées à partir des paramètres précédents nous donnant des équations paramétriques
- Enfin, les actions utilisant le buffer sont identifiées (lecture et écriture) puis l'énergie de chaque événement est déterminée par les capacités se chargeant / déchargeant pendant ladite action. Le coût énergétique d'une action est directement calculée avec une équation dérivée de (1.4) : $P = E \times f$ sachant que $f = \frac{1}{t}$, nous pouvons donc en déduire que l'énergie consommée par une action précise est

$$E = \alpha \times C \times V_{dd}^2. \tag{1.5}$$

De la même manière, l'énergie statique est déterminée pour la durée d'un cycle d'horloge (à une fréquence donnée) pour incrémenter l'énergie à chaque cycle. Ceci peut être réalisé car la micro-architecture du buffer à la porte près est connue, et que les paramètres spécifiques des transistors à une technologie donnée sont connus également. Ainsi, il est possible de déterminer les courants de fuites des buffers car ils dépendent de ces caractéristiques [21].

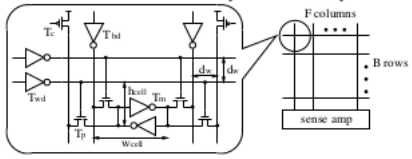
Canonical structure	
A FIFO buffer with 1 read port and 1 write port	
	
Architectural parameters	
B	buffer size in flits
F	flit size in bits
P_r	number of buffer read ports
P_w	number of buffer write ports
Technological parameters	
h_{cell}	memory cell height
w_{cell}	memory cell width
d_w	wire spacing
Capacitance equations	
wordline length	$L_{wl} = F(w_{cell}+2(P_r+P_w)d_w)$
bitline length	$L_{bl} = B(h_{cell}+(P_r+P_w)d_w)$
wordline cap.	$C_{wl} = 2FC_g(T_p)+C_a(T_{wd})+C_w(L_{wl})^\star$
read bitline cap.	$C_{br} = BC_d(T_p) + C_d(T_c) + C_w(L_{bl})$
write bitline cap.	$C_{bw} = BC_d(T_p) + C_a(T_{bd}) + C_w(L_{bl})$
precharge cap.	$C_{chg} = C_g(T_c)$
memory cell cap.	$C_{cell} = 2(P_r + P_w)C_d(T_p) + 2C_a(T_m)$
sense amp energy	E_{amp} from empirical model [28]
Operation energy equations	
δ_{bw}	number of switching write bitlines
δ_{bc}	number of switching memory cells
read energy	$E_{read} = E_{wl}+F(E_{br}+2E_{chg}+E_{amp})$
write energy	$E_{wrt} = E_{wl}+\delta_{bw}E_{bw}+\delta_{bc}E_{cell}$

FIGURE 1.9 – Modèle d'un buffer avec sa structure canonique, les différentes capacités du circuit et les opérations consommant de l'énergie [20].

1.3.2.2 Routage

Le routage est l'intelligence du routeur dont le but est d'indiquer la prochaine direction à la donnée en transit. L'algorithme de routage utilisé dépend grandement de la topologie du réseau, plus la topologie sera complexe et plus les algorithmes seront spécifiques.

Quel que soit l'algorithme de routage, tant qu'il est déployé dans le routeur, la méthode pour modéliser sa consommation reste la même. Cet algorithme doit être décrit via un langage de description matériel puis synthétisée pour avoir une estimation de la puissance et de la consommation d'énergie du routage. Une fois que cela est réalisé, les puissances statique et dynamique sont déterminées pour ensuite en déduire le coût énergétique d'utilisation de la logique de routage. Ces valeurs sont comprises dans le modèle global du routeur et incrémentent l'énergie totale lorsque la simulation aura besoin de calculer le chemin d'un paquet arrivé dans un routeur.

1.3.2.3 Arbitre

L'arbitre résout le problème de contention, qui se présente lors d'un conflit entre différentes données, qui veulent accéder à la même ressource. Cette partie du routeur gère également le contrôle du flux du routeur. Le choix de la technique d'arbitrage dépend de la politique de mémorisation, donc de la taille des tampons mémoires des composants du réseau. La technique d'arbitrage choisie doit fournir le même temps d'accès à toutes les requêtes qui ont la même priorité. Par conséquent, les conflits sont résolus en appliquant l'une des techniques suivantes :

- Arbitrage à priorité tournante (round-robin) : cette technique donne l'accès à la communication qui détient le jeton. Et à chaque cycle le jeton change de propriétaire
- Arbitrage à priorité calculée : cette technique permet d'affecter à chaque paquet un niveau de priorité en fonction de l'importance et/ou de l'urgence des données transportées. Cette priorité est stockée dans l'en-tête du paquet.
- Sans arbitrage : cette technique est utilisée lorsque les ressources sont réservées comme dans les NoCs avec une commutation de circuits où il ne peut y avoir de conflits. Il est possible de se passer d'un arbitrage lorsque le trafic qui est injecté dans le NoC est connu à l'avance. De cette manière, il est ordonnancé pour qu'aucun conflit ne se produise.

Dans les SoCs qui ont des contraintes temps réel fortes sur certains messages, il faut donner la priorité à certains messages. Par conséquent, ces configurations utilisent une priorité calculée. Dans les autres cas, où il faut maximiser le débit du réseau, c'est la priorité tournante qui est généralement utilisée.

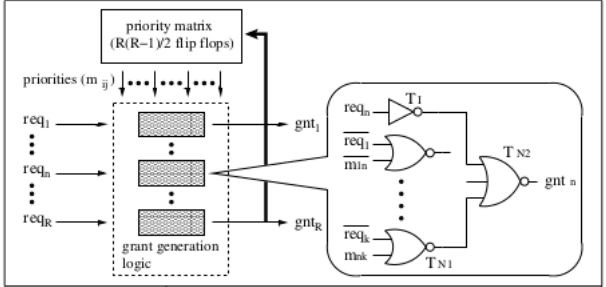
La modélisation de l'arbitre suit le même procédé que pour le buffer :

- Sa micro-architecture est définie au plus bas niveau possible pour que le modèle de l'arbitre soit l'union des éléments de cette micro-architecture. La micro-architecture est illustrée à la figure 1.10.
- Les dimensions et paramètres faisant varier la consommation énergétique sont énumérés
- Les diverses capacités de l'arbitre sont calculées à partir des paramètres précédents nous donnant des équations paramétriques
- Enfin, les actions utilisant l'arbitre sont identifiées (arbitrage) puis l'énergie de chaque événement est déterminé par les capacités se chargeant/déchargeant pendant ladite action. De même que pour le buffer, l'énergie statique est déterminée avec les paramètres technologiques [21].

1.3.2.4 Crossbar

Le crossbar est le composant du routeur reliant les entrées du routeur à ses sorties. Il est configuré par le bloc logique arbitre qui décide de la priorité des

Canonical structure and notation



C_{FF}	switch capacitance of a flip flop
C_{FC}	clock capacitance of a flip flop

Architectural parameters	
R	number of requesters

Capacitance equations	
request cap.	$C_{req} = C_a(T_I) + (R-1)C_g(T_{N1}) + C_g(T_{N2})^*$
grant cap.	$C_{gnt} = C_d(T_{N2})$
priority cap.	$C_{pri} = C_{FF} + 2C_g(T_{N1})$
internal cap.	$C_{int} = C_d(T_{N1}) + C_g(T_{N2})$
clock cap.	$C_{clk} = C_{FC}$

Operation energy equations	
δ_{ar}	number of switching request signals
δ_{ap}	number of switching priority bits
δ_{ai}	number of switching internal nodes
arbitration energy	$E_{arb} = \delta_{ar}E_{req} + \delta_{ap}E_{pri} + \delta_{ai}E_{int} + (E_{gnt} + E_{xb\_ctr})$

FIGURE 1.10 – Modèle de l'arbitre avec sa structure canonique, les différentes capacités du circuit et les opérations consommant de l'énergie [20].

entrées pour l'accès aux sorties. Il existe plusieurs méthodes pour concevoir un crossbar et la plus simple est l'utilisation de multiplexeurs. Si nous considérons un routeur de NoC avec 5 entrées/sorties dont 1 locale, le crossbar peut être représenté par le symbole à la figure 1.11.

La modélisation du crossbar reprend le principe utilisé pour l'arbitre et le buffer :

— Sa micro-architecture est définie au plus bas niveau possible pour que le modèle du crossbar soit l'union des éléments de cette micro-architecture comme illustré à la figure 1.12
— Les dimensions et paramètres faisant varier la consommation énergétique sont énumérés
— Les diverses capacités du crossbar sont calculées à partir des paramètres précédents nous donnant des équations paramétriques
— Enfin, les actions utilisant le crossbar sont identifiées (traversée) puis l'énergie de chaque événement est déterminée par les capacités se chargeant/déchargeant pendant ladite action.

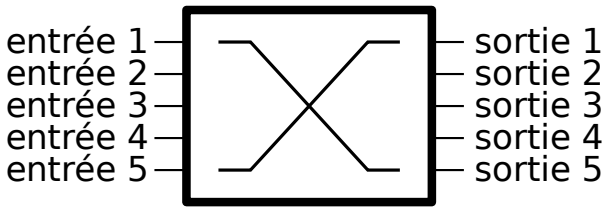
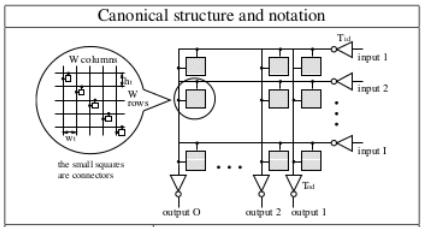


FIGURE 1.11 – Symbole d'un crossbar 5 vers 5. Il est réalisé avec des multiplexeurs 5 vers 1.



Canonical structure and notation	
connector	either a tri-state buffer or a transmission gate
C_{in_cnt}	input node cap. of a connector
C_{out_cnt}	output node cap. of a connector
C_{ctr_cnt}	control node cap. of a connector
Architectural parameters	
I	number of crossbar input ports
O	number of crossbar output ports
W	port width in bits
Technological parameters	
h_t	track height
w_t	track width
Capacitance equations	
input line length	$L_{in} = O \cdot W \cdot w_t$
output line length	$L_{out} = I \cdot W \cdot h_t$
input line cap.	$C_{xb\_in} = O \cdot C_{in\_cnt} + C_a(T_{id}) + C_w(L_{in})^*$
output line cap.	$C_{xb\_out} = I \cdot C_{out\_cnt} + C_a(T_{od}) + C_w(L_{out})$
control line cap.	$C_{xb\_ctr} = W \cdot C_{ctr\_cnt} + C_w(\frac{L_{in}}{2})$
Operation energy equations	
δ_{xi}	number of switching input bits
δ_{xo}	number of switching output bits
traversal energy	$E_{xb} = \delta_{xi} E_{xb\_in} + \delta_{xo} E_{xb\_out}$

FIGURE 1.12 – Modèle du crossbar avec sa structure canonique, les différentes capacités du circuit et les opérations consommant de l'énergie [20].

1.3.2.5 Bilan sur le routeur

Pour conclure cette section, une approche "descendante" a été employée dans la littérature pour modéliser complètement le routeur. Cela commence par une vue d'ensemble sur l'architecture du routeur et sa composition, puis nous descendons au niveau de chaque composant pour modéliser finement sa micro-architecture. Enfin, les équations paramétriques, utilisées pour calculer l'énergie, permettent d'obtenir une estimation de l'énergie consommée par le routeur. A présent, nous allons passer à la modélisation de la consommation des interconnexions du NoC.

1.3.3 Modélisation des interconnexions

Dans le but de modéliser la consommation d'énergie des interconnexions, il est nécessaire de connaître les caractéristiques physiques de ces interconnexions. Pour cela, il y a 3 grandeurs physiques qui permettent de caractériser le comportement en termes de délai et de consommation d'énergie des interconnexions :

— R, la résistance du fil
— C, la capacité du fil
— ℓ, l'inductance du fil

Ces grandeurs physiques dépendent des caractéristiques du fil (sa composition métallique : cuivre, aluminium..., la couche de métal utilisée : couche basse, haute ou centrale...) ainsi que de ses dimensions (hauteur T, largeur W et longueur L) illustrées sur la figure 1.13. Parmi ces 3 grandeurs physiques, la modélisation de l'inductance ℓ n'est pas toujours obligatoire car cette grandeur est significative dans les liens d'une puce seulement lorsqu'une des conditions suivantes est remplie. La première est que les liens subissent des pics de tension ou de courant forts [14]. La deuxième est que les liens soient extrêmement longs (parcourant par exemple toute la puce) [28]. Les courants forts peuvent se retrouver dans les interconnexions utilisées pour l'alimentation de la puce et les liens très longs dans la distribution de l'arbre d'horloge du circuit [29]. Dans le cas des interconnexions d'un NoC, les lignes ne subissent pas ou peu de fortes variations du courant. De plus, la longueur des lignes est de plus en plus petite à mesure que la taille du NoC augmente. Ainsi, la prise en compte de la résistance et de la capacité du fil est suffisante. Dans la suite de cette section, nous allons voir le modèle d'interconnexion utilisé pour les NoCs.

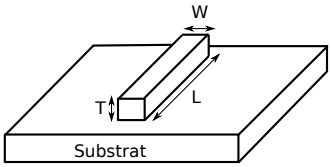


FIGURE 1.13 – Illustration des dimensions W,T et L d'un fil dans un circuit.

1.3.3.1 Modèle utilisé dans la littérature des NoCs

Les modèles d'interconnexions utilisés dans les simulateurs de systèmes complexes tels que les réseaux sur puces sont simplifiés dans le but d'avoir des temps de simulations courts. C'est le cas pour les modèles de NoC tels qu'ORION 2.0 et DSENT vus précédemment dont le modèle d'interconnexion est issu de [30].

La puissance statique dans les interconnexions est due aux inverseurs qui

y sont insérés pour limiter l'atténuation du signal. Les courants de fuites des répéteurs apparaissent quel que soient leurs états. Lorsque la sortie est à l'état haut, c'est le transistor NMOS qui fuit et lorsque l'état de sortie est bas c'est le PMOS. La proportion du courant de fuite dépend linéairement de la taille des transistors et de leurs caractéristiques, les résultats sont obtenus par des simulations sous SPICE et le modèle est ensuite généré par régression linéaire. La puissance statique des fils p_{liens} peut être calculée de la manière suivante :

$$p_{liens} = \frac{p_s^n + p_s^p}{2} \tag{1.6}$$

$$p_s^n = k_0^n + (k_1^n \times w_n) \tag{1.7}$$

$$p_s^p = k_0^p + k_1^p \times w_p \tag{1.8}$$

— p_s^n et p_s^p représentent respectivement la puissance statique des transistors NMOS et PMOS
— k_0^n, k_1^n, k_0^p et k_1^p représentent les coefficients déterminés par régression linéaire.
— w_n et w_p représentent respectivement la largeur des transistors NMOS et PMOS

Bien que la puissance statique des interconnexions ne soit pas nulle, elle est très faible comparée à la puissance dynamique des liens et à la puissance statique des autres composants du réseau. En effet, le nombre de transistors utilisé dans les interconnexions est très faible comparé aux routeurs et encore plus faible comparé aux blocs de calculs utilisant le réseau. La consommation dynamique des liens est due à la charge et la décharge des capacités de charge des fils et des inverseurs. Sachant que la puissance statique des liens est très faible, la puissance totale générée par les liens est principalement dynamique.

La puissance dynamique des liens P_{liens} est donnée par l'équation suivante qui est dérivée de (1.3) :

$$P_{liens} = \frac{1}{2} \times (C_{fils} + C_{IN}) \times V_{dd}^2 \times f \tag{1.9}$$

$$C_{fils} = C_s + C_c \tag{1.10}$$

— $\frac{1}{2}$ représente le taux de commutation α des liens qui est fixe
— C_{fils} représente la capacité totale des fils (F)
— V_{dd} représente la tension d'alimentation (V)
— f représente la fréquence du circuit (Hz)
— C_{IN} représente la capacité d'entrée des inverseurs à la fin des fils (F)
— C_s représente la capacité des fils par rapport au substrat (F)
— C_c représente la capacité de couplage des fils (F)

C_{IN} et C_{fils} sont déterminés selon la configuration des interconnexions, le nombre de répéteurs et les dimensions des liens. Ceux sont donc des valeurs

fixes dans ce modèle. Ce qui est également important de constater avec l'équation (1.9) c'est que le taux de commutation donné par $\frac{1}{2}$ est une constante assignée à tous les fils de chaque lien inter-routeur du réseau. En conclusion, pour une simulation donnée où V_{dd} et f sont fixes, P_{liens} est une constante. Ainsi, la consommation d'une interconnexion du NoC ne dépend que de son utilisation, sans être en mesure de distinguer l'activité fil par fil, ce qui rend ce modèle insensible à la nature des données traversant les liens. Cela veut dire que ce modèle considère que la consommation d'énergie d'un fil est la même si l'on ne fait, par exemple, transférer sur ce fil que des 0 ou des données commutant beaucoup. Or, il a été mentionné précédemment que c'est la commutation de ces données qui consomme de l'énergie. De plus, si l'on considère un bus de plusieurs fils, la commutation des fils de proche en proche génèrent un effet de diaphonie capacitive (C_c) qui va donc varier selon les motifs observés (nous verrons ces effets en détail dans le chapitre 2).

L'énergie consommée par tous les composants du NoC a été modélisée, la deuxième étape consiste à incorporer ces modèles dans un simulateur de NoC pour estimer la consommation d'énergie sous divers trafics et configurations. La prochaine section présente des simulateurs de NoC de la littérature incorporant des modèles de consommation d'énergie.

1.4 Simulateurs de NoC

1.4.1 Introduction

Au début de la section 1.3, nous avons évoqué le fait que le niveau d'estimation architectural est utilisé dans la littérature à cause de la complexité du réseau qui demanderait des temps de calcul trop importants à des niveaux de granularité plus fins. Nous allons maintenant passer à la deuxième étape de la modélisation du NoC au niveau architectural, qui est l'utilisation d'un simulateur de NoC pour reproduire le comportement du réseau au cycle d'horloge près. Ce simulateur est associé aux modèles de consommation pré-cités pour estimer la consommation. Nous allons commencer par présenter le fonctionnement d'un simulateur.

La figure 1.14a présente l'agencement des modèles de comportement des composants du réseau [3]. Ils sont sensibles aux événements déclenchés par leurs modèles voisins. Chaque élément est associé à un modèle de consommation illustré figure 1.14b. Lorsque le trafic dans le réseau est émulé, le simulateur de NoC peut ainsi estimer la consommation d'énergie au cours de la simulation. Le transfert d'un flit est décrit à la figure 1.14c, chaque événement est

3. Les NAs sont représentés dans ce schéma pour montrer l'arrivée et la sortie du flit du réseau bien que dans cette thèse nous ne traitions pas leur consommation pour les raisons énoncées à la section 1.3.1.

répertorié et génère donc une consommation d'énergie se basant sur le modèle de consommation des composants impliqués par l'événement.

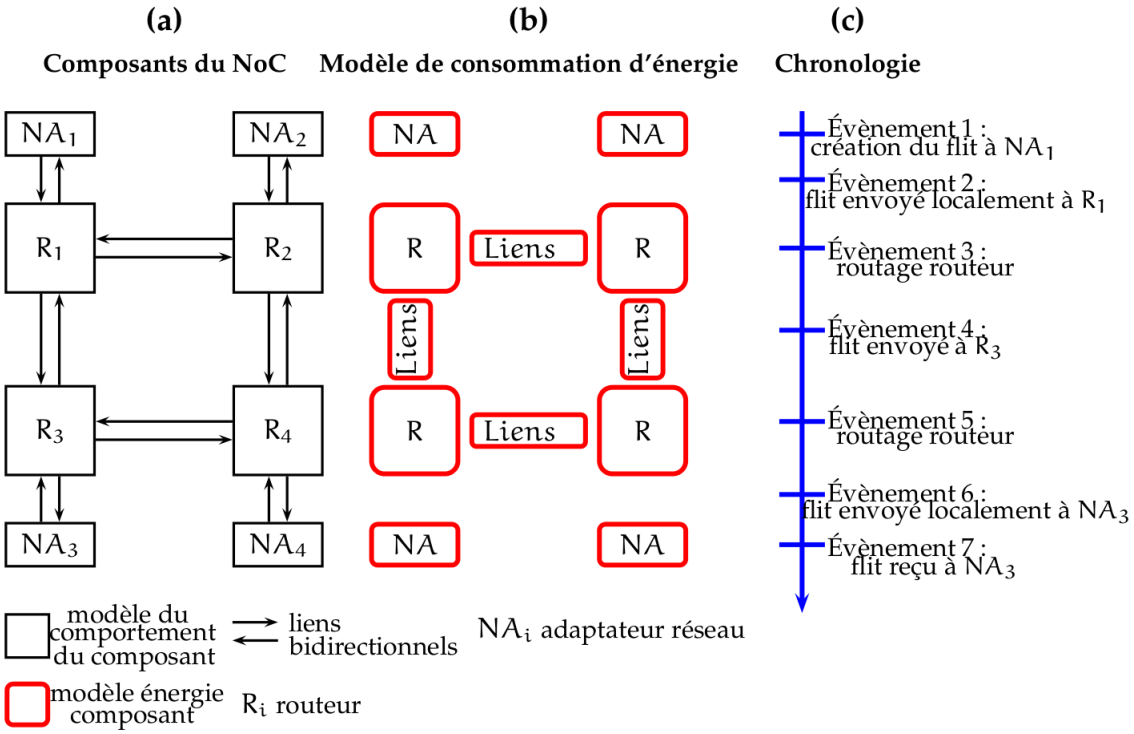


FIGURE 1.14 – Structure d'un simulateur de NoC. (a) Modèles comportementaux des composants du réseau. (b) Modèles de consommation d'énergie associés aux modèles comportementaux. (c) Chronologie du transfert d'un flit de NA_1 à NA_3.

Il existe de nombreux simulateurs de NoCs qui ont chacun leurs spécificités. Dans le cadre de cet état de l'art, nous allons nous focaliser seulement sur les simulateurs capables de mesurer la consommation d'énergie des réseaux et qui sont cycle-près afin de garantir un minimum de précision sur le comportement temporel.

La suite de cette section présente un tableau récapitulatif des simulateurs capables d'estimer la consommation d'un NoC.

1.4.2 Tableau récapitulatif

Les informations récupérées sur les différents simulateurs présentés pouvant estimer l'énergie d'un NoC sont synthétisées dans le tableau 1.2. Au total, 4 simulateurs sont sélectionnés de la littérature pour déterminer leurs points communs et savoir comment la consommation d'énergie est estimée. Les simulateurs sont les suivants :

— NIRGAM créé en SystemC par une collaboration entre l'université de Southampton (Royaume Uni) et le Malaviya National Institute of Technology (Inde).
— GARNET est développé en C++ par l'Université de Princeton
— TOPAZ est développé en C++ par l'Université de Cantabria en Espagne
— Noxim créé par l'Université de Catania en Italie et est codé en SystemC

TABLEAU 1.2 – Caractéristiques des simulateurs de NoC

Nom	**Topologies**	**Algorithmes**	**Paramètres routeurs**	**Trafics**	**Modèle de consommation d'énergie**	**Simulation CABA ?**
NIRGAM [31]	mesh-2D, torus-2D	XY, source, odd-even, Qrouting, dyad	profondeur buffer et canaux virtuels (VC)	hotspot, MPEG-4	ORION 2.0 [21]	non
GARNET [32]	fat-tree, mesh-2D, butterfly	tables de routage	profondeur buffer et VCs	aléatoire	ORION 1.0 [20]	non
TOPAZ [33]	mesh/torus-2D/3D	source, DOR, adaptatifs	profondeur buffer, VCs et plusieurs modèles de routeurs	aléatoire	Doit être ajouté par l'utilisateur	non
Noxim [34]	mesh-2D, torus-2D	XY, dyad, odd-even, routing tables	profondeur buffer	aléatoire, transposé	ORION 2.0 ou son propre modèle	non

Cette étude sur les différents simulateurs de NoC permet de tirer plusieurs conclusions. Premièrement, tous les simulateurs présentés qui ont un modèle de consommation d'énergie déjà incorporé utilisent ORION (1.0 ou 2.0), un des modèles de consommation de NoC référence avec DSENT. DSENT n'est pas utilisé par ces simulateurs car il a été développé après les simulateurs présentés ci-dessus. Ce que ces simulateurs proposent en termes de simulation et de configuration est très similaire. De plus, aucun de ces simulateurs n'est CABA [4] (voir le tableau 1.2), rendant impossible l'utilisation de modèles de consommation d'énergie considérant les données pouvant être plus précis.

Deuxièmement, il est possible de distinguer 2 groupes de simulateurs de NoC :

— Le groupe des simulateurs visant à être associés à un simulateur de plus haut niveau reproduisant un système complet. GARNET et TOPAZ se retrouvent ici.
— Le groupe des simulateurs s'utilisant seuls tels que NIRGAM et Noxim. Ils n'ont pas été conçus pour s'associer avec un simulateur. Ces simulateurs sont plus flexibles et faciles à modifier du fait qu'ils fonctionnent sans interface avec des composants externes.

Enfin, un point important pour les simulateurs est leur facilité de prise en main et d'ajout de fonctionnalités du fait qu'ils soient open source. Ceci est

4. Cycle-Accurate Bit-Accurate

particulièrement vrai avec Noxim, dont le code a été pensé pour qu'il soit modifié par l'utilisateur selon ses besoins.

1.5 Bilan

Ce chapitre a présenté la modélisation de la consommation du NoC et a exposé ses défauts. En effet, la puissance dynamique estimée par le modèle d'interconnexion de la littérature est une constante car les données ne sont pas considérées. Or, nous savons que d'après les travaux d'Antoine Courtay [23], il est essentiel de considérer les données pour établir des estimations fiables de la consommation d'énergie des bus de communication. Sachant que les liens entre les routeurs sont de simples bus comme le représente la figure 1.15, il est donc important de considérer les données pour modéliser la consommation des liens inter-routeur. Pour cela, il faut incorporer dans un simulateur de NoC un modèle d'interconnexion considérant les données.

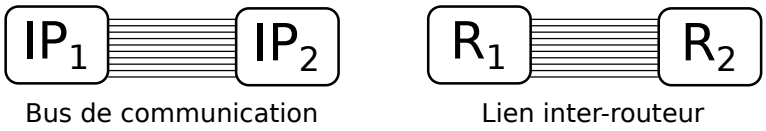


FIGURE 1.15 – Comparaison entre bus de communication et lien inter-routeur.

Parmi les simulateurs de NoC existants, nous avons pu constater que leurs propositions sont proches et ce qui les départage est principalement leur facilité d'accès et de modification. Pour ces raisons, Noxim est le simulateur choisi pour embarquer un modèle d'interconnexion plus précis. Pour tester un nouveau modèle considérant les données avec des trafics d'applications, il est nécessaire d'avoir des traces de données provenant de l'exécution d'applications réelles, ce qui n'est pas disponible actuellement dans la littérature.

Le deuxième chapitre débute ainsi avec la présentation d'un modèle d'interconnexion considérant les données. Il est ensuite implanté dans Noxim pour permettre la comparaison du modèle d'interconnexion de la littérature avec un modèle plus précis. L'extraction de traces d'application est expliquée et les autres améliorations apportées sont détaillées dans le chapitre 2.

2

Amélioration de l'estimation de la consommation d'énergie du NoC

Sommaire

Introduction

Suite à l'étude des modèles de consommation du NoC de la littérature, nous avons constaté que la modélisation des interconnexions est une constante car indépendante de la nature des données transmises. Cependant, nous pensons intuitivement que la consommation des interconnexions dépend des données transmises. Afin de vérifier cette hypothèse, nous proposons d'utiliser un modèle d'interconnexion considérant les données. Pour ce faire, il doit être intégré dans un simulateur de NoC.

Dans ce deuxième chapitre, nous allons étudier un modèle d'interconnexion considérant les données. Nous proposons donc d'incorporer le modèle d'interconnexion considérant les données dans un simulateur de NoC. Enfin, une comparaison des deux modèles d'interconnexions est effectuée.

2.1 Présentation du modèle d'interconnexions considérant les données

L'objectif de l'étude de ce modèle d'interconnexion est de déterminer l'impact qu'ont les données sur la consommation des interconnexions du NoC. Si cet impact est significatif, alors l'implantation de ce modèle dans un simulateur de réseau permettra d'améliorer l'estimation de la consommation du NoC.

2.1.1 Modélisation de la consommation des interconnexions considérant les données

Le modèle d'interconnexion présenté qui considère les données est issu des travaux d'Antoine Courtay dont la thèse était l'étude de la consommation d'énergie des bus de communication [23]. Sachant que nous avons vu à la fin du chapitre 1 avec la figure 1.15 qu'un lien entre deux routeurs peut être considéré comme un bus de communication, ce modèle peut être utilisé pour estimer la consommation des liens du NoC. L'équation du calcul de la puissance dynamique des liens au bit près est la suivante :

$$P_{liens} = \sum_{i \in N_{bit}} \alpha_i \times (C_{fil,i} + C_{in}) \times V_{dd}^2 \times f \tag{2.1}$$

— N_{bit} représente la largeur du bus, en nombre de bits
— α_i représente le taux de commutation du $i^{\text{ème}}$ fil
— $C_{fil,i}$ représente la capacité du $i^{\text{ème}}$ fil
— C_{in} représente la capacité d'entrée de l'inverseur à l'extrémité du fil

La différence de (2.1) avec (1.9) est que la puissance est déterminée pour chacun des fils au lieu d'utiliser une activité moyenne. L'intérêt réside dans le fait que $C_{fil,i}$ varie et l'activité du fil α_i également. Par conséquent, la puissance des fils d'un même bus peut varier significativement. Nous allons maintenant voir comment est modélisé un fil et quels sont les paramètres qui influent sur $C_{fil,i}$.

2.1.2 Modélisation d'un fil

Nous avons mentionné en section 1.3.3 que la modélisation des liens de communication nécessite de représenter la résistance R et la capacité C. Nous rappelons également que l'inductance ℓ peut être négligée dans le cas de liens utilisés pour la communication intra-puce car ces fils ne sont pas suffisamment longs, et ne subissent pas de fortes variations de courant. Ainsi, la modélisation d'un fil se fait avec un modèle RC.

Un modèle RC peut être construit en mettant simplement une résistance et une capacité en série mais il en ressort une modélisation peu précise. Pour plus de précision, ce modèle peut être distribué en utilisant plusieurs résistances et capacités montées en étage. En effet, dans [23], la modélisation d'une interconnexion est réalisée avec un modèle distribué $RC\Pi 3$. Il représente une interconnexion avec un circuit de 3 condensateurs et de 3 résistances comme le montre la figure 2.1. Un couple RC représente un étage. Dans la figure 2.1, nous pouvons en constater trois. Nous pouvons également remarquer la présence de 2 inverseurs pour limiter l'atténuation du signal (répéteurs).

Ce modèle est utilisé car il représente un compromis entre complexité et précision. En effet, le comportement du fil est ainsi modélisé avec une marge d'erreur inférieure à 3% que ce soit en termes de délai ou de consommation [35]. L'ajout d'autres étages RC n'apporterait que très peu de précision supplémentaire alors que les temps de simulation subiraient de fortes augmentations [35].

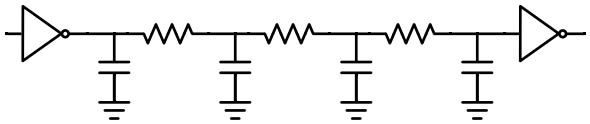


FIGURE 2.1 – Modèle $RC\Pi 3$ d'une interconnexion avec 2 répéteurs.

A partir de ce modèle, les valeurs des résistances et des capacités (dont celles requises par l'équation (2.1)) sont calculées en fonction des dimensions de chaque fil, de son état, de celui de ses voisins et des caractéristiques du matériau utilisé. Dès lors que nous avons le modèle d'un fil, il est alors possible de modéliser un bus de communication.

2.1.3 Regroupement des lignes

Afin de former un bus de communication, nous regroupons plusieurs fils. Lorsque plusieurs fils sont parallèles, il y a apparition d'un phénomène appelé diaphonie capacitive ou *crosstalk*. La diaphonie capacitive est une perturbation qui va ajouter du bruit sur le signal passant dans les interconnexions. Lorsque les fils sont parallèles et suffisamment longs, cela crée une capacité virtuelle entre chaque fil. L'intensité du crosstalk varie selon les changement d'états des fils de proche en proche. Sachant que le crosstalk modifie la valeur de la capacité des fils, ce phénomène doit être pris en compte. Ainsi, le crosstalk est illustré sur la figure 2.2 par les capacités rouges C_c entre chaque modèle $RC\Pi 3$. Les effets du crosstalk des fils au-delà des 2 plus proches voisins sont très faibles et peuvent être donc négligés [14]. C'est pourquoi la prise en compte des deux voisins du fil central suffit pour modéliser le crosstalk que peut subir un fil. Le prochain paragraphe détaille l'impact du crosstalk sur les interconnexions.

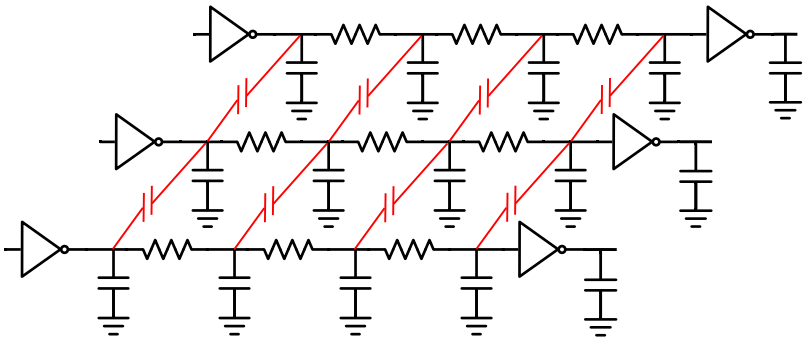


FIGURE 2.2 – Modèle complet $RC\Pi 3$ pour 3 fils avec capacités de couplage (en diagonale).

2.1.3.1 Conséquences du crosstalk sur les interconnexions

Le crosstalk a trois effets : il peut engendrer des erreurs de transmission, impacter le délai de transmission et enfin, peut générer une surconsommation des liens.

Source d'erreurs : Du bruit peut être généré avec le crosstalk, lorsqu'un fil commute, la ligne voisine est en effet perturbée. Nous définissons la ligne victime (la ligne perturbée) et l'agresseur (la ligne perturbatrice). La figure 2.3 montre les différents cas de bruits pouvant se produire sur un fil victime du couplage capacitif. Les figures 2.3a,b montrent un bruit positif lorsque l'agresseur effectue une transition montante. Ce bruit est représenté par la perturbation augmentant temporairement la tension du fil victime alors que sans perturbation, la tension du fil devrait rester constante, comme montré en pointillé. Les

figures 2.3c,d quant à elles montrent un bruit négatif causé par la transition descendante de la ligne agresseur.

Le bruit provoqué par ces pics ou chutes de tensions peut fausser le signal. Si ce pic est suffisamment important pour faire avoisiner la tension du fil victime de la tension de seuil comme dans le cas des figures 2.3a,d, le signal peut être alors mal interprété par le récepteur et causer une erreur.

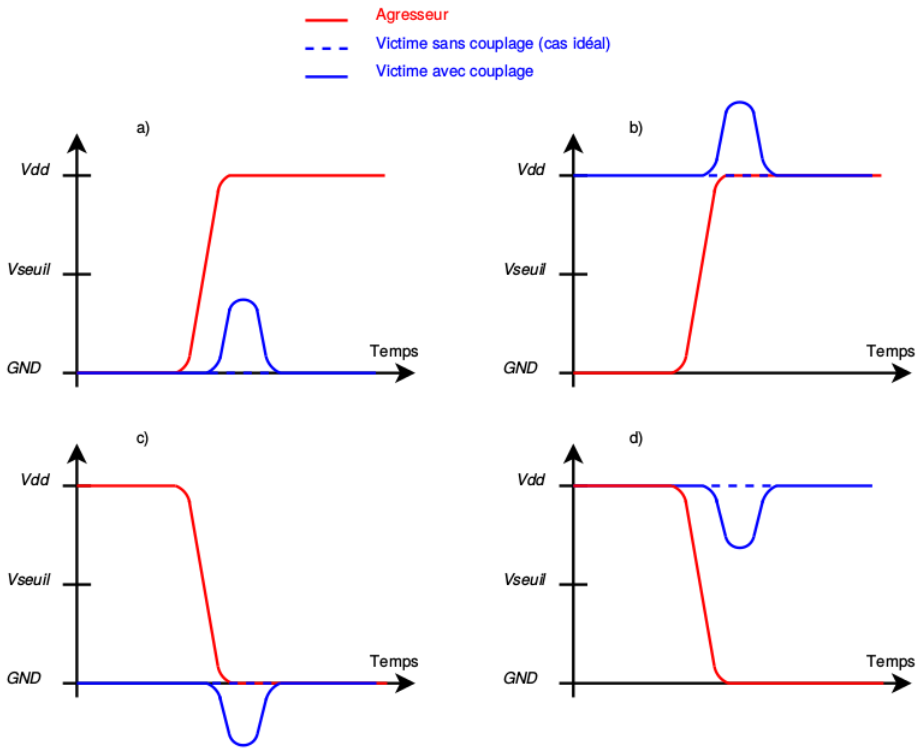


FIGURE 2.3 – Bruits potentiels sur le fil victime (avec transition) [23].

Retard de transmission : La partie précédente a montré qu'une transition sur un fil affecte son voisin en créant un pic de tension sur celui-ci. Lors de transitions simultanées sur le fil victime et agresseur, un pic de tension sera également généré ; ce pic peut selon la configuration des transitions légèrement accélérer ou considérablement ralentir la propagation de la donnée sur le fil victime (figure 2.4). Il existe quatre cas de couplage possibles.

— Sur la figure 2.4a, l'agresseur et la victime effectuent des transitions montantes. Le cas sans couplage représenté en pointillés montre que les transitions se font identiquement. Avec couplage, la transition sur le fil agresseur va générer sur le fil victime un pic de tension dans le sens de la transition ; les transitions sur le fil victime et l'agresseur vont en quelque sorte s'accélérer mutuellement.

— Sur la figure 2.4b, l'agresseur effectue une transition montante alors que la victime effectue une transition descendante. Avec couplage, la transition sur le fil agresseur va générer sur le fil victime un pic de tension dans le sens inverse de la transition du fil ; la transition sur le fil victime va donc être ralentie.
— Sur la figure 2.4c, l'agresseur et la victime effectuent des transitions descendantes. Avec couplage, la transition sur le fil agresseur va générer sur le fil victime un pic de tension dans le sens de la transition ; les transitions sur le fil victime et agresseur vont en quelque sorte s'accélérer mutuellement.
— Sur la figure 2.4d, l'agresseur effectue une transition descendante alors que la victime effectue une transition montante. Avec couplage, la transition sur le fil agresseur va générer sur le fil victime un pic de tension dans le sens inverse de la transition du fil ; la transition sur le fil victime va donc être ralentie.

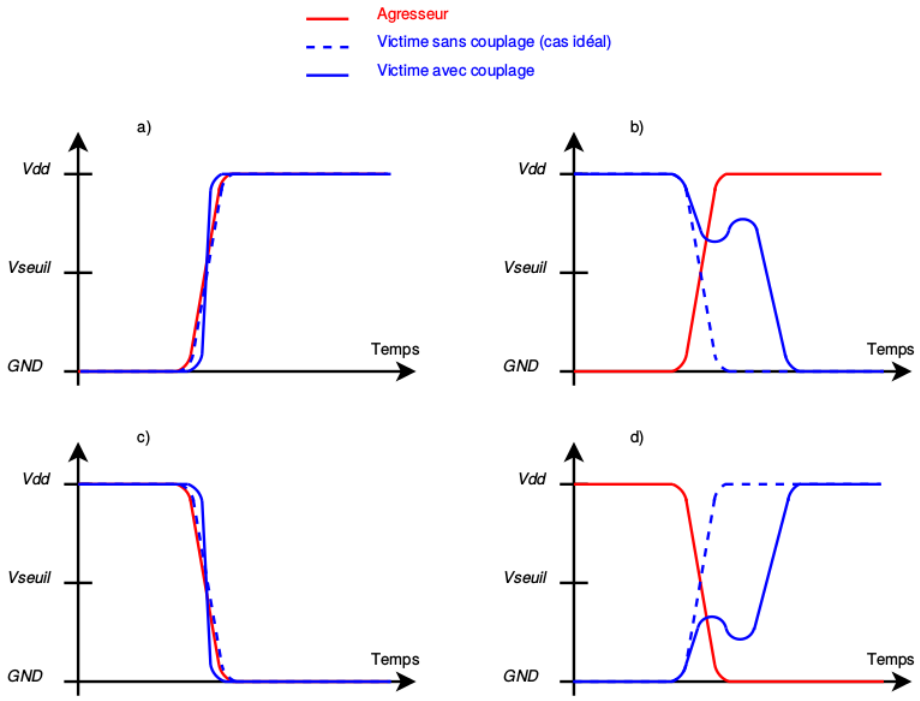


FIGURE 2.4 – Retards potentiels sur le fil victime (avec transition) [23].

Un fil victime du crosstalk peut avoir jusqu'à deux agresseurs (un seul s'il se trouve à une extrémité du bus) qui sont les fils voisins les plus proches. Il est donc possible ici d'effectuer un classement des types de transitions selon la rapidité de propagation des transitions.

Trois symboles sont définis pour identifier les états d'un fil : $\downarrow$ est une transition descendante (de 1 vers 0), $\uparrow$ en représente une montante (de 0 vers 1) et un $-$ est une absence de transition.

Prenons le cas de trois fils qui commutent dans la même direction (↑,↑,↑ ou ↓,↓,↓). La transition sur le fil victime (c'est-à-dire le fil central) se verra accélérée par les perturbations de ses deux agresseurs. Maintenant, si un de ses agresseurs ne transite pas (−,↑,↑ ou −,↓,↓), la transition sur le fil victime va se voir accélérée par la perturbation d'un seul de ses agresseurs et sera donc moins rapide que dans le cas précédent.

Prenons maintenant le cas de trois fils où le fil victime et l'un de ses agresseurs commutent dans la même direction tandis que son autre agresseur commute dans le sens inverse (↑,↑,↓ ou ↓,↓,↑). Un de ses agresseurs va accélérer la transition et l'autre la ralentir. Maintenant, si l'agresseur qui commute dans le même sens que la victime, ne transite pas (−,↑,↓ ou −,↓,↑), la transition sur le fil victime va se voir ralentir par la perturbation d'un seul de ses agresseurs et donc la transition sera moins rapide que dans le cas précédent.

Poursuivons ce raisonnement en prenant le cas où les agresseurs commutent dans le sens inverse de la victime (↓,↑,↓ ou ↑,↓,↑). Ici, la victime va se voir perturbée par ses deux agresseurs et la transition sera encore plus lente que dans le cas précédent ; ceci représente le pire cas pour le temps de propagation sur le bus.

Avec ces différents cas, nous pouvons effectuer un classement des transitions selon le temps de propagation sur le fil victime tel que le montre le tableau 2.1. Le paramètre g représente le facteur de retard, r le rapport entre la capacité induite par le crosstalk C_c et la capacité de fil au substrat C_s.

TABLEAU 2.1 – Capacité $C_{fil,i}$ et facteur de délai g du fil victime i selon les transitions

$C_{fil,i}$	**Types de transitions**				g
C_s	(↓,↓,↓)	(↑,↑,↑)			1
$C_s + C_c$	(−,↓,↓)	(↓,↓,−)	(−,↑,↑)	(↑,↑,−)	$1 + r$
$C_s + 2C_c$	(−,↓,−) (↓,↓,↑)	(−,↑,−) (↑,↑,↓)	(↓,↑,↑)	(↑,↓,↓)	$1 + 2r$
$C_s + 3C_c$	(−,↑,↓)	(↓,↑,−)	(−,↓,↑)	(↑,↓,−)	$1 + 3r$
$C_s + 4C_c$	(↓,↑,↓)	(↑,↓,↑)			$1 + 4r$

Afin de vérifier la tendance de l'augmentation du délai dû aux interconnexions avec la technologie, le tableau 2.2 représente l'évolution du paramètre r sur les différentes couches de métal dans trois technologies (130, 90 et 65 nm). Il peut être remarqué dans le tableau 2.2 que r a tendance à augmenter lors d'un changement technologique et donc le crosstalk aussi. La hausse de ces paramètres est due au fait que les interconnexions sont de plus en plus rapprochées et de plus en plus hautes avec les technologies plus fines [36]. En effet, le crosstalk est une capacité virtuelle entre deux fils car ceux-ci ont deux surfaces en regard, comme une capacité. Ainsi, si nous rapprochons ces fils, alors le crosstalk va perturber plus fortement les interconnexions et si la surface de regard des fils entre eux augmente également, alors la capacité de couplage augmente aussi comme le montre la figure 2.5. Outre les problèmes

de bruit et de retard, le crosstalk est également à l'origine de l'augmentation de la consommation.

TABLEAU 2.2 – Évolution du paramètre r en fonction de la technologie et de la couche de métal.

	r_{130nm}	r_{90nm}	r_{65nm}
M1	0.92	0.8	0.8
M2	1.84	1.84	1.79
M3	3.06	3.34	3.42
M4	4.29	4.91	5.06
M5	2.70	6.44	6.70
M6	3.8	3.23	8.33
M7		4.47	3.13
M8			4.44

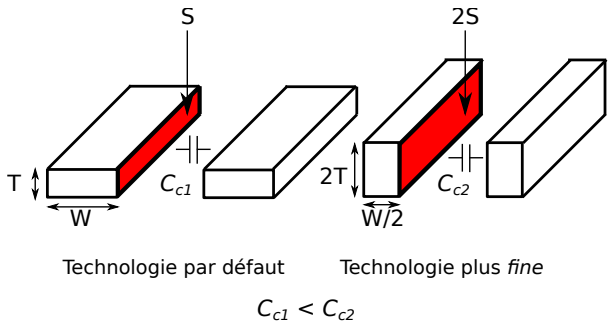


FIGURE 2.5 – Illustration des conséquences de l'évolution des dimensions des fils avec la technologie.

Surconsommation : Nous avons vu avec l'équation (2.1) que la puissance dynamique d'un fil dépend linéairement de C_c. Par conséquent, plus la capacité de couplage induite par le crosstalk sera grande et plus la consommation sera importante.

Dans le but de quantifier l'impact sur la consommation d'énergie des interconnexions du crosstalk, des simulations électriques ont été menées dans [37] en utilisant le modèle de bus composé de 3 fils représentés avec le modèle $RC\Pi 3$ illustré à la figure 2.2. Ce modèle de bus est injecté dans SPICE pour effectuer des simulations de ce circuit. Ces simulations permettent aussi de vérifier si le classement énergétique est le même que le classement temporel donné dans le tableau 2.1,

Le tableau 2.3 présente la classification temporelle et énergétique des transitions. Ces résultats ont été produits à l'aide de simulations qui ont été effectuées avec les paramètres suivants : technologie 65 nm , longueur de fil 1

TABLEAU 2.3 – Classement énergétique et temporel des transitions du tableau 2.1

$C_{fil,i}$	**Type de transition**	**Coût Temporel (ps)**	**Coût Énergétique (fJ)**
0	(−,−,−)	0	0.21
C_s	(↑,↑,↑)	49	13.29
C_s	(↓,↓,↓)	49	33.77
$C_s + C_c$	(−,↑,↑)	67	13.43
$C_s + C_c$	(−,↓,↓)	67	92
$C_s + 2C_c$	(↑,↑,↓)	99	13.89
$C_s + 2C_c$	(−,↑,−)	99	13.45
$C_s + 2C_c$	(−,↓,−)	99	150.35
$C_s + 2C_c$	(↓,↓,↑)	99	150.73
$C_s + 3C_c$	(−,↑,↓)	139	14.1
$C_s + 3C_c$	(−,↓,↑)	139	207.76
$C_s + 4C_c$	(↓,↑,↓)	173	14.86
$C_s + 4C_c$	(↑,↓,↑)	173	265.07

mm et sur une couche métallique intermédiaire (couche utilisée pour les bus de communication), sachant que cette classification est la même dans toutes les configurations.

Les résultats du tableau 2.3 montrent que la classification temporelle en fonction de l'importance de la capacité présentée par le fil victime est la même que celle présentée dans le tableau 2.1. Cependant, il est important de remarquer que le classement des transitions du point de vue de la consommation n'est pas le même que le classement des transitions du point de vue de l'impact temporel. Le classement en consommation du tableau 2.3 peut s'expliquer en divisant les types de transitions en deux parties :

— les transitions montantes sont les transitions les moins coûteuses ; elles sont ordonnées de la plus faible capacité à la plus élevée.
— les transitions descendantes sont les transitions les plus coûteuses ; elles sont ordonnées de la même manière.

Il est important de noter que la consommation des transitions montantes est similaire, alors que pour les transitions descendantes, la consommation augmente avec le crosstalk. Afin de comprendre pourquoi les transitions descendantes consomment plus que les montantes, il est nécessaire de savoir quand le courant consommé provient de l'alimentation. En fonction du type de transition (montante ou descendante), la capacité présentée par la ligne est chargée ou non. Lors d'une transition montante, le transistor NMOS de l'inverseur est actif et donc la capacité présentée par la ligne n'est pas chargée par l'alimentation puisque la sortie est inversée. Lors d'une transition descendante, le

transistor PMOS est actif donc la capacité présentée par la ligne est chargée par le courant extrait de l'alimentation.

Pour conclure, ce modèle nous montre que l'activité de chacun des fils peut faire énormément varier la consommation d'énergie des interconnexions du fait du crosstalk (jusqu'à un facteur 20 entre les deux transitions respectivement la moins et la plus consommatrice) . En ne prenant pas en compte le crosstalk, les modèles de la littérature des NoCs sont incapables de modéliser le comportement réel du fil d'une interconnexion et peuvent donc sous ou sur estimer la consommation d'énergie des liens. Afin de quantifier l'imprécision potentielle du modèle utilisé dans l'état de l'art, nous avons réalisé une étude comparant le modèle avec et sans prise en compte du crosstalk. Pour être en mesure de réaliser cette étude, nous avons dans un premier temps modifié un simulateur de NoC, Noxim.

2.2 Préparation du simulateur

Plusieurs modifications sont apportées à Noxim et notre version modifiée est nommée Noxim-XT [1]. Le flot d'estimation de Noxim-XT est résumé à la figure 2.6 illustrant aussi les changements apportés au simulateur ainsi que les différentes entrées nécessaires. Il peut être observé que Noxim-XT peut prendre en entrée des fichiers de traces d'applications, l'extraction de ces traces est d'abord détaillée avant d'aborder les modifications du simulateur.

2.2.1 Extraction de traces d'applications

2.2.1.1 Extraction de traces

L'extraction de traces d'applications réelles contenant le trafic et les données des paquets permet d'avoir les informations suffisantes pour reproduire le trafic dans Noxim-XT au bit près. Ainsi, au lieu d'avoir seulement accès aux trafics synthétiques dont le comportement n'est pas représentatif de la réalité, nous pouvons reproduire le trafic issu d'une application. Les trafics issus d'applications disponibles dans la littérature ne font que reproduire le trafic sans leur contenu, la méthode d'extraction des traces est décrite dans les paragraphes suivants.

Afin de réaliser une trace du trafic avec les données d'une application, il est nécessaire que l'application soit exécutée sur un SoC utilisant un NoC. La littérature dans le domaine des simulations d'applications dans un SoC est divisée en deux parties. Premièrement, il existe des suites de benchmarks tels que SPLASH-2[39] et ParMiBench [40] qui contiennent un ensemble d'appli-

1. Une version bêta est disponible sur github en tant que branche du Noxim original [38].

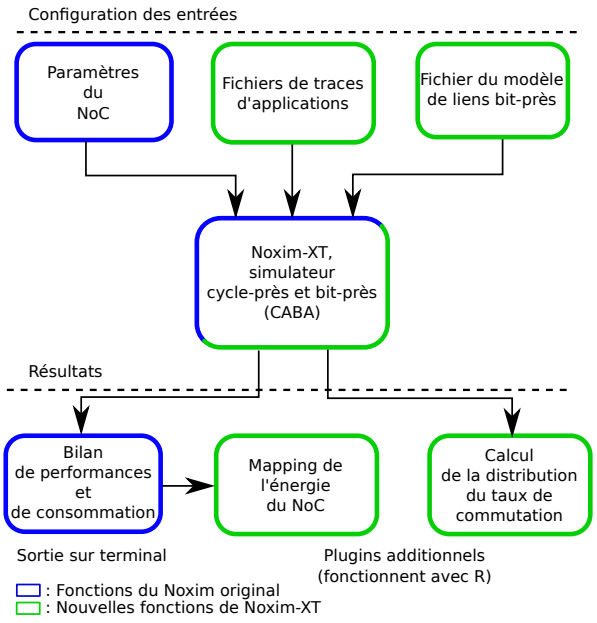


FIGURE 2.6 – Flot de l'analyse de la consommation d'énergie de Noxim-XT représentant son fonctionnement.

cations parallèles. SPLASH-2 a été conçu en 1995, et bien qu'il commence à dater comparé aux autres benchmarks, il est toujours utilisé dans la littérature comme d'autres benchmarks tels que ParMiBench et PARSEC [41]. PARSEC est un benchmark simulant les mémoires partagées et la synchronisation des threads d'une application. Parmi les benchmarks présentés, aucun d'eux ne considère les composants matériels pour l'évaluation du design, c'est pourquoi ces suites d'applications sont associées avec des plate-formes de simulation de SoC pour permettre une meilleure analyse.

La seconde partie de cette littérature concerne les plate-formes de simulation de SoC. SoCLib [42] est un simulateur d'architecture qui permet à l'utilisateur de personnaliser sa configuration matérielle. SoCLib contient une librairie en SystemC des composants virtuels. Gem5 [43] est une plate-forme dite "full system" donnant la possibilité de simuler un système complet comprenant une application gérée par un système d'exploitation (OS), le tout sur une architecture définie préalablement. Le dernier outil présenté est MPSoCBench [44], une plate-forme de simulation développée en 2014 par les chercheurs de l'université de Campinas proposant un simulateur de MPSoC associé directement à des suites d'applications telles que SPLASH-2 et ParMiBench. Cela permet à l'utilisateur de configurer directement l'architecture voulue avec l'application désirée. Étant donné que MPSoCBench associe l'exécution d'une application d'un benchmark à une architecture, il s'agit du meilleur choix pour extraire le trafic et les données d'applications.

La figure 2.7 montre un exemple de la configuration d'une architecture de SoC exécutant une application. Dès qu'un nouveau paquet est injecté dans le NoC, un événement apparaît et les informations du paquet sont enregistrées dans un fichier global de traces. Chaque ligne de ce fichier correspond à un paquet transféré durant la simulation de l'application. Le temps (en ns) du premier flit relâché, le couple source/destination de ce paquet et le contenu de chaque flit contenant les données utiles à l'application sont enregistrées. Quand l'exécution de l'application est terminée, le fichier de trace global est découpé selon l'origine de chaque paquet, ainsi, il y a un fichier de trace pour chaque IP qui a envoyé un paquet dans le NoC pendant la simulation. Cette méthode permet d'extraire les traces de diverses applications.

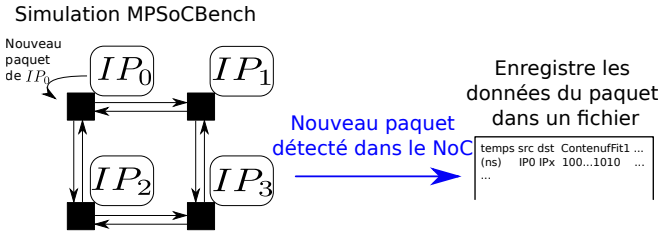


FIGURE 2.7 – Extraction des traces d'applications issues des simulations de MPSoCBench.

2.2.1.2 Applications retenues pour l'extraction de traces

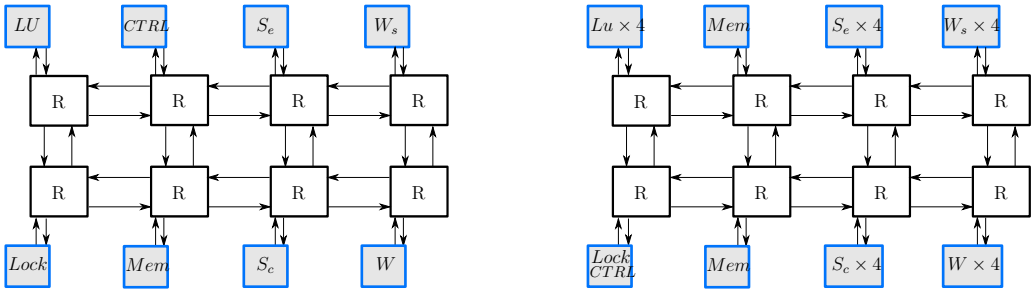
Afin d'effectuer des simulations avec des trafics d'applications, plusieurs ensembles de traces ont été extraits. Leurs noms sont les suivants : Dijkstra, Basicmath, 5 applications, clusters applications et Full HD. La description de Dijkstra et Basicmath peut être trouvée dans le tableau 2.4. L'application nommée 5 applications utilise les applications suivantes exécutées en même temps : LU, susan-corners, susan-edges, water et water-spatial. La description de ces applications est disponible dans le tableau 2.4. Chaque application est exécutée par un processeur associé avec un routeur du NoC. Ils utilisent tous la même mémoire partagée. Le mapping de 5 applications et clusters applications est présenté sur la figure 2.8. L'IP CTRL lance l'application tandis que Lock permet le contrôle de l'accès à la mémoire partagée Mem.

Clusters applications utilise quatre cœurs par routeur en parallèle pour exécuter chaque application (les mêmes que 5 applications) et génère un débit dans le NoC plus important. Il y a deux mémoires partagées dans le NoC pour éviter la congestion.

Parmi les applications disponibles dans les benchmarks de MPSoCBench, il n'y a pas d'application vidéo, donc nous avons développé une application nommée Full HD afin de compléter la suite de benchmarks. Full HD génère

TABLEAU 2.4 – Description des applications

Application	Benchmark	Description
Dijkstra	ParMiBench [40]	Calcul le chemin le plus court de toutes les paires de sommets dans un graphe représenté par une matrice d'adjacence, la stratégie de décomposition est utilisée de manière à ce que chaque processeur s'occupe d'un sommet pour obtenir le plus court chemin selon le sommet d'origine.
Basicmath	ParMiBench [40]	Exécute des calculs mathématiques tels que le calcul de fonctions cubiques, la conversion d'angles du degré au radian ou la racine carré d'entiers. La parallélisation est réalisée avec le partitionnement des données.
LU	SPLASH-2 [39]	Calculs matriciels sur de grandes matrices.
Susan-corners (Sc)	ParMiBench [40]	Permet de faire la détection des coins d'une image, méthode de traitement d'image utilisée dans le domaine médical (IRM).
Susan-edges (Se)	ParMiBench [40]	Détection de bords d'une image, méthode de traitement d'image utilisée dans le domaine médical (IRM).
Water (W)	SPLASH-2 [39]	Modélise le comportement d'un système de molécules d'eau au cours du temps.
Water-spatial (Ws)	SPLASH-2 [39]	Fait la même chose que l'application Water mais avec un algorithme différent.



(a) 5 applications (b) Clusters applications

FIGURE 2.8 – Mapping de deux groupes d'applications de MPSoCBench dans un NoC grille 2D 4×2. Voir le tableau 2.4 pour les acronymes des applications.

un flux vidéo HD qui est traité par six IPs. Ces IPs fabriquent chacune un rectangle se déplaçant dans l'image avec à l'intérieur un traitement différent. L'application Full HD a été testée et validée sur FPGA (Field Programmable

Gate Array) avec une architecture utilisant un NoC permettant également l'extraction du trafic et des données.

Une fois que ces fichiers de traces sont réalisés, ils peuvent être utilisés dans Noxim-XT pour reproduire un trafic d'application au bit près.

2.2.2 Modifications de Noxim-XT

2.2.2.1 Insertion du modèle d'interconnexion tenant compte du crosstalk

Utiliser le modèle d'interconnexion présenté en section 2.1 nécessite d'avoir un simulateur bit-près en plus d'être cycle-près (CABA). Pour ce faire, il faut connaître les bits de données transférés par les flits pour déterminer les types de transitions et déterminer l'impact du crosstalk. Le contenu des flits est ajouté dans la classe représentant les flits.

Lorsque le contenu des flits est accessible, chaque routeur doit mémoriser le dernier flit qui a traversé chacun de ses liens vers les autres routeurs. De cette manière, avec l'arrivée d'un nouveau flit passant à travers un lien inter-routeur, les transitions qui ont lieu peuvent être identifiées puisque l'on connaît l'état précédent et l'état courant de chaque fil. Un vecteur de transition est alors créé, il s'agit d'un vecteur comprenant toutes les transitions qui ont lieu entre deux flits comme le montre l'étape 2 de la figure 2.9. Ensuite, les différents motifs de transitions sont identifiés pour chaque fil selon leurs voisins respectifs. Dans le cas particulier des fils situés aux extrémités du bus, nous considérons qu'ils ont un second voisin agresseur n'effectuant pas de transition, en plus de leur unique voisin réel. Une fois que toutes les transitions ont été identifiées, la consommation d'énergie du lien inter-routeur est incrémentée selon le coût de toutes les transitions répertoriées. Cette méthodologie nous donne la consommation d'énergie locale d'un lien inter-routeur et est appliquée à toutes les connexions entre les routeurs. Pour compléter l'insertion du modèle de liens bit-près, lors de trafics synthétiques, le contenu des flits est généré aléatoirement pour alimenter le modèle considérant le crosstalk. D'autres modes de génération de flits synthétiques seront présentés en section 2.3.1.

2.2.2.2 Simuler un trafic basé sur des applications réelles au bit-près

Dans Noxim-XT, le Processing Element (PE) est le composant qui injecte les paquets dans le réseau. Chaque PE a été adapté afin de lire les fichiers de traces générés précédemment en section 2.2.1. Les PEs ont une adresse qui est associée à celle de l'IP de manière à ce que le PE_0 puisse seulement lire le fichier de trace de l'IP_0 et ainsi de suite, comme illustré à la figure 2.10. Les PEs envoient un paquet à l'intérieur du NoC seulement lorsque le temps de

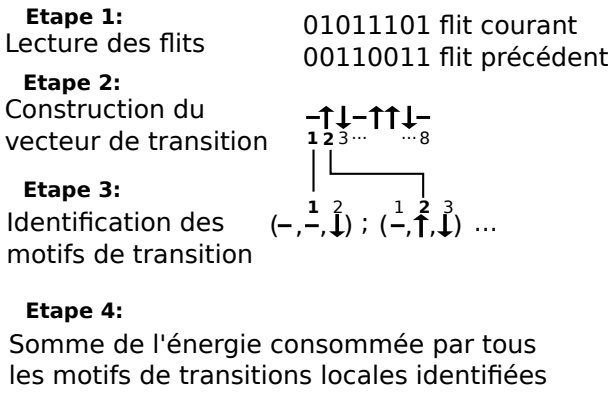


FIGURE 2.9 – Méthodologie pour identifier le taux de commutation courant des interconnexions et les motifs de transition.

simulation atteint le temps d'envoi enregistré dans le fichier de trace généré lors de la simulation MPSoCBench. Ainsi, si la configuration du NoC dans Noxim n'est pas en mesure de supporter le trafic généré par l'application, les paquets sont envoyés plus tard que prévu par le fichier de traces. Par conséquent, les résultats de simulations de Noxim indiqueront une hausse du délai moyen ce qui nous permettra de détecter les aléas temporels.

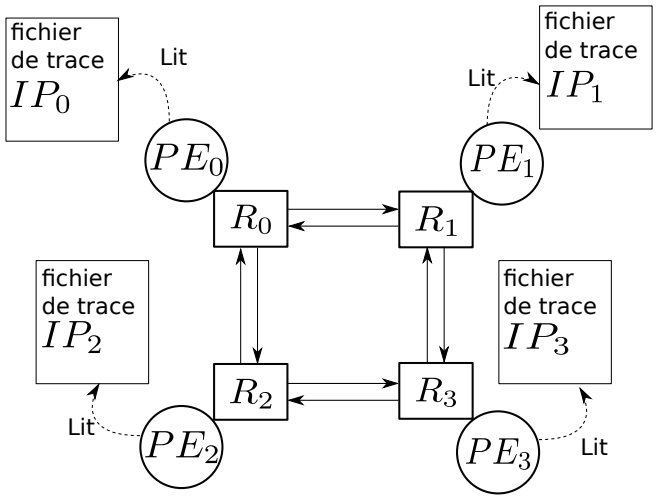


FIGURE 2.10 – Utilisation des fichiers de traces d'applications dans Noxim-XT. Chaque Processing Element (PE) lit un fichier correspondant à son IP respective dans le but de reproduire le trafic de l'application. Ensuite, les PEs injectent dans le réseau les paquets issus de la lecture du fichier de traces.

Les limitations de cette technique de reproduction de trafics d'applications sont les suivantes :

— Dès que l'on souhaîte changer de configuration, nous devons générer de nouveau en ensemble de traces pour la même application. Il en est de même pour l'exécution d'une application avec des entrées différentes.
— La dépendance des messages est garantie seulement si le NoC testé a une configuration suffisante en termes de débit et de délai pour transmettre les données issues des fichiers de traces.

2.2.2.3 Distribution du taux de commutation

Nous avons également ajouté l'option `-alpha_trace` calculant le taux de commutation à chaque fois qu'un flit traverse une interconnexion, ces données sont ensuite enregistrées dans un fichier. Une fois le fichier généré, Noxim-XT est associé avec un script écrit en R [45] générant un graphique tel que celui présent à la figure 2.11. Cet histogramme indique le pourcentage d'occurrences des différentes valeurs du taux de commutation prises pendant la simulation.

Cet outil nous permettra de montrer la véracité d'un taux de commutation moyen de 0.5 pris usuellement par les différents simulateurs.

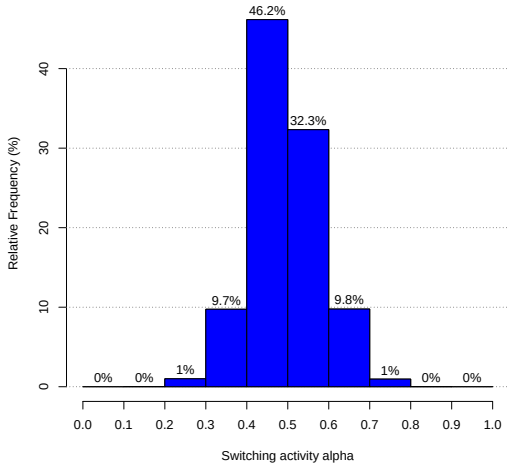


FIGURE 2.11 – Histogramme de la distribution du taux de commutation pendant la simulation d'un trafic uniforme aléatoire avec contenu des données aléatoire.

2.2.2.4 Mapping énergétique du NoC

Dans le but de compléter les améliorations apportées à l'analyse de la consommation d'énergie du NoC, nous associons Noxim-XT avec R pour générer un mapping énergétique du NoC après une simulation. Cette fonction ajoute à

Noxim-XT un outil supplémentaire pouvant être utile pour alimenter des modèles thermiques tels que Hotspot [46].

La figure 2.12 montre un exemple de mapping énergétique avec l'application Dijkstra dans un NoC en grille 2D 4×3. La métrique est l'énergie moyenne par cycle d'horloge pour chaque composant ; les carrés représentent les routeurs, et les rectangles les liens.

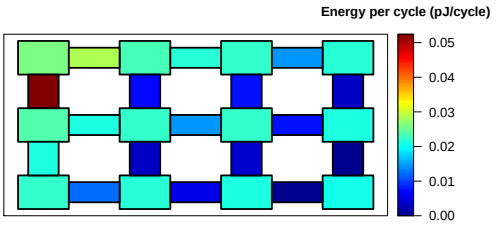


FIGURE 2.12 – Mapping énergétique de l'application Dijkstra dans un NoC.

A partir de ces améliorations apportées au simulateur, nous allons pouvoir réaliser l'étude de la précision du modèle d'interconnexion de la littérature des NoCs en le comparant au modèle de la section 2.1 prenant en compte les données.

2.3 Évaluation du modèle d'interconnexions des NoCs

Dans le but de comparer les deux modèles d'interconnexions, ils sont évalués par différentes études utilisant Noxim-XT :

— une étude de la précision
— une étude du surcoût temporel
— une étude du poids des interconnexions dans la consommation du NoC
— une étude des trafics d'applications.

Les études proposées ci-dessus exploitent les résultats de simulations réalisées avec des trafics synthétiques et d'applications.

2.3.1 Analyse des résultats

2.3.1.1 Configuration du simulateur pour trafics synthétiques

Les modèles de consommation du routeur et des interconnexions utilisés sont ceux de Noxim. Ces modèles de consommation ont été synthétisés et mappés dans la librairie technologique CMOS Bulk d'UMC à 65 nm, la même technologie utilisée par le modèle d'interconnexion tenant compte du crosstalk

présenté en section 2.1. Étant donné que le trafic est synthétique, il n'y a pas de réelles IP utilisant le NoC, cette étude de la consommation considèrera donc la consommation d'énergie d'un NA générique issu de XPIPES [7]. Ceci ne sera réalisé que pour cet ensemble de simulations dans le but de vérifier si la consommation du NA est significative ou non.

Les expérimentations sont exécutées avec un NoC pour une topologie grille 2D 4×4 cadencé à 700 MHz avec un algorithme de routage déterministe XY. Les autres paramètres sont des flits de 32 bits de largeur, une profondeur de buffer à chaque entrée de 4 flits, la taille des paquets est de 8 flits et la longueur des interconnexions entre les routeurs est de 3 mm. L'origine et la destination des paquets sont sélectionnées aléatoirement avec un trafic uniforme aléatoire. Les simulations sont exécutées pendant 100 000 cycles avec différents taux d'injection de paquets (PIR, Packet Injection Rate), ce qui nous donne en moyenne entre 50 ko de données injectées dans le réseau pour le PIR le plus faible et jusqu'à 1.5 Mo pour le plus élevé. Le PIR est le taux d'injection de paquets dans le réseau, il s'exprime en nombre de paquets émis par routeur au cours d'un cycle, il est compris entre zéro et un. Chaque configuration de simulation est répétée 20 fois du fait de l'aspect aléatoire du trafic. Les résultats présentés sont donc une moyenne de ces 20 itérations.

2.3.1.2 Étude de la précision

Pour déterminer la précision du modèle indépendant des données, deux principaux types de motifs de données sont utilisés pour alimenter le contenu des flits, illustrés dans le tableau 2.5. Ces motifs de données sont créés par la succession systématique de deux flits nous donnant un taux de commutation connus dans le réseau pendant la simulation.

Le premier type de motifs de données a pour but de générer la consommation d'énergie la plus faible en fonction du taux de commutation α, il est le type nommé Meilleur cas (du point de vue de l'énergie). Pour accomplir cet objectif, il faut que la succession systématique de deux données provoque des transitions avec l'énergie la plus faible possible. D'après le tableau 2.3, la transition montante la moins coûteuse en énergie est ($\uparrow$, $\uparrow$, $\uparrow$) tandis que pour les transitions descendantes, ($\downarrow$, $\downarrow$, $\downarrow$) est la moins coûteuse. Ainsi, si l'on considère un bus de 4 bits, alors les deux données se succédant sont 0000 et 1111 et ainsi de suite.

Le deuxième type de motifs de données est nommé Pire cas. Ce sont les motifs dont l'énergie consommée par les liens est maximale selon le taux de commutation α. De la même manière que le premier type de motifs, nous déterminons quelles sont les deux données donnant les transitions montantes et descendantes les plus coûteuses. Etant donné que les transitions montantes et descendantes les plus consommatrices en énergie sont respectivement ($\downarrow$, $\uparrow$, $\downarrow$) et ($\uparrow$, $\downarrow$, $\uparrow$), les deux mots de données générant toujours ces transitions, sur

4 bits, sont 1010 suivi de 0101.

TABLEAU 2.5 – Classement des motifs de données selon le taux de commutation α sur un bus de 8 bits

Type de motif des données	α	**Flit précédant**	**Flit courant**
Meilleurs cas	0	**00000000**	**00000000**
	0.25	00**000000**	11**000000**
	0.5	0000**0000**	1111**0000**
	0.75	000000**00**	11111**00**
	1	00000000	11111111
Pires cas	0	**00000000**	**00000000**
	0.25	10**000000**	01**000000**
	0.5	1010**0000**	0101**0000**
	0.75	101010**00**	010101**00**
	1	10101010	01010101

0 : bits qui ne changent pas pendant la simulation

Estimation de l'erreur en faisant varier le PIR La figure 2.13 montre l'estimation de l'énergie par le modèle de liens de l'état de l'art nommé Statique, ainsi que le modèle prenant en compte le crosstalk nommé Crosstalk. Les simulations sont réalisées sous différents taux d'injections de paquets (PIR). Comme nous pouvons le constater, l'écart de consommation d'énergie des liens estimé par le modèle Crosstalk entre le Meilleur et le Pire cas est le même quel que soit le PIR dans le réseau. De plus, cet écart est significatif car l'estimation de l'énergie double en passant du Meilleur au Pire cas. Cette importante variation n'est pas perceptible par le modèle Statique ignorant les données fluctuant dans le NoC. Cela montre également l'impact significatif des données sur la consommation des interconnexions.

Il peut être noté que le modèle Statique se situe environ au centre des deux estimations fournies par le modèle Crosstalk. Cependant, la figure 2.13 montre que cela est vrai pour $\alpha = 0.5$. Or, nous savons que le taux de commutation des données peut varier au cours d'une simulation. Il serait donc intéressant de comparer les estimations de ces deux modèles avec un PIR fixe, tout en faisant varier le taux de commutation α. Sachant que l'écart de consommation d'énergie est le même quel que soit le PIR, la valeur choisie du PIR est $PIR = 0.017$.

Estimation de l'erreur en faisant varier α Maintenant, nous effectuons des simulations avec différents taux de commutations α observés sur les liens. Dans la figure 2.14, le PIR est fixé à 0.017 et le taux de commutation α varie

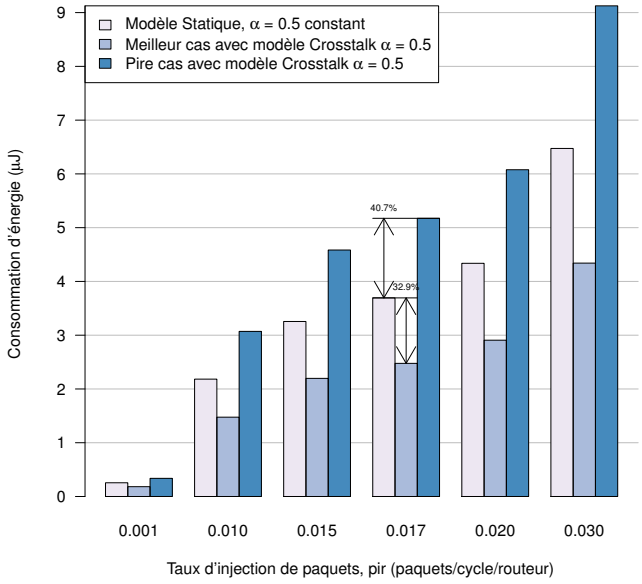


FIGURE 2.13 – Estimation de l'énergie des liens avec le modèle Statique et crosstalk sous différents motifs de données et différents PIR. Le taux de commutation α est fixé à 0.5.

dans l'intervalle [0, 1]. Les motifs des Meilleurs et des Pires cas sont construits en utilisant les motifs de données décrits dans le tableau 2.5.

La figure 2.14 montre deux droites correspondant aux estimations du modèle Crosstalk Ces droites nous donnent les limites de la variation de la consommation d'énergie des liens du NoC à ce PIR, puisque les motifs de Meilleurs et Pires cas de consommation sont utilisés. Ainsi, entre ces deux droites se trouve une surface où la consommation des liens peut varier avec le modèle Crosstalk en fonction des données traversant le réseau.

Quant au modèle Statique, ne considérant pas les données en supposant un taux de commutation fixe $\alpha = 0.5$, il ne propose qu'une seule valeur possible représentée par la croix au centre de la figure 2.14. Lorsque le taux de commutation à l'intérieur du NoC est véritablement égal à 0.5, l'estimation fournie par le modèle Statique se situe à la valeur moyenne des extrémums estimés par le modèle Crosstalk. Néanmoins, pour ce même taux de commutation, la consommation d'énergie des liens peut doubler, ce qui échappe au modèle Statique. Si le taux de commutation s'écarte de 0.5, l'estimation fournie par le modèle Statique est forcément fausse. En effet, son estimation est en permanence au-dessus du modèle Crosstalk de $\alpha = 0$ à $\alpha = 0.5$ et constamment en dessous de $\alpha = 0.75$ à $\alpha = 1$ (Voir figure 2.14). En conclusion, le modèle Statique n'apporte une estimation convenable que dans une seule configuration dans un continuum de possibilités à l'intérieur de laquelle la consommation

d'énergie peut varier. Cette configuration où le modèle Statique est correct est à $\alpha = 0.5$ lorsqu'il y a une répartition moyenne entre les transitions fortement consommatrices d'énergie et les autres. Dans tous les autres cas, le modèle Statique ne peut pas, par construction, estimer l'énergie des liens du NoC.

L'étude menée sur la précision des modèles des interconnexions a montré l'intérêt d'utiliser le modèle Crosstalk puisque les données impactent grandement la consommation d'énergie des liens du NoC. Cependant, ce modèle nécessite plus de calculs pour fournir une estimation. La prochaine étude concerne le surcoût temporel du modèle de liens Crosstalk.

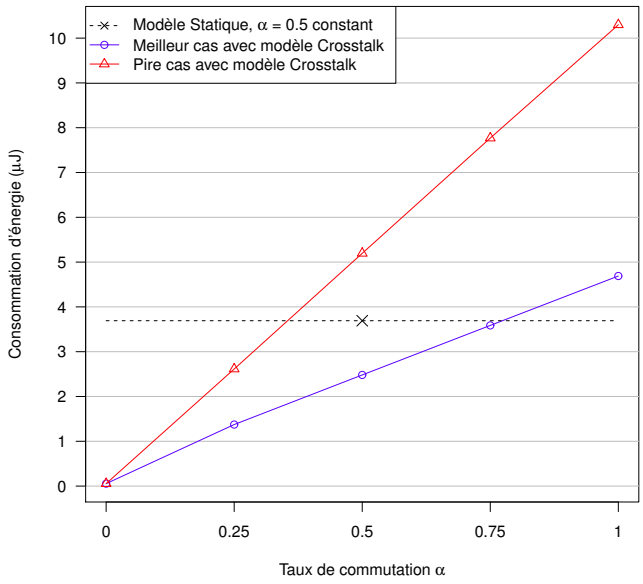


FIGURE 2.14 – Consommation d'énergie des liens avec le modèle Statique et crosstalk sous différentes valeur de α avec $pir = 0.017$. α étant calculé selon les données des flits pendant les simulations.

Estimation de l'erreur avec des trafics d'applications Pour cet ensemble de simulations, Noxim-XT est configuré pour exécuter des simulations avec des trafics basés sur des fichiers de traces d'applications.

Les expérimentations sont menées avec un NoC en grille 2D 4×3 pour les applications Dijkstra et Basicmath dans le but de se conformer au mapping de MPSoCBench utilisant 8 cœurs de processeurs, une mémoire partagée, une IP de contrôle et une IP de Lock gérant l'accès mémoire. Un NoC en grille 2D 4×2 est utilisé avec les autres applications pour la même raison. La fréquence du NoC est de 700 MHz avec un algorithme de routage XY. Les autres paramètres sont des flits de 32 bits, une profondeur des buffers de 4 flits, et des liens inter-routeur de 3 mm. Les simulation sont exécutées jusqu'à la fin de

l'application. Sachant que les temps d'exécutions varient selon les applications, au lieu de considérer la consommation d'énergie totale du NoC, nous considèrerons l'énergie totale de la simulation divisée par le temps de simulation afin de pouvoir comparer ces applications sur la base de leur consommation moyenne.

La figure 2.15 montre l'énergie moyenne par cycle du NoC selon l'application et le modèle d'interconnexion utilisé. Nous remarquons que l'estimation de l'énergie moyenne du NoC avec le modèle de liens Statique surestime constamment l'énergie consommée par les liens comparé au modèle Crosstalk. Cette surestimation va de 33% pour l'application Full HD jusqu'à 50% avec l'application Basicmath. Sachant que le modèle Statique surestime constamment l'énergie consommée des interconnexions, nous pouvons en déduire avec la figure 2.14 la nature des données. Ainsi, le taux de commutation α des trafics issus d'applications réelles semble être largement inférieur à la valeur moyenne $\alpha = 0.5$, supposée par le modèle Statique.

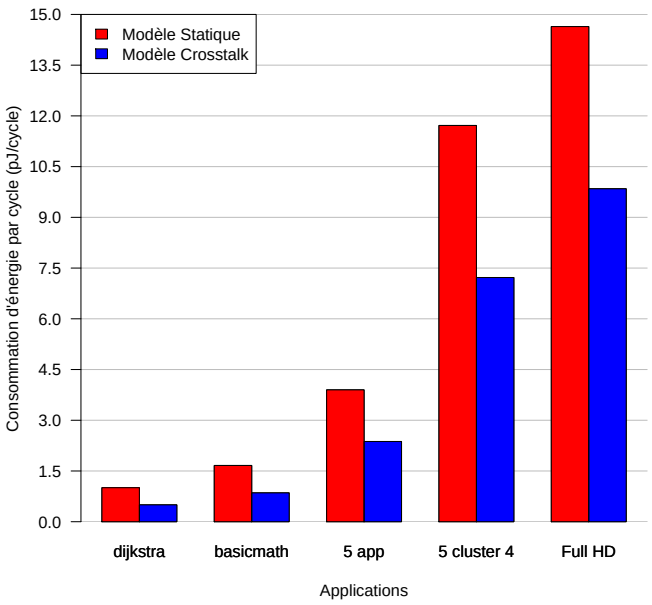


FIGURE 2.15 – Estimation de la consommation moyenne d'énergie du NoC par cycle d'horloge en fonction de l'application utilisée et du modèle d'interconnexion. 5 app est l'acronyme pour l'application *5 applications* et 5 cluster 4 pour *clusters applications*.

2.3.1.3 Étude du surcoût temporel

Nous avons vu précédemment que le modèle Crosstalk permet de prendre en compte l'ensemble des variations de consommation des liens inter-routeurs ;

variations liées à la nature des données transportées sur ces liens. Pour être acceptable, ce nouveau modèle ne doit pas générer un surcoût temporel trop important au niveau de la simulation. Nous avons donc repris l'ensemble des expériences précédentes des trafics synthétiques et mesuré le temps de simulation en fonction du modèle d'interconnexions.

La figure 2.16 représente le temps de simulation de Noxim-XT selon le modèle de liens utilisé. Nous faisons également varier la taille du NoC allant de 4×4 à 8×8 et le PIR qui va de 0.001 à 0.030 avec un trafic uniforme aléatoire. Nous pouvons constater que le modèle Crosstalk a des temps de simulation plus importants que le modèle Statique. L'augmentation du temps de calcul va de 2% pour le PIR le plus bas jusqu'à 22% pour le plus haut. Il peut également être observé que le surcoût temporel pour un NoC de taille 8×8 devient stable et atteint une limite lorsque le PIR est élevé. Arrivé à une certaine densité de trafic dans le NoC, il arrive à un état de congestion où le transfert de données ne peut plus augmenter, limitant par conséquent les calculs pour le modèle. Ainsi, le surcoût temporel maximal du modèle Crosstalk étant de 22%, cela est tout à fait acceptable aux vues des améliorations d'estimation de la consommation d'énergie.

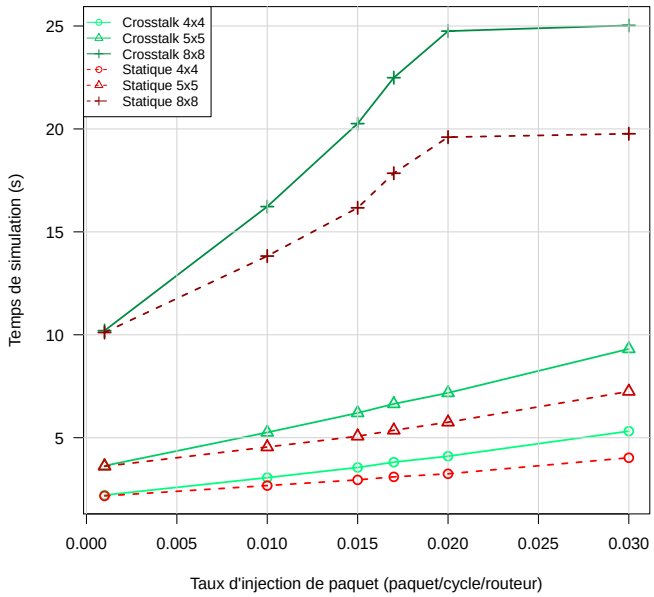


FIGURE 2.16 – Temps de simulation de Noxim-XT selon le modèle d'interconnexion utilisé, la taille du NoC et le PIR. Le terme Crosstalk est pour le modèle bit-près et Statique pour le modèle indépendant des données.

L'imprécision du modèle d'interconnexion Statique a été constaté et le surcoût temporel du modèle de liens proposé est raisonnable, la prochaine étape de cette évaluation est de vérifier si la consommation d'énergie des liens est

suffisamment importante pour justifier un effort supplémentaire sur sa modélisation.

2.3.1.4 Étude de la part des interconnexions dans la consommation du NoC

Nous abordons maintenant la question de la part de la consommation d'énergie des liens dans l'ensemble de la consommation du NoC. Deux motifs de données sont utilisés représentant deux scénarios extrêmes, nous permettant d'évaluer la variation d'énergie maximale qui peut être attendue sur l'ensemble de la consommation du NoC. Le premier motif de données est le Meilleur cas avec $\alpha = 0$ du tableau 2.5 et le second motif de données, est le Pire cas avec $\alpha = 1$.

La figure 2.17 montre l'impact de la consommation d'énergie des liens sur l'ensemble de la consommation d'énergie du réseau. Nous pouvons constater que l'écart de la consommation d'énergie du NoC entre le Pire cas et le Meilleur cas estimé avec le modèle Crosstalk augmente à mesure que le PIR augmente, contrairement à ce qui a été observé à la figure 2.13 ne montrant que l'énergie des liens. Cela peut être expliqué comme suit : un PIR faible implique un trafic faible dans le NoC et donc une faible utilisation des interconnexions. En conséquence, la part de consommation des liens est faible quand le PIR est bas, ce qui se traduit par un faible écart entre le Meilleur et le Pire cas. Cet écart devient significatif dès $PIR = 0.010$. La variation de l'énergie totale peut aller du simple au double lorsque le PIR est élevé ($PIR = 0.017$). Ainsi, la contribution des liens est suffisamment importante pour que la consommation d'énergie totale du NoC puisse doubler en fonction de la nature des données quand le trafic est important.

La figure 2.18 montre la consommation d'énergie totale du NoC avec le modèle Crosstalk sous un trafic uniforme aléatoire avec données également aléatoires. La consommation d'énergie de chaque composant y est présenté. Nous observons que la consommation d'énergie des liens devient le deuxième plus grand consommateur d'énergie à partir de $PIR = 0.010$ et devient le premier consommateur d'énergie lorsque le PIR est au-delà de 0.020. Il peut également être constaté que le NA consomme jusqu'à 13% de la consommation totale et est stable quel que soit le PIR. Du fait de sa stabilité, et de l'absence de modèle générique pour des trafics d'applications, le NA ne sera plus considéré dans la consommation totale d'énergie du NoC.

Une étude supplémentaire est effectuée afin de vérifier si l'hypothèse du modèle Statique fixant le taux de commutation des données à 0.5 est pertinente.

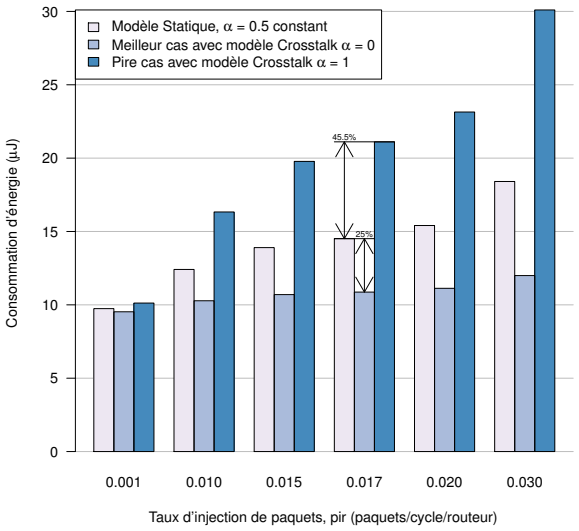


FIGURE 2.17 – Consommation d'énergie totale du NoC avec le modèle de liens statique et dynamique. Les résultats sont sous différents PIR et avec deux motifs de données.

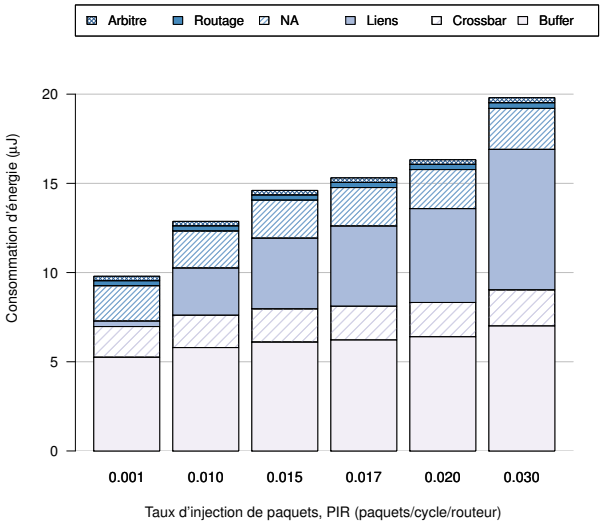
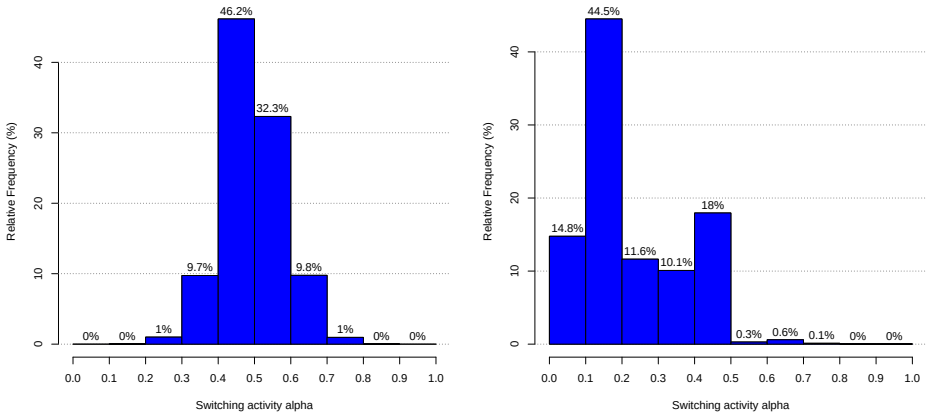


FIGURE 2.18 – Bilan de consommation d'énergie du NoC utilisant le modèle de liens proposé et avec des données aléatoires.

2.3.1.5 Étude des trafics d'applications

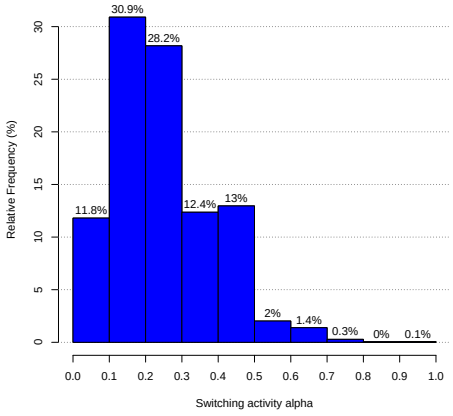
Cette étude reprend les résultats de simulations effectués en section 2.3.1.2, ceux évaluant la précision des modèles avec des trafics d'applications. La fi-

gure 2.19a montre la distribution du taux de commutation lorsque le NoC est sous un trafic de données aléatoire uniforme avec un contenu des flits aléatoire également. Il n'est donc pas surprenant de constater que la distribution est une gaussienne centrée sur $\alpha = 0.5$. Concernant la distribution de l'application Dijkstra (figure 2.19b), 60% de la distribution a un taux de commutation inférieur à $\alpha = 0.2$. De plus, comme le montre les figures 2.19c,d,e,f, le taux de commutation des applications oscille entre 0.2 et 0.3, confirmant nos suppositions émises en section 2.3.1.2 avec les trafics d'applications. Ce sont des taux faibles d'activité comparé à l'hypothèse émise par le modèle Statique fixant le taux à 0.5. Par conséquent, le modèle Statique ne peut fournir une estimation correcte de l'énergie pour ces applications.

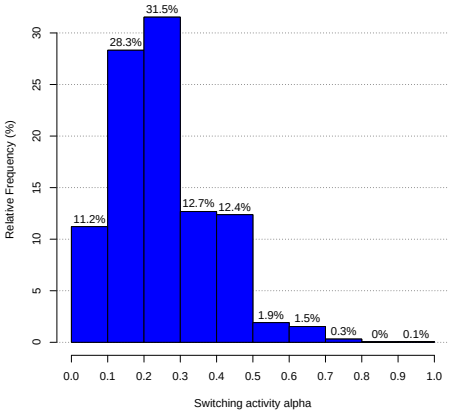


(a) Trafic uniforme aléatoire

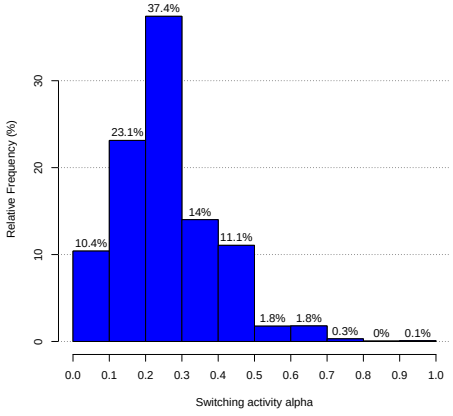
(b) Dijkstra



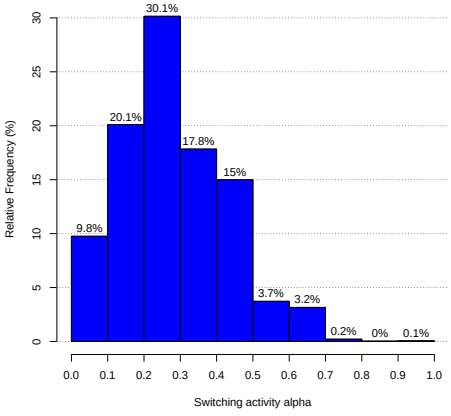
(c) Basicmath



(d) 5 applications



(e) Clusters applications



(f) Full HD

FIGURE 2.19 – Distribution du taux de commutation selon le trafic à l'intérieur du NoC.

2.4 Bilan

Ce chapitre a évalué la précision du modèle d'interconnexion utilisé dans la littérature des NoCs. Nous avons montré que ce modèle n'est précis en moyenne que lorsque $\alpha = 0.5$, et nous avons également montré que pour un même taux, l'énergie consommée par les liens peut doubler en fonction de la nature des données. Dans tous les autres cas, où le taux diffère de 0.5 ou varie pendant la simulation, ce modèle est imprécis. Ainsi, notre première contribution partant sur l'amélioration du modèle de consommation des liens inter-routeurs démontre tout son intérêt.

Notre seconde contribution est l'intégration de ce modèle dans le simulateur Noxim-XT ainsi que l'ajout de nouvelles fonctionnalités aidant le concepteur dans l'exploration d'architectures du réseau. En fournissant de meilleures estimations de l'énergie, Noxim-XT permet l'investigation d'optimisations énergétiques.

Dès lors, l'estimation désormais plus précise de l'énergie consommée par un NoC ouvre la voie à l'optimisation de la consommation énergétique de ces réseaux. Le prochain chapitre s'intéressera aux optimisations de la consommation d'énergie des NoCs. Un état de l'art sera réalisé pour déterminer quelles sont les méthodes existantes et quelles sont les plus efficaces afin d'extraire les pistes à suivre.

Deuxième partie

Optimisation

3

État de l'art de l'optimisation de la consommation d'énergie des NoCs

Sommaire

3.1 Introduction

Le chapitre 3 présente un état de l'art non exhaustif sur les principales techniques d'optimisation de la consommation d'énergie des NoCs. Dans un premier temps, l'étude se porte sur les contributions au niveau de la topologie du NoC. Nous nous intéresserons ensuite aux optimisations pouvant être apportées au routeur, et enfin, nous abordons les techniques d'optimisation applicables aux interconnexions. Nous utiliserons ce découpage hiérarchique pour présenter cet état de l'art.

3.2 Optimisation au niveau de la topologie

Dans la littérature des NoCs, de nombreuses topologies sont proposées, présentant chacune des atouts et des défauts qui leurs sont propres. Certaines topologies requièrent un plus grand nombre d'interconnexions et de nœuds afin d'améliorer les performances temporelles au détriment de la consommation d'énergie. A l'inverse, des topologies sont pensées pour consommer moins d'énergie.

Cette section présente plusieurs topologies 2D avec leurs spécificités. Ensuite, l'intérêt des topologies en 3D est discuté.

3.2.1 Exploration des topologies 2D

La recherche sur la topologie des NoCs bénéficie des travaux réalisés sur la topologie des réseaux en général tel que dans [47]. La recherche dans ce domaine existe depuis plusieurs décennies et ces travaux ont été adaptés pour les NoCs.

Parmi les propositions de topologie en deux dimensions dans les NoCs, quatre d'entre elles sont particulièrement étudiées. Ces topologies sont illustrées à la figure 3.1 et décrites ci-dessous :

— Grille 2D, un maillage régulier de routeurs. C'est la topologie la plus utilisée et la plus simple à réaliser car les interconnexions sont identiques ainsi que les routeurs.
— SPIN [8], porte le nom du NoC associé. Cette topologie est un arbre, les feuilles étant les IPs et les nœuds sont les routeurs. Cette topologie en arbre a la particularité de contenir des nœuds parents dupliqués. Par conséquent, elle présente le même nombre de routeurs à chaque étage de l'arbre.
— BFT, cela signifie Butterfly Fat Tree, il s'agit comme son nom l'indique d'une topologie en arbre. La différence avec SPIN est que les nœuds parents ne sont pas dupliqués, réduisant le nombre d'interconnexions

du NoC.

— Octogone [48], cette topologie regroupe les nœuds du réseau par huit en les connectant avec douze interconnexions. Cela permet de réduire le nombre de sauts [1] nécessaire à un paquet pour atteindre sa destination. Il est possible d'agrandir le réseau en associant plusieurs groupes de huit nœuds liés par un nœud commun appelé nœud pont.

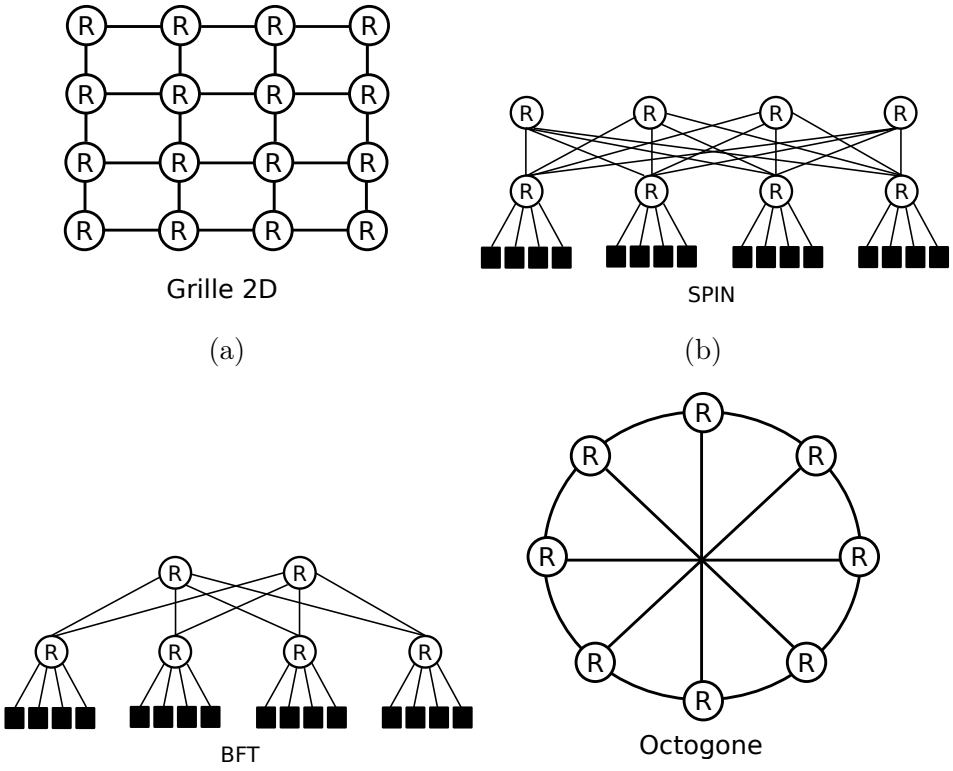


FIGURE 3.1 – Topologies en deux dimensions les plus représentées. Les carrés noirs des topologies SPIN et BFT sont les IPs, les autres topologies possèdent une IP par routeur.

Ces topologies ont fait l'objet de plusieurs études de performance que ce soit en termes de débit ou de puissance [49, 50, 51]. Le tableau 3.1 résume les études de [49, 50, 51] réalisées sur les topologies présentées. Il peut être observé que la topologie SPIN est la meilleure en termes de délai et de débit, cela au prix d'une puissance très élevée comparée aux autres topologies. Ce débit élevé est permis par un important nombre de liaisons entre les routeurs ainsi qu'un nombre de routeurs plus important qu'une topologie en arbre classique.

La topologie BFT est quant à elle la topologie fournissant le débit le plus

1. Nombre de routeurs par lequel le paquet doit passer pour arriver à destination.

faible mais dont la puissance est la plus faible également. Cette faible puissance est expliquée par son coût matériel plus bas que celui des autres topologies, ainsi que par son faible nombre d'interconnexions. De plus, le débit ne pouvant être aussi élevé que pour les autres topologies, cela induit des pics de puissance plus faibles.

La topologie Octogone est un compromis entre puissance et débit, son débit est supérieur à la topologie en grille 2D mais a également une consommation d'énergie légèrement plus élevée. Il peut être noté que la topologie en grille 2D est la topologie avec la latence la plus élevée. Cela est dû au nombre plus important de sauts que le paquet doit accomplir pour arriver à destination.

Enfin, une corrélation peut être observée dans le tableau 3.1 entre le débit et la puissance consommée. En effet, plus le débit supporté est grand et plus sa puissance le sera également. Cela nous permet de déduire qu'il semble utopique de proposer une topologie peu consommatrice tout en conservant un débit élevé. En conclusion, si la puissance et la consommation d'énergie sont les contraintes principales du système, alors la topologie BFT est recommandée. Néanmoins, la topologie en grille 2D et Octogone sont de meilleurs compromis puissance / débit.

Les progrès technologiques permettent de concevoir des circuits en trois dimensions au lieu de seulement deux. Avec cette nouvelle dimension entrant dans le design d'architecture des SoCs, des chercheurs proposent des topologies de NoCs exploitant cette approche au moyen de topologies en trois dimensions.

TABLEAU 3.1 – Tableau récapitulatif des performances en débit, latence et consommation des topologies

Topologie	**Classement des topologies**		
	Latence	**Débit**	**Puissance**
Grille 2D	La plus haute	Haut	Moyenne
SPIN	La plus basse	Le plus haut	La plus haute
BFT	Moyenne	Le plus bas	La plus basse
Octogone	Basse	Très haut	Moyenne haute

3.2.2 Topologies en 3D

Réaliser un NoC en 3D apporte plusieurs avantages. Pour commencer, il est possible de disposer plus de routeurs dans une surface réduite grâce aux étages supplémentaires. Cela augmente la densité du NoC et réduit la distance entre les nœuds, réduisant la latence et également la consommation d'énergie comparé à son homologue en 2D puisque les interconnexions sont plus petites à chaque couche [52].

La condition obligatoire afin de concevoir un NoC en 3D est d'avoir des in-

terconnexions verticales entre les couches. Ces interconnexions verticales sont nommées TSV pour Through-Silicon-Via [53], ces liaisons supplémentaires augmentent la connectivité du réseau et la bande passante est également augmentée.

Ce type de topologie apporte cependant de nouvelles problématiques. L'augmentation de la densité du NoC provoque une forte hausse de la température à cause de composants superposés dissipant beaucoup de puissance. Une température élevée a des conséquences critiques dans un circuit. Cela peut nuire à la fiabilité de la puce en générant des erreurs et réduire aussi sa durée de vie. Il existe différentes méthodes pour réduire la température, tel qu'un placement intelligent considérant la température [54] ou encore de nouvelles structures de refroidissement [55].

Nous présentons brièvement deux topologies 3D prometteuses de la littérature vis-à-vis de la consommation d'énergie :

— 3D mesh [56], ou grille 3D comme représenté à la figure 3.2. C'est l'adaptation de la grille 2D, apportant une topologie 3D simple à réaliser. Cependant, la taille du routeur augmente car deux nouvelles directions peuvent être empruntées par les paquets ce qui implique un crossbar 7×7 au lieu d'un crossbar de taille 5×5 en 2D.
— Honeycomb [57] est un NoC exploitant une structure d'interconnexions alvéolée. Cela permet de réduire le coût matériel du réseau car le nombre d'interconnexions est moins important que dans une grille 2D classique comme le montre la figure 3.3a. De cette manière le nombre de directions par routeur est réduit, diminuant la taille du crossbar. Ainsi, ce procédé est reproduit avec une dimension supplémentaire, la figure 3.3b montrant une structure alvéolée en 3D. Cette topologie réduit le coût de 40 % comparé à une topologie en grille 3D. Ces réductions se traduisent par une réduction significative de la consommation d'énergie.

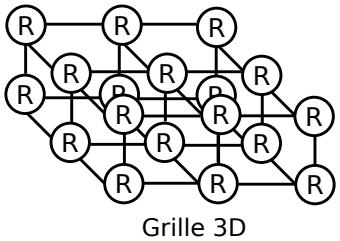


FIGURE 3.2 – Exemple de topologie 3D.

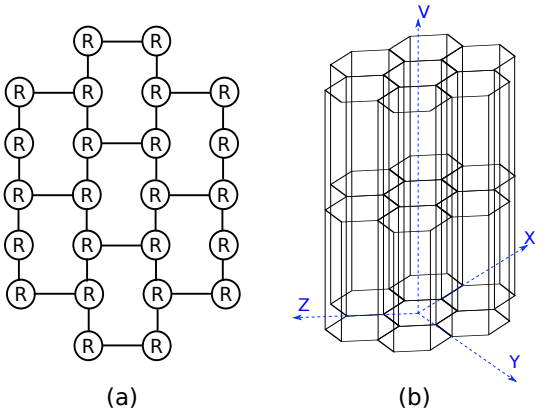


FIGURE 3.3 – (a) Structure alvéolée basée sur une grille 2D. (b) Aperçu en 3D d'une topologie alvéolée en 3D (les sommets sont les routeurs).

3.3 Optimisation au niveau du routeur

Le routeur est un composant complexe pouvant être optimisé globalement ou sur un de ses sous-composants. Le routeur étant principalement composé de mémoires (buffers) utilisées par intermittence, la proportion d'énergie statique consommée par le routeur est importante. C'est pourquoi les techniques d'optimisation de la consommation du routeur sont principalement axées autour de la réduction de l'énergie statique.

3.3.1 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. Cependant, afin d'éviter d'importantes baisse des performances, il est indispensable d'avoir une bonne compréhension du circuit pour savoir quand en éteindre certaines parties. La contribution [58] présente les différents avantages et inconvénients du Power gating.

L'extinction des parties du circuit peut être réalisée à différents niveaux de granularité. En prenant l'exemple d'un NoC, le routeur entier peut être éteint comme proposé dans [59, 60], ou seulement une partie du routeur tel que montré dans [61, 62]. Nous allons détailler ces deux niveaux de granularité en analysant une contribution pour chacun d'eux.

3.3.1.1 Power gating gros grain

Dans le cas des routeurs, l'utilisation du power gating à gros grain se fait en éteignant complètement un routeur. Dans [60], les auteurs souhaitent utiliser le power gating sur un NoC à très forte demande en bande-passante avec des interconnexions de 512 bits de large. La topologie du NoC est en grille 2D concentrée, cela signifie que chaque routeur est associé à plusieurs IP (quatre par routeur dans cet exemple).

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. Néanmoins, avec des routeurs aussi imposants, ceux qui sont alimentés sont deviennent quand le trafic est faible. 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.

La figure 3.4 de gauche montre bien l'inefficacité du power gating sur un NoC unique. En effet, le trafic du réseau étant représenté par trois communications étiquetées 1, 2 et 3 nécessite l'utilisation de 8 des 9 routeurs ce qui permet une faible économie d'énergie. Avec un multi-NoC composé de quatre sous-réseaux (figure 3.4 de droite), si le besoin en bande-passante des communications 1, 2 et 3 est bas, on peut voir qu'il est possible de n'utiliser que 8 routeurs sur les 36 disponibles. Le gain d'énergie atteint est alors important.

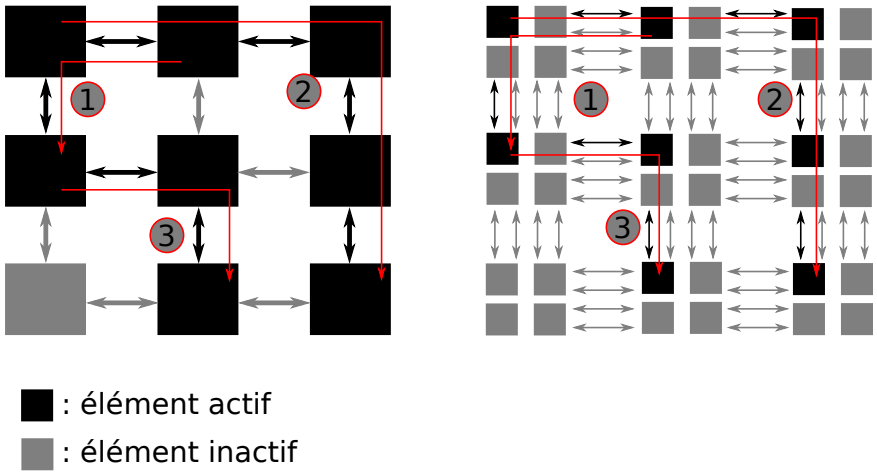


FIGURE 3.4 – La figure montre le nombre potentiel de routeurs pouvant être éteints avec un NoC unique (à gauche) et avec quatre sous-réseaux (à droite).

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 [2], 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. Puisque la subdivision du NoC est indispensable pour exploiter cette méthode, elle ne peut être déployée que sur un NoC disposant d'une très importante bande-passante. Pour les NoC plus usuels, le power gating n'est plus appliqué sur tout le routeur mais sur une partie de ses composants seulement.

3.3.1.2 Power gating grain fin

Les auteurs de [62] ont conçu une technique visant à couper l'alimentation de parties spécifiques d'un routeur tout en garantissant le fonctionnement du réseau. Dans un NoC, certaines interconnexions entre deux routeurs peuvent être inexploitées et le crossbar ainsi que les buffers consomment inutilement de l'énergie statique. Cette contribution propose d'éteindre les composants d'un segment de donnée inutilisé. Comme le montre la figure 3.5, un segment de donnée représente tous les composants du NoC compris entre la sortie du crossbar d'un routeur émetteur et l'entrée du crossbar du routeur récepteur d'une voie.

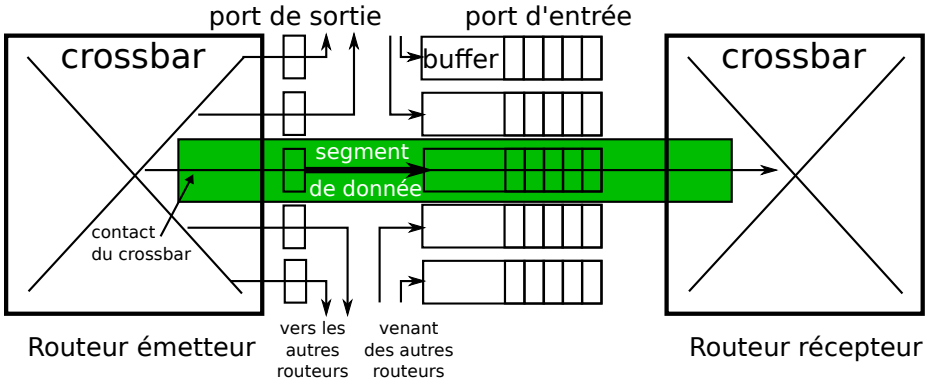


FIGURE 3.5 – Schéma de représentation du segment de donnée, montrant les extrémités des crossbars entre deux routeurs avec le port de sortie, les buffers d'entrées d'une direction et l'interconnexion entre les deux routeurs.

L'extinction de l'alimentation se déclenche avec un mécanisme d'étude du trafic pour chaque segment de donnée. Pendant une époque [3] donnée, le trafic

2. Lorsque le taux d'utilisation de ses buffers dépasse un seuil, comme 75 % d'occupation par exemple.

3. Une époque représente plusieurs milliers de cycles.

est analysé avec un compteur de messages. Si l'activité passe en dessous d'un certain seuil durant une époque, alors le segment est désactivé. Le seuil utilisé est dynamique, il s'adapte aux retours qu'a reçu l'algorithme de décision avec l'époque précédente. Afin d'éviter de bloquer le réseau en redirigeant les données vers des chemins de données désactivés, l'algorithme de routage s'adapte lors de chaque époque à la nouvelle configuration modifiant la topologie. Cela permet d'éviter les interblocages (deadlock). Il est, bien sûr, impossible de bloquer complètement un routeur en coupant tous ses chemins de données.

Le gain d'énergie apporté par cette technique oscille entre 14.5 % et 37 % selon la taille du réseau et le nombre de segments de données en veille. Il peut être noté que le réveil d'un segment n'est pas instantané et ce réveil entraîne une baisse des performances de 2 %. Néanmoins, cette contribution ne mentionne pas ce qu'il se passe pour un paquet en transit qui rencontre un routeur venant de s'éteindre. De plus, la consommation des transistors gérant l'alimentation des segments n'est pas comptabilisée dans le surcoût de cette proposition.

3.3.2 Buffer avec mémoire non-volatile

La contribution proposée dans [63] réduit la consommation d'énergie des buffers du routeur, qui sont les plus grands consommateurs d'énergie. Pour cela, ils utilisent une architecture hybride composée de deux types de mémoires :

- La drowsy SRAM (Static Random Access Memory) est une mémoire statique non-volatile qui a constamment besoin d'être alimentée pour fonctionner. Sa différence avec la SRAM couramment utilisée dans les NoCs est que cette mémoire bénéficie d'un mode supplémentaire, le *drowsy mode*. La cellule est alors alimentée avec une tension sous-nominale afin de consommer moins d'énergie tout en conservant l'information stockée. Cependant, pour accéder à cette donnée il faut obligatoirement rétablir la tension d'alimentation normale. Ce changement de mode entraîne cependant une latence, qui reste moins conséquente que le réveil d'une SRAM en power gating.
- La STT-RAM (Spin-Transfer Torque-Random Access Memory) est une mémoire magnétique qui mémorise une donnée en exploitant l'orientation du spin des électrons. Cette méthode permet de conserver l'information même lorsque le système n'est pas alimenté. Son énergie consommée en fonctionnement est cependant légèrement supérieure à la drowsy SRAM. L'inconvénient principal de la STT-RAM est son temps d'écriture de la donnée, plus long que celui de la SRAM.

Afin d'utiliser cet ensemble de mémoire hybride, les buffers du NoC proposés dans [63] emploient des canaux virtuels à quatre étages. Deux étages utilisent de la drowsy SRAM tandis que les deux autres utilisent de la STT-RAM. Cela permet d'activer les étages en fonction des besoins du réseau. La

figure 3.6 nous montre la configuration par défaut des canaux mémoires, le premier étage de mémoire drowsy SRAM est en mode actif (tension normale), le deuxième étage de drowsy SRAM est en *drowsy mode* et enfin les deux étages de STT-RAM utilisent du power gating.

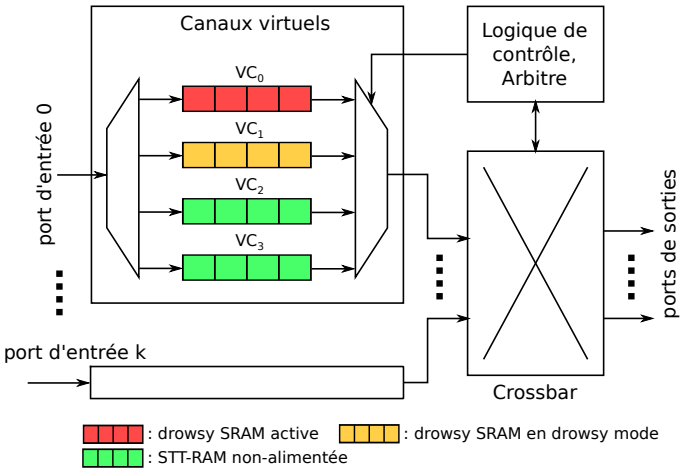


FIGURE 3.6 – Disposition hiérarchique de la mémoire hybride. VC_0 et VC_1 sont les mémoires drowsy SRAM et VC_2 et VC_3 sont les STT-RAM.

Avec cet agencement de mémoire, il est possible de réduire jusqu'à 30.9 % l'énergie consommée avec également une réduction de la surface de 7.6 % grâce à la haute densité des cellules STT-RAM. Cette technique est très efficace lorsque le trafic dans le NoC est faible. Cependant, dès que le trafic augmente, puisque les temps d'écritures sont plus lents sur les STT-RAM, il faut activer plus d'étages de canaux virtuels qu'avec des mémoires SRAM standard ce qui provoque un surcoût énergétique.

3.3.3 Algorithme de routage

Avant d'étudier les algorithmes de routage de la littérature pouvant réduire la consommation d'énergie, il est important de s'attarder sur l'algorithme XY, le plus utilisé dans les NoCs. Son fonctionnement est le suivant : un paquet est dirigé d'abord horizontalement pour atteindre la même colonne de routeur que sa destination puis, le paquet est routé verticalement pour arriver à destination. Le même principe peut être appliqué sur un NoC en trois dimensions et dans ce cas l'algorithme est nommé XYZ. Cet algorithme est simple à implanter et son coût matériel est très faible, impliquant une faible consommation d'énergie. Néanmoins, l'algorithme XY a quelques inconvénients, le trafic est concentré au centre du NoC comme l'illustre la figure 3.7. Cela crée des points chauds (hot spot) dans le réseau ce qui peut conduire à une élévation locale de la température et cette concentration du trafic n'exploite pas le débit maximal

du NoC. Le second problème est qu'un seul chemin est possible pour le paquet, le rendant sensible aux fautes dans le cas où une interconnexion est défectueuse.

La majorité des contributions réalisées au niveau de l'algorithme du routage des NoCs se trouve dans la résolution de ces problèmes et non dans la proposition d'algorithmes de routage consommant moins d'énergie. En effet, que ce soit pour la résilience aux fautes [64], pour la réduction de la congestion [65, 66] ou pour l'augmentation du débit [67], toutes ces améliorations impliquent de l'intelligence supplémentaire dans chaque routeur, ce qui augmente le coût matériel du routage et donc sa consommation d'énergie.

Toutefois, certaines propositions présentent des algorithmes de routage simples, avec un meilleur débit que l'algorithme XY et une consommation d'énergie équivalente à celui-ci. Nous allons présenter un de ces algorithmes.

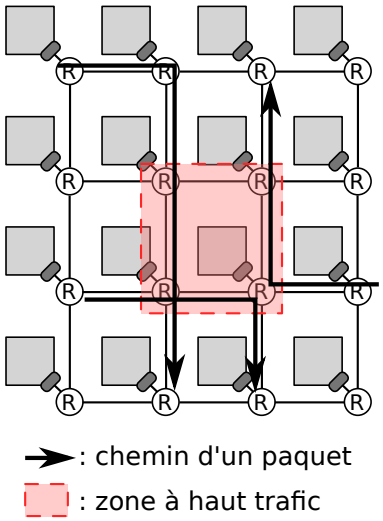


FIGURE 3.7 – Congestion locale du NoC au centre du réseau avec l'algorithme de routage XY.

3.3.3.1 Algorithme DyAD

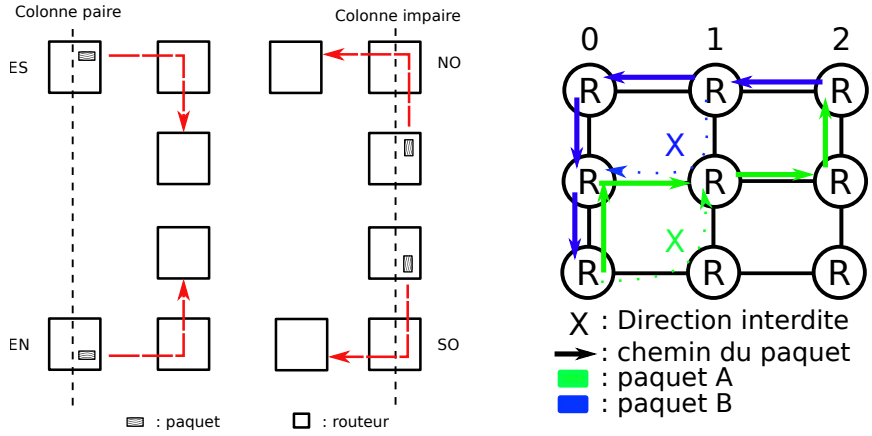
L'algorithme DyAD [68] (Dynamic Adaptive Deterministic switching) alterne entre un mode de routage déterministe et adaptatif. Lorsque le trafic est faible, le routage déterministe XY est utilisé car il a une latence faible du fait de sa simplicité. Quand le trafic devient dense, une version de l'algorithme Odd-Even [69] est utilisée. Le principe du Odd-Even est de restreindre des directions que peut prendre un paquet selon la colonne de routeur où il se trouve.

— Loi 1 : Aucun paquet n'est autorisé à prendre la direction EN (paquet voulant prendre la direction Est puis Nord au routeur suivant) à un rou-

teur d'une colonne paire ni à prendre la direction NO dans les colonnes impaires.

— Loi 2 : Aucun paquet n'est autorisé à prendre la direction ES (paquet voulant prendre la direction Est puis Sud au routeur suivant) à un routeur d'une colonne paire ni à prendre la direction SO dans les colonnes paires.

Les lois sont montrées dans la figure 3.8 ainsi qu'un exemple sur un NoC en grille 2D 3×3.



(a) (b)

FIGURE 3.8 – (a) Illustration des tournants interdits selon la colonne de routeur. (b) Exemple du routage de deux paquets avec Odd-Even.

L'utilisation du Odd-Even à haut trafic permet de gérer un débit plus important puisque le trafic est mieux réparti au lieu de se concentrer au centre avec le routage XY. De plus, le surcoût matériel de l'algorithme DyAD est faible, il est 7 % plus important que le XY laissant supposer une faible consommation. Néanmoins, les expérimentations proposées dans la contribution n'estiment pas l'énergie consommée de la technique. Une implantation des algorithmes XY et DyAD a été réalisée dans la version originale de Noxim, et le surcoût de l'algorithme DyAD est 8 % supérieur à XY pour une augmentation du débit maintenable de 53.3 %.

La prochaine section aborde les techniques d'optimisation pouvant être menées sur les interconnexions.

3.4 Optimisation au niveau des interconnexions

3.4.1 Optimisation architecturale

Les techniques d'optimisation de la consommation d'énergie des liens au niveau architectural consistent toutes en un codage de données tel que le montre la figure 3.9. Le but du codage est de transmettre l'information originale codée sur n bits en une information codée sur m bits avec $n \leq m$ tel que le taux de commutation α des données codées soit inférieur à celui des données non codées. On trouve plusieurs travaux dans la littérature des bus de communication proposant des adaptations au contexte des NoCs.

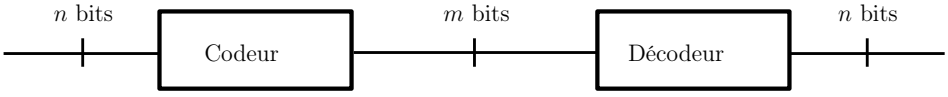


FIGURE 3.9 – Principe des techniques d'encodage des interconnexions.

3.4.1.1 Bus Invert/Bus Invert partiel

Le Bus Invert et le Bus Invert partiel sont des techniques adaptées dans un NoC dans [70] et viennent de deux contributions antérieures sur les bus de communication [71, 72]. L'idée du Bus Invert est de calculer la distance de Hamming entre la donnée courante et la précédente. Lorsque le résultat est supérieur à la moitié de la largeur du bus, alors la donnée courante est inversée. Cela nécessite l'ajout d'un fil supplémentaire sur le bus pour indiquer si la donnée est inversée ou non. Cette technique fonctionne particulièrement sur des bus larges où le taux de commutation α des données est élevé. Néanmoins, le surcoût matériel du Bus Invert est important et cette technique a été déclinée (Bus Invert partiel) pour être utilisée seulement sur les bits de poids faibles (LSB) où le taux de commutation est le plus important [23].

3.4.1.2 Codage des données selon leur taux d'apparition

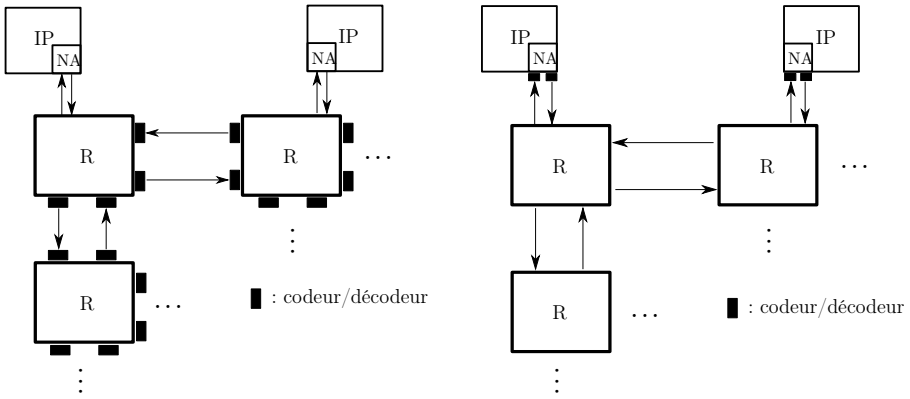
L'algorithme nommé MFLP (Most Frequent Least Power) est présenté dans [73]. Ces travaux proposent d'utiliser un algorithme codant les mots les plus fréquents par les valeurs binaires basses (contenant beaucoup de zéros) afin de limiter le nombre global de transitions. Pour ce faire, une analyse des données est effectuée dans le NoC en temps réel. Dans une fenêtre glissante temporelle, l'algorithme compte l'occurrence des différents mots qui sont apparus et un arbre est généré assignant aux mots un code en évoluant dans l'arbre. Par exemple, si le mot de 4 bits 1100 est le plus fréquent dans une fenêtre de

temps donnée, il est codé par la valeur 0000. Par conséquent, l'activité des interconnexions du réseau est réduite, ce qui permet une économie d'énergie. Néanmoins, cette méthode n'empêche pas l'apparition de transitions croisées et l'implantation de cet algorithme dans chaque routeur du NoC génère un surcoût matériel de 15.43 %.

3.4.1.3 Codage des données end-to-end

Les auteurs de [74, 75] proposent une technique de codage des données effectuée à l'entrée et à la sortie du réseau (end-to-end) dans le but de réduire le surcoût matériel induit par les codeurs et les décodeurs. Les données sont encodées dès qu'elles arrivent dans le NA au lieu d'être codées et décodées à chaque liaison inter-routeur, réduisant ainsi le nombre de codeurs / décodeurs (codecs) nécessaires. La figure 3.10a montre le placement des codeurs / décodeurs (codecs) pour la méthode routeur à routeur. Elle requiert un plus grand nombre de codecs qu'en utilisant la méthode end-to-end illustrée à la figure 3.10b. Cependant, l'inconvénient avec la technique end-to-end est que le codage ne gène pas le routeur pour identifier la destination des paquets.

La technique proposée dans [74] pour réduire l'énergie des liens, tout comme le Bus-Invert, a pour but d'inverser les données qui génèrent des transitions coûteuses. La différence avec celui-ci est la décision de l'inversion. Elle est réalisée selon les transitions qui sont générées avec ou sans inversion du flit. Si l'inversion du flit génère moins de transitions coûteuses en énergie (transitions croisées), alors l'inversion est effectuée. De ce fait, ce codage end-to-end doit ajouter l'information de l'inversion des flits. Ces informations sont ajoutées dans le contenu des flits puisque l'architecture des liens inter-routeur n'est pas modifiée. Par conséquent, les paquets générés utilisant ce codage sont plus longs et cela augmente le trafic à l'intérieur du NoC.



(a) Router-to-router (b) End-to-end

FIGURE 3.10 – Position des codecs en Router-to-router et end-to-end.

3.4.2 Optimisation du circuit

3.4.2.1 Approximate computing

L'approximate computing est un domaine de l'électronique où la précision d'un calcul est légèrement diminuée au profit d'un gain énergétique significatif. Cette diminution de la précision peut être atteinte de plusieurs manières, soit en utilisant des blocs de calculs plus petits mais moins précis que les blocs originaux, soit en abaissant la tension des composants, ce qui réduit l'énergie consommée et la fiabilité du résultat. L'exploration du compromis entre énergie et précision a commencé sur les NoCs en 2005 avec les travaux D. Bertozzi *et al.* dans [76].

Les auteurs de [77] proposent de reprendre cette idée et de l'utiliser pour la transmission de données dans un NoC. Les interconnexions inter-routeur ont deux modes de fonctionnement :

— Voltage normal, fonctionnement normal du circuit.
— Sous-voltage, réduit l'écart entre l'état haut et l'état bas du signal d'un fil en abaissant la tension.

De cette manière, lorsqu'une donnée doit être envoyée d'un routeur à un autre, un de ces deux modes est sélectionné pour la transmission. L'inconvénient du mode sous-voltage est l'augmentation du taux d'erreur par rapport au fonctionnement normal. Ainsi, lorsque les données ne sont pas critiques (tel que l'information de la couleur d'un pixel d'une image), alors le chemin à faible voltage est emprunté tandis que si le signal est important, le chemin normal est utilisé. La figure 3.11 montre la duplication des interconnexions avec les deux modes de fonctionnement. Cette méthode permet de réduire la consommation d'énergie entre 20 % et 43 % sans impacter les performances au prix du doublement des interconnexions. Cela multiplie donc par deux la surface occupée pour toutes les liaisons inter-routeur, impact qui devient critique dans le cas où la surface du NoC est un critère important.

3.4.3 Optimisation technologique

3.4.3.1 Sérialisation

Le principe de la sérialisation est de passer d'une transmission parallèle à une transmission série dans le but de réduire le nombre d'interconnexions. L'intérêt de cette méthode est double. Premièrement, la sérialisation réduit significativement la surface occupée par les liens de communication. Deuxièmement, les effets de crosstalk sont éliminés puisqu'une liaison série n'utilise qu'un fil, s'épargnant ainsi les effets de couplages de voisins potentiels.

Un NoC utilisant des liaisons séries est proposé dans [78]. Ils utilisent un système asynchrone pour le transfert de la donnée dans le lien inter-routeur afin

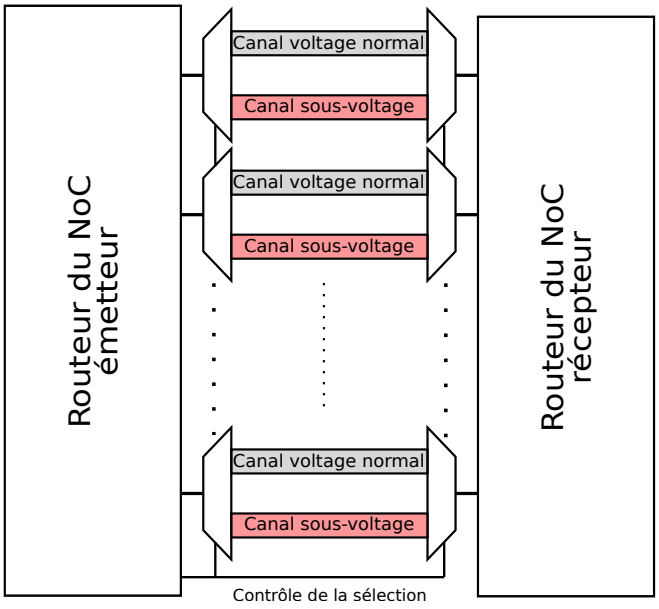


FIGURE 3.11 – Schéma d'implantation du système de transmission au voltage variable.

de rattraper, en termes de bande-passante, une liaison parallèle. Cette technique implique l'addition d'une interface matérielle synchrone / asynchrone et sérialisation / dé-sérialisation pour chacune des liaisons. Elle permet donc de réduire jusqu'à 75 % la puissance dissipée par les liens, cependant ces résultats sont à nuancer. En effet, le flux de données utilisé pour comparer la consommation de la transmission synchrone et asynchrone dans [78] n'utilise qu'une séquence de deux mots (0xA5A5 suivi de 0x5A5A et ainsi de suite...) qui maximise les effets du crosstalk sur les liaisons parallèles, phénomène n'apparaissant pas sur une liaison série. Avec ce procédé d'évaluation, les auteurs se placent dans un cas de figure très favorable à leur proposition, mais qui peut-être considéré comme peu représentatif d'un trafic réel.

Dans [79], les auteurs proposent une technique de codage (optimisation architecturale) créée pour les liaisons séries afin de réduire le nombre de transitions observé sur le lien. Ils changent l'ordre des bits réduisant ainsi le nombre de transitions sur les liens séries, avec comme surcoût matériel l'ajout de lignes en parallèles indiquant les données à permuter au décodeur. Cet encodage des données permet une réduction de 27 % de la dissipation de puissance des liens du NoC, sans pour autant que le surcroît de consommation énergétique du dispositif proposé ne soit pris en compte.

3.5 Bilan

Ce chapitre a présenté un état de l'art sur les différentes techniques d'optimisations de la consommation du NoC. Elles peuvent être appliquées à différents niveaux de l'architecture du NoC. Cette étude nous a permis de faire plusieurs constats :

1. Il y a une corrélation entre les performances d'une topologie et sa consommation d'énergie.
2. Les techniques d'optimisation appliquées aux routeurs visent principalement à réduire la consommation d'énergie statique en limitant les courants de fuite.
3. Il est très difficile de proposer un algorithme induisant une consommation aussi faible que celle du très répandu XY.
4. Les techniques d'optimisation appliquées aux interconnexions cherchent à diminuer la consommation d'énergie dynamique en réduisant le nombre de transitions ou en limitant les transitions croisées.

A partir de ces constats, nous pouvons orienter nos futures propositions visant à minimiser la consommation d'énergie d'un NoC. A priori, la proposition de nouvelles topologies apparaît peu prometteur. D'une part nous avons vu avec le constat 1) que ce que l'on gagne en énergie est systématiquement perdu en performances et d'autre part, la topologie des NoCs a tiré parti de plusieurs décennies de recherches sur les topologies des réseaux informatiques, ce qui rend très difficile la production d'une contribution novatrice et efficace. De la même manière, se lancer dans la proposition d'algorithmes de routage semble compromis. Ce domaine de recherche a également commencé depuis plusieurs décennies et les gains potentiels en énergie sont faibles. Sachant que la contribution en énergie de certains composants du routeur et des interconnexions est très importante comparée à la consommation d'énergie totale, ces options semblent plus prometteuses.

D'après les constats 2) et 4), l'optimisation du routeur revient à diminuer la consommation d'énergie statique du NoC et optimiser les interconnexions intervient sur la consommation dynamique. Le choix peut donc être effectué en fonction de l'évolution de la proportion d'énergie statique et dynamique dans un circuit, pour une technologie donnée. La figure 3.12 montre l'évolution de la puissance dynamique ainsi que les principaux acteurs de la puissance statique que sont I_{grille}, I_{seuil} et I_{diode}. Face à l'augmentation des courants de fuite causée par la miniaturisation des transistors, de nouvelles solutions ont été proposées pour réduire significativement I_{grille}. L'ajout d'un sleep mode dans [80] et par contrôle actif de la polarisation du transistor (pour contrôler sa tension de seuil) avec [81] permet une réduction importante du courant I_{grille}. Concernant I_{seuil} et I_{diode}, il a été impératif de repenser la conception des transistors pour limiter ces problèmes de courants de fuite lorsque la finesse devenait inférieure à 25 nm. La technologie FinFET (Fin Field-Effect Transistor) propose des transistors

conçus pour fonctionner à une extrême finesse de gravure (< 25 nm) et ses propriétés physiques réduisent considérablement les courants de fuite I_{seuil} et I_{diode} jusqu'à 2 ordres de magnitude comparé à un transistor MOSFET Bulk classique [82, 83]. En conclusion, grâce aux améliorations apportées au design des transistors, l'augmentation des courants de fuite a été limité bien que ces courants restent un problème. Par conséquent, la puissance dynamique reste prépondérante dans la consommation d'une puce.

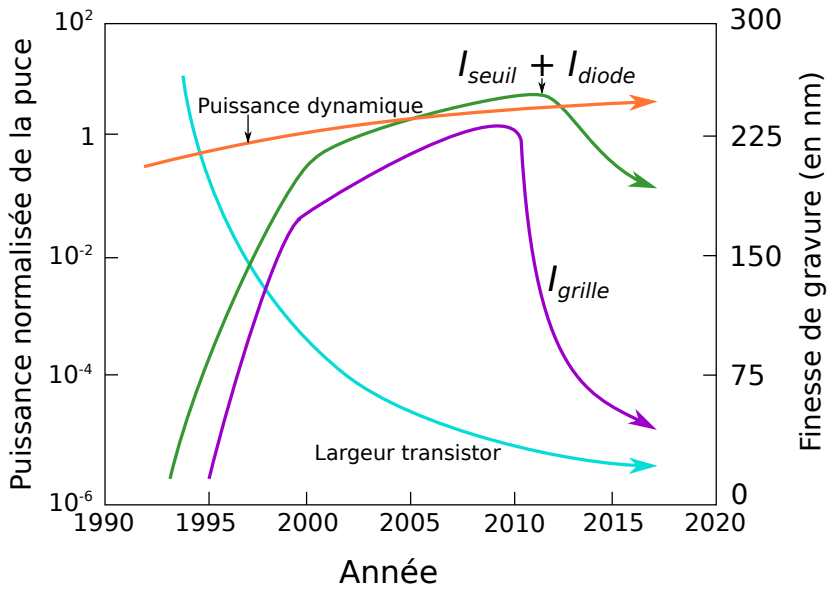


FIGURE 3.12 – Évolution de la consommation dynamique et statique d'une puce. Cette figure a été adaptée de la figure de [84]. Les résultats viennent de prédictions plus récentes d'ITRS [85].

Le dernier constat nous conforte dans le choix consistant à proposer des optimisations au niveau des interconnexions afin de réduire la consommation dynamique. De plus, nous avons vu dans le chapitre 2 que la part de la consommation des interconnexions est conséquente comparée aux autres composants du réseau. Enfin, les optimisations effectuées au niveau des liens sont compatibles avec les autres techniques employées ailleurs pour réduire la consommation du NoC.

4

Propositions d'optimisations de la consommation d'énergie au niveau des interconnexions

Sommaire

4.1 Introduction

Ce chapitre présente les techniques d'optimisations des interconnexions que nous avons développées en nous basant sur les observations du chapitre 2 et les réflexions du chapitre 3. Ces techniques optimisent la consommation d'énergie des liens en réduisant le nombre de transitions croisées et en diminuant le taux de commutation. Les techniques mises en œuvre pour minimiser la consommation d'énergie produisent parfois des effets collatéraux négatifs sur les performances, comme le délai ou la bande passante disponible. Cela fait apparaître la nécessité de trouver un compromis acceptable entre les économies d'énergie et la dégradation de performances. Afin d'évaluer objectivement ces techniques, des simulations sont réalisées en utilisant Noxim-XT, tout en prenant en compte leur surcoût matériel.

4.2 Smart Temporal Shielding

L'optimisation de la consommation des interconnexions présentée ici est basée sur une technique utilisée dans les bus de communication, le Temporal Shielding. Elle est décrite dans le paragraphe suivant.

4.2.1 Temporal Shielding sur bus

Le Temporal Shielding est proposé dans [86], les auteurs émettent une donnée fictive (shield), intercalée entre deux données utiles, afin de supprimer les transitions croisées. La figure 4.1 montre l'insertion d'un shield et comment celui-ci supprime les transitions croisées.Il suffit que ce shield soit un mot constitué uniquement de zéros pour empêcher l'apparition simultanée de transitions montantes et descendantes. Ainsi, les transitions responsables des consommations énergétiques les plus élevées ne peuvent plus apparaître. Pour éviter de doubler le délai de transfert des données, la taille du bus de données est doublée pour envoyer deux données en un seul cycle d'horloge, et au cycle d'horloge suivant un shield est émis afin d'élimer toute transition croisée potentielle.

Il existe également une autre version du Temporal Shielding [87]. Contrairement à la proposition précédente, la taille du bus n'est pas doublée, au lieu de cela, un shield est placé systématiquement entre chaque donnée pour éviter les transitions croisées comme montré à la figure 4.2. L'inconvénient de cette version est que le temps de transmission est doublé.

Ces techniques développées sur les bus présentent soit un important surcoût en surface (bus doublé) ou temporel (blindage systématique toutes les deux données). De plus, la génération du même mot de blindage peut augmenter le taux de commutation des fils. Les deux contributions proposent un

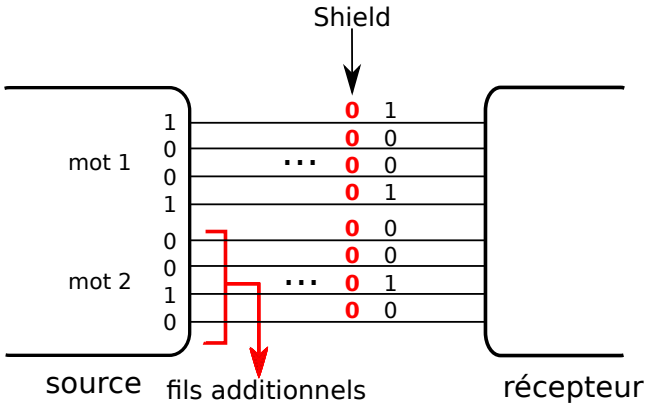


FIGURE 4.1 – Shielding Temporel [86] sur les bus. La taille originale du bus est de 4 bits.

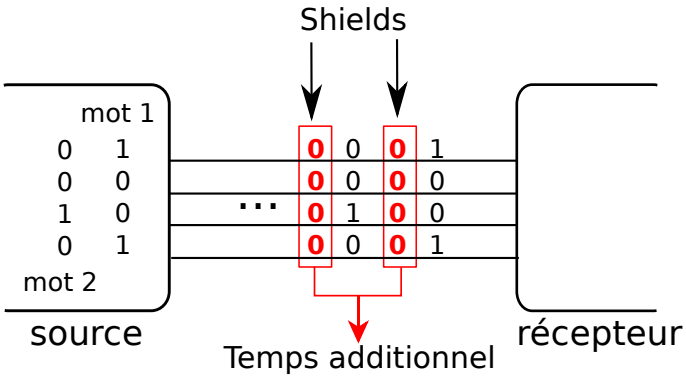


FIGURE 4.2 – Bus encodé utilisant la technique de codage *Code 0* dans [87].

blindage constitué de zéros selon la technique *Code 0* montrée à la figure 4.2. Le tableau 4.1a montre une transmission par défaut sans blindage des données sur 4 bits : `1010` et `1100`. Nous constatons l'apparition d'une transition croisée impliquant le deuxième et le troisième bit, alors que le premier bit et le dernier restent inchangés. Le tableau 4.1b montre que l'insertion du shield provoque une transition descendante suivie d'une transition montante au niveau du premier bit, alors que ce bit serait resté à l'état haut en l'absence de shield. Ce que nous proposons avec le Smart Temporal Shielding est d'adapter le Temporal Shielding (TS) dans les NoCs. Pour ce faire nous avons choisi la version du TS ne doublant pas les interconnexions du bus afin de minimiser le coût en surface. Ensuite, nous améliorons cette technique en réduisant son impact négatif en termes de délai tout en économisant la même quantité d'énergie.

TABLEAU 4.1 – Exemple de l'augmentation de l'activité des liens avec la technique de codage *Code 0*.

Cycle d'horloge i	mot w_i	Type
0	1010	donnée
1	1100	donnée

(a) Transmission par défaut.

Cycle d'horloge i	mot w_i	Type
0	**1**010	donnée
1	**0**000	**shield**
2	**1**100	donnée

(b) Transmission avec *Code 0*.

4.2.2 Analyse de la génération du blindage et de l'optimalité

4.2.2.1 Génération du blindage

Étant donné que la génération d'un shield identique peut augmenter l'activité des fils, il est nécessaire que le blindage soit créé en fonction des données courantes. Nous proposons une méthode simple de blindage avec l'algorithme suivant. Soit A une ligne de 1 bit du signal à transférer et S_A le bit de sortie du blindage de la ligne A. S_A est égal à 0 seulement si la valeur courante et précédente de A est égale à 0, sinon S_A est égal à 1. Le tableau 4.2 reprend l'exemple de transfert de données précédent. Il montre que le blindage utilisé enlève bien les transitions croisées comme les techniques de l'état de l'art et que de surcroît, le taux de commutation du premier bit reste identique à ce qu'il était en l'absence de shield. Cela donne à notre technique de blindage les propriétés suivantes :

- Un shield est généré entre chaque donnée
- Les transitions croisées sont supprimées
- L'activité des fils n'augmente pas
- Le nombre d'interconnexions n'est pas doublé
- Le délai de transfert est doublé

TABLEAU 4.2 – Exemple de blindage avec la méthode proposée.

Cycle d'horloge i	mot w_i	Type
0	**1**010	donnée
1	**1**110	**shield**
2	**1**100	donnée

L'architecture matérielle nécessaire à ce blindage pour 1 bit est illustrée à la figure 4.3. A est la ligne de donnée avec la donnée courante et S_A est le shield généré. SEL est contrôlé par le routeur pour sélectionner la valeur courante ou le shield. Pour ce blindage, seul une porte logique et un registre 1 bit sont requis par ligne de donnée. Ainsi, le surcoût matériel de cette méthode de blindage

est faible. En considérant un routeur avec 4 sorties non-locales, le blindage représente une augmentation de la surface de 4.91% du routeur configuré avec des buffers de profondeur 4 flits et des flits de 32 bits.

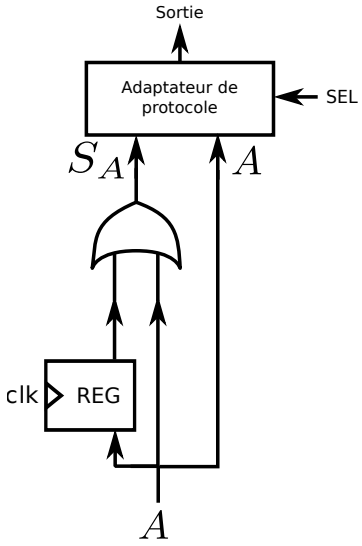


FIGURE 4.3 – Architecture au niveau porte logique de la technique de blindage.

Cette méthode permet de générer un blindage efficace à bas coût (matériel), néanmoins nous ne savons pas si le shield produit conduit à la solution la moins énergivore. Au sein des contributions sur le TS, il n'y a aucune évaluation de la qualité du blindage généré. Ainsi, dans le but d'évaluer notre proposition, nous effectuons une évaluation de la technique de blindage en cherchant le blindage optimal entre deux données.

4.2.3 Étude statistique de l'optimalité

Cette étude est réalisée en énumérant toutes les possibilités de blindage entre chaque couple de mots de N bits. Par conséquent, pour deux mots de N bits, il y a 2^{2N} couples de mots possibles et 2^N blindages différents. Nous considérons que toutes les combinaisons sont équiprobables. Cela nous donne au final 2^{3N} configurations à tester. Sachant que l'ensemble des combinaisons à tester évolue exponentiellement avec la taille des mots N, la longueur maximale évaluée est de 9 bits du fait du temps de calcul devenant trop important au-delà de cette valeur.

La consommation d'énergie de chaque configuration "donnée-shield-donnée" est déterminée en utilisant le modèle d'interconnexion inclus dans Noxim-XT étudié dans le chapitre 2. De cette manière, une évaluation du blindage peut être effectuée en comparant l'économie d'énergie du blindage proposé avec le

blindage optimal. Nous allons maintenant expliquer quelle méthode est utilisée pour trouver le meilleur blindage.

4.2.3.1 Identification du blindage optimal

Le blindage optimal est identifié en transformant le problème en un problème de plus court chemin dans un graphe. La figure 4.4 montre toutes les transitions possibles entre deux mots de taille $N = 2$. Chaque sommet représente un mot tandis que les arêtes symbolisent les transitions dont le poids est l'énergie requise pour la transition. Les valeurs correspondent à une interconnexion de 1 mm, avec une technologie de 65 nm. Nous pouvons observer que le meilleur chemin de `10` à `01` ne consiste pas à utiliser l'arête qui les connecte directement mais plutôt à prendre 2 arêtes dont la somme en énergie donne 150.56 + 13.66 = 164.22 fJ au lieu de 221.86 fJ avec le chemin direct. Sachant que la taille du graphe augmente avec la taille des mots, la meilleure solution n'est peut être pas l'utilisation d'un mot intermédiaire mais l'utilisation d'une série de mots de blindage. Ainsi, l'algorithme utilisé pour explorer toutes les solutions est celui de Dijkstra qui détermine le chemin le plus court (c'est-à-dire le moins coûteux) entre chaque couple de mots.

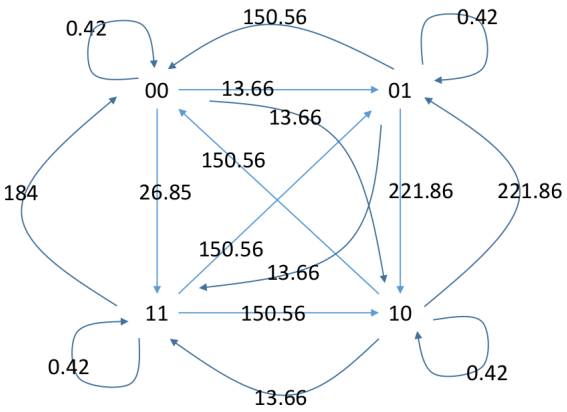


FIGURE 4.4 – Représentation du coût énergétique (en fJ) de chaque transition pour des mots de données de taille $N = 2$. Les sommets représentent les mots et les arêtes sont les transitions avec le coût énergétique indiqué.

La figure 4.5 donne le taux des solutions optimales en fonction de leur nombre de shields. Les expériences sont effectuées de 2 à 9 bits. Nous pouvons voir que le taux de solutions optimales ayant besoin de shields pour minimiser la consommation augmente avec la taille des mots, pour $N = 2$ il y a seulement 10% de solutions avec 1 shield et cela va jusqu'à 70% des mots qui nécessitent au moins un shield pour $N = 9$. La prochaine étude se penche sur la distance séparant l'économie d'énergie réalisée par la solution optimale et la méthode de blindage proposée.

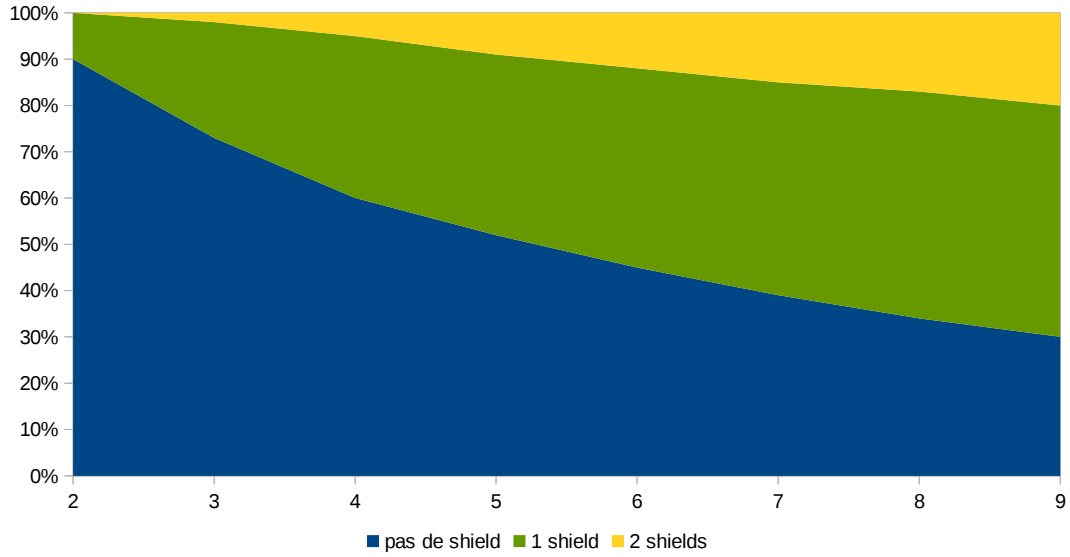


FIGURE 4.5 – Proportion (%) des solutions optimales en fonction du nombre de shield nécessaire. L'abscisse est la taille des mots en bits.

4.2.3.2 Comparaison blindage optimal et le blindage proposé

La figure 4.6 montre la distance moyenne entre les gains d'énergie apportés avec la technique de blindage proposée et le gain maximal pouvant être atteint. Nous rappelons que le gain maximal est déterminé en utilisant l'algorithme de Dijkstra. Comme nous pouvons le constater, la technique proposée est optimale jusqu'à $N = 3$ bits. Ensuite, elle commence à s'éloigner du gain optimal pour se stabiliser autour de 5%. Bien que notre méthode de blindage ne donne pas toujours le gain optimal en énergie, l'écart avec l'optimal est de seulement 5% alors que le surcoût matériel est faible. Ainsi, avec un surcoût matériel faible, la méthode proposée apporte un bon compromis entre efficacité et surface, tout en n'introduisant qu'un seul shield entre deux données utiles, là où la solution optimale peut en requérir deux ou plus.

Étant donné que le blindage permet de gagner de l'énergie que lorsqu'il y a apparition d'une transition croisée, nous proposons de n'activer le blindage que lorsqu'une transition croisée est détectée. De cette manière, le blindage n'est pas activé systématiquement et l'impact de cette méthode sur le délai sera donc réduit. L'architecture est décrite dans le paragraphe suivant.

4.2.4 Architecture du Smart Temporal Shielding

Le Smart Temporal Shielding (STS) regroupe une méthode de blindage couplée avec un détecteur de transitions croisées. Son architecture est illustrée à la figure 4.7. Nous pouvons distinguer trois parties dans le STS : 1) L'étage de mémorisation du mot courant, 2) l'étage du blindage et 3) l'étage d'activation.

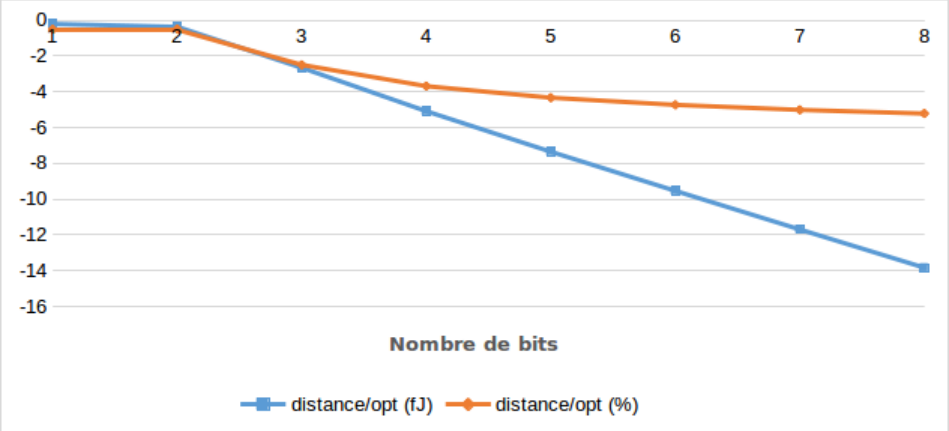


FIGURE 4.6 – Évolution de la distance moyenne entre les gains énergétiques obtenus avec le blindage proposé et la solution optimale.

L'entrée du STS est le flit courant W de N bits et ses sorties sont le shield généré *Shield* et *Shield actif* qui est un signal de 1 bit indiquant si le blindage doit être utilisé. Le premier étage est la mémorisation du dernier flit envoyé, cette information est requise pour la génération du shield et la détection des transitions croisées. Pour cela, nous utilisons un registre 1 bit par ligne de donnée. L'étage 2 est le bloc de blindage qui nécessite une seule porte OU par ligne de donnée. L'étage 3 est le détecteur de transitions croisées, il vérifie si un fil est le siège d'une transition croisée avec chacun de ses voisins. Dès qu'une transition croisée est détectée *Shield actif* passe à 0 [1]. Par conséquent, quand une transition croisée est détectée, l'adaptateur de protocole envoie le flit shield au lieu du flit donnée au routeur suivant. Le signal *Shield actif* est également envoyé au routeur récepteur afin qu'il sache si le flit doit être ignoré ou enregistré. La section suivante implante le STS dans Noxim-XT afin d'évaluer la proposition.

1. Le détecteur est actif-bas car dans ce cas des portes NON-ET peuvent être utilisées nécessitant moins de transistors que les ET.

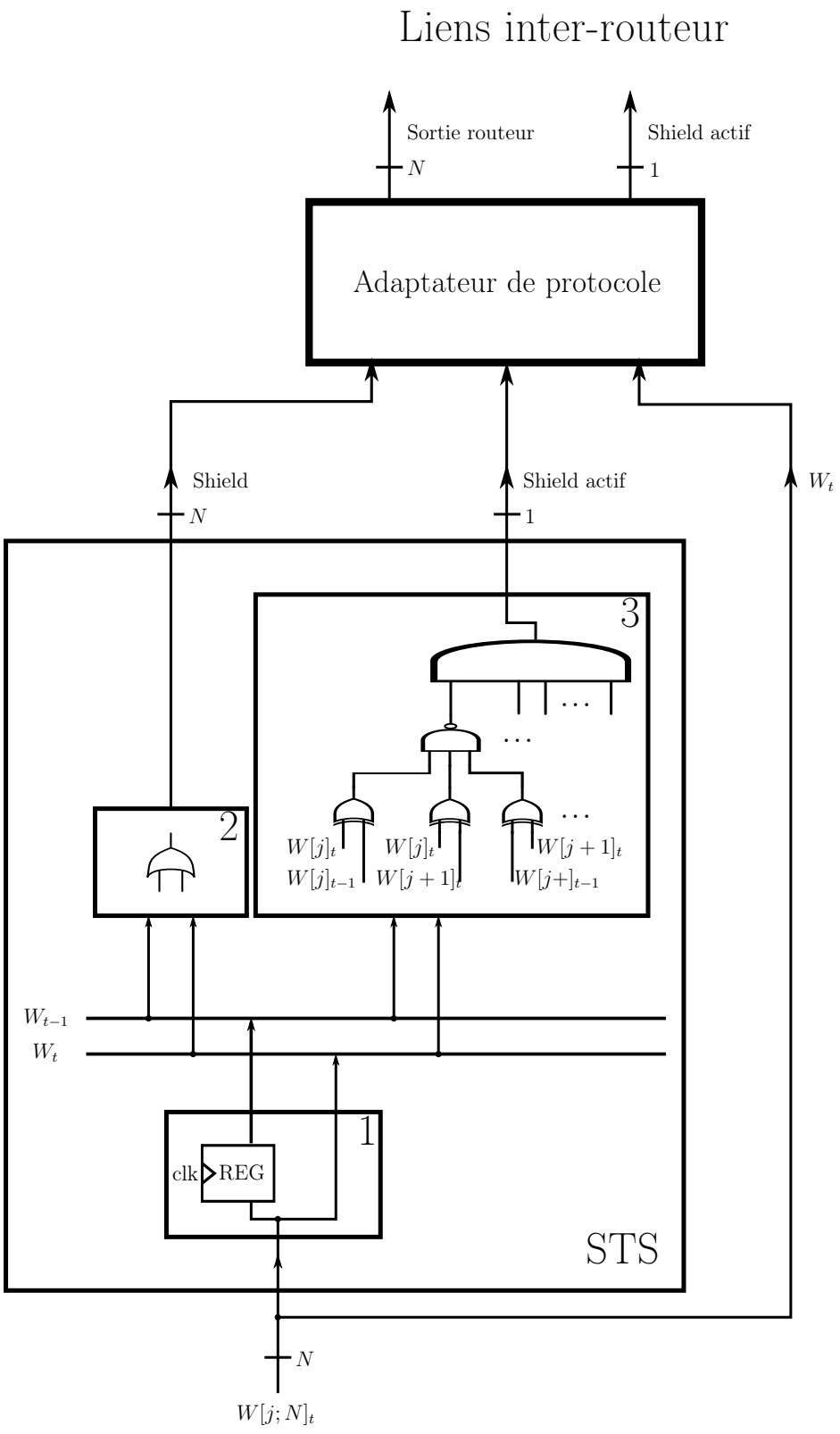


FIGURE 4.7 – Architecture du Smart Temporal Shielding (STS).

4.2.5 Résultats expérimentaux

L'évaluation du STS est réalisée en le comparant au TS de la littérature [87]. Ces deux techniques sont implantées dans Noxim-XT. Le surcoût énergétique (énergie statique et dynamique) des deux techniques est pris en compte et la surface du codeur est estimée. Ces résultats viennent de synthèses réalisées avec Synopsys Design Compiler avec une technologie de 65 nm. La surface d'un bloc STS représente 2.2% par rapport à un routeur. Sachant que le routeur utilisé dans une topologie en grille possède 4 sorties non locales, la surface additionnelle totale que représente quatre fois les blocs de STS est de 8.8%. La puissance totale du routeur est augmentée de 8%, ce qui est faible comparé à la consommation des interconnexions et des buffers du routeur. L'étage de détection de transitions croisées du STS représente 45% de la surface totale de celui-ci (Le TS a seulement besoin des étages nommés mémoire et blindage). Tous les résultats de synthèses sont résumés dans le tableau 4.3.

TABLEAU 4.3 – Surcoût en énergie et en surface du routeur selon sa configuration avec le STS par rapport au routeur classique. La colonne colorée est la configuration utilisée pour les simulations.

Taille des flits (bits)	8	16	32	32	32	64	64	64
Profondeur buffer (flits)	4	4	4	8	12	4	8	12
Surface étage mémoire (%)	2.24	2.56	2.81	1.71	1.519	3.41	2.05	1.8
Surface étage blindage (%)	1.54	1.76	2.1	1.33	1.176	2.64	1.59	1.4
Surface étage d'activation (%)	3.15	3.6	3.91	2.49	2.2	4.9	2.97	2.6
Surface routeur additionnelle (%)	**7**	**8**	**8.8**	**5.53**	**4.9**	**11.02**	**6.612**	**5.8**
Puissance routeur additionnelle (%)	**6.3**	**7.4**	**8**	**4.85**	**4.6**	**9.8**	**5.2**	**5.9**

4.2.5.1 Configuration du simulateur

Les simulations sont effectuées avec un NoC configuré comme suit :
- Topologie en grille-2D de taille 4 × 4, horloge à 1 GHz
- Algorithme de routage XY
- Taille des flits : 32 bits, largeur liens inter-routeur 32 bits
- Profondeur des buffers de 4 flits
- Longueur interconnexions : 2 mm

L'origine et la destination des paquets est choisie aléatoirement avec un trafic uniforme aléatoire. Les simulations se font sur un million de cycles et sont répétées 10 fois dans chaque configuration. Une moyenne des simulations est réalisée afin de lisser les résultats du fait du caractère aléatoire du trafic. Cela nous donne d'après les outils statistiques de R [45] un niveau de confiance de 99%.

Nous utilisons quatre motifs de données dans les expériences utilisant des trafics synthétiques. Les deux premiers motifs sont utilisés pour délimiter la consommation maximale et minimale des interconnexions. Pour cela, sont re-

pris dans le tableau 2.5 du chapitre 2 le motif Meilleur cas ($\alpha = 0$) et le Pire cas ($\alpha = 1$). Dans le but d'évaluer le système d'activation du blindage du STS, deux autres motifs de données sont utilisés. Le troisième motif est First Random, qui remplit aléatoirement chaque flit, ainsi chaque bit du flit a 50% de chance de commuter. Le dernier motif, Second Random, est un motif pseudo-aléatoire qui génère des flits permettant un taux d'apparition d'une transition croisée de 50%.

Nous avons également réalisé des simulations avec des trafics d'applications. Les applications utilisées sont Dijkstra, 5 applications, clusters applications, Full HD et Dual Full HD vu au chapitre 2. Dual Full HD est l'application Full HD sauf que 2 flux vidéos passent à travers le NoC afin de stresser le NoC avec un débit encore plus intense. Les différences de configuration sont les suivantes :

— La topologie utilisée est 4×3 ou 4×2 selon la configuration requise par l'application
— Le temps de simulation va jusqu'à la fin de l'application
— Le contenu des données est issu des traces d'applications

4.2.5.2 Résultats de simulations

Analyse des performances : délai La figure 4.8 montre l'impact du STS et du TS sur le délai de transmission moyen du NoC. La comparaison est réalisée en fonction de la technique utilisée et du motif de donnée alimentant les flits. Pour commencer, le NoC classique et le NoC avec le TS n'ont qu'une seule courbe car leurs performances ne dépendent pas de la nature des données, contrairement au NoC avec STS. Comme nous pouvons le constater, le NoC classique est en état de congestion quand le PIR dépasse 0.030 tandis qu'avec le TS la saturation arrive à $PIR = 0.015$. Il sature donc deux fois plus vite, ce qui est tout à fait normal car nous générons deux fois plus de trafic. En revanche, les performances lors du STS vont du NoC classique au TS. Lorsque le motif de données Meilleur cas est utilisé, le blindage n'est jamais activé puisqu'il n'y a pas de transitions croisées, d'où le fait que les performances soit identiques au NoC classique. De la même manière, avec le motif Pire cas, il y a systématiquement des transitions croisées et le blindage est constamment activé.

Avec le motif de données First Random, le STS surpasse TS en performances d'environ 5%. Avec le motif Second Random, les performances du STS se situent entre le TS et le NoC classique. Dans ce cas, le STS est en état de congestion à $PIR = 0.020$ permettant de gérer un débit 33% plus important qu'avec le TS, limitant ainsi l'impact négatif sur le délai. Le taux d'utilisation des shields est extrait des simulations afin d'évaluer l'impact de la stratégie d'activation du STS.

La figure 4.9 montre le taux d'utilisation du shield du TS et STS selon

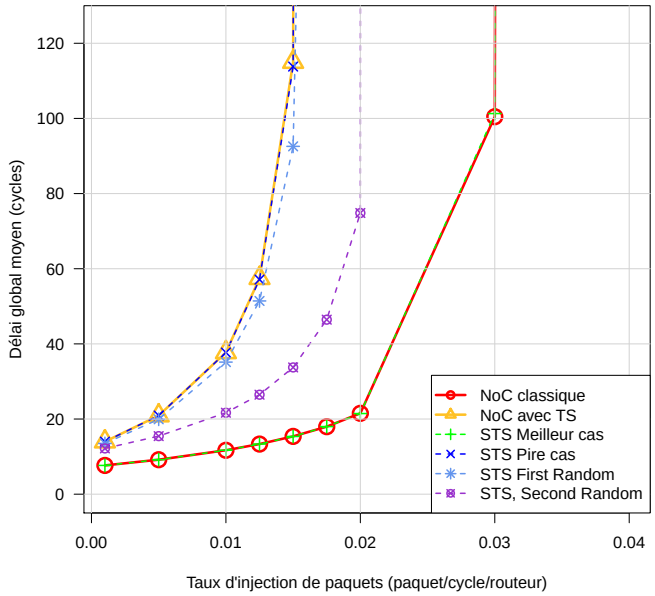


FIGURE 4.8 – Délai global moyen du NoC selon l'intensité du trafic (PIR).

le motif de données et le PIR. Évidemment, le TS a un taux d'utilisation de 100% puisque le blindage est systématique. Quant au STS, son utilisation varie de 0 à 100% pour les motifs de Meilleur et Pire cas respectivement. Le taux d'utilisation est de 95% lorsque le motif First Random est utilisé et passe à 58% avec Second Random. Cette chute du taux d'utilisation du blindage explique les résultats obtenus avec le délai dans la figure 4.8.

La figure 4.10a montre le taux d'utilisation du shield pour le TS et STS avec des trafics issus d'applications. Nous pouvons constater que le taux d'utilisation est divisé par 2 pour le STS. Le bénéfice de cette réduction est illustré à la figure 4.10b où le délai du NoC est affiché en fonction de l'application et de la technique utilisée. Le délai peut être réduit jusqu'à 40% pour l'application 5Cluster4 (cluster applications) et pour les autres applications un gain significatif du délai est constaté, sauf pour Full HD. Cette absence de gain en délai est dûe au fait qu'entre le délai du NoC classique et avec TS, il y a seulement 2 cycles d'écarts (4 cycles pour le classique et 6 cycles pour le TS). Par conséquent, le gain en termes de délai ne peuvent être que de 2 cycles.

Dans le cas où les temps d'exécution des applications pouvaient être raccourcis avec une transmission plus rapide dans le NoC, le gain en délai du STS pourrait réduire ce temps d'exécution et permettre une économie d'énergie. Cependant, nous ne pouvons mettre en évidence cet effet à cause de la méthodologie employée pour générer les traces d'applications.

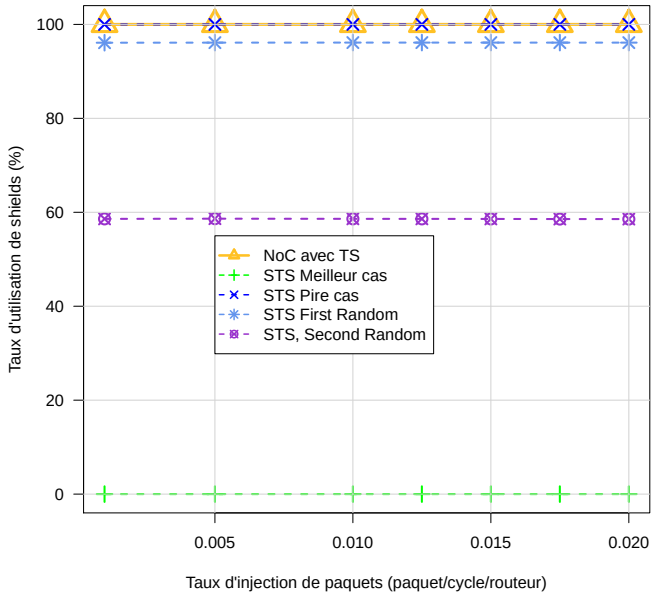
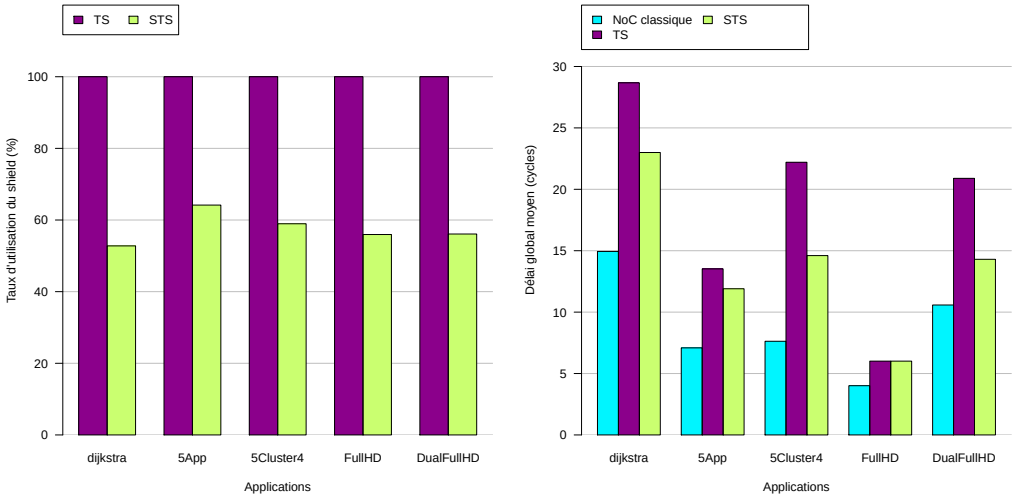


FIGURE 4.9 – Taux d'utilisation du shield selon le PIR et le contenu des flits pour le STS.



(a) Taux d'utilisation du shield (b) Délai global moyen

FIGURE 4.10 – Taux d'utilisation du shield et délai global moyen du NoC avec TS et STS selon l'application simulée.

Analyse des performances : passage à l'échelle Dans le but de compléter l'étude des performances du STS, nous analysons la passage à l'échelle en étudiant l'évolution du débit maximal selon la taille du réseau. Cette métrique est calculée avec le débit de saturation, l'équation est présentée ci-dessous :

$$Débit\ Réseau = Débit\ Saturation \times Nb.\ de\ nuds \tag{4.1}$$

Le débit de saturation est défini comme le taux d'injection de paquet (PIR) où la latence moyenne des paquets devient supérieure à deux fois la latence à charge nulle du NoC (*i.e.*, la latence moyenne des paquets quand il n'y a pas de congestion importante dans le réseau). La figure 4.11 montre l'évolution du débit maximal pouvant être accepté par le réseau avec et sans STS. Comme montré avec l'étude précédente sur le délai, le NoC 4×4 avec le STS a des performances variant avec la nature des données. Pour plus de clarté, seuls les deux motifs de données First Random et Second Random sont affichés pour le STS. Nous pouvons constater avec la figure 4.11 que les performances avec STS évoluent de la même manière que le NoC classique, bien que le débit maximal soit plus bas. En effet, le STS avec le motif Second Random oscille entre 20% (4×4) et 30% (12×12) en-dessous du débit du NoC classique. Cela prouve que le STS peut être déployé sur des réseaux de plus grande taille. Nous allons maintenant nous intéresser au gain d'énergie apporté par le STS.

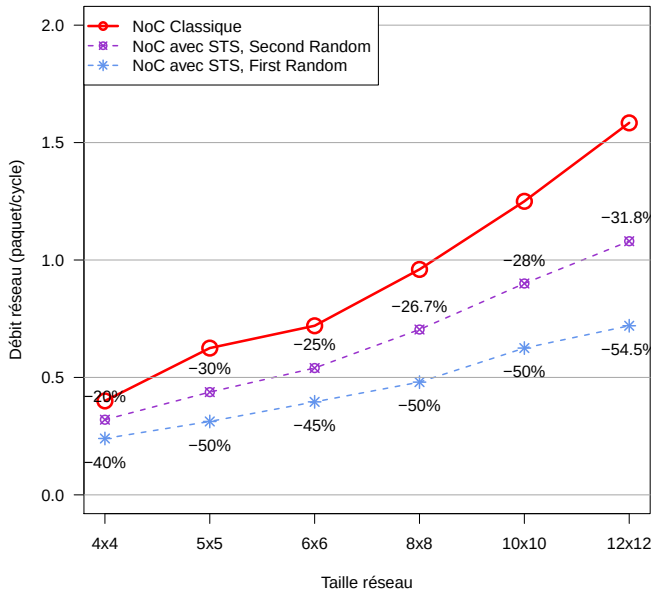


FIGURE 4.11 – Passage à l'échelle du STS en mesurant le débit maximum atteignable sous différentes tailles de réseaux et motifs de données.

Analyse de la consommation d'énergie Cette étude compare les gains d'énergie du TS et du STS dans un NoC. Le tableau 4.4 montre la réduction de la consommation d'énergie des liens pouvant être atteinte avec le TS et STS sous différents motifs de données et PIR.

Avec le motif Meilleur cas, la consommation des liens avec le TS augmente de 50%. Cela est causé par le fait qu'avec ce motif où les données à transmettre ne sont constituées que de zéros, l'énergie consommée par les liens est principalement statique. Puisque le TS double le temps de transmission des données avec le blindage (inutile ici), l'énergie statique des liens est doublée. Concernant le STS sous ce même motif de données, ce dernier n'active pas le shield donc il n'y a pas de pertes d'énergie. De plus, le STS peut fonctionner à un débit équivalent au NoC classique dans ce cas contrairement au TS.

Avec le motif de Pire cas, le TS et STS apportent tous deux un gain d'énergie d'environ 50% et le PIR maximal est limité à 0.015 du fait de l'utilisation constante du blindage. De la même manière, pour le motif First Random, le TS et le STS ont des gains en énergie similaires autour de 15.5%.

Lorsque les données sont générées par le motif Second Random, de $PIR = 0.001$ à $PIR = 0.015$, le TS et STS permettent de réduire l'énergie des liens de respectivement 10.3% et 11.3%. Le STS est capable de maintenir un trafic plus important puisqu'il peut atteindre un PIR de 0.020 tout en conservant un gain énergétique de 11.5%.

TABLEAU 4.4 – Gain en énergie des liens avec le TS et le STS dans un NoC selon le motif de données et l'intensité du trafic (PIR). La référence de ces résultats est le même NoC sans technique d'optimisation.

	Économie d'énergie (%) selon le PIR							
PIR (flit/cycle/routeur) / **Configuration**	**0.001**	**0.005**	**0.01**	**0.0125**	**0.015**	**0.0175**	**0.02**	**0.03**
TS Meilleur cas	-50	-49.9	-50	-50	-50			
TS Second Random	10.3	10.3	10.2	10.2	10.5			
TS First Random	15.9	15.6	16	15.8	16.1			
TS Pire cas	50.7	50.6	51.2	50.6	50.8			
STS Meilleur cas	-0.2	-0.1	-0.01	-0.1	0.25	-0.1	-0.02	-0.01
STS Second Random	11.3	11.5	11.2	11.3	11.3	11.5	11.5	
STS First Random	15.5	15.3	15.5	15.5	15.3			
STS Pire cas	50.6	50.3	50.8	50.9	50.8			

: Trafic non supporté.

Le tableau 4.5 montre les gains sur l'énergie totale du NoC apportés par le TS et STS selon la nature des données et le PIR. Le TS sous motif Meilleur cas engendre une perte d'énergie de 2.2% à 4.1% pour le NoC du fait des shields inutiles transférés et son surcoût matériel. La perte énergétique est plus faible pour le STS puisque le blindage n'est pas activé. Les résultats avec les autres motifs de données sont les suivants :

— Pire cas, le TS et STS apportent un gain d'énergie du NoC de 42% à 49%

— First Random, le TS et STS apportent un gain d'environ 14.5%
— Second Random, le TS et STS apportent un gain d'environ 9%

Si nous regardons de plus près les résultats du STS, nous pouvons voir que les gains sont quasiment les mêmes que le TS, avec une légère différence causée par le surcoût matériel du détecteur de transitions croisées. Cependant, le STS gère un débit plus important qu'en utilisant le TS étant donné le nombre de shields réduit, ce qui permet d'obtenir des gains d'énergie plus élevé. Ces derniers résultats nous montrent que le TS et le STS permettent de réduire significativement l'énergie du NoC. L'avantage étant donné au STS qui avec des gains similaires au TS, permet de gérer un trafic plus important.

TABLEAU 4.5 – Gain d'énergie total avec le TS et le STS dans un NoC selon le motif de données et l'intensité du trafic (PIR). La référence de ces résultats est le même NoC sans technique d'optimisation.

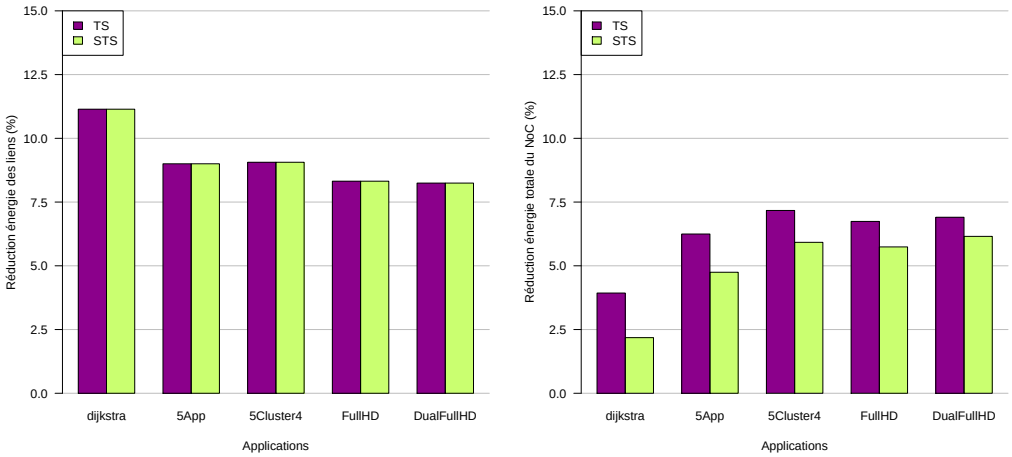
	Économie d'énergie (%) selon le PIR							
PIR (flit/cycle/routeur) / **Configuration**	**0.001**	**0.005**	**0.01**	**0.0125**	**0.015**	**0.0175**	**0.02**	**0.03**
TS Meilleur cas	-2.2	-2.6	-3.3	-4	-4.1			
TS Second Random	6	8.3	8.6	8.9	9.1			
TS First Random	10.5	13.6	14.5	14.4	14.6			
TS Pire cas	42	47.5	49	48.5	48.6			
STS Meilleur cas	-2.2	-2.3	-2.35	-2.3	-2.35	-2.35	-2.3	-2.25
STS Second Random	6.5	8.3	8.7	8.9	9.2	9.4	9.6	
STS First Random	9.5	12.2	12.75	13.1	13.8			
STS Pire cas	41	45.5	47	47.2	47.5			

: Trafic non supporté.

La figure 4.12a montre la réduction d'énergie des liens selon l'application et la technique utilisée. Nous pouvons observer que les gains du STS et ST sont quasiment identiques. Comparés aux données synthétiques, les gains d'énergie sont légèrement inférieurs au motif Second Random avec de 8% à 11% de réduction d'énergie pour les liens.

La figure 4.12b montre les gains apportés par le STS et le TS sur l'énergie totale du NoC. Les gains peuvent aller de 4% à 7.5%. Les gains avec l'application Dijkstra sont faibles car le trafic généré par cette application est également faible, par conséquent l'énergie statique du NoC domine. Pour les autres applications, l'énergie économisée est d'environ 7%, ce qui est légèrement plus faible qu'avec les données synthétiques (motif Second Random). Il peut être noté que le STS génère des gains plus faibles que le TS (entre 0.75% et 1.75%) du fait du surcoût matériel.

Nous venons de présenter le STS, l'amélioration du TS et son adaptation dans un NoC. Cette technique permet de réduire significativement l'impact sur le délai du blindage en ne l'activant que lorsque c'est utile (les gains légèrement plus bas du STS étant dû au surcoût matériel). De plus, les gains observés restent du même ordre qu'avec la technique d'origine. Le STS s'étant concentré sur la suppression des transitions croisées au prix d'un délai plus important.



(a) Réduction d'énergie des liens (b) Réduction d'énergie du NoC

FIGURE 4.12 – Gains d'énergie avec le TS et STS avec trafics d'applications.

Une autre façon d'optimiser l'énergie des liens est de limiter l'activité des liens. La prochaine méthode d'optimisation présentée mettra en œuvre cette solution.

4.3 Communication inspirée par le Cortex

4.3.1 Contexte

L'idée de cette technique de codage est issue d'observations réalisées sur les trafics d'applications vus dans le chapitre 2 et utilisées pour évaluer nos précédentes techniques dans ce chapitre. Nous avons vu que les demandes en bande-passante des applications peuvent être faibles, comme avec Dijkstra et 5 Applications. De plus, certaines applications ont des besoins en bande-passante très variables en cours d'exécution : c'est le cas pour Cluster Applications.

A partir de ces observations, il nous a semblé intéressant de proposer une méthode de transmission à très bas coût énergétique lorsque les besoins en bande-passante du NoC le permettent. La première étape est de trouver une technique de codage efficace d'un point de vue énergétique.

4.3.1.1 Communication cérébrale

La technique Communication Inspirée par le Cortex (CIC) est basée sur le principe du fonctionnement du cortex cérébral. En effet, le cerveau humain est soumis à deux contraintes majeures [88] :

1. Une faible quantité d'énergie disponible (*c-à-d* le sucre dans le sang)
2. Une dissipation de puissance thermique maximale de 25 W.

Dans ces conditions, l'évolution a proposé une méthode très efficace d'un point de vue énergétique pour transmettre et traiter les informations sensorielles. P. Lennie montre dans [89] que le cerveau humain stocke et transfère l'information en utilisant des codes de représentation des données dépendant de l'activité d'un petit nombre de neurones. Le taux d'activité des neurones dans le cortex est estimé à 1% du réseau neural complet et parmi les deux hémisphères, seulement 100 fibres nerveuses sont actives en même temps à chaque instant. C'est de cette utilisation parcimonieuse des ressources dont dispose le cerveau que notre technique s'inspire. A une échelle bien plus faible, nous proposons d'imiter ce comportement en ne faisant commuter que quelques fils à chaque cycle d'horloge, de façon à transmettre des données à coût réduit en énergie. Le principe de ce codage est décrit dans le paragraphe suivant.

4.3.1.2 Principe

Supposons qu'il y ait N interconnexions entre deux routeurs d'un NoC. Dans un transfert de données standard, ces lignes permettent de transmettre N bits d'information par cycle. Si nous décidons, entre deux cycles d'horloge, de commuter une seule interconnexion, nous avons une ligne commutant et toutes les autres étant constantes. De cette manière, à chaque cycle d'horloge,

nous pouvons déterminer une position N du fil commutant, qui correspond à transmettre l'équivalent de $log_2(N)$ bits d'information. Nous nous servons donc de la position de la ligne active comme information.

Par exemple, prenons l'état courant d'un bus de $N = 8$ bits suivant $(w_7 w_6 ... w_1 w_0) = (10011101)$, nous pouvons transmettre $log_2(8) = 3$ bits d'information $(b_2 b_1 b_0)$ en faisant commuter seulement la ligne de donnée numéro $k = 4b_2 + 2b_1 + b_0$. Ainsi, si nous souhaitons transmettre l'information (110), c'est la ligne w_6 qui doit commuter et l'état du bus va donc passer de (10011101) à (11011101) et ainsi de suite. Il est également possible d'améliorer ce procédé en "gelant" la ligne w_0 : quand il n'y a pas de commutation observée sur le bus inter-routeur côté récepteur, alors w_0 est considéré comme la donnée transférée correspondant ici à (000). L'architecture de ce codage est détaillée plus tard dans la section.

Dans l'exemple que nous venons de donner, nous avons considéré toutes les interconnexions d'un bus de 8 bits pour le coder. Cependant, il est possible de définir des sections de bus afin de coder plusieurs groupes d'informations en même temps. Cela permet de moduler le gain énergétique et le temps de transmission potentiel du CIC. La figure de mérite du CIC doit être définie.

4.3.2 Figure de mérite du CIC

Reprenons un bus inter-routeur de N bits de large. Nous rappelons que le transfert standard de l'information permet la transmission de N bits par cycle. Considérons des données aléatoires non corrélées, leur taux de commutation sur chaque interconnexion est de $\alpha = 0.5$. D'après l'équation de la puissance dynamique des liens (2.1) vu dans le chapitre 2, le transfert standard sur le bus dépense en moyenne une quantité d'énergie $E_s = \frac{1}{2}E_0$ par bit transmis, où $E_0 = CV_{dd}^2$ est l'énergie moyenne dépensée pour faire commuter l'état d'une interconnexion. Ce modèle se révèle à la fois optimiste et pessimiste. Optimiste, car l'énergie due au crosstalk n'est pas représentée. Pessimiste, car nous considérons la transmission de données aléatoires, dont le taux de commutation est supérieur aux applications réelles. Néanmoins, connaissant les limites de ce modèle, il est possible de dériver la figure de mérite de la technique proposée en fonction de N.

Dans la transmission CIC, une seule transition permet de transmettre $\log_2(N)$ bits d'information. En outre, si la technique de "gel" est appliquée sur w_0, alors nous avons en moyenne $\frac{N-1}{N}$ transitions requises pour transmettre $\log_2(N)$ bits. Le coût énergétique moyen avec le CIC est $E_c = \frac{N-1}{N\log_2(N)}E_0$ J/bit. Ainsi, le facteur de réduction d'énergie est

$$\gamma(N) = 1 - E_c/E_s = \frac{2(N-1)}{N\log_2(N)}. \tag{4.2}$$

Puis, en termes de débit de données, la réduction du débit est

$$\delta(N) = 1 - \frac{\log_2(N)}{N}. \tag{4.3}$$

Dans un cas d'application réel, le nombre d'interconnexions du bus inter-routeur N_w est fixe. Le seul degré de liberté dans l'application du CIC est le partitionnement $\mathcal{Q}$ des N_w interconnexions en q sous-ensembles $\mathcal{Q} = \{N_k\}_{k=1\ldots q}$, avec la somme des N_k, $k = 1 \ldots q$ qui est égale à N_w. Il est important de noter que puisque w_0 est gelé, il agit comme un shield entre les interconnexions des différentes partitions du bus, évitant ainsi les transitions croisées. Pour une partition $\mathcal{Q}$ donnée, $T_\mathcal{Q}$ est le nombre de bits transmis par cycle d'horloge et $E_\mathcal{Q}$ le coût énergétique moyen par bit ; ils sont exprimés via les équations suivantes :

$$T_\mathcal{Q} = \sum_{k=1}^{q} \log_2(N_k) \quad ; \quad E_\mathcal{Q} = \frac{1}{T_\mathcal{Q}} \sum_{k=1}^{q} \frac{N_k - 1}{N_k} E_0. \tag{4.4}$$

Le tableau 4.6 montre l'impact de différentes partitions d'un bus inter-routeur de $N_w = 32$ bits de large. La configuration $\mathcal{Q} = \{16, 16\}$ est utilisée comme preuve de concept. Comparée à la transmission standard, elle offre une réduction potentielle d'énergie de 53.2% tout en divisant par 4 le débit initial (75%). Cela signifie qu'un flit de 32 bits est transmis en 4 cycles d'horloge. L'architecture de la partition $\mathcal{Q} = \{16, 16\}$ est décrite dans la sous-section suivante.

TABLEAU 4.6 – Débit et coût énergétique par bit selon la partition du bus inter-routeur.

Type de transmission	Énergie par bit	$T_\mathcal{Q}$	γ	δ
Standard	0.5 E_0	32	0 %	0 %
$\mathcal{Q} = \{32\}$	0.2 E_0	5	61.3 %	84.4 %
$\mathcal{Q} = \{16, 16\}$	0.234 E_0	8	53.2 %	75 %
$\mathcal{Q} = \{8, 8, 8, 8\}$	0.292 E_0	12	41.6 %	62.5 %
$\mathcal{Q} = \{4, \ (\times 8)\}$	0.375 E_0	16	25 %	50 %

4.3.3 Architecture

L'architecture du CIC pour $N = 16$ est présenté à la figure 4.13. L'encodeur one-hot reçoit comme entrée 4 bits (f_3, f_2, f_1, f_0) qui donne en sortie le signal a_F à 1, avec $F = 8_f 3 + 4f_2 + 2f_1 + f_0$, et les autres sorties a_i sont à 0 [2]. Ensuite, la valeur des lignes inter-routeur est définie par une porte logique XOR ayant

2. Nous rappelons que pour a_0 la technique du "gel" du signal est utilisée.

pour entrée la valeur courante de la ligne et sa valeur précédente stockée dans un registre 1 bit (illustré par la boîte $[Z^{-1}]$ à la figure 4.13). Cette étape est nommée *Générateur de front*. Du côté du récepteur, le principe du décodeur est symétrique. Premièrement est détecté la ligne qui a commuté avec l'étage du *Détecteur de front*, puis une fois le signal one-hot reconstitué, les 4 bits sont restitués.

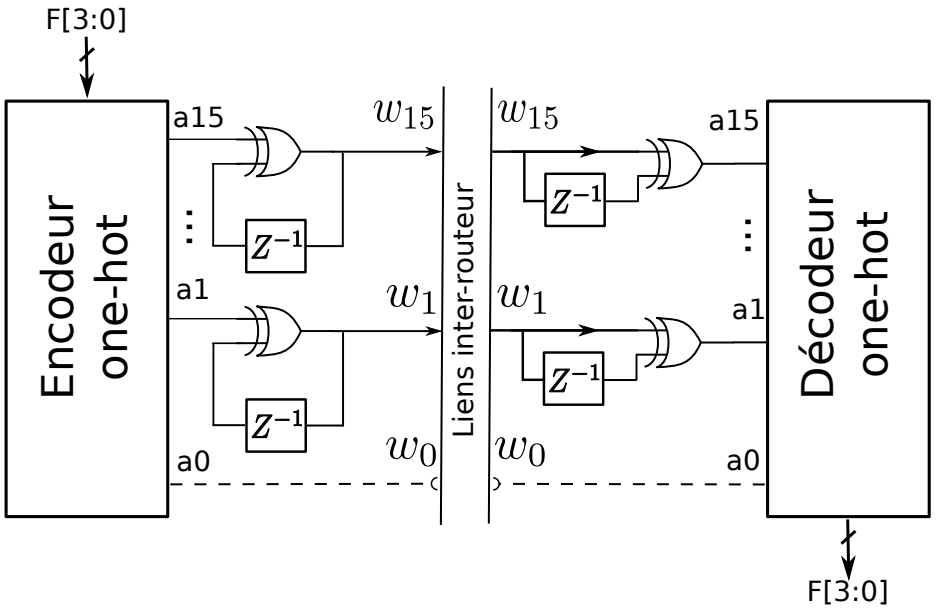


FIGURE 4.13 – Encodeur CIC avec les détails au niveau porte logique du générateur et détecteur de front.

Étant donné que dans le NoC étudié, la taille des flits est de 32 bits, un adaptateur entre le flit et le CIC est requis afin de découper le flit. La figure 4.14 montre deux encodeurs CIC pour $N = 16$ avec leur adaptateur (configuration $\mathcal{Q} = \{16, 16\}$). Du côté de l'émission, des multiplexeurs sérialisent le flit en entrée. Les deux multiplexeurs sélectionnent 4 bits chacun parmi les 32 bits du flit et sont envoyés en parallèle aux deux encodeurs CIC. Pour récupérer le flit complet à la partie décodeur, un registre à décalage est utilisé pour concaténer chaque sous-partie décodée du flit envoyé et le restaurer complètement au routeur.

L'architecture du CIC avec la configuration $\mathcal{Q} = \{16, 16\}$ est implantée dans un simulateur afin d'évaluer les gains potentiels d'énergie sur les liens et son impact sur le débit du réseau.

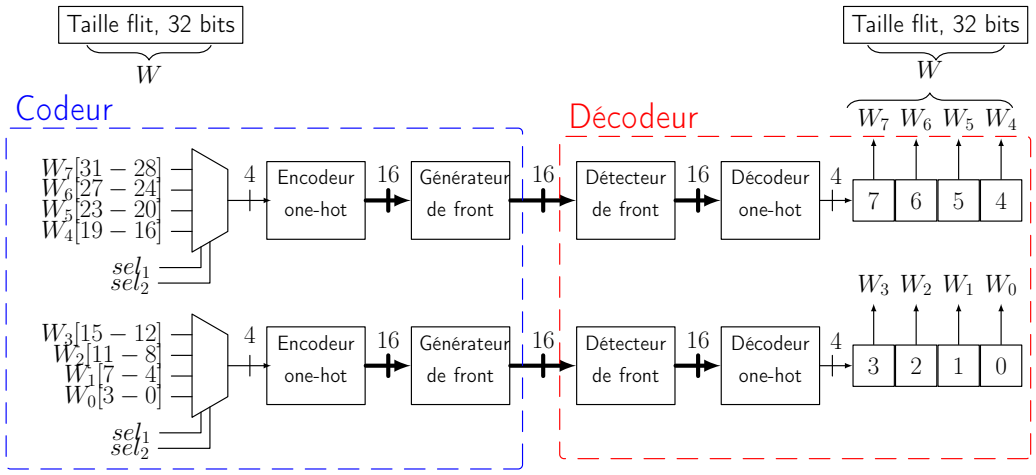


FIGURE 4.14 – Architecture du CIC et son adaptateur, configuration $\mathcal{Q} = \{16, 16\}$

4.3.4 Évaluation du CIC

Cette étude préliminaire vise à présenter l'impact du CIC sur le NoC lorsque cette transmission est utilisée constamment. Les performances en termes de délai et de consommation d'énergie du CIC sont évaluées avec des trafics synthétiques. Dans cette étude, nous ne considérons pas le surcoût matériel car seule l'énergie des interconnexions avec et sans codage est étudiée afin de connaître les gains potentiels de cette technique. Le simulateur de NoC employé pour ces simulations est Noxim-XT.

4.3.4.1 Configuration du NoC

Les simulations sont effectuées avec un NoC configuré comme suit :
- Topologie en grille-2D de taille 8 × 8, horloge à 1 GHz
- Algorithme de routage XY
- Taille des flits : 32 bits, largeur liens inter-routeur 32 bits
- Profonfeur des buffers de 4 flits
- Longueur interconnexion : 2 mm

L'origine et la destination des paquets est choisie aléatoirement avec un trafic uniforme aléatoire. Les simulations se font jusqu'à que 1 Mo de données soit transféré et sont répétées 10 fois dans chaque configuration. Afin d'avoir des résultats fiables, une moyenne des simulations de chaque configuration est réalisée. Cela nous donne d'après les outils statistiques de R [45] un niveau de confiance de 99%.

Nous utilisons trois motifs de données dans les expériences nécessitant des trafics synthétiques. Les deux premiers motifs visent à délimiter la consom-

mation maximale et minimale des interconnexions. Pour cela, le tableau 2.5 est repris du chapitre 2 ainsi que les motifs Meilleur cas ($\alpha = 0$) et Pire cas ($\alpha = 1$). Le troisième motif utilisé pour compléter les expérimentations est Random, dont le contenu des flits est rempli aléatoirement.

4.3.4.2 Résultats de simulations

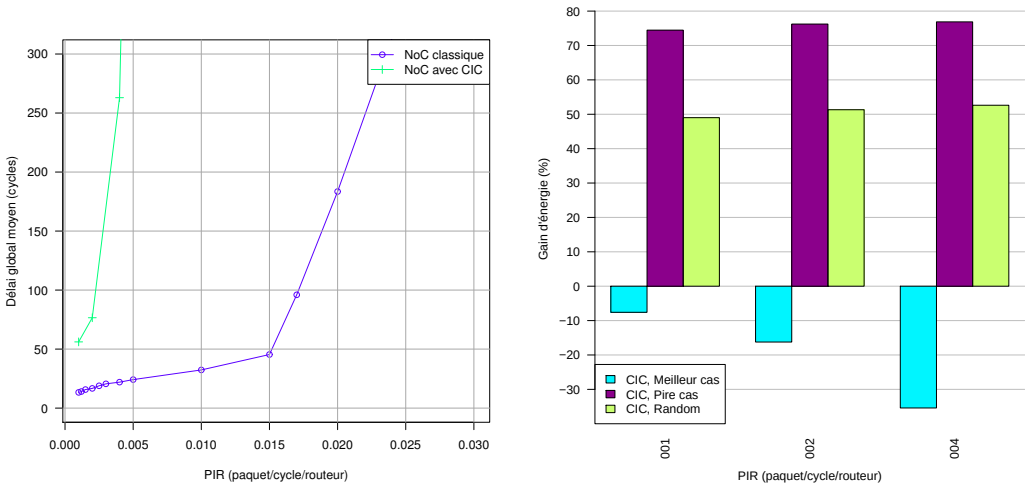
La figure 4.15a montre le délai global moyen du NoC avec et sans utilisation du CIC. Une seule courbe est représentée pour le NoC classique et pour le CIC car leurs performances en termes de délai ne dépend pas de la nature des données. Nous remarquons que l'utilisation constante du CIC accélère significativement la congestion du NoC. Avec le CIC, le NoC commence à être en congestion à environ $PIR = 0.004$ au lieu de $PIR = 0.0175$. Cette valeur est environ quatre fois plus faible, ce qui correspond à la transmission réalisée en 4 cycles du CIC.

La figure 4.15b montre les gains d'énergie pouvant être atteint avec le CIC dans l'intervalle de fonctionnement du NoC (de $PIR = 0$ à $PIR = 0.004$) avec les trois motifs de données présentés précédemment. Un surcoût énergétique peut être constaté avec le motif Meilleur cas n'envoyant que des zéros. Cela est due à une transmission plus lente, qui augmente la durée de fonctionnement, et donc la consommation statique. Nous pouvons remarquer que cette perte augmente avec le PIR, causée par l'accroissement de la latence du fait de la transmission par CIC. Un gain très important est atteint lorsque le motif des données est le Pire cas, la consommation d'énergie des liens peut être réduite jusqu'à 76%. Pour le cas des données Random, une économie d'énergie autour de 50% peut être attendue.

Avec le CIC, nous avons donc la technique réduisant le plus fortement la consommation des liens parmi celles que nous avons proposées mais cela au prix d'une transmission plus lente. Sachant que l'impact du CIC sur le délai du NoC est trop important pour permettre une transmission constante des données par cette technique, la seconde étape de notre contribution est de proposer des stratégies visant à limiter son usage selon l'état courant du réseau.

4.3.5 Stratégies de transmission

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 [60], 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 avec parcimonie via le CIC ou rapidement via le transfert standard, selon son état ou celui de ses voisins. Il est important que le coût matériel



(a) (b)

FIGURE 4.15 – (a) Délai du NoC avec et sans CIC. (b) Réduction d'énergie des liens du NoC (jusqu'au PIR de congestion).

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.

Nous allons nous intéresser à deux métriques locales permettant de connaître l'état local du NoC, la contention et la congestion. Ainsi, l'activation de la transmission par CIC sera décidée sur la base de ces métriques.

4.3.5.1 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 avec le CIC augmente les risques de congestion du NoC.

La figure 4.16a montre comment l'ajout de la stratégie prenant en compte la contention (CONT) interagit avec les autres éléments du routeur. Le bloc contention récupère les directions désirées de chaque paquet via le bloc de routage. De cette manière, quand une sortie est seulement demandée par une entrée, alors le bloc de contention sélectionne la sortie du multiplexeur associée à la transmission codée. Il est important de signaler que le choix de la transmission est envoyé au routeur récepteur via une interconnexion supplémentaire. Ceci dans le but que celui-ci active le chemin vers le décodeur si le CIC est choisi.

La figure 4.16b montre le contenu du bloc contention nécessaire à la stra-

tégie CONT. Le coût matériel est faible pour cette stratégie, une porte OU avec 4 entrées par sortie non-locale. Les entrées de ces portes OU viennent du bloc de routage, nous donnant la direction sélectionnée de chaque entrée. Ensuite, chaque sortie va en direction de : l'arbitre contrôlant le crossbar, les multiplexeurs en sortie et vers les routeurs récepteurs. La prochaine métrique présentée est la congestion locale du réseau, une métrique pouvant également être utilisée pour adapter le mode de transmission.

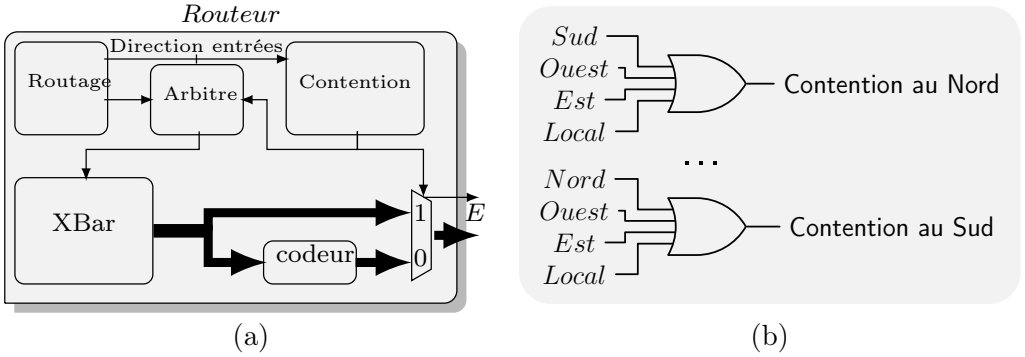


FIGURE 4.16 – (a) Stratégie contention (CONT). Dans cet exemple, nous voyons l'interaction entre les éléments du routeur et le bloc de contention ajouté permettant le contrôle de la transmission à la sortie Est. (b) Bloc contention au niveau porte logique. Si des flits venant du Nord et de l'entrée Local veulent aller au Sud, alors le signal de contention au Sud passe à 1.

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 via le CIC afin de ne pas faire empirer la situation dans le NoC. Il existe une méthode peu coûteuse en matériel permettant 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 avec le CIC peut être envisagée. Afin de savoir s'il est possible de transmettre parcimonieusement à une interconnexion donnée, nous allons contrôler la transmission des flits en sorties (non-locales) du routeur en fonction de l'état d'occupation des buffers d'entrées des routeurs recevant les flits envoyés.

La figure 4.17 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 2), un signal d'alerte (flag) indiquant que le remplissage dépasse un flit est implanté sur chaque buffer du *Routeur*2 de la figure 4.17. Ainsi, nous implantons dans chaque FIFO un signal qui indique au routeur envoyeur (ici *Routeur*1) quand le buffer contient plus d'un flit, afin de contrôler le mode de

transmission. La stratégie OCC nécessite seulement une interconnexion inter-routeur pour fonctionner, ceci est appliqué pour chaque liaison inter-routeur.

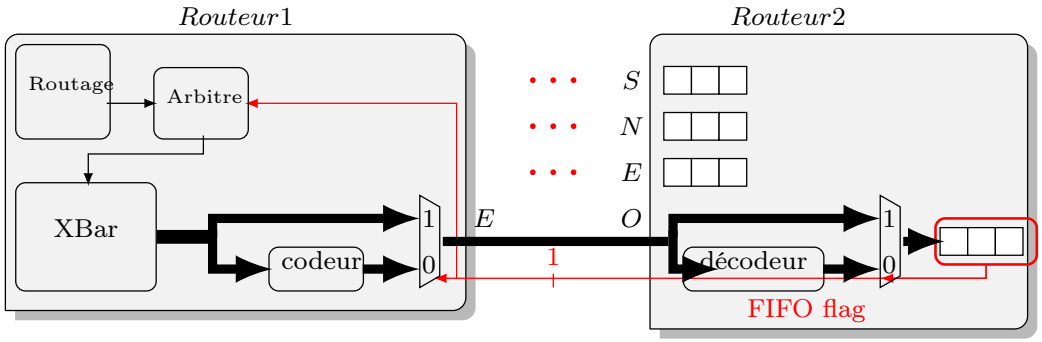


FIGURE 4.17 – Stratégie occupation des buffers (OCC). La sortie Est du *Routeur*1 est contrôlée selon l'occupation des buffers de l'entrée Ouest du *Routeur*2. Les signaux de FIFO des entrées restantes sont connectées aux autres routeurs voisins du *Routeur*2.

En outre, il est possible de combiner les stratégies CONT et OCC pour créer la stratégie CONT+OCC. De cette manière, le trafic en amont (CONT) et en aval (OCC) de l'interconnexion est considéré. Seule une porte OU par sortie est nécessaire en plus du matériel des deux stratégies. Nous allons maintenant évaluer le CIC associé aux différentes stratégies proposées.

4.3.6 Résultats expérimentaux

4.3.6.1 Configuration du NoC

Les conditions de simulations avec un trafic synthétique sont les mêmes que celles mentionnées en section 4.3.4. Nous utilisons pour ces expérimentations seulement le motif de données Random. Afin de considérer la consommation d'énergie totale du NoC, le surcoût matériel du CIC, de son adaptateur et des stratégies sont considérés. Le tableau 4.7 présente le coût matériel généré par le CIC et les différentes stratégies. Ces résultats proviennent de synthèses de fichiers de description matérielle avec l'outil Synopsys Design Compiler.

Nous avons également réalisé des simulations avec des trafics d'applications. Les applications utilisées sont Dijkstra, 5 applications, clusters applications et Full HD vu au chapitre 2. Voici les différences de configuration avec les simulations réalisées avec les trafics synthétiques :

— La topologie utilisée est 4 × 3 ou 4 × 2 selon l'application
— Le temps de simulation va jusqu'à la fin de l'application
— Le contenu des données et le trafic sont issus des traces d'applications

TABLEAU 4.7 – Surcoût matériel et en consommation du routeur selon sa configuration (taille des flits et des buffers) et la stratégie utilisée. Tous les composants ajoutés sont considérés. La configuration utilisée du routeur est la colonne colorée.

Taille flit (bits)	32	32	32	64	64	64
Profondeur buffer (flits)	4	8	12	4	8	12
Surface multiplexeurs (%)	0.89	0.58	0.46	0.98	0.62	0.49
Surface codeur (%)	5.34	3.49	2.75	5.89	3.73	2.94
Surface décodeur (%)	4.65	3.04	2.40	5.13	3.25	2.56
Surface registre à décalage (%)	2.80	1.84	1.45	3.09	1.96	1.54
Surface CONT* (%)	0.10	0.07	0.05	0.06	0.03	0.03
Surface OCC[1] (%)	0	0	0	0	0	0
Surface CONT+OCC[2] (%)	0.03	0.021	0.015	0.017	0.012	0.012
Surcoût surface routeur (%)	**13.81**	**9.04**	**7.12**	**15.17**	**9.62**	**7.57**
Surcoût consommation routeur (%)	**17.2**	**11.4**	**8.6**	**18.4**	**12**	**9.1**

*: Cette stratégie requiert aussi un fil par sortie non-locale.
[1]: Cette stratégie requiert aussi un fil par sortie non-locale.
[2]: C'est le coût supplémentaire pour associer les deux stratégies.

4.3.6.2 Résultats avec trafics synthétiques

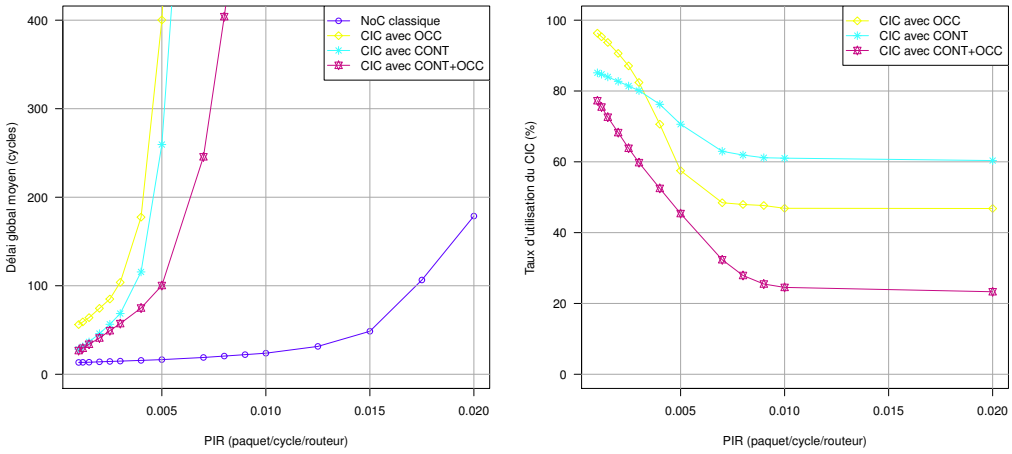
Analyse de performances : délai La figure 4.18a montre le délai du NoC en fonction du PIR et de la stratégie employée. Nous pouvons observer qu'avec la stratégie OCC, le NoC congestionne à $PIR = 0.0045$, ce qui apportent une légère amélioration du délai comparé au CIC seul qui entre en congestion à $PIR = 0.004$. Concernant la stratégie CONT, elle apporte un meilleur gain de performances puisque la congestion n'apparaît qu'à $PIR = 0.005$. Néanmoins, cela reste une faible amélioration comparé aux performances du NoC classique. Avec la stratégie CONT+OCC (combinant les deux précédentes stratégies), les performances du NoC avec le CIC s'améliorent significativement. Le NoC peut maintenir un PIR jusqu'à 0.008 ce qui est le double du CIC seul ; ce fonctionnement maximal avec CONT+OCC représente 45% des performances maximales du NoC classique.

La figure 4.18b montre le taux d'utilisation du CIC selon la stratégie employée. Le taux d'utilisation du CIC avec OCC commence très proche de 100% lorsque le trafic est très faible, ce qui maximise les gains énergétiques. Par la suite, lorsque le PIR augmente, le taux chute brutalement jusqu'à 50% à cause de la congestion qui s'installe due au remplissage des buffers des routeurs. Nous pouvons en déduire que la stratégie OCC devient rapidement insuffisante pour permettre une bonne gestion du mode de transmission.

Nous pouvons voir lorsque le trafic est moindre que la stratégie CONT a un taux d'utilisation plus faible que OCC (autour de 85%). Cependant, le taux d'utilisation ne chute que jusqu'à 60%, ce qui est moins important qu'avec

la stratégie OCC. Or, la figure 4.18a, nous indique que les performances de CONT sont meilleures que OCC. Cette stratégie engendre donc une meilleure gestion du trafic, par un taux d'utilisation supérieur à partir de $PIR = 0.003$.

Enfin, avec la stratégie CONT+OCC, le taux d'utilisation débute à 78% et descend linéairement en fonction du PIR jusqu'à 24% d'utilisation à $PIR = 0.008$, son PIR à la limite de la congestion. Bien que la gestion du mode de transmission soit meilleure, la congestion apparaît rapidement ($PIR = 0.008$). Une analyse de consommation d'énergie est réalisée afin de déterminer les gains atteint avec ces configurations.



(a) (b)

FIGURE 4.18 – (a) Délai du NoC sans CIC ou avec selon la stratégie et le PIR. (b) Taux d'utilisation du CIC selon la stratégie employée et le PIR.

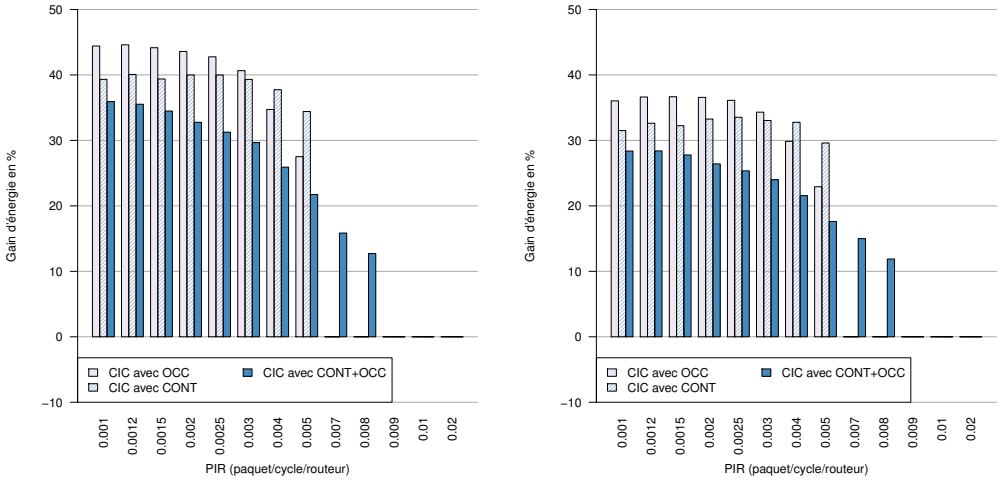
Analyse de consommation d'énergie La figure 4.19a montre le gain en énergie des interconnexions du NoC en fonction du PIR et de la stratégie et ce jusqu'à la congestion. Il est logique d'observer que la technique apportant les plus grands gains d'énergie (environ 45%) quand le PIR est faible est OCC puisque c'est celle dont le taux d'utilisation est le plus grand d'après la figure 4.18b. Néanmoins, quand le PIR dépasse 0.003, la stratégie CONT commence à dominer (de 39% à 35% d'économie d'énergie). Bien que la stratégie CONT+OCC apporte les gains d'énergie les plus faibles (de 35% à 14%), c'est la stratégie qui permet de maintenir le taux d'injection le plus élevé et représente un bon compromis entre performances et énergie.

La figure 4.19b montre le gain d'énergie totale du NoC en fonction du PIR et de la stratégie jusqu'à la congestion. Nous pouvons remarquer que les tendances des résultats sont les mêmes que celles de la figure 4.19a avec des gains moins élevés lorsque le trafic est faible. Cela est dû à deux facteurs : le

premier est le surcoût matériel induit par le CIC et les stratégies employées. Le second concerne le gain amoindri à PIR faible, qui est causé par la part réduite des interconnexions dans la consommation d'énergie du NoC comparée aux autres composants du NoC (le trafic étant faible). Nous obtenons les gains suivants avec les trois stratégies :

— De 22% à 36% de gain d'énergie total pour OCC
— De 29% à 32% de gain d'énergie total pour CONT
— De 12% à 28% de gain d'énergie total pour CONT+OCC

Dans le but de vérifier si les performances du CIC avec stratégie de transmission ne se dégradent pas en fonction de la taille du NoC, une étude du passage à l'échelle est réalisée.



(a) (b)

FIGURE 4.19 – (a) Réduction d'énergie des liens du NoC (jusqu'au PIR de congestion). (b) Réduction d'énergie totale du NoC (jusqu'au PIR de congestion).

Passage à l'échelle du CIC Pour l'étude du passage à l'échelle, nous reprenons les métriques telles que le *Débit Réseau* et *Débit de staturation* définies en section 4.2.5.2. La figure 4.20 montre l'évolution du débit maximal pouvant être accepté par le réseau avec un NoC classique et un NoC utilisant le CIC couplé à la stratégie CONT+OCC. Nous pouvons constater que le débit maximal atteint par la configuration CONT+OCC oscille entre -55% et -60% de débit par rapport au NoC classique quel que soit la taille du NoC. Le CIC avec stratégie CONT+OCC peut donc être utilisé sur des réseaux de plus grandes tailles.

Il est clair que toutes les applications ne peuvent pas se permettre d'utiliser le CIC à cause d'un besoin en débit trop important ou de contraintes en latence. Néanmoins, il existe plusieurs applications dont les besoins en bande-passante

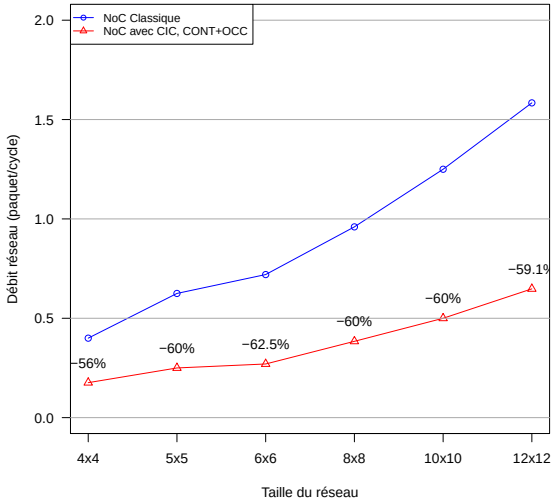


FIGURE 4.20 – Passage à l'échelle du CIC avec stratégie CONT+OCC en mesurant le débit maximum atteignable.

peuvent correspondre aux performances apportées par le CIC et permettre ainsi une forte réduction énergétique. Les prochaines expérimentations présentent l'utilisation du CIC avec les stratégies de transmission par les trafics issus d'applications.

4.3.6.3 Résultats avec trafics d'applications

Le tableau 4.8 montre les résultats de simulation du NoC avec le CIC associé aux différentes stratégies proposées pour les différents trafics d'application. Il est divisé en trois sous-parties, une pour chaque stratégie. Nous pouvons observer que les gains d'énergie réalisé sur les liens avec l'application Dijkstra vont de 50.6% à 69.7%. Cette application nécessitant une très faible bande-passante, le CIC peut être utilisée constamment comme montré avec les 99.6% d'utilisation du CIC avec OCC. La stratégie OCC devient inopérante pour l'application Clusters applications car le délai devient prohibitif. Nous constatons que la stratégie CONT obtient de meilleurs résultats en termes de délai additionnel pour les application Dijkstra et 5 applications avec un gain en énergie légèrement réduit. Néanmoins, il n'est toujours pas possible de maintenir le débit de l'application Clusters applications. Seule la stratégie CONT+OCC permet de gérer le trafic de cette application avec des gains jusqu'à 31.2% pour un taux d'utilisation de 36%. Ces deux stratégies complémentaires ne permettent cependant pas de maintenir le débit du flux vidéo HD requis par l'application Full HD.

La figure 4.21 montre le gain d'énergie total réalisé sur le NoC avec le CIC et

TABLEAU 4.8 – Évaluation du CIC et des stratégies avec trafics d'applications. Les résultats de référence sont ceux du NoC classique.

	CIC avec OCC				
Application	Énergie des liens (%)	Taux d'utilisation CIC (%)	Délai moyen additionnel (%)	Débit application (%)	Latence globale (%)
Dijkstra	-69.7	99.6	+360	0	0
5 applications	-63.6	92	+338	-0.001	+2.5
Clusters applications	-63.2	78.4	>+1000	-27.84	+35
Full HD	-64.1	78	>+1000	-42.2	+77

	CIC avec CONT				
Application	Énergie des liens (%)	Taux d'utilisation CIC (%)	Délai moyen additionnel (%)	Débit application (%)	Latence globale (%)
Dijkstra	-61.05	66.5	+172	-0.001	0
5 applications	-57.5	64.4	+154	-0.001	0
Clusters applications	-62.3	62.5	>+1000	-12.5	+21
Full HD	-57.2	88	>+1000	-29	+61

	CIC avec CONT+OCC				
Application	Énergie des liens (%)	Taux d'utilisation CIC (%)	Délai moyen additionnel (%)	Débit application (%)	Latence globale (%)
Dijkstra	-59.5	62.9	+78	0	0
5 applications	-50.6	58	+161	0	0
Clusters applications	-31.2	36	+319	-0.01	0.2
Full HD	-57.2	87.5	>+1000	-28.9	+61

les stratégies proposées. Les gains vont de 12% pour la stratégie CONT+OCC avec Dijkstra jusqu'à 31% pour la stratégie OCC avec l'application 5 applications. Ces résultats montrent que même avec le surcoût matériel, des gains intéressants peuvent être atteints. De plus, la stratégie CONT+OCC est un excellent compromis puisqu'elle est en mesure de maintenir le débit de Clusters applications (5 clust app) avec une réduction de 15% de l'énergie totale du NoC. Toutefois, aucune des stratégies ne permet de répondre aux besoins de bande-passante de l'application Full HD. La seule solution envisageable pour utiliser le CIC sur l'application Full HD est l'augmentation de la fréquence de fonctionnement du NoC.

Le tableau 4.9 montre l'expérimentation du CIC avec la stratégie CONT + OCC sous le trafic de Full HD avec une fréquence du NoC passant de 1 GHz à 1.6 GHz. Cette augmentation de la fréquence respecte les contraintes de timing imposée par le circuit du NoC. Afin de faire une comparaison honnête, la

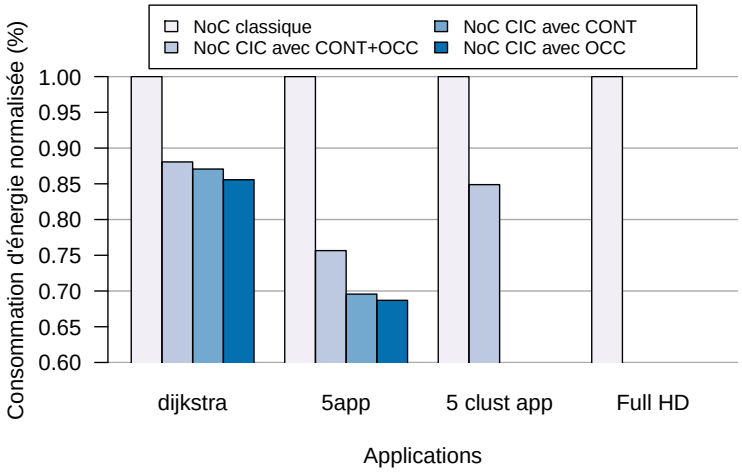


FIGURE 4.21 – Gain d'énergie total du NoC avec CIC en fonction de la stratégie employée et de l'application. Si la stratégie ne peut gérer le débit de l'application alors nous ne présentons pas les résultats.

consommation du NoC classique à 1 GHz est considéré puisqu'une augmentation de sa fréquence est inutile pour cette application. Une fois que la fréquence du réseau est suffisamment augmentée, le NoC avec CIC (CONT+OCC) est assez performant pour pouvoir gérer le débit de Full HD et nous pouvons constater une réduction d'énergie des liens de 45% soit une réduction de l'énergie totale du NoC de 24% sans chute des performances.

Cette dernière expérimentation a aussi été réalisée afin de montrer l'intérêt du CIC associé à un mécanisme de modification dynamique de la fréquence. Bien que cette augmentation de 60% semble importante, il existe depuis plusieurs années de systèmes multi-cœurs à la tension et à la fréquence variable dont la fréquence du réseau peut aller de 60 MHz jusqu'à 2.6 GHz [90]. Une extension des travaux pourrait être le développement d'une meilleure stratégie de transmission afin qu'elle soit utilisée dans un NoC où divers trafics d'applications peuvent apparaître. Ainsi, cette stratégie améliorée pourrait être de surcroît associée à un système de gestion dynamique de la fréquence.

TABLEAU 4.9 – Évaluation du CIC avec la stratégie CONT+OCC avec l'application Full HD. Les résultats de référence sont ceux du NoC classique à 1 GHz.

	NoC avec CIC et CONT+OCC à 1.6 GHz					
Application	Énergie des liens (%)	Énergie du NoC (%)	Taux d'utilisation CIC (%)	Délai moyen additionnel (%)	Débit application (%)	Latence globale (%)
Full HD	-44.9	-23.7	87.5	0	0	0

4.4 Bilan

Nous avons présenté dans ce chapitre plusieurs techniques d'optimisation au niveau des interconnexions du NoC. Ces techniques ont abordé la consommation des liens sous plusieurs angles. Le STS est une technique qui permet une réduction de l'énergie par la suppression des transitions croisées tandis que le CIC se concentre sur la réduction significative du taux de commutation. Afin de confirmer les gains proposés par ces techniques, le surcoût de chacune d'entre elles a été considéré. De plus, nous nous sommes focalisés seulement sur la réduction de la consommation d'énergie des liaisons inter-routeurs. Par conséquent, les optimisations proposées dans ce chapitre peuvent être associées à des optimisations réalisées sur les routeurs afin de réduire davantage la consommation d'énergie totale du NoC.

Ces deux propositions offrent des compromis différents entre latence et énergie. Le STS limite son impact sur la latence tout en conservant des gains d'énergie intéressants. Quant au CIC, le gain d'énergie est plus important au prix d'une réduction des performances maximales du NoC. Étant donné que l'intérêt du NoC est de transférer les données d'un SoC pouvant exécuter plusieurs applications ayant toutes des besoins variés en termes de bande-passante, il est pertinent de proposer des solutions considérant différents compromis latence / énergie. De cette manière, la meilleure technique d'optimisation peut être sélectionnée selon la situation, à condition d'avoir une stratégie de transmission adaptée. La conception de ce type de stratégie pourrait constituer une perspective intéressante à ces travaux.

Conclusion

Cette thèse avait pour visée principale le développement de techniques d'optimisation de la consommation d'énergie dans les NoCs. Cet objectif a dans un premier temps nécessité de modéliser précisément cette consommation, puis de développer les optimisations tout en limitant les effets négatifs potentiels.

Nous avons vu dans la première partie de la thèse la composition d'un NoC et comment est modélisée sa consommation d'énergie. Le premier chapitre montre que les NoCs sont composés de routeurs, d'interconnexions et d'adaptateurs réseaux. Puisque les configurations de ce type de réseau sont nombreuses, nous avons sélectionné les différents NoCs existant afin d'identifier un profil de NoC aussi représentatif que possible. Cet état de l'art est étendu par une étude de la modélisation de la consommation d'énergie de chaque composant du réseau. Les différents paramètres qui influent sur la consommation d'énergie ont été identifiés. Pour cela, les méthodes de la littérature calculent la capacité de chaque composant et déterminent le taux de charge de ces capacités en fonction des actions réalisées dans le NoC. Plusieurs travaux fournissent des modèles de routeurs avec un important niveau de détail concernant sa micro-architecture. Cependant, la modélisation des interconnexions proposée dans la littérature est indépendante de la nature des données. Or, nous savons que les données ont un impact significatif sur la consommation d'énergie des interconnexions comme l'attestent de nombreux travaux dans le domaine des bus de communication. Nous avons donc proposé d'enrichir le modèle de consommation du NoC en y intégrant un modèle de consommation bit-près pour les liens inter-routeurs. Puis nous avons proposé de comparer ce modèle à celui de la littérature.

Le deuxième chapitre présente donc cette comparaison entre les deux modèles de liens inter-routeurs puis évalue l'impact du modèle d'interconnexion dans l'estimation de la consommation d'énergie totale du NoC. A partir des différentes estimations qu'apporte le modèle bit-près, nous avons remarqué que la consommation d'une interconnexion peut être multipliée par vingt (de 13.29 fJ à 265.07 fJ) selon son activité et celle de ses voisins. Cette importante différence est dûe au *crosstalk*, phénomène physique de diaphonie capacitive. Étant donné que le modèle d'interconnexion de la littérature des NoCs est indépendant des données, il ne peut pas tenir compte des effets pourtant significatifs du crosstalk. Ainsi, ce modèle fournit une seule estimation possible parmi une surface de possibilités montrant la nécessité de considérer les données pour l'estimation de l'énergie des interconnexions. De plus, le modèle de la littérature considère un taux de commutation des données constant, égal à 0.5, or ce taux varie entre chaque donnée. Ainsi, la consommation réelle peut s'écarter de **-95%** à **+270%** de la valeur estimée par le modèle de la littéra-

ture.. Le modèle de liens bit-près a ainsi été implanté dans un simulateur de NoC afin d'évaluer l'impact des interconnexions dans la consommation totale du NoC. Les résultats montrent que la consommation du réseau peut doubler en fonction de la nature des données et que les interconnexions sont les composants les plus consommateurs lorsque le débit du NoC dépasse un certain seuil. Enfin, diverses améliorations ont été apportées au simulateur de NoC afin de proposer Noxim-XT. Cet outil de simulation open source peut être utilisé pour l'exploration d'architecture et l'évaluation de techniques d'optimisation de l'énergie.

Une fois que la modélisation de l'énergie a été améliorée, nous avons entreprit l'étude et la mise en œuvre de techniques d'optimisation de l'énergie du NoC dans la deuxième partie de cette thèse. Nous avons réalisé dans le chapitre 3 une étude bibliographique des différentes techniques d'optimisation de la consommation d'énergie du NoC. Cette étude montre les possibilités multiples d'améliorations pouvant être mises en œuvre dans les NoCs. Parmi le nombre important d'orientations possibles, la majorité des techniques d'optimisation du routeur vise la réduction de l'énergie statique du fait de son nombre important de transistors (en particulier des buffers), tandis que les techniques de réduction d'énergie des interconnexions concernent l'énergie dynamique. Nous savons que les interconnexions représentent une part importante de la consommation du NoC et que peu de travaux sont réalisés sur ce sujet, par comparaison avec l'étude de la topologie et le routage. De plus, les techniques d'optimisation au niveau des interconnexions sont indépendantes de l'architecture du NoC. C'est pourquoi nous avons choisi de proposer des techniques d'optimisation au niveau des interconnexions.

Le quatrième chapitre introduit plusieurs propositions de techniques d'optimisation des interconnexions. Elles ont l'avantage d'être compatibles avec d'autres techniques d'optimisation de la littérature. Le principe du Smart Temporal Shielding (STS) est de placer une donnée factice entre deux mots qui auraient généré une transition croisée, très consommatrice en énergie. Nous avons montré que cette technique permet d'économiser jusqu'à **15%** d'énergie sur le NoC complet. De plus, l'impact sur la latence est réduit de **33%** comparé au Temporal Shielding. La Communication Inspirée par le Cortex (CIC) est une technique inspirée de la représentation des données dans le cortex, c-à-d beaucoup d'information pour peu d'activité. Nous avons repris cette idée afin de proposer une technique de codage qui permette de coder plusieurs bits de données en seulement une transition. La consommation d'énergie des interconnexions est diminuée de **50%** pour des données aléatoires. La contre-partie de cette méthode est l'augmentation du temps de transmission. C'est pourquoi une stratégie a été mise en œuvre afin d'utiliser ce mode de transmission avec parcimonie. Les résultats obtenus avec ces deux techniques peuvent être reprises pour être associées à de nouvelles stratégies de transmission afin de sélectionner intelligemment le meilleur mode de transmission selon l'état du NoC.

Perspectives

Cette section présente les perspectives ouvertes par nos travaux. La première partie présente les perspectives à court terme pouvant être développées en améliorant les travaux sous différents points de vue. Enfin, des perspectives à moyen terme sont évoquées.

Perspectives à court terme

Les perspectives à court terme concernent en majorité une extension potentielle de nos travaux sur le CIC car cette technique offre de nombreuses possibilités. En effet, dans le cas d'une technique adaptée à une application, il serait intéressant d'approfondir l'exploration des différentes configurations du codage, ceci afin de trouver le meilleur compromis entre gain d'énergie et bande-passante requise. Nous avons vu avec le CIC que plusieurs variantes de ce codage peuvent être réalisées et chacune d'elles apporte un compromis énergie / latence différent. Ainsi, la sélection du codage pourrait être réalisée en amont dans le design du réseau en fonction de l'application finale du système.

Piste d'amélioration de la stratégie de transmission Nous proposons une amélioration qui est en cours de validation. La nouvelle stratégie proposée pour améliorer la gestion du mode de transmission est nommée Time of Flight (ToF). Le but du ToF est d'adapter la stratégie des routeurs en fonction du temps que les paquets ont passé dans le réseau. Par exemple, plus un paquet restera longtemps dans le NoC, et moins le CIC sera utilisé sur les flits de celui-ci afin d'accélérer sa transmission.

Technologies d'interconnexions alternatives Nos travaux réalisés sur les interconnexions peuvent également être étendus à des technologies d'interconnexions telles que :

— Les interconnexions en 3D
— La transmission optique
— La transmission sans-fils.

Les TSV employés par les NoC en 3D subissent également d'importants effets de crosstalk et il serait donc pertinent de modéliser finement ces liaisons afin de proposer un modèle complet de NoC 3D. Ensuite, si ces effets sont suffisamment conséquents, alors il serait possible d'utiliser les techniques développées au cours de cette thèse afin d'optimiser la consommation sur ce type de support.

Avec l'avancée technologique, le nombre des travaux sur les transmissions optiques est de plus en plus important. Nous avons vu au cours de cette thèse que la consommation des interconnexions est significative pour le NoC tout

entier et cela s'accroît avec la miniaturisation des transistors. De plus, ces solutions émergentes pourraient bénéficier de nos travaux sur la réduction du crosstalk puisque de la diaphonie apparaît également par transmission optique.

La dernière perspective à court terme concerne les interconnexions sans-fils dans un NoC. Au lab-STICC, le projet BBC [3] financé par le Labex Cominlabs a pour objectif de réaliser les composants nécessaires pour une communication sans-fils dans une puce. Les travaux que nous avons réalisés sur le simulateur de NoC pourraient compléter ce projet et permettre le développement d'un simulateur capable de modéliser l'énergie d'un NoC filaire et sans-fils.

Perspectives à moyen terme

Les perspectives à moyen terme concernent des travaux potentiels nécessitant un effort de développement important.

Conception d'un NoC parcimonieux Au lieu d'employer le CIC seulement sur les liens avec codage et décodage à chaque liaison, il pourrait être envisagé de développer un NoC conçu pour n'avoir que le CIC comme mode de transmission. C'est-à-dire le développement d'un routeur adapté pour le CIC. Les intérêts de ce système sont multiples : le premier est la réduction significative de la taille du routeur. En effet, si nous prenons par exemple une largeur de liens $N = 16$ pour coder 4 bits d'information par cycle, la taille du crossbar pourrait être divisée par deux (si nous considérons que la taille par défaut est de 32 bits) et l'agencement des buffers des routeurs serait modifiée avec la possibilité de gains en surface. Bien évidemment, se contenter de cette configuration nous octroierait un NoC avec des capacités bien inférieures au réseau sur 32 bits. Cependant, ces dernières années, l'adaptation dynamique de la fréquence (DVFS) est devenue une technique répandue et en employant le DVFS sur ce réseau parcimonieux, nous pourrions adapter la fréquence selon les besoins en bande-passante de l'application en cours d'exécution.

Simulateur de NoC universel Une dernière perspective à ces travaux de thèse serait de compléter le simulateur Noxim-XT avec les différents types de transmissions émergentes. Nous aurions un outil open source complet fournissant des estimations précises de l'énergie disponible pour toute la communauté.

3. Labex CominLabs (2016-2019): Wireless Interconnect Network-on-Chip or in board for Broadcast-Based parallel Computing

Glossaire

ASIC	Application-Specific Integrated Circuit
CI	Core Interface
CIC	Communication Inspirée par le Cortex
DVFS	Dynamic Voltage Frequency Scaling
DyAD	Dynamic Adaptive Deterministic switching
flit	FLow control unIT
FPGA	Field Programmable Gate Array
IP	Intellectual property
MFLP	Most Frequent Least Power
MPSoC	Multi-Processor System-on-Chip
NA	Network Adapter
NI	Network Interface
NoC	Network-on-Chip
PE	Processing Element
phit	PHysical unIT
PIR	Packet Injection Rate
SoC	System-on-Chip
SRAM	Static Random Access Memory
STS	Smart Temporal Shielding
STT-RAM	Spin-Transfer Torque-Random Access Memory
TS	Temporal Shielding
VC	Virtual Channel

Table des figures

Liste des tableaux

Bibliographie

[1] G. E. Moore, "Cramming more components onto integrated circuits, reprinted from electronics, volume 38, number 8, april 19, 1965, pp.114," *IEEE Solid-State Circuits Society Newsletter*, vol. 11, pp. 33–35, Sept 2006.

[2] P. Coussy and A. Morawiec, *High-level synthesis: from algorithm to digital circuit*. Springer Science & Business Media, 2008.

[3] M. Taylor, J. Kim, J. Miller, D. Wentzlaff, F. Ghodrat, B. Greenwald, H. Hoffman, P. Johnson, J.-W. Lee, W. Lee, A. Ma, A. Saraf, M. Seneski, N. Shnidman, V. Strumpen, M. Frank, S. Amarasinghe, and A. Agarwal, "The raw microprocessor: a computational fabric for software circuits and general-purpose programs," *Micro, IEEE*, vol. 22, pp. 25–35, Mar 2002.

[4] M. Niroomand and H. R. Foroughi, "A rotary electromagnetic microgenerator for energy harvesting from human motions," *Journal of Applied Research and Technology*, vol. 14, no. 4, pp. 259–267, 2016.

[5] L. Benini and G. De Micheli, "Networks on chips: a new soc paradigm," *Computer*, vol. 35, pp. 70–78, Jan 2002.

[6] R. Dafali, *Design of dynamic reconfigurable Network-on-Chip*. Thèse, Université Européenne de Bretagne ; Université de Bretagne-Sud, May 2011.

[7] D. Bertozzi and L. Benini, "Xpipes: A network-on-chip architecture for gigascale systems-on-chip," *IEEE circuits and systems magazine*, vol. 4, no. 2, pp. 18–31, 2004.

[8] A. Adriahantenaina, H. Charlery, A. Greiner, L. Mortiez, and C. A. Zeferino, "Spin: a scalable, packet switched, on-chip micro-network," in *2003 Design, Automation and Test in Europe Conference and Exhibition*, pp. 70–73 suppl., 2003.

[9] K. Goossens, J. Dielissen, and A. Radulescu, "Æthereal network on chip: concepts, architectures, and implementations," *IEEE Design & Test of Computers*, vol. 22, no. 5, pp. 414–421, 2005.

[10] T. Bjerregaard, *The MANGO clockless network-on-chip: Concepts and implementation*. PhD thesis, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, 2005. Supervised by Assoc. Prof. Jens Sparsø, IMM.

[11] D. Siguenza-Tortosa and J. Nurmi, "Proteo: a new approach to network-on-chip," *Proceedings, IASTED-Communication Systems and Networks (CSN 2002)*, 2002.

[12] D. Rostislav, V. Vishnyakov, E. Friedman, and R. Ginosar, "An asynchronous router for multiple service levels networks on chip," in *11th IEEE International Symposium on Asynchronous Circuits and Systems*, pp. 44–53, March 2005.

[13] M. Coppola, M. D. Grammatikakis, R. Locatelli, G. Maruccia, and L. Pieralisi, *Design of cost-efficient interconnect processing units: Spidergon STNoC*. CRC press, 2008.

[14] W. J. Dally and J. W. Poulton, *Digital systems engineering*. Cambridge University Press, 1998.

[15] R. Ben Atitallah, *Modèles et simulation des systèmes sur puce multiprocesseurs : estimation des performances et de la consommation d'énergie*. Thèse, 2008.

[16] K. Roy, S. Mukhopadhyay, and H. Mahmoodi-Meimand, "Leakage current mechanisms and leakage reduction techniques in deep-submicrometer cmos circuits," *Proceedings of the IEEE*, vol. 91, pp. 305–327, Feb 2003.

[17] L. Nagel and C. McAndrew, "Is spice good enough for tomorrow's analog?," in *Bipolar/BiCMOS Circuits and Technology Meeting (BCTM), 2010 IEEE*, pp. 106–112, Oct 2010.

[18] "Synopsys primetime px power analysis solution achieves broad market adoption." Available: `https://news.synopsys.com/index.php?s=20295&item=123041`, 2014. [Online].

[19] "Ieee standard for standard systemc language reference manual," *IEEE Std 1666-2011 (Revision of IEEE Std 1666-2005)*, pp. 1–638, Jan 2012.

[20] H.-S. Wang, X. Zhu, L.-S. Peh, and S. Malik, "Orion: a power-performance simulator for interconnection networks," in *Microarchitecture, 2002. (MICRO-35). Proceedings. 35th Annual IEEE/ACM International Symposium on*, pp. 294–305, 2002.

[21] A. Kahng, B. Li, L.-S. Peh, and K. Samadi, "Orion 2.0: A power-area simulator for interconnection networks," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 20, pp. 191–196, Jan 2012.

[22] C. Sun, C.-H. Chen, G. Kurian, L. Wei, J. Miller, A. Agarwal, L.-S. Peh, and V. Stojanovic, "Dsent - a tool connecting emerging photonics with electronics for opto-electronic networks-on-chip modeling," in *Networks on Chip (NoCS), 2012 Sixth IEEE/ACM International Symposium on*, pp. 201–210, May 2012.

[23] A. Courtay, *On-chip interconnects energy consumption: High-level estimation and architectural optimizations*. Thèse, Université de Bretagne Sud, Nov. 2008.

[24] T. Bjerregaard, S. Mahadevan, R. G. Olsen, and J. Sparso, "An ocp compliant network adapter for gals-based soc design using the mango network-on-chip," in *2005 International Symposium on System-on-Chip*, pp. 171–174, Nov 2005.

[25] A. Agarwal, C. Iskander, and R. Shankar, "Survey of network on chip (noc) architectures & contributions," *Journal of engineering, Computing and Architecture*, vol. 3, no. 1, pp. 21–27, 2009.

[26] J. Cong, C. Liu, and G. Reinman, "Aces: Application-specific cycle elimination and splitting for deadlock-free routing on irregular network-on-chip," in *Design Automation Conference*, pp. 443–448, June 2010.

[27] T. Bjerregaard and S. Mahadevan, "A survey of research and practices of network-on-chip," *ACM Comput. Surv.*, vol. 38, June 2006.

[28] Y. I. Ismail, E. G. Friedman, and J. L. Neves, "Figures of merit to characterize the importance of on-chip inductance," in *Proceedings 1998 Design and Automation Conference. 35th DAC. (Cat. No.98CH36175)*, pp. 560–565, June 1998.

[29] N. Srivastava, X. Qi, and K. Banerjee, "Impact of on-chip inductance on power distribution network design for nanometer scale integrated circuits," in *Sixth international symposium on quality electronic design (isqed'05)*, pp. 346–351, March 2005.

[30] L. Carloni, A. B. Kahng, S. Muddu, A. Pinto, K. Samadi, and P. Sharma, "Interconnect modeling for improved system-level design optimization," in *2008 Asia and South Pacific Design Automation Conference*, pp. 258–264, March 2008.

[31] "Nirgam." Available: `http://www.nirgam.ecs.soton.ac.uk`. [Online].

[32] N. Agarwal, T. Krishna, L.-S. Peh, and N. Jha, "Garnet: A detailed on-chip network model inside a full-system simulator," in *Performance Analysis of Systems and Software, 2009. ISPASS 2009. IEEE International Symposium on*, pp. 33–42, April 2009.

[33] P. Abad, P. Prieto, L. Menezo, A. Colaso, V. Puente, and J.-A. Gregorio, "Topaz: An open-source interconnection network simulator for chip multiprocessors and supercomputers," in *Networks on Chip (NoCS), 2012 Sixth IEEE/ACM International Symposium on*, pp. 99–106, May 2012.

[34] V. Catania, A. Mineo, S. Monteleone, M. Palesi, and D. Patti, "Cycle-accurate network on chip simulation with noxim," *ACM Trans. Model. Comput. Simul.*, vol. 27, pp. 4:1–4:25, Aug. 2016.

[35] J. M. Rabaey, A. P. Chandrakasan, and B. Nikolic, *Digital integrated circuits*, vol. 2. Prentice hall Englewood Cliffs, 2002.

[36] "Ptm, predictive technology model." Available: `http://ptm.asu.edu/`, 2012. [Online].

[37] A. Courtay, O. Sentieys, J. Laurent, and N. Julien, "High-level interconnect delay and power estimation," *Journal of Low Power Electronics*, vol. 4, no. 1, pp. 21–33, 2008.

[38] E. Moréac, A. Rossi, J. Laurent, and P. Bomel, "Noxim-xt: Crosstalk-aware noxim based simulator." Available: `https://github.com/davidepatti/noxim/tree/noxim-XT`. [Online].

[39] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The splash-2 programs: Characterization and methodological considerations," *SIGARCH Comput. Archit. News*, vol. 23, pp. 24–36, May 1995.

[40] S. M. Z. Iqbal, Y. Liang, and H. Grahn, "Parmibench - an open-source benchmark for embedded multiprocessor systems," *IEEE Computer Architecture Letters*, vol. 9, pp. 45–48, Feb 2010.

[41] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The parsec benchmark suite: characterization and architectural implications," in *Proceedings of the 17th international conference on Parallel architectures and compilation techniques*, pp. 72–81, ACM, 2008.

[42] A. Mello, I. Maia, A. Greiner, and F. Pecheux, "Parallel simulation of systemc tlm 2.0 compliant mpsoc on smp workstations," in *2010 Design, Automation Test in Europe Conference Exhibition (DATE 2010)*, pp. 606–609, March 2010.

[43] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, pp. 1–7, Aug. 2011.

[44] L. Duenha, M. Guedes, H. Almeida, M. Boy, and R. Azevedo, "Mpsocbench: A toolset for mpsoc system level evaluation," in *Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS XIV), 2014 International Conference on*, pp. 164–171, July 2014.

[45] R Development Core Team, *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2008. ISBN 3-900051-07-0.

[46] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. Stan, "Hotspot: a compact thermal modeling methodology for early-stage vlsi design," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 14, pp. 501–513, May 2006.

[47] T. Feng, "A survey of interconnection networks," *Computer*, vol. 14, pp. 12–27, Dec 1981.

[48] F. Karim, A. Nguyen, and S. Dey, "An interconnect architecture for networking systems on chips," *IEEE Micro*, vol. 22, pp. 36–45, Sep 2002.

[49] P. Gehlot and S. S. Chouhan, "Performance evaluation of network on chip architectures," in *2009 International Conference on Emerging Trends in Electronic and Photonic Devices Systems*, pp. 124–127, Dec 2009.

[50] M. A. A. E. ghany, M. A. El-Moursy, D. Korzec, and M. Ismail, "Power characteristics of networks on chip," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pp. 3721–3724, May 2010.

[51] G. Reehal and M. Ismail, "A systematic design methodology for low-power nocs," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 22, pp. 2585–2595, Dec 2014.

[52] I. Loi, F. Angiolini, and L. Benini, "Supporting vertical links for 3d networks-on-chip: Toward an automated design and analysis flow," in *Pro-*

ceedings of the 2Nd International Conference on Nano-Networks, NanoNet '07, (ICST, Brussels, Belgium, Belgium), pp. 15:1–15:5, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2007.

[53] W. R. Davis, J. Wilson, S. Mick, J. Xu, H. Hua, C. Mineo, A. M. Sule, M. Steer, and P. D. Franzon, "Demystifying 3d ics: the pros and cons of going vertical," *IEEE Design Test of Computers*, vol. 22, pp. 498–510, Nov 2005.

[54] B. Goplen and S. Sapatnekar, "Efficient thermal placement of standard cells in 3d ics using a force directed approach," in *ICCAD-2003. International Conference on Computer Aided Design (IEEE Cat. No.03CH37486)*, pp. 86–89, Nov 2003.

[55] B. Dang, P. Joseph, M. Bakir, T. Spencer, P. Kohl, and J. Meindl, "Wafer-level microfluidic cooling interconnects for gsi," in *Proceedings of the IEEE 2005 International Interconnect Technology Conference, 2005.*, pp. 180–182, June 2005.

[56] B. S. Feero and P. P. Pande, "Networks-on-chip in a three-dimensional environment: A performance evaluation," *IEEE Transactions on Computers*, vol. 58, pp. 32–45, Jan 2009.

[57] A. W. Yin, T. C. Xu, P. Liljeberg, and H. Tenhunen, "Explorations of honeycomb topologies for network-on-chip," in *2009 Sixth IFIP International Conference on Network and Parallel Computing*, pp. 73–79, Oct 2009.

[58] H. Jiang, M. Marek-Sadowska, and S. R. Nassif, "Benefits and costs of power-gating technique," in *2005 International Conference on Computer Design*, pp. 559–566, Oct 2005.

[59] H. Farrokhbakht, M. Taram, B. Khaleghi, and S. Hessabi, "Toot: an efficient and scalable power-gating method for noc routers," in *2016 Tenth IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, pp. 1–8, Aug 2016.

[60] R. Das, S. Narayanasamy, S. K. Satpathy, and R. G. Dreslinski, "Catnap: energy proportional multiple network-on-chip," in *Proceedings of the 40th annual international symposium on Computer architecture*, pp. 320–331, ACM, 2013.

[61] S. T. Muhammad, M. A. El-Moursy, A. A. El-Moursy, and A. M. Refaat, "Optimization for traffic-based virtual channel activation low-power noc," in *5th International Conference on Energy Aware Computing Systems Applications*, pp. 1–4, March 2015.

[62] R. Parikh, R. Das, and V. Bertacco, "Power-aware nocs through routing and topology reconfiguration," in *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6, June 2014.

[63] J. Zhan, J. Ouyang, F. Ge, J. Zhao, and Y. Xie, "Hybrid drowsy sram and stt-ram buffer designs for dark-silicon-aware noc," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, pp. 3041–3054, Oct 2016.

[64] Y. Y. Chen, E. J. Chang, H. K. Hsin, K. C. . Chen, and A. Y. . Wu, "Path-diversity-aware fault-tolerant routing algorithm for network-on-chip systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, pp. 838–849, March 2017.

[65] M. Palesi, G. Longo, S. Signorino, R. Holsmark, S. Kumar, and V. Catania, "Design of bandwidth aware and congestion avoiding efficient routing algorithms for networks-on-chip platforms," in *Second ACM/IEEE International Symposium on Networks-on-Chip (nocs 2008)*, pp. 97–106, April 2008.

[66] F. A. Samman, T. Hollstein, and M. Glesner, "Runtime contention and bandwidth-aware adaptive routing selection strategies for networks-on-chip," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, pp. 1411–1421, July 2013.

[67] A. Charif, N. E. Zergainoh, and M. Nicolaidis, "A new approach to deadlock-free fully adaptive routing for high-performance fault-tolerant nocs," in *2016 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, pp. 121–126, Sept 2016.

[68] J. Hu and R. Marculescu, "Dyad: Smart routing for networks-on-chip," in *Proceedings of the 41st Annual Design Automation Conference*, DAC '04, (New York, NY, USA), pp. 260–263, ACM, 2004.

[69] G.-M. Chiu, "The odd-even turn model for adaptive routing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 11, pp. 729–738, Jul 2000.

[70] M. Taassori and S. Hessabi, "Low power encoding in nocs based on coupling transition avoidance," in *Digital System Design, Architectures, Methods and Tools, 2009. DSD '09. 12th Euromicro Conference on*, pp. 247–254, Aug 2009.

[71] M. R. Stan and W. P. Burleson, "Bus-invert coding for low-power i/o," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 3, pp. 49–58, March 1995.

[72] Y. Shin, S.-I. Chae, and K. Choi, "Partial bus-invert coding for power optimization of system level bus," in *Low Power Electronics and Design, 1998. Proceedings. 1998 International Symposium on*, pp. 127–129, Aug 1998.

[73] M. Taassori, M. Taassori, and S. Uysal, "Mflp: a low power encoding for on chip networks," *Design Automation for Embedded Systems*, vol. 20, pp. 191–210, Sep 2016.

[74] M. Palesi, G. Ascia, F. Fazzino, and V. Catania, "Data Encoding Schemes in Networks on Chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, pp. 774–786, May 2011.

[75] N. Jafarzadeh, M. Palesi, A. Khademzadeh, and A. Afzali-Kusha, "Data Encoding Techniques for Reducing Energy Consumption in Network-on-Chip," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, pp. 675–685, Mar. 2014.

[76] D. Bertozzi, L. Benini, and G. D. Micheli, "Error control schemes for on-chip communication links: the energy-reliability tradeoff," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, pp. 818–831, June 2005.

[77] A. Mineo, M. Palesi, G. Ascia, P. P. Pande, and V. Catania, "On-chip communication energy reduction through reliability aware adaptive voltage swing scaling," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, pp. 1769–1782, Nov 2016.

[78] S. Ogg, E. Valli, B. Al-Hashimi, A. Yakovlev, C. D'Alessandro, and L. Benini, "Serialized asynchronous links for noc," in *2008 Design, Automation and Test in Europe*, pp. 1003–1008, March 2008.

[79] S. Ghosh, P. Ghosal, N. Das, S. P. Mohanty, and O. Okobiah, "Data correlation aware serial encoding for low switching power on-chip communication," in *2014 IEEE Computer Society Annual Symposium on VLSI*, pp. 124–129, July 2014.

[80] A. Abdollahi, F. Fallah, and M. Pedram, "Leakage current reduction in cmos vlsi circuits by input vector control," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, pp. 140–154, Feb 2004.

[81] M. Olivieri, G. Scotti, and A. Trifiletti, "A novel yield optimization technique for digital cmos circuits design by means of process parameters run-time estimation and body bias active control," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 13, pp. 630–638, May 2005.

[82] M. Agostinelli, M. Alioto, D. Esseni, and L. Selmi, "Leakage delay tradeoff in finfet logic circuits: A comparative analysis with bulk technology," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, pp. 232–245, Feb 2010.

[83] F. Moshgelani, D. Al-Khalili, and C. Rozon, "Ultra low leakage structures for logic circuits using symmetric and asymmetric finfets," in *10th IEEE International NEWCAS Conference*, pp. 385–388, June 2012.

[84] N. S. Kim, T. Austin, D. Baauw, T. Mudge, K. Flautner, J. S. Hu, M. J. Irwin, M. Kandemir, and V. Narayanan, "Leakage current: Moore's law meets static power," *Computer*, vol. 36, pp. 68–75, Dec 2003.

[85] "International Technology Roadmap for Semiconductors. (2013)." Available: `http://www.itrs.net`. [Online].

[86] S. Salerno, E. Macii, and M. Poncino, "Crosstalk energy reduction by temporal shielding," in *2004 IEEE International Symposium on Circuits and Systems (IEEE Cat. No.04CH37512)*, vol. 2, pp. II–749–752 Vol.2, May 2004.

[87] J.-M. Philippe, S. Pillement, and O. Sentieys, "Area efficient temporal coding schemes for reducing crosstalk effects," in *Proceedings of the 7th International Symposium on Quality Electronic Design*, ISQED '06, (Washington, DC, USA), pp. 334–339, IEEE Computer Society, 2006.

[88] E. R. Kandel, J. H. Schwartz, T. M. Jessell, S. A. Siegelbaum, and A. Hudspeth, *Principles of neural science*, vol. 4. McGraw-hill New York, 2000.

[89] P. Lennie, "The cost of cortical computation," *Current Biology*, vol. 13, no. 6, pp. 493 – 497, 2003.

[90] J. Howard, S. Dighe, S. R. Vangal, G. Ruhl, N. Borkar, S. Jain, V. Erraguntla, M. Konow, M. Riepen, M. Gries, G. Droege, T. Lund-Larsen, S. Steibl, S. Borkar, V. K. De, and R. V. D. Wijngaart, "A 48-core ia-32 processor in 45 nm cmos using on-die message-passing and dvfs for performance and power scaling," *IEEE Journal of Solid-State Circuits*, vol. 46, pp. 173–183, Jan 2011.

Liste des publications

Revues internationales

1. Erwan Moréac, Pierre Bomel, Johann Laurent, André Rossi. **Energy Savings in Networks-on-Chip with Smart Temporal Shielding.** *Journal of Low Power Electronics*, American Scientific Publishers, Sep 2017, 13 (3), pp. 441-455.
2. Erwan Moréac, André Rossi, Johann Laurent, Pierre Bomel. **Bit-Accurate Energy Estimation for Networks-on-Chip**. *Journal of Systems Architecture*, Elsevier, 2017, 77, pp. 112-124.

Conférences internationales avec actes

1. Erwan Moréac, Johann Laurent, Pierre Bomel, André Rossi, Emmanuel Boutillon, Maurizio Palesi. **Energy Aware Networks-on-Chip Cortex Inspired Communication.** *PATMOS*, Sep 2017, Thessaloniki, Greece.
2. Erwan Moréac, André Rossi, Johann Laurent, Pierre Bomel. **Crosstalk-aware link power model for Networks-on-Chip.** *DASIP*, Oct 2016, Rennes, France.

Conférences nationales avec actes

1. Erwan Moréac, Johann Laurent, André Rossi, Pierre Bomel. **Modèle de consommation d'énergie des liens sensible au crosstalk pour les réseaux sur puces**. *COMPAS*, Jul 2016, Lorient, France.

Conférences internationales sans actes

1. Erwan Moréac, Johann Laurent, Pierre Bomel, André Rossi, Emmanuel Boutillon, Andrea Mineo, Maurizio Palesi. **Networks-on-Chip Cortex Inspired Communication To Reduce Energy Consumption**. *Design Automation Conference (DAC 2017)*, Jun 2017, Austin, TX, United States.
2. Erwan Moréac, Johann Laurent, Pierre Bomel, André Rossi. **Bandwidth vs power comsumption tradeoffs for coding strategies on NoC's links**. *Design Automation Conference (DAC 2017)*, Jun 2017, Austin, TX, United States.
3. Erwan Moréac, Johann Laurent, Pierre Bomel, André Rossi. **A bit-accurate power estimation simulator for NoCs**. *DATE 2017 Design Automation and Test in Europe*, Mar 2017, Lausanne, Switzerland.

Communications orales nationales

1. Erwan Moréac. **Optimisation de la latence et de la consommation dans les réseaux sur puce**. *CNRS-GdR RO*, Journée du Groupe de Travail "Optimisation pour les Systèmes Intégrés", Nov 2014, Paris, France.

N ° d'ordre : 467

Résumé

Les progrès dans le domaine des semi-conducteurs ont permis la miniaturisation des puces et l'extension considérable de leurs capacités de calcul et de mémorisation. Cela s'est accompagné d'un accroissement très important du volume des données échangées à l'intérieur de ces puces, limitant les performances au débit de données dans la puce. Ainsi, les concepteurs ont proposé le réseau sur puce (ou NoC : Network-on-Chip) afin de répondre à ces besoins. Cependant, l'accroissement du trafic permis par ce réseau se traduit par une consommation énergétique plus importante engendrant une hausse de la température et une diminution de la fiabilité de la puce. L'élaboration de techniques d'optimisation de l'énergie du NoC est alors nécessaire.

La première partie de cette thèse est consacrée à l'étude de la modélisation des NoCs afin d'estimer leur consommation et d'identifier les composants les plus consommateurs. Ainsi, la première contribution de cette thèse a été d'améliorer la modélisation du NoC en modifiant le modèle d'interconnexions d'un simulateur de NoC existant (Noxim), pour le rendre bit-près (Noxim-XT), et ainsi permettre au simulateur d'incorporer un modèle d'interconnexions considérant les effets du crosstalk, phénomène physique faisant varier leur consommation d'énergie.

La seconde partie de la thèse traite de l'optimisation de la consommation d'énergie du NoC. Ainsi, la recherche d'optimisation s'est orientée vers la réduction d'énergie des liens étant donné leur importante contribution énergétique dans la consommation d'énergie dynamique du réseau. De plus, la part de l'énergie dynamique tend à augmenter avec l'évolution de la technologie. Nous avons proposé à l'issue de cette étude deux techniques d'optimisation pour les interconnexions du NoC. Ces deux optimisations proposent des compromis énergie / latence différents et une extension possible de ces travaux pourrait être la mise en œuvre de la sélection de l'optimisation selon les besoins de l'application en cours.

Mots-clés : NoC, interconnexions, modélisation, consommation d'énergie, latence, simulation, optimisation

Abstract

Thanks to the technology's shrinking, a considerable amount of memory and computing capacity can be embedded into a single chip. This improvement leads to an important increase of the bandwidth requirements, that becomes the bottleneck of chip performances in terms of computational power. Thus, designers proposed the Network-on-Chip (NoC) as an answer to this bandwidth challenge. However, the on-chip traffic growth allowed by the NoC causes a significant rise of the chip energy consumption, which leads to a temperature increase and a reliability reduction of the chip. The development of energy optimization techniques for NoC becomes necessary.

The first part of this thesis is devoted to the study of NoCs power models in order to estimate accurately the consumption of each component. Then, we can identify which ones are the most power consuming. Hence, the first contribution of this thesis has been to improve the NoC power model by replacing the lilnk power model in a NoC simulator (Noxim) by a bit-accurate one (Noxim-XT). In this way, the simulator is able to consider Crosstalk effects, a physical phenomenon that increases links energy consumption.

The second part of the thesis deals with NoC energy optimization techniques. Thus, our research of optimization techniques is focused on inter-router links since their energy contribution regarding the NoC dynamic energy is significant and the dynamic energy tends to stay prominent with the shrinking technology. We proposed two optimization techniques from the study of NoC links optimizations. These two techniques present different energy / latency compromises and a possible extension of this work could be the development of a transmission strategy in order to select the right technique according to the application requirements.

Keywords: NoC, links, modeling, energy consumption, latency, simulation, optimization