Architectures Flexibles pour la Validation et L'exploration de Réseaux-sur-Puce

A. Kouadri-Mostefaoui

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

N°attribué par la bibliothèque
978-2-84813-135-1

THESE

pour obtenir le grade de
DOCTEUR DE L'INP Grenoble

Spécialité : « Micro et Nano Electronique »
préparée au laboratoire TIMA
dans le cadre de l'Ecole Doctorale « Electronique, Electrotechnique, Automatique, Télécommunications, Signal »(EEATS)

présentée et soutenue publiquement

par

Abdellah Medjadji Kouadri Mostéfaoui

Le 24 Août 2009

Titre :

Architectures Flexibles pour la Validation et l'Exploration de Réseaux Sur Puce

Directeur de thèse :
Fréderic Pétrot

JURY :

M. Régis Laveugle	Président
M. Lionel Torres	Rapporteur
M. Smail Niar	Rapporteur
M. Frédéric Pétrot	Directeur de thèse
M. François Pécheux	Examinateur

Remerciements

Je suis reconnaissant à Frédéric Pétrot d'avoir accepté l'encadrement de ma thèse, pour tous ses conseils avisés, ces encouragements, et pour l'ambiance de travail très agréable qu'il a su entretenir.

Je remercie aussi Frédéric Rousseau, avec qui j'ai eu un réel plaisir à travailler, merci pour la confiance, la disponibilité et les nombreuses relectures.

Je remercie vivement Monsieur Régis Laveugle, président du jury, Messieurs Smail Niar et Lionel Torres, mes rapporteurs, ainsi que Monsieur François Pécheux, examinateur.

Je tiens à remercier tous mes collègues du groupe System-Level-Sythesis, en particulier : Benaoumeur Senouci, Katalin Popovici, Hao Shen et Marius Bonaccio, aussi bien pour leurs aides, que pour les bons moments qu'on a passé ensemble.

Enfin, merci à tous les membres du laboratoire TIMA que j'ai eu le plaisir de côtoyer durant mes trois années de thèse.

à mes parents,
à toute ma famille,
à boolbool,

Résumé

L'infrastructure de communication pour un système multiprocesseur mono-puce (MPSoC) est un organe central et de première importance. Cette importance s'explique par la place importante que tiennent les communications dans de tels systèmes distribués. Alors qu'il est maintenant admis que les réseaux-sur-puce (NoCs) constituent une solution théoriquement idéale, il se pose le problème de la validation de telles architectures complexes. En effet, malgré la régularité de leurs architectures, les réseaux-sur-puce restent des systèmes dont les interactions internes sont très difficiles à appréhender. Par ailleurs, les approches de validation classiquement employées sont très mal adaptées aux systèmes à base de NoC car très peu flexibles et très peu scalables.

Cette thèse introduit un nouveau concept dans la validation matérielle des réseaux-sur-puce, ce concept que nous avons appelé « émulation imprécise » contraste avec les approches d'émulation matérielles classiques qui sous-entendent toutes une précision au « cycle près, bit près ». Notre approche hérite de tous les avantages liés au prototypage matériel sur les plateformes reconfigurables et y ajoute un degré de flexibilité très élevé. En effet, l'étude menée au cours de ce travail sur le comportement des réseaux-sur-puce à commutation de paquets en régime non congestionné montre que, sous certaines conditions, des modifications des caractéristiques du NoC (introduites par la plateforme d'émulation elle même) peuvent être tolérées sans que pour autant le comportement du réseau ne change de façon radicale.

La technique d'émulation multi-FPGA étudiée dans cette thèse est une technique très flexible car basée sur un mode d'interconnexions inter-FPGA série. Les interconnexions séries sont beaucoup moins sensibles aux phénomènes de parasitage que les interconnexions parallèles et par conséquent les vitesses de transferts sont beaucoup plus élevées. D'autre part la technique d'émulation que nous proposons ne pose aucune condition sur la vitesse du processus d'émulation lui-même. Considérant les délais additionnels induits pas les liaisons séries et les vitesses d'émulation très élevées, un phénomène de déviation des performances peut être observé d'où l'imprécision de l'émulation. Ce phénomène a été étudié dans le cadre de cette thèse et nous avons proposé plusieurs solutions afin d'y remédier.

Abstract

For a multiprocessor system-on-chip (MPSOC), the communication backbone is a central component of prime importance. This is due to the importance of the communications on such distributed systems. Now that networks-on-chip (NoCs) are admitted to be the solution which theoretically best solves the problem of on-chip communications, an important problem which rises consists in providing the designer with fast validation techniques able to tackle such complexes systems. Indeed, despite their regular architectures networks-in-chip internal interactions are difficult to formalize. On the other side, classical validation approaches are far from being suited for NoC-based systems due to their lack of flexibility and scalability.

This thesis introduces a new concept in the field of hardware validation of networks-on-chip; we have called this new concept "Inaccurate Hardware Emulation" in contrast with most existing hardware emulation approaches which assume a "cycle accurate bit accurate" precision. Our approach inherits from all advantages of hardware prototyping on reconfigurable devices and adds new scalability features. Study conducted during this thesis showed that under the non-congested regime a NoC may admit a number of alterations on its characteristics (introduced by the emulation platform) without adopting a completely different behavior.

The multi-FPGA emulation technique proposed in this thesis is highly flexible since it relies on serial inter-FPGA interconnections. Serial interconnections are less sensitive to noises than parallel style of interconnections, and allow then for higher transfer rates. On the other hand, our emulation approaches does not poses any constraint on the emulation speed. If we consider the fact that serial interconnection schemes may introduce additional delays and the high speeds of the emulation process, performance of the NoC being emulated on the multi-FPGA emulator may deviate from the original NoC. We have studied this phenomenon and we have proposed various solutions for it.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1 : Introduction & Problématique

1.1 Introduction et contexte de la thèse

Dans la perspective de fournir aux applications logicielles des plateformes matérielles toujours plus performantes capables de répondre aux exigences d'interactivité et de fluidité de traitement, les recherches se sont dirigées très tôt vers les architectures multiprocesseurs et cela bien avant l'avènement de l'informatique embarquée. De prime abord, un system-on-chip (SoC) à multiprocesseurs constitue une solution idéale à la fois en termes de puissance de calcul, de bilan énergétique et de productivité. En effet l'intégration de toutes les fonctionnalités d'un système informatique sur une seule puce tel que les capteurs analogiques, les composants d'acquisition de données, les processeurs de traitement de signal, les processeurs de traitements logiques, réduit la complexité des processus de fabrication et permet donc de réduire les coûts de production. Cependant il est vrai qu'à ce jour plusieurs problématiques principalement liées au parallélisme restent sans solutions. Cela empêche l'exploitation pleine des potentialités de telles architectures.

Un des aspects les plus problématique lié aux architectures multiprocesseur sur puce (MPSoC), consiste à faire communiquer tous les composants de façon cohérente et sans impact sur les performances globales du système. Bien que ce problème soit récurent et était déjà bien identifié dès les débuts des systèmes sur puce (SoC), il restait toujours possible de le contourner par des solutions ad-hoc. Néanmoins, les systèmes sur puce actuels ont des contraintes de performances (toutes métriques confondues) telles qu'il a été nécessaire de repenser complètement les solutions de communications jusqu'alors utilisées. Les recherches dans ce domaine se sont donc dirigées vers les architectures de réseaux sur puce (NoC) [1], qui par certains aspects sont très proches des macro-réseaux. Les réseaux sur puce constituent actuellement la seule proposition d'architecture capable de répondre aux exigences des applications actuelles. Cependant une exploitation pleine des potentialités d'un réseau sur puce nécessite que le concepteur dispose d'un certain nombre d'outils et de méthodes d'aide à la conception.

Afin de bien cerner les problématiques liées à l'intégration efficace d'un réseau-sur-puce sur une architecture SoC, il est nécessaire de rappeler que tous les processus de conception associés aux systèmes SoC et MPSoC actuels posent un problème d'optimisation efficace des performances [2][3]. Ce problème typique est engendré par trois facteurs importants :

1. Un espace de recherche très large car chaque organe du système considéré a son propre ensemble de paramètres de fonctionnement ds_i (ex : largeur de ligne de cache, fréquence de fonctionnement, taille mémoire cache) ; paramètres dont l'impact sur les performances globales du système se manifeste de façon plus ou moins prononcée. L'espace de recherche global DS peut être vu comme étant le produit cartésien de tous les ds_i correspondants à chacun des composants.
2. La complexité des interactions entre composants d'un système MPSoC est telle qu'il est impossible ou du moins très difficile de proposer une organisation logique de l'espace de recherche. Une telle organisation rendrait possible l'application de méthodes formelles [4] pour la recherche de solutions optimales, mais les coûts matériels pour garantir les propriétés nécessaires à ces interactions seraient exorbitants.
3. La mesure même des performances d'un système sur puce reste encore un problème très difficile. Quelle que soit la méthode de mesure des performances adoptée, il se pose le problème de la non-intrusivité et de la précision des mesures.

Pour revenir aux réseaux-sur-puce (NoCs), étant eux même des organes constitués de composants (interfaces, routeurs, liens, buffers) ils héritent de toutes les problématiques d'exploration [5], mesure et optimisation des performances énoncées précédemment pour les MPSoC. Une conséquence à cela est que l'intégration d'un NoC dans un système MPSoC rend encore plus grand l'espace de recherche et ajoute donc un degré de difficulté au processus d'exploration.

Le premier facteur (1) étant intrinsèquement lié aux systèmes complexes (MPSOC et NoC), il n'est possible d'aborder les problèmes de conception/optimisation que sur les deux derniers plans à savoir : essayer de formaliser autant que possible les interactions et le

comportement d'un NoC, et deuxièmement proposer des méthodes rapides et précises pour la mesure des performances.

La formalisation du comportement d'un NoC en vue de l'évaluation analytique de ses performances reste une tâche difficile, modéliser et reproduire des conditions de trafic à l'entrée du réseau identiques aux conditions d'exploitation reste encore très difficile à mettre en œuvre. A cela s'ajoute le manque d'outils mathématiques pour modéliser et évaluer les performances d'éléments constituant le NoC dont le comportement évolue de façon dynamique (files bloquantes [6], routages adaptatif…).

En ce qui concerne les méthodes de validation des architectures de NoCs, elles peuvent être classées en trois catégories :

1) Description dans des langages de haut niveau : Ces méthodes permettent d'atteindre des vitesses de validation très élevées ; cependant seuls quelques aspects fonctionnels peuvent être effectivement adressés. Il n'est pas possible par exemple d'avoir des mesures de performances (latence, congestion) à ce niveau de description mais juste de vérifier si une politique d'ordonnancement n'est pas interbloquée, par exemple.
2) Par simulation : Ici le NoC est entièrement décrit puis modélisé dans un langage exécutable et/ou interprétable. Dans ces cas-ci en plus de la validation fonctionnelle, il est possible d'effectuer des mesures de performances dont la précision dépendra du niveau d'abstraction du modèle de simulation (niveau transactionnel TLM **Erreur ! Source du renvoi introuvable.**, niveau RTL). La simulation nécessite des délais pour modéliser tous le NoC, et les temps de simulations sont d'autant plus longs que la précision souhaitée est grande.
3) Par prototypage ou émulation sur plateformes configurables : ici le NoC est implémenté de façon matérielle sur une plateforme reconfigurable (typiquement une plateforme FPGA). Le principal avantage du prototypage réside dans les vitesses de validation pouvant être atteintes, vitesses comparables aux vitesses de fonctionnement réelles du NoC. La vitesse à laquelle une instance de NoC peut être testée et validée est un paramètre clé dans la conception des systèmes SoCs, MPSoCs et NoCs car il peut être nécessaire de tester plusieurs configurations de NoC (dizaines, centaines, milliers) avant d'arriver à une implémentation satisfaisante sur le plan des performances (meilleur compromis). L'approche par prototypage elle-même n'introduit aucune imprécision sur les performances temporelles mesurées (latence, congestion, débit), et ne dépend que des conditions de test appliquées au NoC.

En pratique, et particulièrement dans le contexte industriel, la simulation reste majoritairement employée comme méthode de validation des NoCs. Un modèle exécutable du

NoC est initialement construit, et partant des mesures de performances obtenues, des modifications seront apportées à l'architecture originale. Comme nous l'avons indiqué précédemment les temps de simulation ne sont pas négligeables et peuvent être très longs, ce qui limite les possibilités d'exploration architecturales. Le prototypage étant un processus plus rapide et au moins aussi précis que la simulation RTL, même si les possibilités d'introspection sont plus limitées, nous nous intéressons dans cette thèse au prototypage des réseaux-sur-puces sur les plateformes FPGA en vue de l'exploration architecturale.

1.2 Problématiques de la thèse

Dans le paragraphe précédent nous avons introduit plusieurs problèmes liés à l'intégration d'un NoC dans un système MPSoC; notamment le problème d'exploration architecturale pour de tels systèmes qui nécessite des outils de validation et de mesure de performances très rapides et aussi précis que possible. Nous avons également montré que le prototypage sur plateforme FPGA pouvait être utilisé pour répondre aux besoins de vitesse et de précision de la validation. Les problématiques adressées dans cette thèse s'articulent donc autour du prototypage matériel des NoCs.

Optimisation des architectures de NoCs pour le prototypage FPGA

Comme nous l'avons souligné auparavant, les NoCs sont des systèmes électroniques relativement complexes. Cette complexité se traduit par un grand besoin en ressources (cellules logiques, cellules mémoires et interconnexions) lors de l'implémentation matérielle. Lors du processus de prototypage et afin d'être en mesure de tester le NoC sous des conditions aussi proches que possible de celles de son utilisation réelle, il devient primordiale d'économiser un maximum de ressources pour l'intégration des autres composants du système MPSoC sur la plateforme FPGA (générateurs de trafic, processeurs, mémoires, IPs). Ce problème d'intégration est crucial et il est nécessaire de tirer partie à la fois des propriétés des éléments constitutifs des réseaux intégrés et des caractéristiques des FPGA pour faciliter et optimiser le processus de prototypage.

Emulation des réseaux intégrés à grande échelle

Cependant il reste vrai que pour certains systèmes MPSoC à base de NoCs les besoins en ressources matérielles sont tels qu'ils ne peuvent être satisfaits par une seule puce FPGA en particulier pour les systèmes modernes comprenant un nombre de composants élevé. Jusqu'à maintenant il existait deux solutions à ce problème : le recours aux émulateurs industriels, dispositifs extrêmement coûteux et dont les vitesses maximales sont de l'ordre de quelques Mhz seulement; ou encore le recours à des plateformes multi-FPGA qui offrent un meilleur compromis coût/vitesse (FIGURE 1.1) (il existe d'autres méthode hybride basées sur le principe de co-émulation [7]).

Partant de ce constat, la deuxième problématique qui se pose est la définition et la mise en œuvre d'une plateforme multi-FPGA adaptée au prototypage de NoC ainsi que le flot associé. Le cahier des charges d'une telle plateforme comporte principalement trois exigences :

- La flexibilité
- La précision
- La vitesse (ou fréquence de fonctionnement)

La flexibilité est une exigence capitale, et cela afin d'être en mesure d'offrir les capacités d'intégration nécessaires quelque soit la taille du NoC (ou MPSoC) considéré. D'un autre point de vue, la flexibilité se traduit par la facilité de construction et de configuration de la plateforme. La connexion entre plusieurs cartes FPGA est un problème central, en termes de vitesse de transfert, à cause de la difficulté d'échantillonner simultanément sur une nappe de fils parallèles. Pour cela au lieu d'utiliser des interconnexions de type parallèles le plus souvent employées, nous avons opté pour des interconnexions de type série à très haute vitesse. Nous détaillerons plus en avant les motivations ayant conduit a ce choix dans les chapitres suivants.

D'autre part, il est clair que la précision des mesures de performance permises par la plateforme devait être maximale pour garantir la convergence du processus d'exploration. Cependant, comme nous l'aborderons par la suite, l'utilisation de liens séries et notre choix de ne poser aucune restriction sur la fréquence du processus d'émulation introduisent sous certaines conditions des variations sur les mesures de performances (performances

réelles/performances mesurées). Pour pallier ce phénomène nous proposons un flot de prototypage multi-FPGA et une approche analytique destinée 1) à réduire au minimum la variation des performances et 2) à corriger les mesures et inférer les performances réelles (telle que mesurées sur un émulateur).

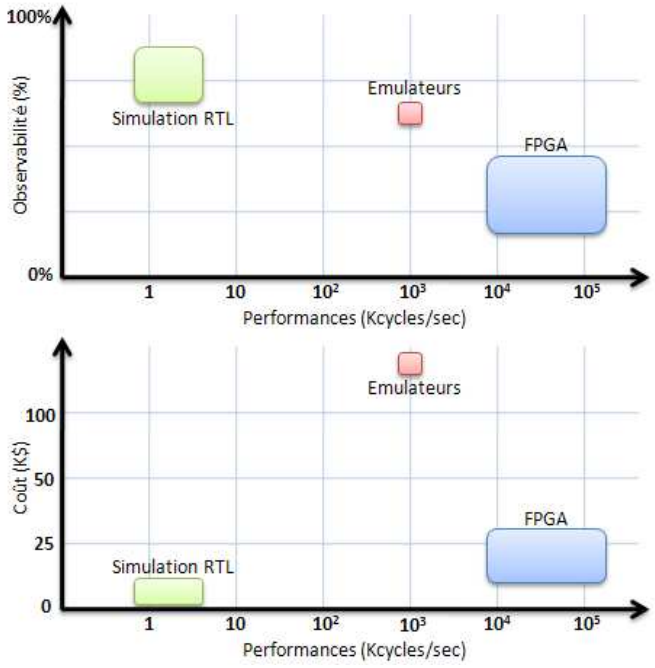


FIGURE - 1.1 Comparaison entre la validation par Simulation/Emulation et Prototypage FPGA

1.3 Contributions de la thèse

La première contribution de cette thèse propose un flot générique de prototypage de NoCs tenant compte de certaines de leurs spécificités architecturales afin d'optimiser leurs implémentation sur FPGA. À titre d'exemple, la régularité des topologies de NoC peut être efficacement exploitée lors de la phase de placement routage sur FPGA, il est aussi possible d'optimiser l'architecture interne des routeurs par l'utilisation de certaines primitives FPGA spécifiques. Une autre perspective consiste en l'optimisation des ressources du NoC selon les besoins de l'application cible. En outre, et concernant le prototypage il nous a été nécessaire de développer un certain nombre de composants aussi bien matériels que logiciels permettant justement de mesurer les performances d'un NoCs une fois implémenté. Ces moniteurs

destinés être implémentés de façon matériel pour la plupart devaient bien sûr nécessiter un minimum de ressources.

La deuxième contribution correspond à la proposition d'une architecture de plateformes multi-FPGA d'émulation de NoCs, ainsi qu'un flot de prototypage multi-FPGA. Ce flot de prototypage comprend trois étapes : 1°) une étape de partitionnement du NoC 2°) Une étape d'implantation et 3°) une étape de mesure et de correction des performances.

Pour le développement de cette plateforme, il nous a été nécessaire de résoudre un certain nombre de problèmes liés principalement à l'exigence d'avoir une fréquence de fonctionnement aussi élevée que possible ; cette exigence se traduit en pratique par l'impossibilité d'avoir un mécanisme de synchronisation globale inter-FPGA pour garantir la cohérence du processus d'émulation. Pour pallier cela, nous avons développé des interfaces inter-FPGA spéciales réduisant au maximum les besoins en synchronisation. La phase de partitionnement de NoCs tient aussi compte de cette limitation et implémente un algorithme qui vise à limiter l'impact du manque de synchronisme global. Pour résumer la FIGURE 1.2 explicite les contributions la thèse :

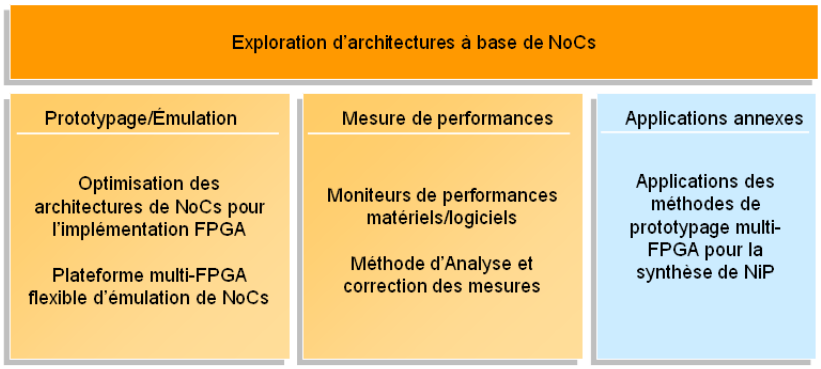


FIGURE - 1.2 Contributions de la thèse sur l'exploration d'architectures à base de NoCs

Une dernière contribution de ce travail de thèse comme indiqué sur la FIGURE 1.2 concerne l'exploration et la synthèse d'architectures de réseaux-empaquetés [9] *«NiP : networks-In-package»*, ce nouveau concept émergeant correspond à l'avènement de nouvelles technologies de puce silicium empilées [10] *« SiP : System-In-package »*. Un SiP est constitué de plusieurs circuits intégrés (puces) empilés, l'avantage de tels circuits par rapport au style d'assemblage standard connu sous le nom de MCM [11]*«Multi-Chip-*

Module» est un gain considérable en surface et en consommation d'énergie. Dans la dernière partie de cette thèse nous exposerons une méthodologie de synthèse de NiP directement inspirée de la méthodologie de prototypage multi-FPGA. Un NiP n'est autre qu'un NoC éclaté sur plusieurs puces silicium, tous comme pour la plateforme d'émulation multi-FPGA que nous proposons où le NoC est partitionné sur plusieurs puces. Une autre analogie découle des limitations d'interconnexions sur les NiP qui imposent que les liaisons inter-puces soient des liaisons sérielles ; Par conséquent toutes les problématiques de partitionnement, mappage et de choix de schémas de sérialisation de la méthodologie multi-FPGA ont été adaptées afin de proposer un flot de synthèse de NiP.

Dans le reste de ce mémoire nous développerons chacune des problématiques exposées dans ce chapitre et nous détaillerons aussi les solutions qui y ont été proposées, en plus du présent chapitre d'introduction, ce mémoire comporte quatre autres chapitres :

- Le deuxième chapitre établit un état de l'art détaillé sur les architectures et les mécanismes internes des réseaux intégrés.
- Dans le troisième chapitre nous exposerons quelques techniques de synthèse et d'exploration de NoCs et nous introduirons un flot de prototypage FPGA optimisé.
- Le quatrième chapitre sera consacré à la présentation de la technique de prototypage multi-FPGA des NoCs.
- Le cinquième chapitre traite de la synthèse et l'exploration d'architectures de NiP.

Le sixième et dernier chapitre analyse les résultats de ces travaux et présente les conclusions et les perspectives de ce travail de thèse.

Chapitre 2 : Etat de l’art sur les réseaux-sur-puce

Sommaire

2.1 Introduction

Il est clair que les réseaux-sur-puce ou NoC offrent de réels avantages [12] par rapport aux structures d'interconnexion utilisées jusqu'à présent pour les SoCs. Cependant un certain nombre de caractéristiques principalement architecturales rendent le comportement des NoC faiblement déterministe et donc difficilement cernable surtout aux niveaux d'abstraction les plus élevés. La nature distribuée et parallèle des NoC, où les événements à l'entrée sont faiblement corrélés entre eux, induit encore plus de complexité dans les interactions régissant le comportement global du NoC. Dans le contexte actuel où la réutilisation et l'assemblage rapide d'IP «*Intellectuel Property*» sont la règle, il serait très couteux pour un concepteur de s'attarder sur des expérimentations longues afin d'essayer de caractériser les performances d'un réseau pour telle ou telle application. De plus c'est une tâche dont l'intérêt resterait somme toute très relatif au vu de l'ultra sensibilité des NoCs aux variations de trafic à l'entrée.

Dans ce chapitre d'introduction, nous détaillerons les principaux mécanismes et aspects relatifs aux NoCs, afin de mieux comprendre leur impact sur le comportement global. Seront exposés aussi bien les macro-caractéristiques telles que la topologie du réseau et les techniques de commutations, que des mécanismes plus fins (Control de flux, routage, bufférisation, mécanismes d'horloge).

Nous commencerons par une présentation rapide des solutions d'interconnexion classiques, donnée ici afin de bien préciser le contexte d'émergence des NoCs. Pour illustrer les concepts clés associé au NoCs nous terminerons ce chapitre par une présentation non exhaustive de quelques réseaux à vocation académiques aussi bien qu'industrielle.

2.2 Schémas d'interconnexions standards pour SoC

La communication est un aspect très important pour les performances globales d'un système sur puce, au même titre que la vitesse et la technologie du processeur. Le besoin de disposer de mécanismes d'interconnexions efficaces pouvant relier les processeurs aux mémoires et aux différents dispositifs d'entrées/sorties est apparu très vite. Dans les paragraphes qui suivent nous introduirons les trois types d'interconnexions majoritairement

employés jusqu'à présent, à savoir : les bus partagés, le crossbar et les réseaux-sur-puce. Parallèlement à cette présentation nous mettrons en avant à chaque fois les avantages et inconvénients liés à chaque solution.

2.2.1 Le Bus partagé

Ce type d'interconnexions reste encore très largement utilisé dans beaucoup de systèmes sur puce. Plusieurs standards de bus sont actuellement disponibles tel que AMBA de ARM [13], le CoreConnect [14] d'IBM ou encore le STBus [15] de STMicroelectronics. Sans trop rentrer dans les détails d'implémentation physique, un bus peut être vu comme étant fonctionnellement équivalent à un multiplexeur qui sélectionne, par une fonction d'arbitrage, un maitre unique qui se voit attribuer le droit d'accéder à un esclave pointé par son adresse; et cela pendant une durée déterminée (généralement le temps d'un transfert). Il est clair que l'atomicité des accès induit de fait une sérialisation des requêtes concurrentes pouvant être émises par différents maitres. La sérialisation dans le temps des accès se traduit par des latences moyennes d'autant plus élevées que le nombre de maitres augmente, ce problème est connu dans la littérature sous l'appellation de scalabilité limitée [16]. En outre, la structure même du bus limite le nombre maximal de modules pouvant être connectés y être connectés, les longueurs des fils et donc la consommation électrique [17] et la surface sont les rasions principales qui limitent ce nombre.

Quand le nombre d'*IPs* qu'il est nécessaire d'intégrer ne peut être supporté par un seul bus, la solution consiste à utiliser plusieurs bus reliés entre eux par des ponts (Bus Bridges). Cette solution se décline en fait sous plusieurs variantes : les bus hiérarchiques et les matrices de bus. L'idée principale étant de permettre des accès simultanés autant que possible dès lors que les accès se font sur des nœuds esclaves distincts. L'architecture décrite en FIGURE 2.1 (a) du bus « *AMBA AHB Multi-layer* » est donnée ici pour illustrer ce principe.

Bien que plus flexible en terme de nombre d'IP pouvant être connectées, et plus performante que l'approche par bus unique, cette solution pose un certain nombre de problèmes. Notamment ceux de la surface silicium nécessaire pour la matrice d'interconnexion, et la cohérence de l'adressage globale.

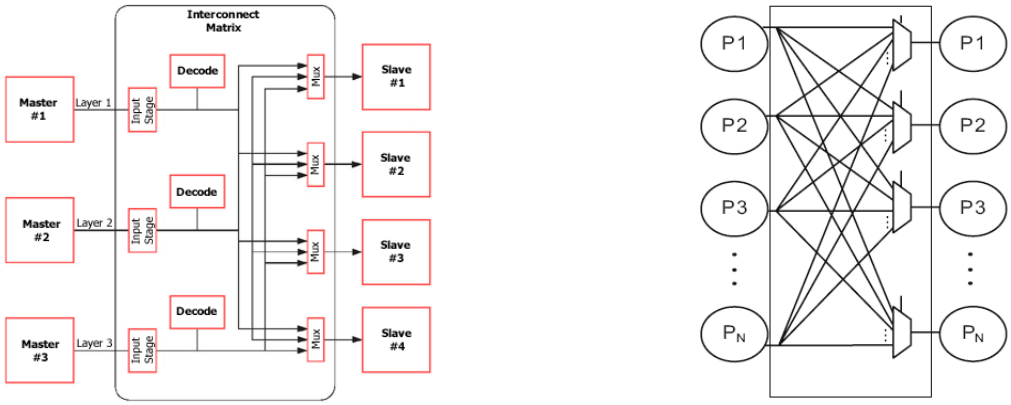


FIGURE 2.1 - (a) Matrice d’interconnexions AMBA-AHB (b) Crossbar Complet NxN

2.2.2 Le Crossbar

Une architecture crossbar (FIGURE 2.1 (b)) permet d’avoir un schéma d’interconnexion global, c'est-à-dire qu’il existe un chemin relient n’importe quel couple maitre-esclave. Comparé à un bus hiérarchiques, le crossbar offre une meilleure bande passante et donc une meilleure latence du fait que les chemins maitres-esclaves sont séparés les uns des autres. Néanmoins le nombre de connexions physiques nécessaires à l’implémentation d’un crossbar complet limite grandement les champs d’application de tels réseaux. Il est communément admis qu’au delà de dix IP le crossbar n’est plus adapté [18].

Dans certains travaux de recherche tel que [19][20], des études sur l’exploitation du schéma de communication au niveau de l’application pour la synthèse de crossbars partiels, montrent qu’il est possible de repousser encore plus loin la taille critique du crossbar. Cependant comme tous les flot de conception se basant sur des spécificités de l’application cible, le domaine d’applicabilité de cette approche demeure restreint.

2.2.3 Les réseaux sur puce

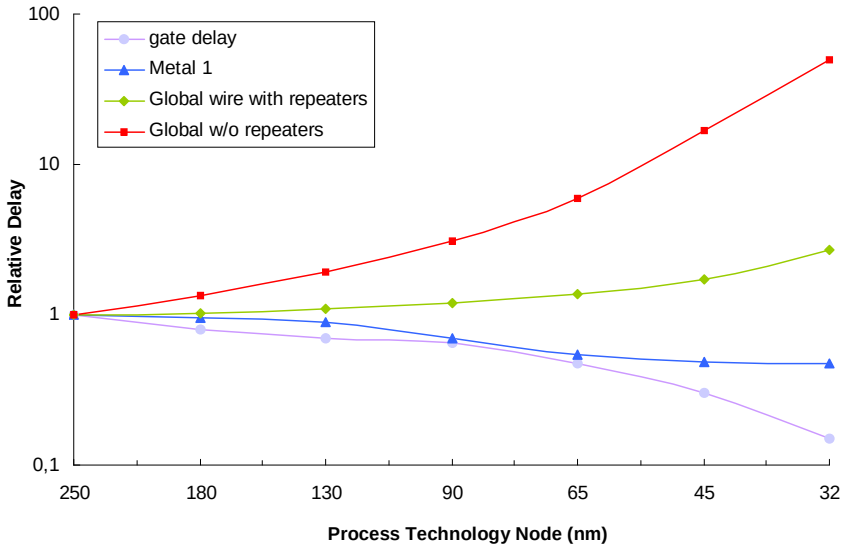


FIGURE 2.2 - Prédictions de l'évolution des délais des interconnexions par rapport aux délais des portes logiques [21]

Plus qu'un nouveau paradigme pour les communications intra-puce [22], les réseaux-sur-puce marquent le passage net entre la génération de systèmes centrés sur le calcul aux systèmes centrés sur la communication. En effet, alors que les technologies d'intégration actuelles rendent abordables des puissances de calcul inimaginables jusqu'à peu ; elles introduisent de nouvelles contraintes physiques pour les interconnexions intra-puce. Ces contraintes, caractérisées principalement par des délais de propagation sur les interconnexions dominants par rapport aux délais des portes logiques (FIGURE 2.2), rendent l'implémentation des solutions de communication classiques très difficile. À cela il faut ajouter qu'aux fréquences de fonctionnement élevées (de l'ordre du GHz pour les systèmes actuels), et considérant les délais de propagation, la gestion et la distribution d'un signal d'horloge unique induit un surcout non négligeable en terme de surface et surtout de dissipation d'énergie [23]. Outre les contraintes technologiques, les cycles de conception et de production très courts tolèrent de moins en moins les solutions de communication ad-hoc (typiquement les bus hiérarchiques) nécessitant pour chaque nouveau système des efforts de vérification et de caractérisation. Sont privilégiées les solutions plus scalables et plus flexibles se rapprochant

des réseaux-sur-puce. Sur le tableau 2.1 sont résumés les principaux avantages liés à l'utilisation des réseaux-sur-puce au vu des problématiques induites par le contexte technologique actuel.

TAB 2.1 Principales problématiques des SoC et avantages apportés par les réseaux sur puce

Problématique	***Solution apportée par les réseaux sur puce***
Délais de propagation	Impact minime car les interconnexions globales sont fractionnées en chemins plus courts avec possibilité de pipelining
Distribution d'horloge	Architecture modulaire et mode de transport par paquets : adaptés aussi bien aux approches synchrones qu'aux approches multi-synchrones ou asynchrones
Bande passante	Globalement élevée car plusieurs chemins de transport peuvent être utilisés en parallèle
Scalabilité/ Flexibilité	Architectures régulières facilement extensibles en fonction du nombre d'IP, sans dégradation majeur des performances temporelles et électriques.

Comme nous l'avons montré ici, le recours aux réseaux-sur-puce tend d'une part à répondre aux nouvelles contraintes d'intégration, et d'autres part à proposer des solutions adaptées au besoins des nouvelles architectures de SoC à la fois en terme de capacités de communications et de flexibilité. Cependant, cette position à cheval entre plusieurs domaines d'optimisation, comme dans la plupart des problèmes de codesign d'ailleurs, rend la conception et la validation d'architectures de réseaux efficaces très difficile. Après cette introduction, et avant de discuter en détails les apports réels et les difficultés liées à la conception et l'intégration d'un réseau-sur-puce, nous introduisons les principaux concepts des réseaux sur puce.

2.3 Architectures et mécanismes des réseaux sur puce

2.3.1 La topologie

La topologie définie l'organisation physique du réseau, en terme de placement et de connectivité des nœuds du réseau les uns par rapport aux autres. Les nœuds du réseau sont assimilés ici aux routeurs (ou encore Switch) éléments constitutifs de base des NoCs. Du point de vue de l'implémentation matérielle (ASIC) la topologie à un impact direct à la fois

sur les performances temporelles, la surface et dans une moindre mesure la consommation [24].

En général les performances sont d'autant plus dégradées que le nombre de liens ainsi que leurs longueurs relatives augmentent, le placement et routage de telles structures produisant généralement des interconnexions avec des longueurs proportionnelles à celles de la topologie d'origine. L'impact sur la consommation électrique, étant dû au phénomène des capacités parasites et du cross-talk [25], dépend lui de la configuration des liens (largueur en bit, fréquences). Cela explique la préférence actuelle des concepteurs pour les topologies planes et régulières dont la plus représentative est celle de la grille à deux dimensions. Cependant il est à noter ici que la régularité d'une topologie n'est pas une garantie de performances en soit, car le déséquilibre dans les surfaces des différentes IP peut quand même induire des interconnexions longues après placement et routage.

En plus des caractéristiques ayant trait à l'implémentation physique, un certain nombre de métriques abstraites sont couramment utilisées, parmi lesquelles :

- *Le degré des nœuds :* correspond au nombre maximal ou moyen d'E/S associés aux nœuds du réseau. Reflète la complexité de la topologie.
- Le diamètre de la topologie : représente la distance maximale en nombre de sauts (liens parcourus) dans la topologie. Donne une indication sur la latence maximale à vide du réseau.
- *La bande-passante de bissection :* C'est le nombre minimal de liens qu'il est nécessaire de neutraliser afin de diviser le réseau en deux sous-réseaux équivalents. Cette métrique permet d'évaluer de façon empirique la bandes passante maximale dans le cas d'un trafique uniformément distribué.
- *La régularité :* c'est la possibilité de décrire les interconnexions de la topologie sous forme mathématique. Une topologie régulière permet d'implémenter des algorithmes de routages simples et de surcroit nécessitant des ressources limitées contrairement à une topologie irrégulière. La régularité se traduit généralement par des architectures de routeurs homogènes.

En plus de la grille à deux dimensions, parmi les topologies les plus utilisées on retrouve le tore [26] l'arbre élargie [27] et l'octogone [28]. Jusqu'à présent ces topologies planaires avaient la préférence des concepteurs et des chercheurs du fait qu'il était possible d'en faire assez facilement une implémentation matérielle. Les topologies non planaires

incluant des interconnexions sur un troisième plan ont naturellement toujours étaient écartées. Cependant un regain d'intérêt pour ces topologies en 3D est à noter avec l'avènement des technologies à puces empilées [29] [30].

2.3.2 Mécanismes de commutations

La commutation exprime la façon par laquelle les flux de données sont acheminés d'une IP émettrice à une autre destinataire à travers le réseau. En fonction des contraintes sur les délais d'acheminement [31] on peut avoir principalement deux modes de commutation :

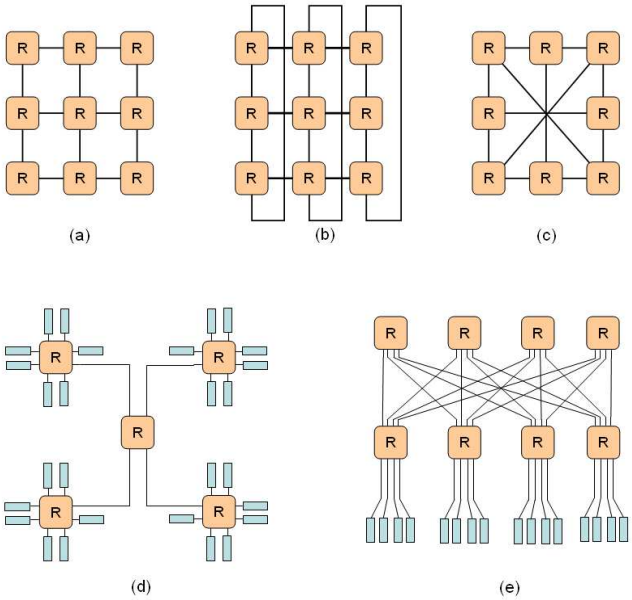


FIGURE 2.3 - Topologies usuelles des réseaux sur puce (a) Grille-2D (b) Tore (c) Octogone (d) Etoile (e) Arbre

a. ***La commutation de circuit :*** Au préalable à chaque transmission toutes les ressources (liens, ports) nécessaires sur le chemin devant être parcourus sont réservées par des messages de signalisation spéciaux. Les messages de signalisation peuvent être acheminés via des signaux de contrôles dédiés ou bien transiter sur le chemin des données sous forme de paquets d'ouverture de chemin. Les ressources ainsi allouées ne sont libérées qu'une fois la transmission terminée. La commutation de circuit assure une excellente qualité de service et

est particulièrement adaptée aux trafiques temps réels. Néanmoins, la phase de négociation initiale nécessitant des délais assez importants cette technique n'est vraiment efficace que pour des gros volumes de données.

b. ***La commutation de paquets :*** Ce mode de commutation requiert qu'un message soit préalablement divisé en plusieurs paquets (eux même divisés en flit). Les paquets sont injectés dans le réseau en se basant sur des informations de disponibilité locales, c'est-à-dire qu'aucune garantie de disponibilité du chemin dans sa globalité jusqu'à l'IP destination n'est nécessaire. Une décision est prise au niveau de chaque routeur pour déterminer si un paquet doit être renvoyé vers le port de sortie correspondant ou bien s'il doit attendre en cas d'indisponibilité du port concerné. Ce fonctionnement implique que chaque paquet contienne un certain nombre d'informations servant à déterminer le chemin devant être suivi. En outre, des buffers sont nécessaires au niveau de chaque routeur, afin de pouvoir conserver les paquets en cas d'indisponibilité de ressources. Ce type de fonctionnement induit forcément des délais plus importants comparé au mode par commutation de circuit, délais pouvant même devenir imprévisible en cas de forte charge du réseau. Il est donc clair que la commutation de paquets est plutôt adaptée à des trafics à faible exigences en qualité de service. Comme nous l'avons indiqué plus haut, le mode de transport par paquet impose que des buffers de sauvegarde soient présents tout le long du chemin de transport. Les stratégies de gestion de la bufférisation de paquets peuvent être catégorisées en trois familles :

Store and Forward (SF): Les paquets sont considérés ici comme entités indivisibles, et avancent dans un buffer en amont que si l'espace disponible est au moins assez suffisant pour le paquet courant. Il est donc nécessaire que chaque buffer soit de taille au moins égale à celle du paquet le plus long pouvant circuler sur le réseau.

Virtual Cut-Through (VCT) : Cette stratégie reprend le principe du Store and Forward et a donc les même besoins en espace de bufférisation, à la différence près qu'un paquet peut commencer à être expédié au buffer suivant sans qu'il soit nécessairement arrivé dans sont intégralité au nœud courant. L'avantage de cette approche par rapport à la précédente réside dans le gain substantiel en latence.

Wormhole (WH): Le but ici étant de réduire les espaces de bufférisation au niveau des routeurs, un paquet peut être expédié au buffer suivant sans qu'il soit nécessaire que ce dernier puisse accepter tout le paquet. En effet il suffi que l'espace équivalent à la taille d'un

seul flit soit disponible. Le flit d'entête du paquet ouvre le chemin et réserve en même temps les ressources nécessaires à tout le paquet, les autres flits du message le suivent en séquence. Au passage du dernier flit le chemin est libéré et devient donc disponible aux autres paquets. La stratégie Wormhole réduit à l'extrême l'espace de bufférisation total et offre de meilleures latences de transfert que SF et VCT, cependant les risques d'inter- blocage et de contention se voient accrus d'autant.

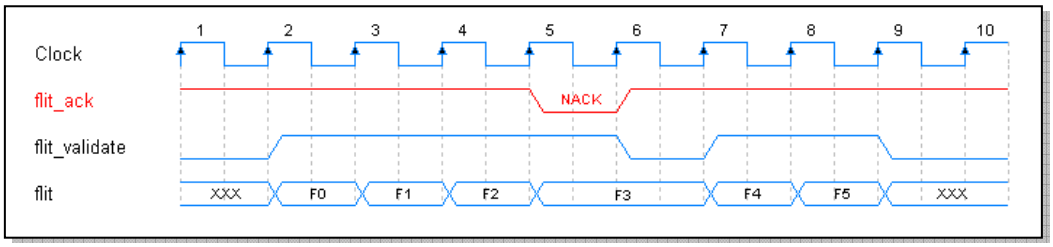
La commutation de paquet est le plus souvent préférée à la commutation de circuit, car elle offre une meilleure gestion de la bande passante globale par le mécanisme d'ordonnancement entrelacé possible uniquement en mode paquet. Bien que la latence à vide en mode commutation de circuit soit bien meilleure, les temps de négociation de connexions donnent des latences moyennes bien supérieures à celles par commutation de paquets. La stratégie Wormhole est aussi prédominante dans la multitude d'implémentations de réseaux existantes. Cela s'explique par des besoins en ressources moindres ce qui se traduit concrètement par une réelle diminution de la surface et de l'énergie. En effet les cellules mémoires sont responsables de la plus grande partie de la surface et de la consommation électrique (en particulier sa composante statique) dans un routeur.

2.3.3 Protocoles de contrôle de flux

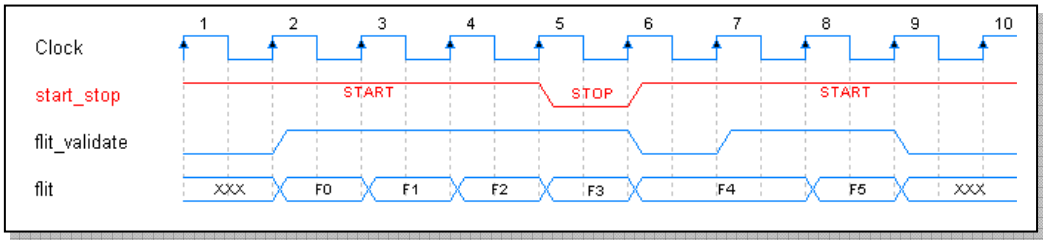
Les protocoles de contrôle de flux ont pour but de garantir un transport de paquets sûr depuis une IP source jusqu'à l'IP destination. La notion de sureté réfère ici a la garantie qu'aucun paquet ne risque pas d'être perdu à cause d'un dépassement de capacité de l'un des buffers parcourus. Aussi appelés protocoles de régulation au niveau liaison de données, ils ont pour but d'assurer la cohérence de la bufférisation afin de gérer le déséquilibre entre le taux d'injection des paquets et le taux effectif auquel ils sont éjectés du réseau. Comme pour les stratégies de gestion de la bufférisation, il existe plusieurs protocoles de contrôle de flux [32], nous détaillerons dans ce qui suit les trois principaux protocoles :

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

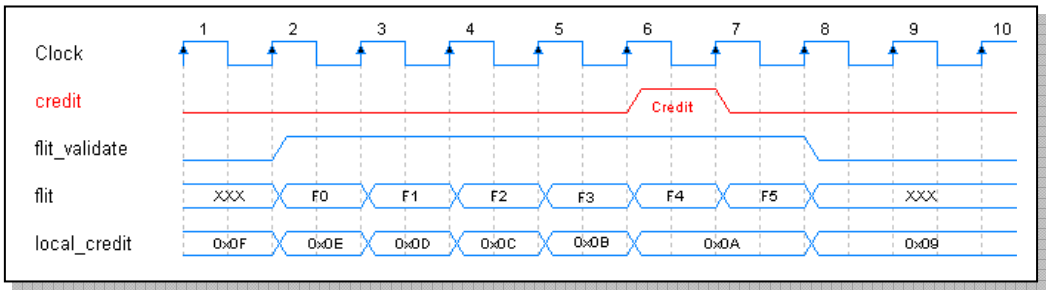
un NACK est notifié. Ce protocole est typiquement implémenté par le mécanisme du bit alterné «*Alternanting Bit Protocol*».



(a)



(b)



(c)

FIGURE 2.4 - Chronogrammes temporels pour les protocoles (a) ACK/NACK, (b) START/STOP, (c) à Crédits

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

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

Sous des conditions de trafiques équilibrés (injection/éjection), où les capacités des buffers ne sont jamais dépassées tous les protocoles sont fonctionnellement équivalents ; c'est-à-dire qu'ils permettent d'avoir exactement les mêmes débits de données. Cependant sous un régime de trafic plus soutenu des disparités apparaissent au niveau des performances; en effet sous de telles conditions de trafic les mécanismes de synchronisation sont activés plus souvent et induisent naturellement plus de pénalités sur le débit de données utiles. Afin d'illustrer le mode de fonctionnement de chaque protocole nous donnons la FIGURE 2.4 illustrant chacun des protocoles.

Le taux d'activité des protocoles de synchronisation joue aussi un rôle très important lorsque les délais des lignes de transport deviennent grand, par exemple en cas de ligne série ou de liaisons off-chip. Dans ces cas-ci le taux d'activité et d'autant plus grand que les délais sont élevés avec tous ce que cela implique en terme de diminution de débit et d'accroissement de la consommation dynamique.

2.3.4 Fonction de routage

Cette notion réfère à la fonction d'ordonnancement (□) au niveau de chaque nœud du réseau, qui permet l'acheminement d'un message de sa source jusqu'à sa destination. Une condition essentielle pour une fonction de routage consiste donc à assurer que n'importe quel nœud du réseau est accessible depuis n'importe quel autre nœud, cette condition est aussi appelée couverture totale [33]. À ce point il est important de noter que nous limiterons ici la notion de routage à l'ordonnancement spatial des messages sur le réseau. En effet il existe une forte corrélation entre la cohérence des mécanismes de routage (inter-blocage) et la stratégie de bufférisation ; cette forte corrélation amène certains travaux à considérer les deux concepts comme faisant partie d'un seul. Il existe plusieurs critères selon lesquels on peut classifier les algorithmes de routage [34], le plus haut niveau de classification significatif définit deux classes données ci-après :

Routage déterministe : Encore appelés statique, cette classe d'algorithmes regroupe les fonctions de routage qui associent une route unique reliant chaque paire (source, destination). Selon que la topologie du réseau est plus ou moins régulière, il est possible d'implémenter le routage déterministe soit par tables de routage, soit par une suite d'opérations arithmétiques sur les adresses source et destination de chaque message. Cette classe est caractérisée par sa grande simplicité d'implémentation, ce qui se traduit par un nombre de ressources matérielles très limité. Parmi les algorithmes les plus souvent utilisés on retrouve le routage appelé *«Dimension Ordered Routing »,* littéralement routage par ordre de dimensions. Cet algorithme se décline sous plusieurs variantes dont la plus usuelle est celle du X-puis-Y. Le principe est fort simple, au niveau de chaque nœud traversé l'aiguillage du message se fait sur la base de l'adresse destination et l'adresse locale, et tant que le paquet n'a pas atteint sa destination il est acheminé sur la dimension sur laquelle il reste un minimum de distance (par rapport à la destination). Comme déjà indiqué, l'avantage des algorithmes statiques réside dans leur simplicité d'implémentation, cependant le manque de flexibilité dans l'attribution des routes les rend tributaires d'une sensibilité accrue aux phénomènes de congestion.

Routage adaptatif : Dans un algorithme adaptatif, la route emprunté par un message n'est pas connue a priori, mais est déterminée au fur et à mesure de sa propagation dans le réseau. Au niveau de chaque nœud traversé la fonction de routage (non déterministe ici) sélectionne un port de sortie en fonction d'un certain nombre de critères. Ces critères sont principalement liés à la charge du réseau telle qu'elle est perçue localement par le nœud courant. L'idée principale derrière ce concept consiste à éviter/contourner les régions du réseau déjà fortement chargées. De prime abord la flexibilité du routage dynamique tend à améliorer les performances globales du réseau (charge acceptée et latence), cependant l'implémentation de tels algorithmes nécessite d'une part de plus importante ressources matérielles, d'autre part elle peut induire sous certaines scénarios de trafic des cas d'inter-blocage. Un inter-blocage peut être de deux types soit statique aussi appelé « *Dead-lock »,* soit dynamique « *Live-Lock* ». Les cas d'inter-blocages statiques se produisent lorsque deux messages sont en attente l'un de l'autre mutuellement à cause d'une dépendance cyclique dans les ressources qu'ils occupent ; il est cependant intéressant de noter ici que la stratégie de bufférisation joue aussi un rôle très important dans l'occurrence de tels événements. Par exemple la stratégie

« *Wormhole »* combiné à certaines fonctions de routage peut induire des inter-blocages [35] mais pas le *« Store & Forward »* avec les même fonctions. L'inter-blocage dynamique quand à lui se produit lorsqu'un message ne peut être acheminé jusqu'à sa destination mais est continuellement routé sur des chemins ne menant pas à la destination. Typiquement les causes responsables d'un *live-lock* sont : une fonction de routage donnant sous certaine conditions une route divergente, ou une mauvaise gestion des priorités entre messages pouvant conduire à un phénomène de famine.

Il existe à ce jour plusieurs travaux de recherche ayant trait aux mécanismes de routages et d'inter-blocage sur les réseaux sur puce [36] [37]; Néanmoins les topologies les plus utilisées étant régulières les routages déterministes (principalement un maillage 2D avec routage X-Y) sont utilisés de préférence comme pour le réseau Hermes [38], ou encore le réseau FAUST [39]. Les travaux sur les mécanismes de routage adaptatifs [40] montrent que le gain en performances reste généralement limité et ne peut justifier le surcout matériel induit par de tels mécanismes. Concernant le phénomène d'inter-blocage, il existe actuellement des approche formelles [41] permettant de déterminer si une combinaison Fonction de routage / Stratégie est interbloquée ou pas.

2.4 Caractérisation des performances pour les réseaux sur puce

Il est clair qu'un certain nombre de métriques sont nécessaires pour mesurer et quantifier les performances d'un réseau sur puce. Les performances étant fortement liés à l'architecture du réseau lui-même, il est primordial d'avoir une bonne compréhension de ces métriques et de la façon de les évaluer, afin d'être en mesure optimiser l'architecture du réseau. Dans ce qui suit, nous donnerons la définition de quelques métriques usuellement utilisée pour l'évaluation des réseaux sur puce :

2.4.1 Latence

La latence est l'un des critères les plus importants servant à la caractérisation des réseaux sur puce, elle correspond aux délais (généralement exprimés en nombre de cycles) nécessaires à l'acheminement d'un message depuis un nœud émetteur jusqu'au nœud

destinataire. Jusqu'à maintenant, nous nous sommes volontairement limités à la notion abstraite de message; cependant pour bien cerner le concept de latence, il est nécessaire de bien définir le format et le mode de transport des messages. La commutation de paquets étant la plus souvent utilisée sur les réseaux-sur-puce, nous ferons l'hypothèse par la suite qu'un message correspond à un paquet qui lui-même est formé de plusieurs flits (*Flow Control Unit*). Individuellement pour un seul paquet, la latence est le nombre de cycles écoulés depuis la première tentative d'injection du premier flit (flit d'entête) jusqu'à la réception du dernier flit par la destination. La latence peut être calculée à vide pour chaque paire (source, destination), dans ce cas là elle ne dépend que la longueur du message, des délais de routage et de la distance entre la source et la destination. Dans un cas plus général il peut être intéressant d'estimer la latence moyenne du réseau, la latence est exprimée ici comme fonction de la charge injectée à l'entrée du réseau. En plus des caractéristiques statiques du réseau, la latence moyenne tient compte du comportement dynamique sous régime conflictuel pour lesquels plusieurs flots de données se trouvent dans l'obligation de partager les mêmes ressources (buffers, port de sortie). Dans la FIGURE 2.5 nous donnons le détail des principales interactions internes du réseau influant sur la latence de transfert, il est à noter ici que les délais de bufférisation dépendent de la technologie d'implémentation et sont indépendants de leurs tailles respectives (FIGURE 2.6).

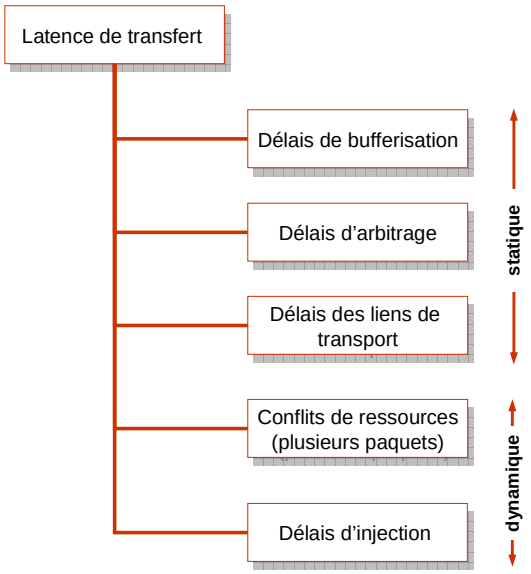


FIGURE 2.5 - Composantes statique & dynamique de la latence de transfert

2.4.2 Débit de données

Le débit de données reflète les capacités d'écoulement de trafic du réseau en terme de quantité de données par unité de temps. Bien que plus ou moins liée à la notion de latence de données, il est important de bien distinguer les deux métriques. Tout comme pour la latence, le débit sur un réseau sur puce comporte deux composantes, l'une statique et l'autre dynamique. Quand au débit statique (Octet/seconde) il correspond à la quantité de données maximale pouvant circuler entre deux nœuds par unité de temps, et dépend principalement des caractéristiques physiques des liens (fréquence de fonctionnement, largeur de données, schéma de sérialisation) mais prend en compte aussi les temps de routage et de bufférisation. La composante dynamique du débit de signe négatif résulte des fluctuations des délais de transport induites par les conflits de ressources potentiels sur les chemins de transport. Ces fluctuations se traduisent par une variation de la charge pouvant être injectée au réseau. Dès lors que le débit comporte une composante dynamique fortement dépendante de l'état du réseau, une problématique intéressante consiste à définir une méthodologie de mesure de débit tenant compte justement de ces variations. Une approche très simple consisterait à échantillonner la mesure de débit sur plusieurs intervalles de temps, donnant, plutôt qu'une valeur moyenne, une courbe caractéristique d'évolution du débit plus facilement exploitable. Cependant l'approche la plus couramment utilisée prenant en compte la composante dynamique consiste à exprimer le débit en termes de rapport entre le taux d'injection totale et le taux de charge effectivement acceptée (FIGURE 2.7).

2.4.3 Point de saturation

Comme montré sur la FIGURE 2.6, le réseau-sur-puce (Grille 2D 4x4 avec stratégie Wormhole) est caractérisé par deux zones de fonctionnement. Sur la première zone que nous appellerons zone de linéarité, les réactions du réseau en terme de latence de transfert et charge acceptée, sont assez prédictibles ou du moins facilement cernables en fonction de la charge de trafic à l'entrée. Par contre, une fois une certaine charge seuil dépassée commence la zone de non-linéarité (régime congestionné) où les performances du réseau commencent à se dégrader

très rapidement. La charge critique qui marque la frontière entre les deux modes de fonctionnement (linéaire et congestionné), est appelée « point de saturation ».

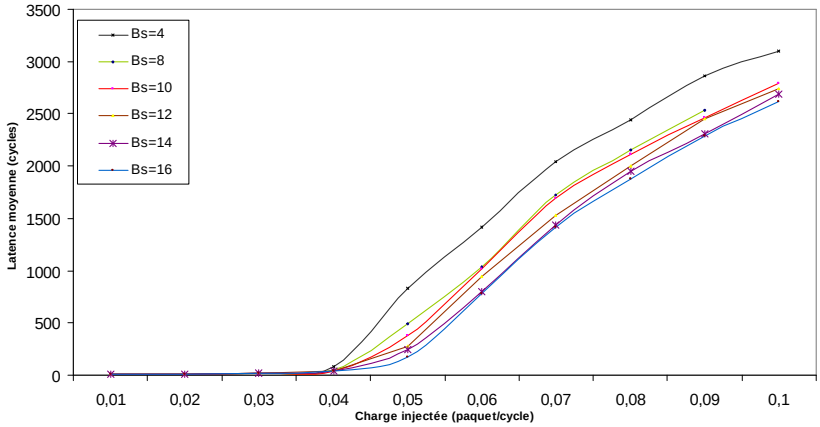


FIGURE 2.6 - Illustration de la non corrélation entre la taille des buffers internes du réseau sur puce et la latence moyenne

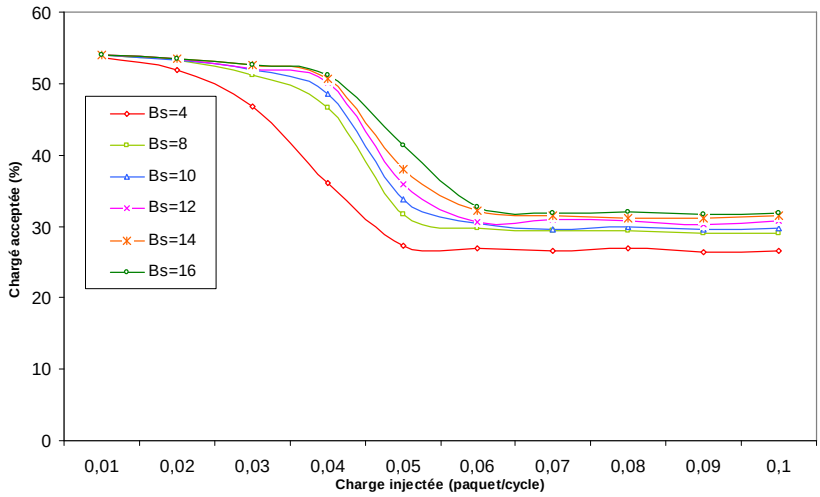


FIGURE 2.7 - Représentation des débits effectifs en termes de charges acceptées

La connaissance du point de saturation peut être très utile pour déterminer si l'utilisation d'un réseau donné est appropriée pour telle ou telle application; cependant une limitation importante réside dans le fait que le point de saturation dépend grandement du schéma de communication (distribution spatiale des communications) ce qui exclu le recours à une seule valeur critique. Cela nécessite plutôt de connaître une valeur pour chaque schéma de communication. En plus du schéma de communication, la valeur seuil dépend aussi de la longueur des paquets, de la distribution de l'espace de bufférisation, du taux d'éjection

(consommation) des paquets, de la taille du réseau et aussi de la diversité des chemins sur le réseau qui est fixée par sa topologie.

Jusqu'à présent, la majorité des travaux de recherche s'étant intéressés aux performances des réseaux-sur-puce tendent à affirmer qu'une fois le point de saturation franchi la latence croie de façon exponentielle. Cette affirmation n'est pas tout à fait vraie, en effet nous avons remarqué qu'après la phase d'accroissement très rapide caractéristique du franchissement du point de saturation la latence tend à se stabiliser. Ce phénomène s'explique par l'état d'équilibre qu'atteint le réseau en phase de forte congestion. En faisant abstraction de tous les mécanismes internes du réseau, et sans perde en généralité on peut considérer que son comportement en régime saturé se rapproche de celui d'une seule file de messages à capacité de traitement limité où toutes les injections de paquets ont lieu au même endroit. Il devient alors très aisé de démontrer que quelque soit le taux d'injection et de consommation des paquets, et quelque soit les délais de services nécessaires pour chaque paquet, le système atteindra un état de pseudo-équilibre ou la latence sera relativement stable. Afin de bien illustrer ce phénomène nous donnons la FIGURE 2.8 qui montre l'évolution de la latence pour une seule file de message en fonction du taux d'injection (régime saturé), le taux de consommation étant constant. Cette remarque, importante théoriquement, n'a cependant, qu'un intérêt pratique limité vu la latence moyenne atteinte lors de la stabilisation.

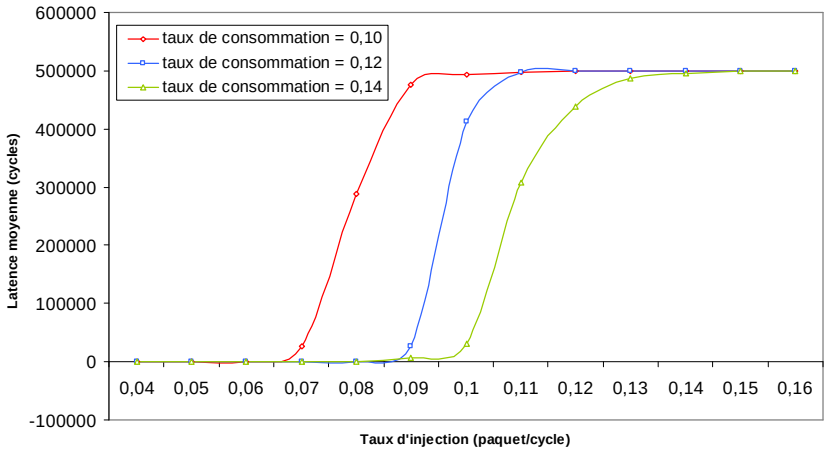


FIGURE 2.8 - Etat d'équilibre de la latence après la phase d'accroissement rapide

2.4.4 Ressources et surface silicium

Comme pour tout système destiné à une implémentation matérielle, la surface silicium tient une importance majeure pour les réseaux-sur-puce et cela quelle que soit la technologie ciblée, ASIC ou FPGA. La surface totale (ou ressources si technologie FPGA) d'un réseau est la résultante des surfaces de deux éléments : les routeurs et les interconnexions entre routeurs. Les routeurs sont eux même composés d'une logique de contrôle (crossbar, contrôle de flux, routage) et d'un certain nombre de buffers (FIFOs), ces derniers tiennent pour la plus grosse part dans la surface totale du routeur.

La surface induite par les interconnexions dépend de la topologie du réseau lui-même [42] [43], de la largeur des interconnexions (nombre bits) mais aussi grandement de la technologie de placement routage utilisée. Notons ici que la surface du réseau est aussi grandement influée par la régularité des IPs de l'architecture matérielle du SoC (processeurs, mémoires, accélérateurs); en effet une estimation précise de la surface du réseau sur puce n'est possible qu'après l'intégration des différentes IPs.

2.4.5 Profil énergétique

Une autre métrique non moins importante que la surface à trait à la consommation électrique sur le réseau, nous l'appellerons ici profil énergétique. Tout d'abord il est nécessaire de rappeler que pour n'importe quel circuit électronique la consommation électrique totale est due d'une part aux courants de fuites correspondant à la part statique de la consommation, et d'autre part à l'activité électrique (transitions) sur le circuit qui elle correspond à la part dynamique de la consommation. Pour un réseau-sur-puce, l'énergie statique dépend grandement de la finesse de la technologie de gravure (90 nm, 45 nm…) et est relative au nombre de ressources du réseau (portes, bascules, cellules mémoires). Une optimisation de la consommation statique d'un réseau-sur-puce passe donc par une optimisation des ressources du réseau : optimisation des tailles des buffers, optimisation de la microarchitecture des routeurs…etc (optimisation niveau système [44]). La proportion de la consommation statique par rapport à l'énergie totale varie entre 6% et 20% et peut même atteindre 30 % [45] pour les technologies de gravure les plus récentes. En ce qui concerne la

consommation dynamique, elle est induite par les transitions sur les interconnexions (+V_{dd} vers *0v* et inversement). Dans le cas d'un réseau-sur-puce complètement synchrone l'arbre de distribution de l'horloge est le principal organe responsable de la consommation dynamique (jusqu'à 50% [46]), viennent en suite les interconnexions entre routeurs et la logique interne de ces derniers.

Un certains nombre d'outils permettant la mesure de la consommation existent, et peuvent être exploités avec les réseaux-sur-puce. Les outils commerciaux pour la plupart permettent de faire des estimations de bas-niveau, i.e. au niveau netlist de porte logique ou même au niveau transistors (Ex : *Synopsys PrimePower* [47]). Ces outils effectuent des simulations électriques en utilisant des bibliothèques de cellules logiques précaractérisées. D'autres outils moins précis, mais moins contraignants du point de vue de l'utilisation, se servent d'un modèle analytique simplifié adapté aux réseaux-sur-puce [48]. Ce modèle tient compte uniquement de la consommation dynamique et est donné comme suit :

$$E_{hope} = E_{switch} + E_{\text{int}erconnexions} \quad (1)$$

$$E_{switch} = \alpha_{switch} \quad C_{switch} \quad V^2 \quad (2)$$

$$E_{\text{int}erconnexions} = \alpha_{nterconnexions} \quad C_{nterconnexions} \quad V^2 \quad (3)$$

$$E_{paquet} = n \quad \sum_{j=1}^{h} E_{hop}(j) \quad (4)$$

Les paramètres α et C correspondent respectivement au taux d'activité et la caractéristique capacitive des routeurs et des liens. L'énergie totale nécessaire pour un paquet traversant le réseau sur *n* sauts, est donnée par la formule (4). Il est clair que ce modèle ne peut concurrencer les approches par simulation en termes de précision, néanmoins il a le mérite de permettre des estimations très rapides.

2.5 Les interfaces réseaux

Les interfaces réseau (NI) constituent les points d'accès au réseau-sur-puce, elles fournissent aux IPs qui lui sont connectées un certains nombre de services pour leur assurer des communications bidirectionnelles. Les NIs assurent en premier lieu une abstraction

complète des mécanismes internes du réseau (topologie, routage, contrôle de flux), ainsi le réseau est complètement transparent du point de vue des IPs. Une autre fonctionnalité des NIs consiste à adapter les protocoles de communications physiques propres à chaque IP à ceux du réseau. En dernier lieu les NIs prennent en charge la paquétisation et la dépaquétisation des données vers/depuis le réseau, étape nécessaire pour toute transaction (FIGURE 2.9).

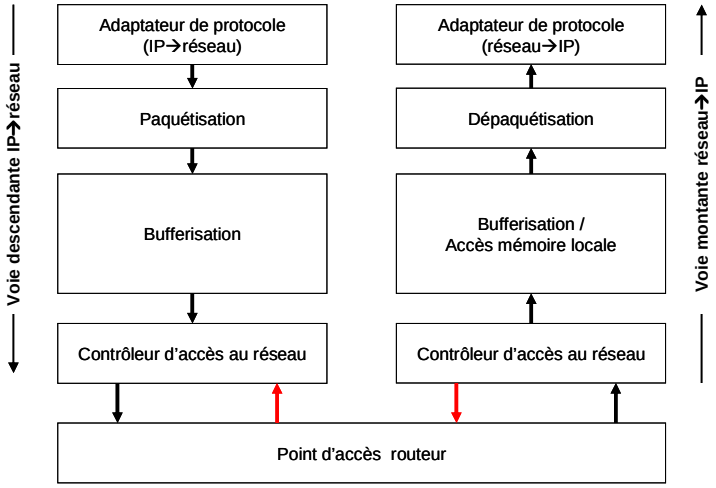


FIGURE 2.9 - Architecture de l'interface réseau

2.6 Paradigmes multi-synchrones et asynchrones pour les réseaux-sur-puce

Le remise en cause de l'approche du *« tout synchrone »* prédominante jusqu'à maintenant trouve sa justification dans plusieurs états de fait découlant tous des nouvelles avancées technologiques mais aussi méthodologiques. La conjonction de deux phénomènes est à l'origine de ce changement de concept ; le premier étant l'émergence de technologies de gravure de plus en plus fines (45 nm, 32nm) avec des contraintes physiques donnant une part plus importante aux interconnexions dans les délais de propagation ainsi que dans la dissipation d'énergie. Le deuxième phénomène réside dans l'augmentation des fréquences de fonctionnement des systèmes-sur-puce ; augmentation motivée par des besoins croissants en performances. Il devient alors très difficile de distribuer un signal d'horloge unique avec un minimum de gigue sur toute la surface de la puce. Les paradigmes multi-synchrones et

asynchrone [49] ont fait l'objet de plusieurs études, et ont été implémentés dans plusieurs circuits : dont des processeurs [50]. Ils ont aussi tôt été adoptés pour les réseaux-sur-puce[51] [52], étant donné que ces derniers occupent généralement la totalité de la surface de la puce. Dans une approche multi-synchrone plusieurs domaines d'horloges coexistent sur le même réseau, alors que dans ce cas-ci l'architecture interne du réseau sur puce est identique à quelques changements prés au cas synchrone; une seule problématique réside dans le passage des signaux de données et de contrôle entre des régions d'horloge différentes. En effet, les différences de phase et de fréquences peuvent conduire à des cas de figures ou un signal synchrone à une horloge H_1 est échantillonné dans le domaine d'horloge H_2 alors qu'il est en transition (fenêtre T_{setup} T_{hold}). L'échec de synchronisation peut conduire à un état de métastabilité. Dans cet état la valeur du signal échantillonné peut prendre une valeur indéterminée entre les états logiques '0' et '1' ou même osciller entre les deux états. Le retour à un état stable dépend de plusieurs conditions dont le bruit électromagnétique ambiant, la température…etc. (FIGURE 2.10). Il est à noter que la probabilité qu'un signal passant d'un domaine d'horloge à un autre, entre dans un état métastable n'est jamais nulle ; et cela quelque soit l'approche utilisée. Cependant les solutions actuellement disponibles rendent une telle probabilité infime. Les FIFOs bi-synchrones permettant des lectures et des écritures avec des fréquences différentes sont couramment utilisées dans les réseaux-sur-puce comme interfaces entre domaines d'horloge.

Comme vu précédemment, l'approche multi-synchrone résout efficacement le problème de distribution d'horloge, et permet ainsi d'atteindre des fréquences de fonctionnement élevées comparées à celle d'un système qui serait complètement synchrone. L'approche asynchrone quand à elle se base sur des circuits ne nécessitant aucun signal d'horloge, ceci résout aussi efficacement les problèmes liés à la distribution d'horloge sur de larges surfaces (délais de propagation, gigue, chemins critiques), mais en plus améliore grandement le profile énergétique global, aucun réseau de distribution d'horloge n'étant alors nécessaire. Cependant la conception de circuits asynchrones amène son lot de problèmes dont le principale réside dans le manque d'outils de conception et de validation matures. Ceci rend le développement de tels circuits très fastidieux. A cela il faut ajouter un surcout non négligeable en surface totale induit par les protocoles spécifiques nécessaires pour l'échange et l'acquittement des données.

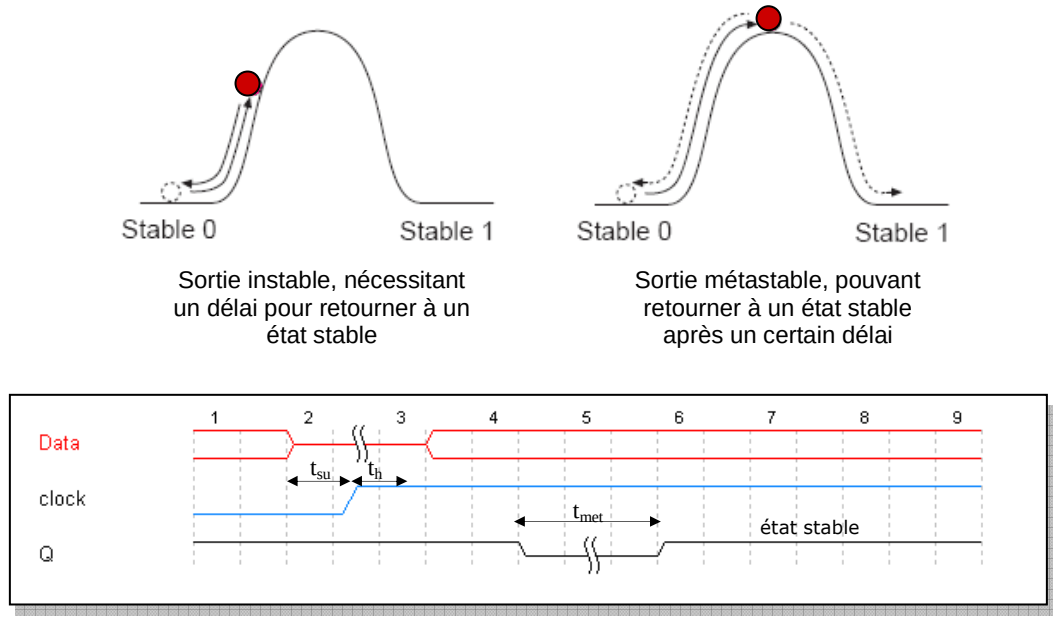


FIGURE 2.10 - Métastabilité causée par une violation T_{setup} T_{hold}

2.7 Etat de l'art sur les réseaux sur puce existants

Il existe actuellement un nombre important d'implémentations de réseaux-sur-puce aussi bien académiques destinés à la recherche [53] [54], qu'industriels à vocation commerciale. Au-delà des détails d'implémentation de chaque réseau [55], il est tout aussi important de s'intéresser aux outils et méthodologies de conception qui y sont associés. En effet, la réutilisation et l'intégration efficace d'un réseau dans un SoC existant dépend en grande partie des outils d'aide à la conception et d'exploration qui y sont associés [56]. Dans ce qui suit nous présenterons quelques réseaux-sur-puce, choisis afin de bien refléter les tendances architecturales et les pratiques de conception dans ce domaine.

2.7.1 ARTERIS

Plutôt qu'une implémentation de réseau-sur-puce, c'est une suite d'outils développés par la compagnie ARTERIS [57]. Nous avons choisi d'exposer cet outil car c'est le premier

outil de conception de réseaux-sur-puce à avoir été commercialisé, bien que cette vocation purement commercial rende l'obtention de documentation technique quasiment impossible.

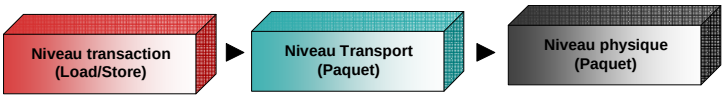


FIGURE 2.11 - Les trois couches d'abstraction du réseau Arteris

Cette suite comprend deux outils d'aide à la conception et à l'exploration architecturale «*NoCexplorer* » et « *NoCcompiler* ». Le premier outil va permettre la construction d'une topologie de réseau à partir d'un cahier des charges, où sont spécifiés les besoins en performances de l'application visée ainsi que les limites de cout et de surface pouvant être admis. Cet outil est basé sur un modèle de simulation SystemC du routeur ARTERIS reprenant les mécanismes de routage, de bufférisation ainsi que les mécanismes de contrôle de flux et de qualité de service. Bien que le niveau d'abstraction du modèle de simulation ne soit pas explicitement indiqué, la vitesse de simulations et la précision des résultats annoncées laissent supposer un niveau se rapprochant du TLM Cycle Accurate. Les rapports de performances incluant les débits aux entrées/sorties ainsi que la latence sur le réseau sont produit à chaque simulation à partir de modèles de trafic synthétiques. Le concepteur a la possibilité de modifier ensuite manuellement la topologie donnée en entrée en fonction des performances obtenues afin d'arriver à la meilleures adéquation (besoins/couts) [58]. L'intérêt majeur de ce genre d'outils est de permettre la modélisation et la simulation de plusieurs instances de réseaux-sur-puce de façon visuelle et très rapide, ce qui se traduit par un gain substantiel en temps de conception. Il est à noter ici que la liberté donnée à l'utilisateur de choisir une topologie arbitraire écarte de fait la possibilité d'utiliser des algorithmes de routage algébriques et impose l'utilisation de tables de routages au niveau de chaque routeur. Une fois la topologie fixée *« NoCcompiler »* va permettre de générer une description matérielle du réseau. Il est aussi possible d'interfacer directement des composants de la librairie d'IP fournie appelée « *Danube IP Library*» : interfaces de bus (AHB, AXI, OCP), modèles mémoires…etc. Le code HDL généré (VHDL ou Verilog) est directement exploitable dans des outils d'EDA tierces au travers de scripts de synthèse générés eux aussi automatiquement.

Comme nous l'avons indiqué plus-haut les mécanismes internes de fonctionnement du réseau relèvent du secret industriel, cependant sur [59] on peut lire que le réseau peut fonctionner aussi bien en mode complètement synchrone qu'on mode GALS «*Globalement Asynchrone et Localement Synchrone*» avec des modifications des mécanismes de bufférisation. Il y est aussi indiqué que les mécanismes de qualité de service sont sans garanties et limités seulement à un mécanisme de priorité géré au niveau des routeurs.

2.7.2 STNOC

Le réseau STNoC [28] est un réseau à commutation de paquets propriétaire de la société STMicroelectronics. L'originalité du STNoC réside dans sa topologie en Spidergon (FIGURE 2.12 (a)), topologie régulière supportant des algorithmes de routage minimalistes très simples nécessitant un minimum de ressources matérielles. Le point fort de la topologie Spidergon découle de l'encombrement minimal des interconnexions qu'elle engendre ; en effet quelle que soit la taille du réseau STNoC il est toujours possible de le transformer en une implémentation planaire avec un minimum de croisements entre les liens. Des interconnexions fortement encombrées avec beaucoup de croisements nécessitent plus de surface et sont à l'origine d'une surconsommation électrique. Une autre caractéristique de la topologie est qu'elle peut être adaptée aux besoins de l'application ciblée en retirant des liens inutilisés sans pour autant modifier les algorithmes de routage, ce qui ajoute un degré de flexibilité non négligeable.

Le STNoC utilise une approche se basant sur une séparation stricte entre la logique interne des routeurs et les liens servant au transport des paquets. En plus du support implicite du mode *GALS*, cette approche donne la possibilité d'avoir plusieurs configurations de liens en fonction des besoins en performances. En effet, selon les contraintes physiques, il est possible d'utiliser des liens séries ou parallèles, il est aussi possible de choisir la fréquence de sérialisation/fonctionnent en fonction des débits souhaités indépendamment de la fréquence utilisée par la logique interne des routeurs.

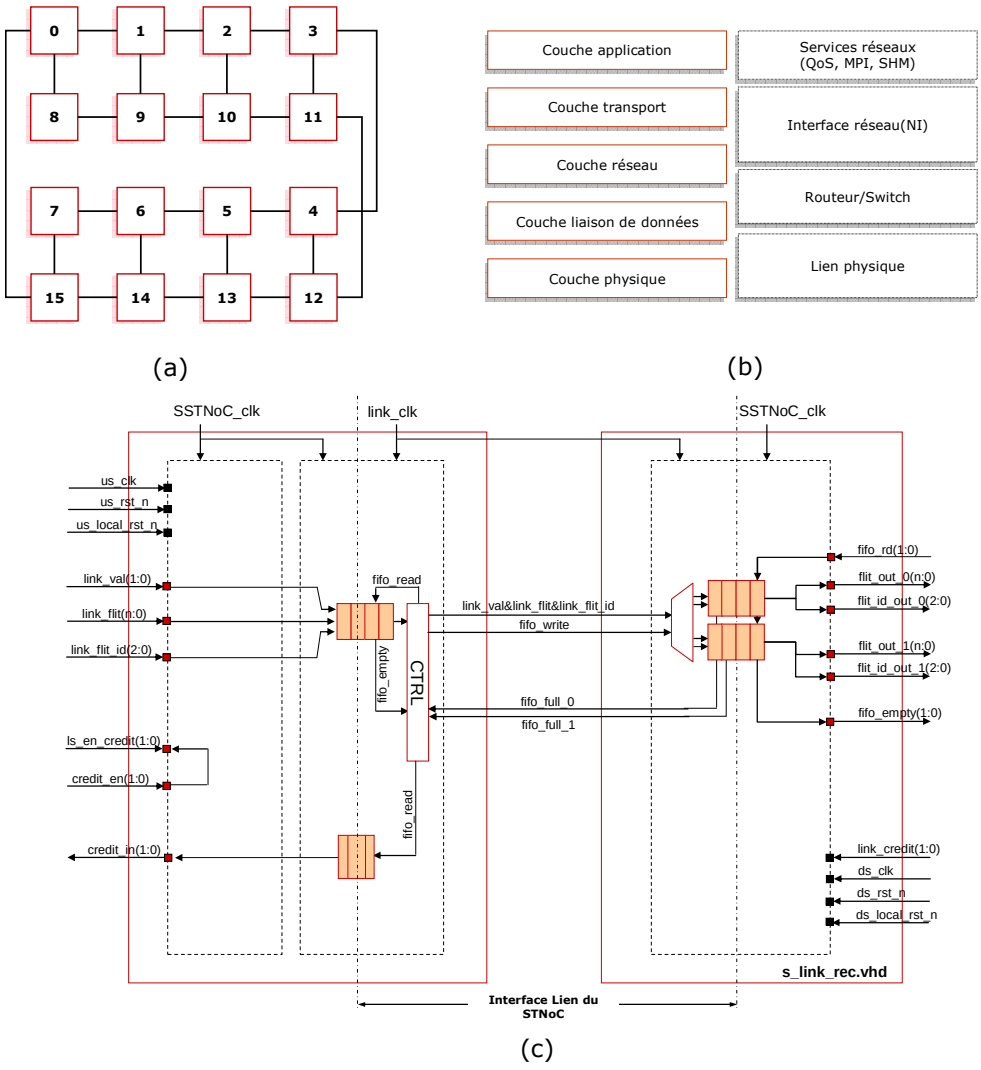


FIGURE 2.12 - (a) La topologie Spidergon du STNoC (b) Couches réseau du STNoC (c) Interface Lien du STNoC

2.7.3 HERMES

Hermes [38] est un réseau avec stratégie de bufférisation Wormhole et un routage X-Y sur une grille 2D dont la taille peut être fixée en fonction des préférences utilisateur. Divers paramètres peuvent être modifiés lors de la phase de conception, tels que la largeur des flits, la longueur des paquets, ou encore la profondeur des buffers internes. Le réseau est conçu pour supporter uniquement le mode de fonctionnement synchrone. C'est d'ailleurs l'une de ses principales limitations. La particularité de ce réseau est qu'il est en accès libre, il est ainsi

possible d'ajouter et/ou modifier certaines fonctionnalité directement sur les sources HDL du réseau.

Une interface très intuitive est fournie, et permet de générer une instance spécifique du réseau en fixant chaque paramètre. Il est aussi possible à travers la même interface de tester l'instance du réseau sous plusieurs scénarios de trafic (Uniforme, Pareto ON/OFF, Poisson) au moyen de générateurs écrit en SystemC. Cependant comme aucune librairie d'IPs n'est fournie avec le réseau, particulièrement des interfaces aux bus standard ; il est nécessaire de développer une interface réseau pour chaque protocole utilisé.

Une dernière caractéristique importante du réseau Hermes est qu'il est compatible avec une implémentation matérielle (ex : FPGA); en effet le code VHDL généré est complètement synthétisable. Toutefois l'adjonction de moniteurs de performances matérielles (non inclus dans Hermes) est nécessaire afin de permettre une étude des performances sous conditions de trafics réels.

2.7.4 Ætheral

[60] [61]Est un réseau développé au sein du département R&D de la société Philips (maintenant NxP) dont le but principal est d'offrir une garantie de qualité de service. Ainsi le réseau défini deux classes de qualité de service : la classe GS *"Guaranteed Service"* et BE *"Best Effort"*. Les mécanismes de garantie de services sont implémentés au niveau des routeurs et se basent sur une technique d'allocation statique des ressources (TDMA) du réseau et cela pour chaque flux de données. L'allocation se fait par des tables appelées « *Slot Tables* » où chaque ligne correspond à un intervalle temporel (ou slot), et chaque colonne correspond à un port de sortie du routeur. La valeur à l'intersection ligne/colonne indique le numéro de port d'entrée pouvant émettre durant un slot donné. Cette approche évite de fait toute contention mais au prix d'une latence moyenne pouvant être plus élevée à celle de la classe de service BE. Initialement et avant l'allocation des slots au flux de données, le réseau fonctionne en mode BE uniquement. Trois paquets de signalisation servent à l'allocation des ressources :

– Le paquet « *SetUp* » : ce paquet ouvre le chemin depuis la source jusqu'à la destination, le chemin parcouru par ce paquet est le même que celui qui sera utilisé pour le

flux de données. Il contient l'adresse source, l'adresse destination ainsi que la succession de tous les nœuds qui seront traversés. Au niveau de chaque routeur traversé si le port de sortie correspondant est libre durant le slot demandé une réservation est faite, et le paquet *SetUp* est transmis aux routeurs suivants en incrémentant le numéro de slot demandé.

– Le paquet « *TearDown* » : Si le paquet *SetUp* ne parvient pas à atteindre la destination à cause d'un échec de réservation, il est retiré du réseau et un paquet *TearDown* est renvoyé à la source sur le chemin inverse et cela afin de signaler l'échec de l'allocation des ressources. Cela implique bien sûre que tous les chemins de la topologie du réseau soient bidirectionnels c'est d'ailleurs la seule contraintes de topologie. Ce paquet libère sur le chemin retour les slot préalablement réservés.

– Le paquet « *AckSetUp* » : Ce paquet signale le succès de l'établissement d'une connexion source/destination, il suit aussi le chemin inverse du paquet *SetUp*.

Il est important de noter ici qu'étant donné que pour la classe GS on connaît à chaque slot de temps quel port d'entrée communique avec quel autre port de sortie, il n'y a plus besoin d'informations de routage, et celles-ci sont fixées par la source au tout début de l'établissement des chemins. Cela a pour effet d'augmenter le débit de données effectif.

TAB 2.2 Comparaison entre quelques réseaux sur puce

Réseau	**Topologie**	**Routage**	**Support de la QoS**	**Performances maximales Gb/s**	**GALS**
DSPIN[62]	Grille 2D	Déterministe/ Adaptatif	Oui	2	Oui
STNoC[28]	Octogone	Déterministe	Oui	-	Oui
QNoC[63]	Grille non symétrique	Déterministe	Oui	0.601	Oui
Arteris[57]	-	Déterministe	Non	-	Oui
Ætheral[61]	-	Déterministe	Oui	4	Non
FAUST[39]	Grille	Déterministe	Oui	-	Non
ANoC[51]	Grille	Déterministe	Oui	5	Oui
XPipes[64]	Grille	Déterministe	Non	2.2	Non
µSpider[65]	-	Déterministe	Oui	-	Oui

2.8 Conclusion et synthèse sur les réseaux sur puce

Dans ce chapitre nous avons exposé dans un premier temps les principales raisons qui mettent les réseaux-sur puces (NoCs) sur le premier plan des schémas d'interconnexions pour SoCs. Les besoins en performances ainsi que les contraintes liées aux nouvelles technologies d'intégration en justifient l'intérêt. Ainsi les bus largement utilisés jusqu'à maintenant montrent très rapidement leurs limites avec l'accroissement des besoins en bande passante. Cela est dû principalement au caractère séquentiel des communications sur de tels supports. Les crossbars quand à eux supportent très bien des charges élevées de trafics mais leur implémentation matérielle devient quasiment impossible au-delà d'une certaine taille. Les réseaux-sur-puce avec leur support intrinsèque du parallélisme ainsi que la régularité de leurs architectures constituent donc l'évolution naturelle des infrastructures de communications sur puce.

Nous avons ensuite détaillé les mécanismes internes des réseaux-sur-puce, cette étude avait un double objectif ; le premier consistait à donner un aperçu du large éventail de possibilités d'implémentations pouvant exister pour une même fonctionnalité. Le second était de faire ressortir la difficulté d'optimisation de telles architectures. En effet la taille de l'espace d'exploration et par conséquent la complexité des algorithmes de recherche sont directement liés au nombre de paramètres devant être pris en compte. D'où la nécessité d'avoir des outils d'aide à la conception pour les NoCs. Ces outils, en facilitant la mise en œuvre et l'implémentation d'une instance d'un réseau, accélèrent le processus de recherche et d'optimisation. Etant donné que pour une exploration efficace il est tout aussi important de pouvoir quantifier les performances d'un réseau, nous avons fait une étude exhaustive des métriques usuellement employées.

En fin, nous avons présenté quelques implémentations de NoCs qui nous semblaient les plus significatives. Il est important de souligner qu'il existe actuellement un grand nombre d'implémentations de NoC [66] avec des fonctionnalités de plus en plus avancées telle que l'évitement de la congestion [67], des mécanismes de garantie de qualité de service ou encore des fonctionnalités de transport temps réel. Cela monter bien l'engouement actuel que suscitent les NoCs dans la communauté de la recherche en micro-électronique.

Chapitre 3 :
Flot optimisé pour le prototypage FPGA des réseaux intégrés

Sommaire

3.1. Introduction

Comme nous l'avons vu dans le chapitre précédent, les NoCs sont des systèmes extrêmement complexes, dont le comportement global dépend fortement à la fois des choix architecturaux et des conditions de trafic qu'ils subissent. Dès lors le recours à des flots de conceptions usuelles, mais intégrant cette dimension, devient nécessaire. Par définition le recours à un flot de conception a pour objectifs principaux : 1) de réduire la durée des cycles de conception et de validation, 2) De permettre au concepteur d'arriver à la meilleure solution cout/performances et 3) de réduire au minimum la fréquence d'apparition et les conséquences des bugs.

Dans ce chapitre nous nous intéresserons à l'intégration dans un processus de conception d'une phase de prototypage de réseaux-sur-puce, le prototypage étant basé sur une plateforme reconfigurable. Nous aborderons dans un premier temps les modèles de spécifications couramment utilisés lors de la conception qui sont respectivement : Un modèle de spécifications des besoins, un modèle de spécifications des contraintes et un modèle de spécifications fonctionnelles. Bien que le modèle de spécification ne soit qu'un conteneur pour les fonctionnalités, besoins en performances et les contraintes sur le NoC, nous avons jugé utile d'aborder cet aspect. Le but étant de bien souligner son importance car il est le point d'entrée de tout flot de conception. Ce modèle de spécifications sera aussi utilisé par la suite lors de la partie consacrée au processus de prototypage optimisé.

La validation est une étape clé de tous processus de conception de NoC. Elle a pour principal objectif de vérifier que le réseau-sur-puce répond bien aux exigences de performances et aux contraintes, autrement dit qu'il est bien conforme au modèle de spécification (fonctionnelles/besoins/contraintes). Il existe trois approches plus ou moins adaptées aux NoCs. La simulation logicielle, qui constitue la solution la moins coûteuse et la plus flexible (mais ne passe pas à l'échelle), l'émulation matérielle qui nécessite des équipements industriels hautement spécialisés, et finalement le prototypage sur plateformes reconfigurables. La seconde partie de ce chapitre sera donc consacrée à l'introduction des plateformes reconfigurables et à la technologie FPGA. Nous essayerons de démontrer dans cette partie du chapitre l'intérêt de ces plateformes pour la validation de NoCs, nous exposerons aussi les limitations de cette approche.

3.2. Flot générique de conception de réseaux intégrés

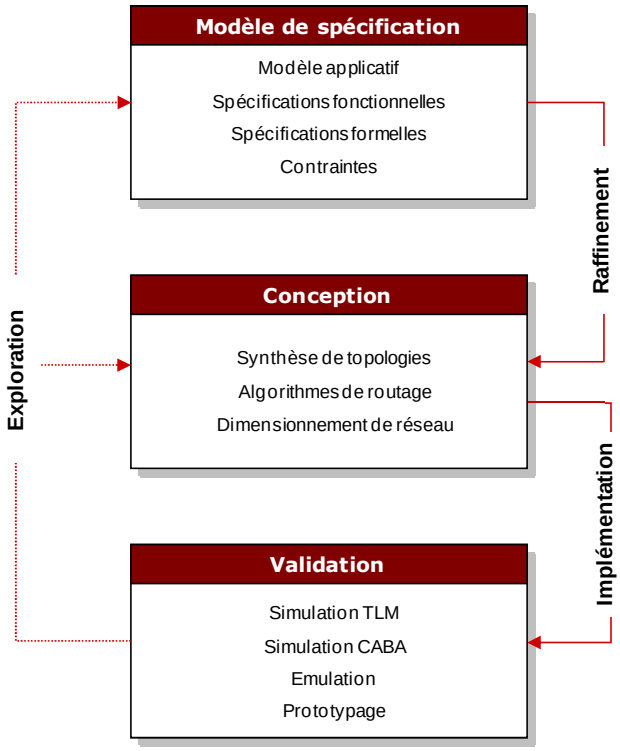


FIGURE 3.1 - Les trois étapes d'un flot de conception de NoC

Une proposition d'un flot générique descendant dédié au NoC est illustrée par la FIGURE 3.1. Ce flot de réalisation est assez semblable dans sa structure générale avec les flots standards de réalisation de SoCs, le but étant d'arriver à une implémentation finale fonctionnelle avec le niveau de performances souhaitées.

Lors de l'étape de spécification, trois éléments importants doivent être clairement identifiés de la part du concepteur. Ces éléments sont respectivement : les fonctionnalités du réseau, le niveau de performance et les contraintes. Les fonctionnalités du NoC représentent toutes les macro-caractéristiques pouvant être incluses dans l'implémentation finale : typiquement le support de la qualité de service, le contrôle de flux, la gestion des contentions…etc. Généralement le modèle de spécifications fonctionnel reflète un modèle de NoC fonctionnel existant; car il est très difficile de concevoir des flots générant automatiquement des NoCs avec une combinaison de fonctionnalités quelconque. Vient

ensuite l'étape de conception qui a pour but de traduire le modèle de spécification en une implémentation de NoC conforme au modèle de spécification fonctionnel. A ce point il n'est pas possible de vérifier l'adéquation avec le modèle des performances ou de contraintes. Cela n'est possible que durant l'étape de validation en utilisant la technique appropriée.

3.3. Modèles de spécification des réseaux intégrés

Dans cette partie nous nous intéresserons aux trois modèles de spécification introduits dans le paragraphe précédent :

3.3.1 Modèle de spécification des besoins

Généralement, dans un système sur puce, tous les besoins en performance, quels qu'ils soient, sont dictés par le logiciel embarqué et les accélérateurs matériels nécessaires à l'implémentation de la spécification applicative. Ainsi les besoins en performances sur un NoC ne reflètent en fait que des besoins en communication du niveau applicatif [68]. Une approche très employée consiste donc à partir d'une description de haut niveau d'une application parallèle, généralement sous forme d'un graphe de tâches (FIGURE 3.2), et en déduire les paramètres de performances requis. À partir du graphe des tâches, il est possible d'extraire le nombre de nœuds du réseau ainsi les bandes passantes nécessaires entre chaque nœud du NoC. Il est aussi possible pour certaines applications avec des échéances temporelles (deadlines) de fixer des limites sur les latences maximales et par conséquent de déterminer les mécanismes de qualité de service. Le graphe de tâches annoté s'avère être très utile pour la construction des spécifications du NoC. Cependant il n'est pas toujours facile d'obtenir un graphe avec toutes les informations nécessaires partant d'une application parallèle. De plus, cela fait des hypothèses pas toujours réalistes sur le placement des tâches.

3.3.2 Modèle de spécification des contraintes

Les contraintes sur un NoC sont quasiment identiques à celles qu'on peut rencontrer sur un système SoC ou MPSoC. À savoir des contraintes de performances temporelles, des

contraintes de surface silicium et des contraintes de consommation d'énergie. Bien que les besoins et les contraintes correspondent à des métriques de performances, la distinction que nous faisons ici entre les deux vient du fait qu'une contrainte doit être satisfaite au plus près ce qui est requis, alors qu'un besoin doit toujours être maximisé ou minimisé. En fonction du contexte et des choix concepteur, il est toujours possible de basculer un besoin en contrainte et inversement.

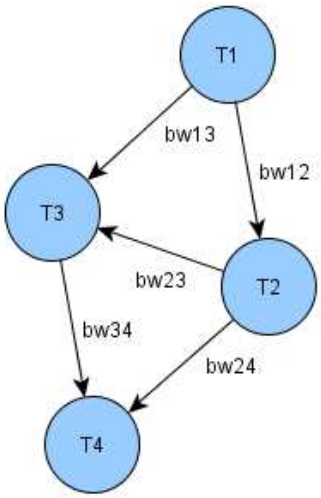


FIGURE 3.2 - Représentation en graph de tâches du modèle applicatif

3.3.3 Modèle de spécification fonctionnel

C'est un modèle abstrait qui reprend toutes les fonctionnalités présentes/souhaitées dans le NoCs, il sert de modèle de référence pour la phase de conception pour les processus à raffinements successifs.

3.3.4 Modèle de spécification formel

L'objectif de ce modèle est double, le premier consiste en la mise en place d'un certain nombre de règles sur le NoC afin de garantir un fonctionnement cohérent des modèles générés dans la phase de conception. Le second objectif est de faciliter la phase d'exploration, en effet une description formelle de certaines interactions et fonctionnalités du NoC facilite l'exploration de son architecture et permet de faire plus facilement des prédictions sur les

performances. De plus, un modèle formel offre une couverture plus exhaustive que toute autre approche. Les modèles formels pour les NoCs sont surtout utilisés pour s'assurer que les algorithmes de routage ne présentent pas de cas d'inter-blocages, mais il existe des modèles de prédiction des performances : tel que [69] pour la latence, [48] pour l'estimation de l'énergie, et [70] pour la surface. Cependant ces modèles imposent généralement des modifications au niveau architectural du NoC, ce qui très souvent induit un surcoût important en surface et en consommation (car les architectures sont plus complexes).

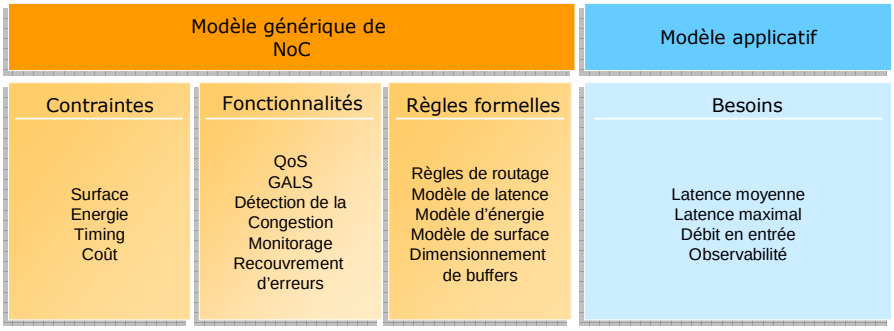


FIGURE 3.3 - Modèle de spécification d'un NoC

3.4. Le prototypage basé sur une plateforme reconfigurable

Dans le cas général, le prototypage consiste à réaliser un système sur une architecture différente de celle pour laquelle il a été prévu à la base (architecture cible). Typiquement, et dans le contexte des systèmes intégrés multiprocesseurs l'avantage des plateformes reconfigurables est double : Elles permettent d'une part de valider les composants logiciels très rapidement et cela sans qu'il soit nécessaire d'avoir à disposition le circuit final. D'autre part, et c'est ce qui nous intéresse principalement dans le cadre actuel des réseaux intégrés, il est possible de valider les fonctionnalités matérielles (absence de bugs) et à effectuer des mesures de performance et cela très rapidement [71][72]. Les possibilités de reconfiguration rapide seront utilisées ici afin de réduire les coûts et d'accélérer l'exploration architecturale. Dans ce qui suit nous nous intéresserons au prototypage des réseaux intégrés sur la technologie FPGA, nous commencerons par exposer les principaux avantages de cette approche.

3.4.1 Avantages et limitations du prototypage FPGA

A. Vitesse

Les avantages à utiliser une plateforme reconfigurables pour le prototypage d'un NoC sont évidents. Par rapport à la simulation RTL, le prototypage FPGA apporte un gain considérable en termes de vitesse de validation (exécution). A titre d'exemple le tableau 3.1 montre le gain en vitesse obtenu pour une simple instance de NoC 3x3 routeurs. En plus du gain en vitesse énorme un point important à noter réside dans le fait que les performances des simulateurs logiciels se dégradent à mesure que la taille du NoC augmente, alors que la fréquence maximale du processus d'émulation FPGA reste relativement constante a mesure que la complexité du système augmente. La vitesse de validation est importante à plus d'un titre, en premier lieu elle diminue la durée de la boucle d'exploration en réduisant la durée de validation de chaque instance. Un second avantage découle de la possibilité de tester rapidement le NoC considéré sous différents scénarios de trafic et avec plus de paquets, cela permettrait de déceler d'éventuels disfonctionnements ou bugs beaucoup plus rapidement que sur une approche par simulation.

TAB 3.1 Gain en vitesse de validation

	Simulation HDL (Modelsim)	Simulation SystemC (niveau cycle)	Prototypage FPGA (Virtex II pro)
Vitesse (cycles/seconde)	**3.0×10^3**	**20×10^3**	**100×10^6**
Gain (%)	**~**	**666%**	**33333 %**

B. Observabilité

Cependant même si le prototypage FPGA permet d'atteindre des vitesses très élevées, les possibilités d'observabilité des signaux à des fins de mise au point des composants matériels sont relativement limitées par rapport à celles offertes par la simulation. L'observabilité définit le degré d'efficacité et la puissance de débogage de la plateforme. Des

technologies de débogage temps réel « *in-circuit* » existent effectivement *« Xilinx Chipscope Pro »* [73] et *« Synplicity Identify »* [74] en sont des exemples, il est ainsi possible d'enregistrer et de suivre les états et transitions d'un ou plusieurs signaux dans le temps à des fins de mise au point. Des analyseurs logiques sont greffés aux modules utilisateur à cette fin. Toutefois, ces techniques nécessitent de mémoriser les états/transitions pour les restituer à l'utilisateur plus tard sous forme de chronogrammes temporels, et ont besoins donc d'autant plus de ressources que la durée d'échantillonnage (enregistrement) nécessaire est allongée. La logique de contrôle nécessaire au débogage in circuit peut elle aussi réduire la vitesse d'exécution à cause du surcoût matériel.

C. Coût

Il existe une gamme très étendue de plateforme de prototypage, dont les coûts financiers varient très largement. Cependant en général les plateformes de prototypage à usage général telles que celle considérées dans ce travail nécessitent des investissements initiaux très attractifs. A titre d'illustration la plateforme FPGA Xilinx XUPV2Pro et la suite d'outils logiciels associée à un coût comparable à une licence individuelle pour un simulateur commercial VHDL/Verilog. Naturellement, les possibilités de reconfiguration quasi infinies jouent aussi un rôle très important pour la réduction des coûts. Une seule plateforme peut ainsi être utilisée pour la validation de plusieurs systèmes sans induire aucun surcoût.

D. Flexibilité/ Mise à l'échelle

Par flexibilité nous entendons ici les capacités d'adaptation d'une plateforme de prototypage à divers modèles. A priori les possibilités de reconfiguration des plateformes FPGA tendent à faire penser qu'elles sont extrêmement flexibles. En réalité, il n'existe actuellement aucun circuit FPGA pouvant accommoder n'importe quel système.

Sans perte en généralité, il est possible de considérer la flexibilité comme étant la somme de deux composantes : capacités d'intégration propres et capacités d'interconnexion (évolutivité). Les capacités d'intégration propres d'une puce FPGA peuvent être relativement bien approximées par le nombre de portes logiques équivalentes pouvant être effectivement implémentées sur la puce. Evidemment, plus les capacités d'intégration sont élevées, plus

large sera l'ensemble des systèmes pouvant être supportés. Comme nous l'avons indiqué précédemment, les puces FPGA les plus courantes sont de capacité limitée cependant les possibilités d'interconnexions grâce à des connecteurs externes sont très nombreuses pour la plus grande majorité de ces plateformes. En conclusion, on peut dire qu'en général les plateformes de prototypage FPGA ont de bonnes propriétés de flexibilité (du moins si on considère l'aspect évolutivité). Cependant il est important de signaler à ce point que très peu d'outils existent actuellement pour exploiter de façon simple les possibilités d'évolutivité, et souvent les concepteurs ont recours à des solutions ad-hoc aux champs d'application très restreints.

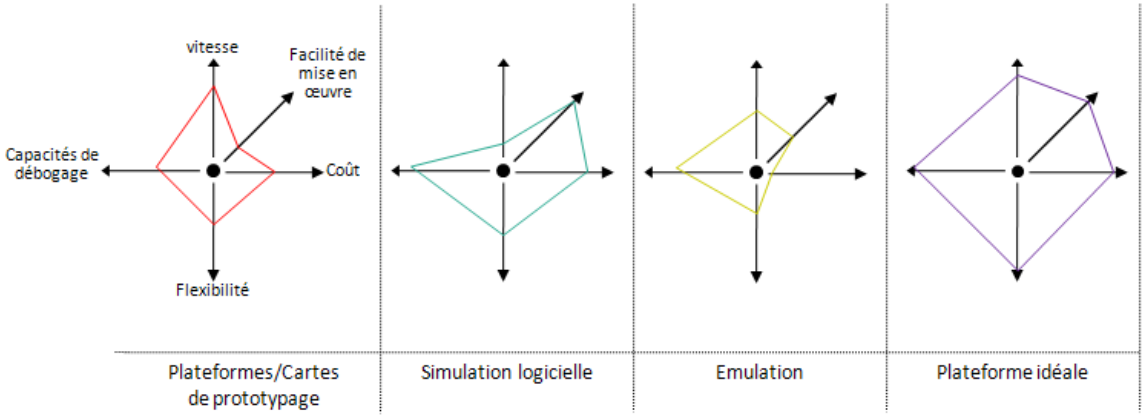


FIGURE 3.4 - Positions respectives des solutions de validation par rapport à la solution idéale

3.4.2 Flot de prototypage FPGA générique

La FIGURE 3.5 montre le flot générique de prototypage basé sur une plateforme FPGA. Comme indiqué la première étape dans le flot est une étape de spécification du modèle ciblé par le prototypage. Il convient de signaler ici que quelque soit le langage utilisé pour la description du modèle (VHDL, SystemC, langage de description FSM…) une condition nécessaire au bon déroulement du processus de prototypage est que la description soit synthétisable. La notion de synthétisabilité FPGA impose d'une part que la description du modèle se fasse au niveau RTL *« Register Transfert Level »* ; et que d'autre part elle tienne

compte de certaines spécificités des circuits FPGA ciblés. Sans trop entrer dans le détail, le but de ce mémoire n'étant pas de présenter la technologie FPGA, il convient de dire qu'une description complètement synthétisable sur une plateforme FPGA donnée peut s'avérer être incompatible avec une autre plateforme (problèmes de ressources, circuit non routable). Cependant, il est utile de préciser que le développement des technologies FPGA tend à uniformiser leur architectures ce qui réduit considérablement l'impact de cette limitation.

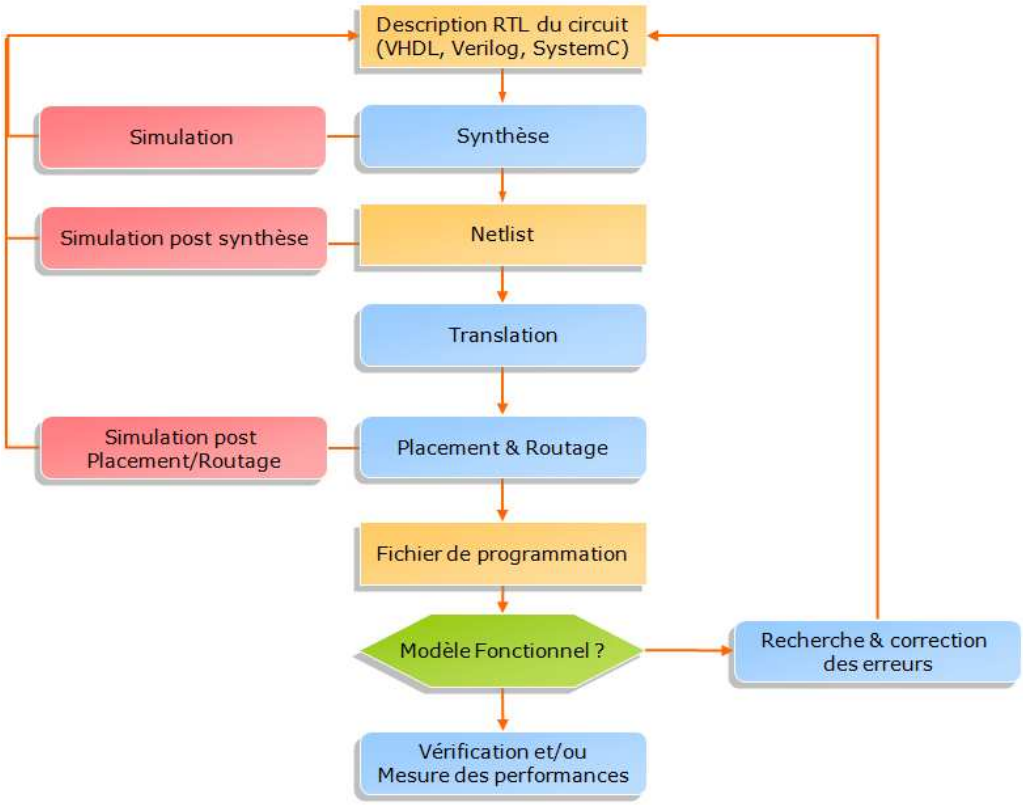


FIGURE 3.5 - Flot générique de prototypage FPGA

Lors de l'étape de synthèse, la description HDL du circuit est transformée en un assemblage (Netlist) de primitives de base (portes logiques, bascules…etc.). A ce point il est possible d'effectuer une étape de simulation post-synthèse, l'intérêt de cette étape étant de vérifier l'équivalence du comportement du circuit pré/post synthèse. En particulier, il est important de vérifier les propriétés temporelles du circuit.

La translation transforme la Netlist issue de la phase de synthèse en des primitives FPGA. En plus des primitives standards (AND, NAND, XOR…) la netlist issue de cette phase peut contenir des primitives plus élaborées tel que : des cellules mémoires RAM, des blocs encodeurs/décodeurs, multiplexeurs/démultiplexeurs et des additionneurs. La translation est généralement optimisée de façon à tirer profit au mieux des ressources FPGA.

Avant la phase de placement & routage, chaque élément de la netlist post-translation est associé à un type de cellule logique FPGA. Vient ensuite le placement qui place les éléments de la netlist (correspondant maintenant à des cellules) physiquement sur le circuit FPGA, en tenant compte d'un certain nombre de contraintes : délais de propagation, placement par rapport au E/S...etc. Une fois l'emplacement physique de chaque cellule fixé, le routeur effectue le calcul des interconnexions entre les cellules physiques en respectant la connectivité des cellules comme indiqué dans la netlist donnée en entrée. Il est possible ici aussi d'effectuer une simulation post placement & routage du circuit afin de vérifier une fois de plus l'équivalence avec les spécifications d'entrée.

Une fois chacune des cellules de la netlist mappées sur une cellule physique, et une fois que les interconnexions entre cellules ont été calculées, le fichier de configuration du FPGA peut être généré. Le fichier de configuration est une image binaire du circuit FPGA, il y détermine pour chaque élément (cellule logique, table de correspondance LUT, élément de routage, cellule mémoire) l'état logique. L'exécution a proprement dite du circuit est possible après que le fichier ait été chargé sur le circuit FPGA. Il est alors possible de vérifier les fonctionnalités du modèle et d'effectuer des mesures de performances.

La présentation faite ici du flot de prototypage FPGA a pour but de mettre en avant les différentes étapes de ce processus. Cette présentation convient pour tout circuit du moment qu'une description synthétisable de ce dernier est disponible. Dans le paragraphe qui suit, nous nous intéresserons plus particulièrement au prototypage FPGA de réseaux intégrés, et nous présenterons un flot adapté à ce type de circuits prenant en compte leurs caractéristiques architecturales.

3.5 Mise en œuvre d'un flot de prototypage de NoCs

3.5.1 Objectifs

Pour rappel, le prototypage matériel des réseaux intégrés vise deux objectifs principaux: En premier lieu une validation « rapide » des fonctionnalités, et en second lieu une évaluation toute aussi rapide des performances. Concernant la validation fonctionnelle d'un NoC, dans les sections précédentes de ce chapitre nous avons introduit brièvement les possibilités de mise au point (débogage) *«In-Circuit»* liées aux plateformes reconfigurables. Ces possibilités peuvent être mises à profit d'une part pour la mise au point/vérification des fonctionnalités d'un NoC, et d'autre part pour la mesure de certaines métriques de performance. Nous nous limiterons dans le cadre de ce travail à la seconde possibilité. Le second volet concernant la mesure de performances nécessite quand à lui le recours à des composants spécifiques et aux outils de conception dédiés aux FPGA.

Dans les paragraphes suivants nous donnerons une proposition de flot de prototypage adapté aux NoCs, et nous détaillerons la mise en œuvre de chacune de ces étapes. Au fur et à mesure nous introduirons aussi les possibilités et limitations du prototypage FPGA appliqué aux NoCs.

3.5.2 Définition du flot de prototypage pour les NoCs

A. Configuration Initiale du modèle du NoC

Ce sont les besoins du niveau applicatifs qui fixent les besoins en communications sur le NoC. Il est donc nécessaire de considérer ces besoins, pour une configuration initiale du modèle de NoC. Bien sûr, selon que l'on utilise tel ou tel modèle de NoC les paramètres pouvant être modifiés à cette étape varient. Nous donnons ici une liste non exhaustive des paramètres directement influencés par le modèle des besoins et pouvant être modifiés à ce niveau.

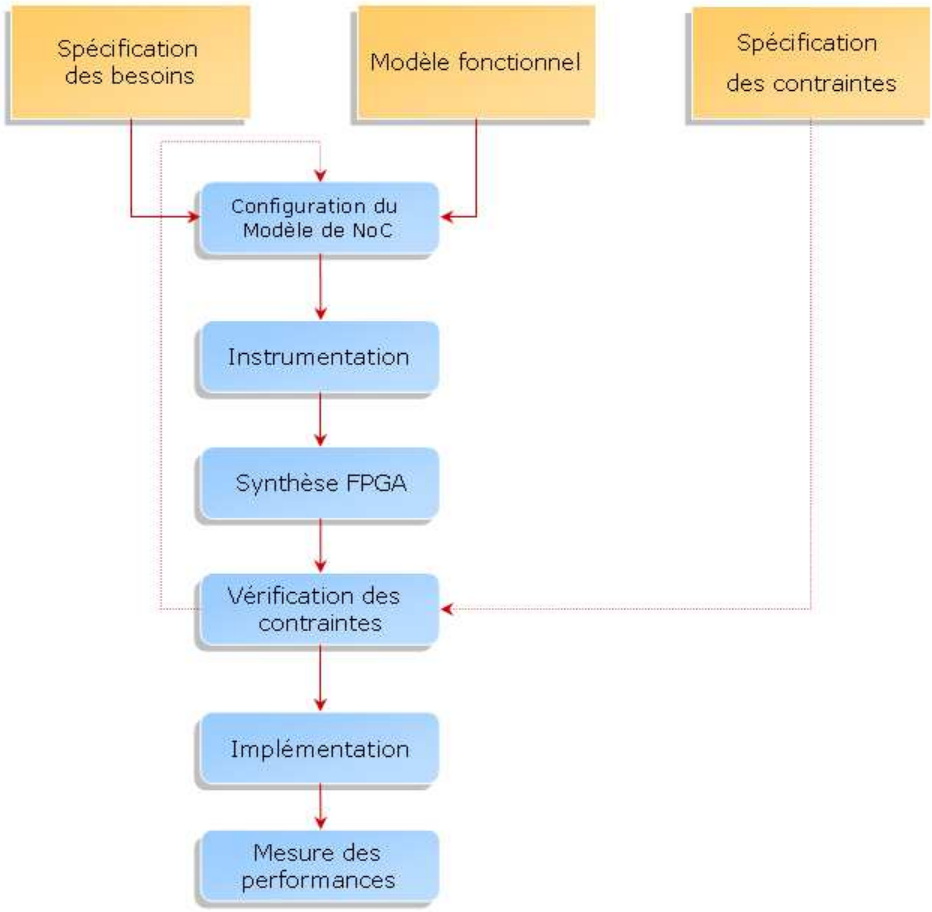


FIGURE 3.6 - Flot de prototypage FPGA pour réseaux intégrés

- ***Taille du réseau et Topologie :*** Comme nous l'avons vu les besoins sont exprimés le plus souvent sous forme d'un graphe de tâches avec annotation du volume des communications (ou autre contrainte relative à la communication) entre chaque couple de tâches. Partant de ce modèle, le nombre de nœuds composant le réseau peut être calculé [75][24]. Avant de donner une définition semi-formelle du dimensionnement d'un réseau intégré partant d'un graphe des tâches nous donnons au préalable quelques définitions :

Définition i : *Le graphe des tâches correspondant à une application donnée est un graphe A(T,F) orienté et valué. T représente l'ensemble des tâches et F l'ensemble des dépendances de données et/ou de contrôle entre chaque couple de taches (t_i,t_j). A chacun des arcs f_i de F est associé un couple de valeurs représentant les contraintes de communication (débit, latence).*

Définition ii : *Un réseau intégré, est un graphe R(C, L, $\Re$) orienté et valué. Où C représente l'ensemble de nœuds de ce réseau (routeurs) $|C| = n$; et L l'ensemble des liens existant entre chaque couple de nœuds (c_i, c_j). Un lien l_i est annoté par une ou plusieurs informations relatives à ses capacités de communication (débit, latence). $\Re = \{r_i : CxCxC \rightarrow C\}$ est l'ensemble des fonctions de routage donnant pour chaque nœud du réseau et pour chaque couple (c_i, c_j) le nœud suivant c_k selon la politique de routage.*

Définition iii : *Un mappage M d'un graphe des tâches A(T, F) sur un réseau R(C, L,$\Re$) est donné par la fonction $\pi : T \rightarrow C$ associant à chaque tache $t_i \in T$ un nœud $c_i \in C$.*
Un mappage est dit valide si :

$$\forall (t_i,t_j) \in T \text{ et, } \pi(t_i) = c_k \text{ , } \pi(t_j) = c_t \text{ S'il existe un arc } f_{ij} \in F$$

alors il existe un sous ensemble de $\Theta \subset \Re$: $\Theta = \{ r_k , r_l, , r_m \ldots, r_t\}$ et tel qu'il existe un chemin

$$r_k \circ r_l \circ r_m \ldots \circ r_t (c_k, c_t) = c_t \text{ dans le réseau}$$

Partant des définitions i, ii et iii il devient clair que le nombre de nœuds (taille du réseau) nécessaire et suffisant pour un graphe de taches donné, correspond au nombre de nœuds du sous-réseau issu du réseau initial qui permet d'assurer toutes les communications du graphe des tâches après mappage de celui-ci. Cependant, modifier la taille d'un réseau intégré peut entrainer une modification de sa topologie ; dans le cas où les mécanismes de routages sont dépendants de la topologie (ex : X-après-Y) on choisira une taille de réseau suffisante pour respecter la topologie initiale et assurer ainsi des mécanismes de routage cohérents. Dans le contexte du prototypage FPGA, le choix d'une taille de réseau optimale permet de préserver des ressources (Slices, BRAM) qui pourrons être utilisées pour l'implémentation de la logique additionnelle (cœurs de processeurs, IPs).

- ***Largeur de l'unité de transfert :*** La taille de l'unité de transfert correspond au nombre de bits qui composent un flit (8, 16, 32). Dans le cas ou le réseau intégré permet de paramétrer la taille des flit, il convient de choisir une largeur de transfert qui permette de satisfaire le débit le plus élevé nécessaire pour une application donnée. Cependant, il faut aussi considérer que le nombre de ressources FPGA est corrélé à la taille de l'unité de transfert.

- ***Profondeur des buffers :*** Tout comme pour la taille de l'unité de transfert, le choix judicieux des tailles de buffers optimise les ressources utilisées par le NoC car se sont les organes nécessitant le plus de surface. En règle générale, il faut savoir que la taille des buffers influence la charge maximale pouvant être acceptée sur le réseau et à une influence moindre sur le débit maximal, car ce dernier paramètre dépend aussi des conditions de congestion. La latence quant à elle n'est pas influencée par la taille des buffers sous les conditions de trafic normales. Il existe des méthodes plus ou moins formelles [76] [77] qui partant d'une description d'un graphe des tâches, permettent de calculer une distribution de l'espace de bufférisation, d'autres approches notamment [78] considèrent un ensemble de métriques plus complet (énergie et latence en plus de la taille des buffers).

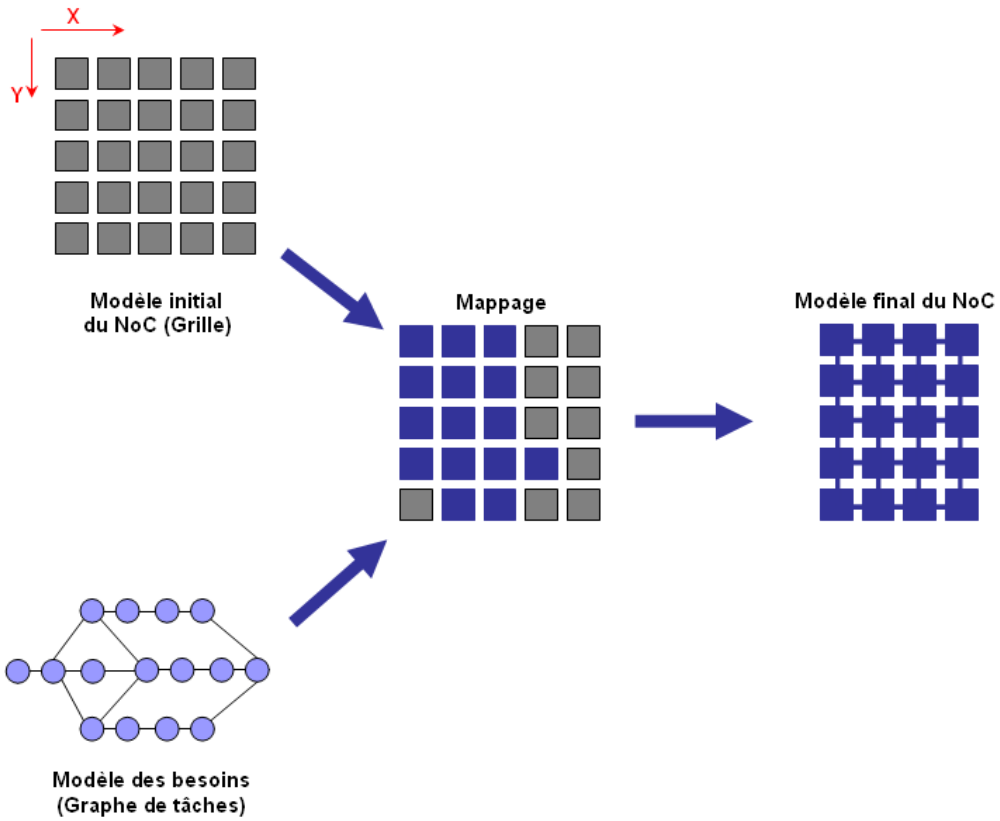


FIGURE 3.7 - Configuration initiale de la taille du réseau

- ***Fréquence de fonctionnement :*** C'est ce paramètre qui détermine la vitesse d'exécution du NoC sur FPGA. Au stade initial du processus de prototypage, il convient de choisir la fréquence maximale permise par la plateforme considérée. La fréquence effective à laquelle

il sera possible d'exécuter le NoC, ne sera elle, déterminée qu'après placement routage et dépendra de la longueur du chemin critique (NoC et IPs connectées). Si la contrainte sur la fréquence initialement choisie ne peut être satisfaite une erreur de violation de timing sera déclenchée. Il est important de commencer par choisir une fréquence aussi élevée que possible afin d'obliger l'outil de placement routage à faire toutes les optimisations possibles. Enfin, dans certains cas de figures, il peut être intéressant de choisir une fréquence de fonctionnement aussi proche de celle qui sera utilisée sur l'implémentation finale (ASIC) afin de garantir une cohérence de comportement du NoC avec les autres composants.

B. Instrumentation

L'instrumentation consiste à intégrer des modules matériels additionnels au modèle du NoC original (FIGURE 3.8) afin de vérifier son comportement à l'exécution et aussi pour mesurer ses performances (moniteurs) [79]. En plus des deux fonctions précédentes, dans certains cas de figures des modules « générateurs de trafic » peuvent être nécessaires ; un générateur de trafic est un composant le plus souvent matériel qui va permettre de générer des messages aléatoires suivant une loi de probabilité donnée, ou bien qui va régénérer un trafic correspondant à une application donnée. L'intérêt des générateurs de trafic est double. D'une part ils permettent d'accélérer le processus de prototypage et d'autre part ils présentent un faible coût en ressources FPGA. Les moniteurs quand à eux enregistrent l'activité à des endroits bien précis du NoC afin de mesurer ses performances tel que : la latence des paquets, les points de congestion …etc. Sous forme d'analyseurs logiques ils servent aussi à détecter d'éventuelles violations de protocoles.

C. Vérification des contraintes

L'étape de vérification des contraintes consiste à vérifier avant l'implémentation FPGA que le modèle de NoC respecte bien les limitations spécifiées dans le modèle initial. Généralement dans un processus de prototypage standard les contraintes sont celles imposées par la plateforme utilisée, comme par exemple la surface ou encore la fréquence d'horloge maximale pouvant être employée. En effet, un NoC n'est presque jamais destiné à être implémenté de façon définitive sur FPGA ce qui dans un sens limite la signification d'une

contrainte pendant le prototypage. Cependant dans certains cas le concepteur peut essayer d'optimiser l'architecture du NoC pendant le prototypage, et dans ce cas un modèle de contraintes est très utile. La plupart des contraintes physiques (sur FPGA) comme la surface, la fréquence maximale et même l'énergie peuvent être estimées avant de passer à la phase d'implémentation après les phases de synthèse/mappage.

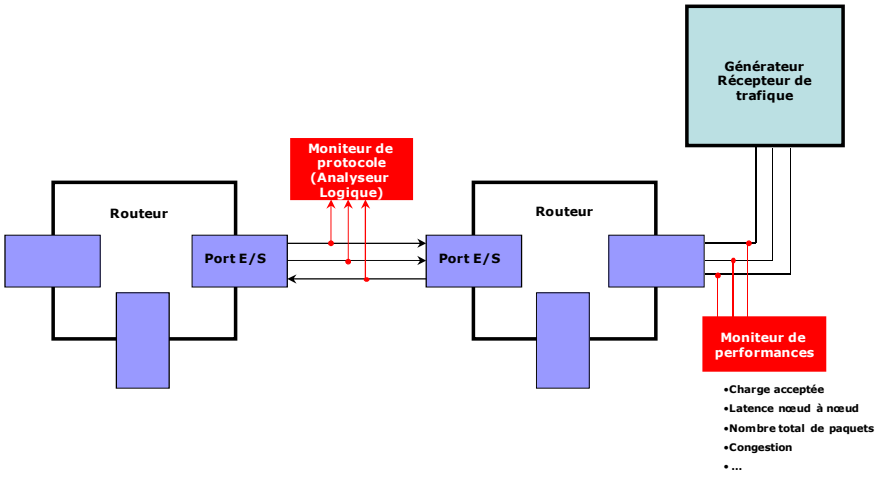


FIGURE 3.8 - Instrumentation du modèle de NoC avant prototypage

D. Mesure des performances

La dernière étape est la mesure des performances qui consiste à rassembler les informations issues des moniteurs de trafic pendant la phase d'exécution. Ces mesures de performances seront ensuite utilisées pour guider la phase d'exploration architecturale.

3.6 Scénarios de test des NoCs sur FPGA

Il existe principalement trois scénarios de test d'un NoC sur FPGA. Le choix de tels de ou tels scénarios de test doit être guidé d'une part par le niveau recherché de précision et d'autre part par la capacité d'intégration de la plateforme disponible. Du point de vue de la précision, le scénario le plus complet consiste à intégrer le réseau ainsi que tous les composants tels qu'ils seront présents dans l'architecture finale. Cette solution nécessite qu'une description adaptée à l'implémentation FPGA de chaque composant soit disponible.

Par ailleurs, la plateforme FPGA ciblée doit être suffisamment large pour intégrer tous les composants (à defaut des mécanismes de synchronisation doivent être prévus [80]). L'intérêt d'une telle approche réside dans le niveau de précision pouvant être atteint. En effet les conditions de trafic réalistes permettent un dimensionnement du réseau au plus près de l'optimum. Cependant, comme il n'est pas toujours possible de recourir à cette solution pour indisponibilité de l'une ou l'autre des pré-conditions précédemment énoncées, il devient dès lors nécessaire de recourir à des solutions alternatives.

Les solutions alternatives, passent par une intégration partielle des composants du système avec comme conséquence une diminution du niveau de précision. Le NoC étant l'organe visé par le prototypage, en plus d'être un composant clé sans lequel aucune exécution n'est possible, seuls les autres composants peuvent être abstraits. Quelque soit le type de composant connecté au NoC (CPU, Mémoire, IP, organe E/S), il peut être remplacé par un couple générateur/récepteur de trafic reprenant son comportement. Une question importante consiste alors à définir pour un système donné quels composants doivent être remplacés par des générateurs de trafic, et quel comportement assigner à ces générateurs. La dernière solution consiste à remplacer tous les composants par des générateurs de trafic, bien sûre cette solution est la moins précise de toutes, cependant elle présente l'avantage d'être la plus flexible avec une mise en œuvre très rapide. D'un point de vue purement technique, l'émulation avec des générateurs de trafic se décline sous deux variantes décrite ci-dessous.

3.6.1 Approche de test centralisée

Dans ce cas, les générateurs et récepteurs de trafic sont tous contrôlés par un organe central (typiquement un processeur). L'injection des paquets par chaque GT (générateur de trafic) est commandée par le processeur central, les récepteurs de trafic (RT) sont eux aussi contrôlés par le processeur central (FIGURE 3.9).

Compte tenu du contrôle centralisé, toute la configuration de la plateforme de test peut être accomplie par le processeur central au moyen d'un programme logiciel (loi de génération des paquets, loi de consommation des paquets). La flexibilité logicielle peut être mise à profit pour tester le NoC sous différentes configurations de trafic (longueurs de messages, fréquence des burst) sans qu'aucune intervention au niveau de la couche matérielle ne soit nécessaire.

Les possibilités de configuration in-situ sont très intéressantes pour l'accélération du processus de prototypage et de test. De plus le processeur de contrôle à une vision globale du NoC, chose qui permet d'effectuer des mesures extrêmement précises au niveau de chaque entrée du NoC.

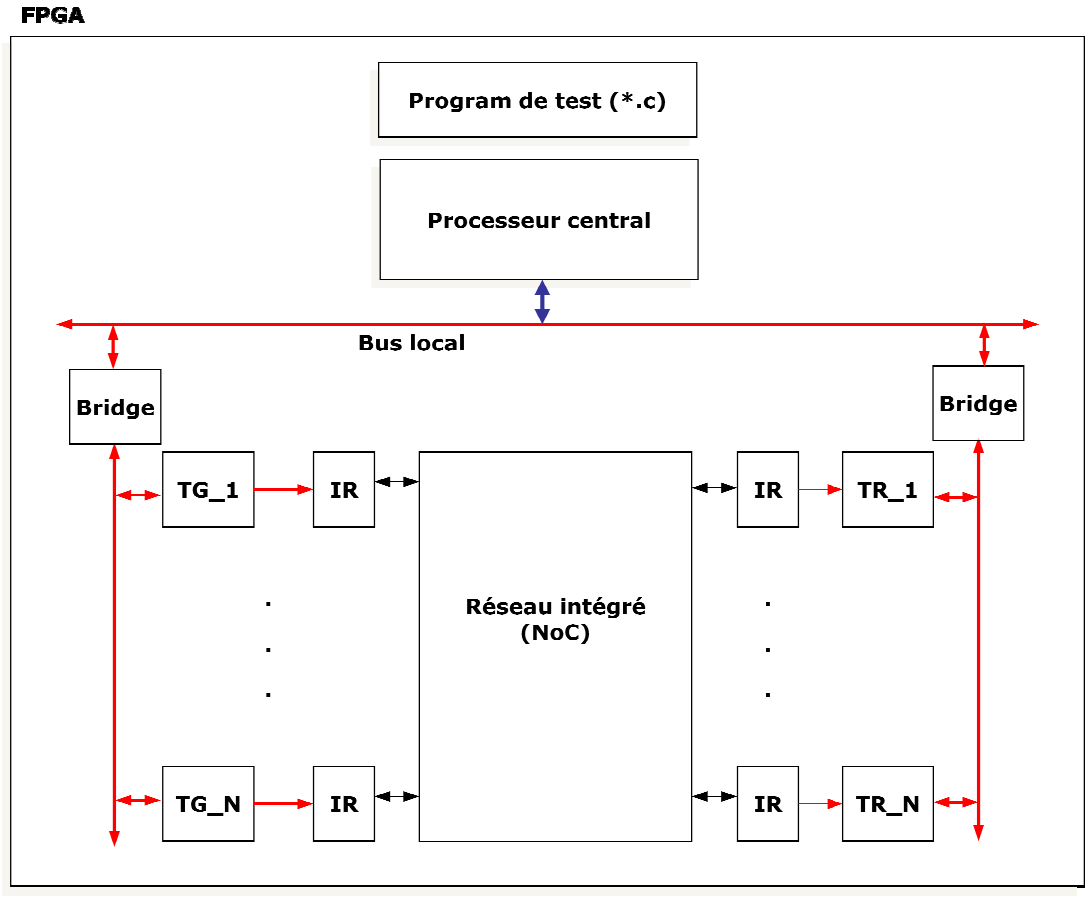


FIGURE 3.9 - Approche centralisée d'émulation de NoCs

Etant donnée un NoC, la mise en place d'une telle plateforme ne nécessite que des modifications au niveau des interfaces réseau (NI) afin d'effectuer l'adaptation nécessaire au protocole physique des messages issus des TG. Pour des raisons de vitesse d'exécution les composants TG et TR sont relativement autonomes, c'est-à-dire qu'ils sont capables de construire des messages suivant une loi de distribution de probabilité donnée partant uniquement d'un nombre restreints de paramètres envoyés par le processeur central.

Du point de vue des performances globales (vitesse d'émulation du NoC, ressources) de cette approche, il est difficile de donner des estimations précises car cela dépend de beaucoup de paramètres. Cependant on peut s'attendre à ce que l'implémentation FPGA du NoC sous cette configuration centralisée (possible avec des directives de placements spécifiques) nécessite moins de ressources de routage que si le NoC été implémenté de façon éclatée.

3.6.2 Approche de test distribuée

Dans cette approche (FIGURE 3.10), contrairement à la précédente aucun contrôle centralisé n'est prévu. Les générateurs et récepteurs de trafic sont configurés avant l'implémentation finale sur FPGA. Ils agissent de façon complètement autonome durant l'exécution du modèle de NoC. De plus, la configuration se faisant au niveau code HDL de chaque composant, il devient dés lors nécessaire de refaire une étape Synthèse/Placement&Routage/Implémentation pour chaque modèle de trafic. Par configuration nous entendons :

- Loi de distribution de la longueur des messages
- Loi de distribution des destinations des messages
- Loi de distribution de la taille des burst
- Débit maximal/moyen d'injection

Ici, tout comme pour l'approche précédente il est nécessaire d'adapter les interfaces réseau du coté des générateurs de trafic afin d'être en mesure d'injecter des messages sur NoC au bon format.

Le seul avantage de cette approche par rapport à la précédente réside dans les possibilités d'adaptation à une plateforme de prototypage multi-FPGA comme nous le verrons dans le chapitre suivant. Une approche centralisée est mois facilement portable sur une telle plateforme à cause des besoins accrus en synchronisation globale. Un autre problème induit par l'absence d'un contrôle centralisé, réside dans la collecte des mesures de performances qu'elles soient effectuées par les récepteurs de trafic ou par des moniteurs additionnels. Pour cela nous proposons deux solutions. La première est une solution d'appoint permettant de collecter les mesures au niveau de chaque récepteur/ moniteur de trafic par la technologie de

debug « *in-circuit* ». L'idée étant d'utiliser des analyseurs logique intégrés très simples afin d'accéder aux valeurs de chacune des mesures (préalablement sauvegardées dans des registres) ; cette solution n'introduit qu'un surcout limité en ressources car il n'est pas nécessaire de mémoriser toute une suite de transitions mais seulement une seule valeur.

La seconde solution tient place dans le corps même des générateurs et récepteurs de trafic. Elle nécessite que le processus de test soit divisé en deux phases. La première phase consiste en une exécution et mesure des performances classique. Et une seconde phase où toutes les mesures au niveau de chaque récepteur/moniteur sont formatées et envoyées à un nœud maître au travers du réseau lui-même. Le nœud maître se charge ensuite d'envoyer toutes les mesures vers l'extérieure en passant par une interface d'entrée/sortie (interface RS232, Ethernet ou bien JTAG).

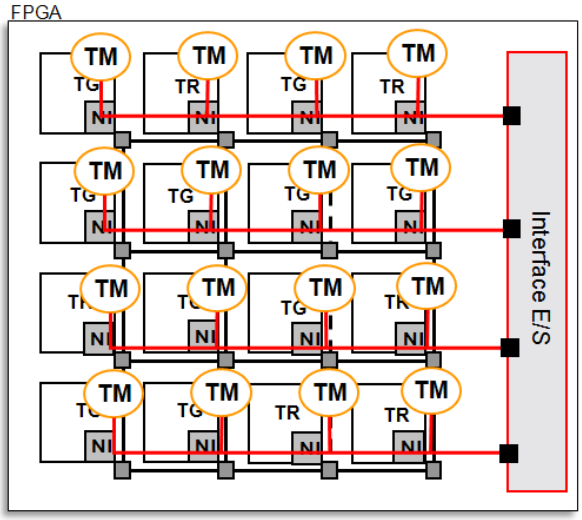


FIGURE 3.10 - Approche distribuée d'émulation de NoCs

3.7 Les générateurs de trafics

3.7.1 Présentation

Avant de présenter l'architecture proposée pour les générateurs de trafic, il est très important de bien déterminer le rôle délégué à ces composants dans le processus de prototypage. De façon très simple, un générateur de trafic est connecté au NoC et est destiné à produire un flot de messages afin de permettre d'avoir un retour sur les performances sous différentes conditions. Partant de cela, un cahier des charges préliminaire peut être fixé pour

ces composants. Premièrement, ces composants étant destinés à une implémentation matérielle, il devient primordial que leur architecture reste aussi simple que possible. Deuxièmement, les contraintes de vitesse sur le prototypage imposent que ces composants aient des possibilités de paramétrage poussées, et ceci afin d'être en mesure de reproduire des conditions de trafic représentatives d'un large éventail d'applications.

Classiquement, les techniques de prédiction et d'estimation pour les macro-réseaux étaient basées sur des modèles de lois statistiques (loi de Poisson, loi Normale, Modèle d'Erlang). D'un point de vue matériel, reprendre des lois statistiques aussi élaborées nécessiterait des composants très coûteux en ressources. De plus l'intérêt de tels modèles reste très limité dans le contexte des réseaux intégrés. En effet, les NoCs à commutation de paquets présentent une hyper sensibilité aux variations des conditions de trafic, quand on sait que les lois statistiques classiques sont plutôt destiné à donner une modélisation d'ensemble il devient claire que leur apport serait très restreints sinon insignifiant [81] [82]. Bien que nous ayons montré qu'il n'est pas nécessaire de disposer de lois de distributions élaborées, il n'en reste pas moins que les générateurs doivent être très flexibles et doivent être capables de générer des conditions de trafic variées. Ci-après nous donnons les paramètres les plus importants pour un générateur de trafic destiné à un réseau à commutation de paquets :

- *Longueur des paquets :* Représente le nombre de flits composants un paquet. Selon l'application et l'architecture du NoC la longueur peut être soit fixe soit variable (longueur Min/Max).
- *Longueur des messages :* C'est le nombre de paquets appartenant au même message. La taille des messages étant variable il convient de fixer un intervalle [taille min, taille max]. Il est à noter que pour les réseaux permettant des longueurs de paquets variables la notion de message se confond avec celle de paquet.
- *Intervalles inter-messages :* Le nombre de cycles entre deux messages successifs (Min/Max).
- *Fréquences de Burst :* Un burst est une succession de message avec un délai inter-messages minimum. La fréquence est exprimée par un nombre réel $f \in [0,1]$.

La FIGURE 3.11 montre la compostions d'un paquet, d'un message et finalement d'un burst, à quelques nuances près cette représentation reste valable pour la plupart des réseaux à commutation de paquets.

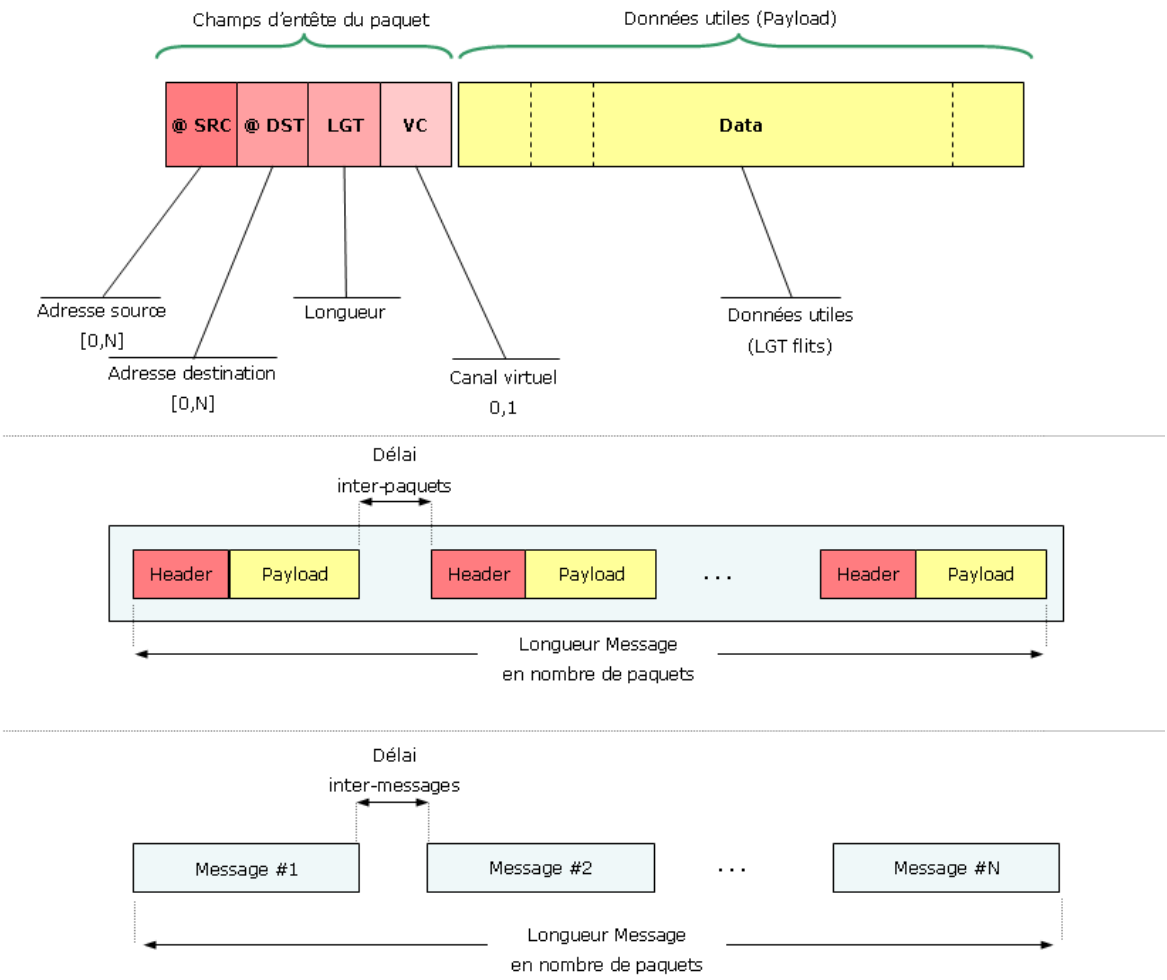


FIGURE 3.11 - Structure (a) d'un paquet (b) d'un message et (c) d'un burst sur un réseau intégré

Enfin, un examen des architectures MPSoC montre que leurs composants constitutifs peuvent être classés en deux catégories : la catégorie des composants initiateurs et celles des composants cibles par analogie avec les composants maîtres et esclaves sur un bus. D'une façon très simple, un initiateur est un composant pouvant initier des envois de messages sur le réseau (typiquement les nœuds de calcul) et une cible se contente de répondre aux messages reçus (les mémoires). Nous avons donc repris les deux comportements pour l'implémentation des générateurs de trafic.

3.7.2 Implémentation

A. Les générateurs de trafic initiateurs

L'architecture que nous proposons ici, est construite autour de deux modules distincts, le premier module est lui-même composé de plusieurs registres à décalage avec rétroaction linéaire connus aussi sous la dénomination de LFSR «*Linear Feedback Shift Register*» [83]. Dans ce qui suit nous détaillerons les mécanismes des LFSR ainsi que leurs propriétés en tant que générateurs de nombres pseudo-aléatoires. Nous introduirons ensuite la micro architecture des générateurs de trafics initiateurs.

i. Les registres à décalage avec rétroaction linéaire

Tout d'abord les LFSR permettent de générer de façon très simple une suite de nombres pseudo-aléatoires avec de bonnes propriété statistiques dans le cadre de nos besoins actuels [84], et cela sous réserve d'un choix judicieux des coefficients de rétroactions C_i. La FIGURE 3.12 illustre le mécanisme de rétroaction des LFSR :

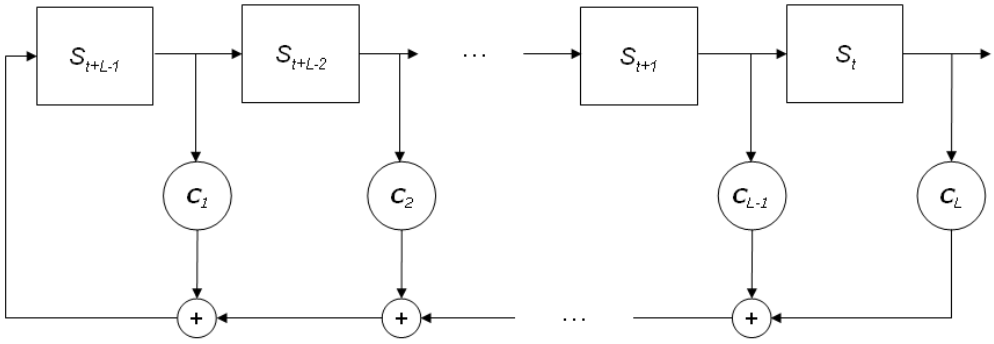


FIGURE 3.12 - Registre à décalage avec rétroaction

A chaque top d'horloge la nouvelle valeur du LFSR dont la longueur est L et les coefficients de rétroaction ($C_1 \ldots C_L$) peut être calculée par la formule suivante :

$$S_{t+L} = \sum_{i=1}^{L} C_i \cdot S_{t+L-i} \qquad (1)$$

Dans le cadre du présent travail, nos besoins initiaux sont de disposer de générateurs de nombres pseudo-aléatoires avec une distribution uniforme, et nécessitants un minimum de ressources matérielles. Notre choix c'est donc naturellement porté sur les LFSR car ils répondaient aux deux exigences.

ii. Architecture du générateur de trafic

L'intégration des registres LFSR dans le module générateur de trafic est telle que montrée dans la FIGURE 3.13. A chacun des paramètres du générateur énoncés dans le paragraphe 3.7.1 nous avons associé un registre LFSR.

Pour les paramètres ayant un intervalle comme la longueur des messages par exemple, nous dérivons une valeur $V_p \in [V_{min}, V_{max}]$ à partir de la valeur du registre LFSR comme suit :

$$V_p = V_{min} + LFSR \qquad (2)$$

La taille du LFSR en nombre de bits, étant choisie de façon à ce que l'inégalité (3) soit toujours vérifiée

$$V_{max} \geq V_{min} + LFSR \qquad (3)$$

Les mécanismes précédents permettent de s'assurer que les paramètres du trafic se situent bien dans les intervalles choisis; et ainsi que le trafic est bien conforme au profil attendu. La machine à états finis en aval enregistre les valeurs contenues dans les LFSR à chaque début d'envoi (envoi de paquet et de message), et utilise les valeurs précédentes pour l'envoi courant. Cette approche bien que comparable en termes de ressources et de comportement, est bien plus simple à implémenter que l'approche classique consistant en une chaine de Markov à deux états [85].

La génération d'un nouveau message se fait en trois étapes ; en premier lieu on récupère la longueur du message courant i.e. le nombre de paquets (et on enregistre la valeur contenue dans le LFSR correspondant). Deuxièmement, on récupère la longueur du paquet courtant et on enregistre aussi la valeur du LFSR correspondant, et finalement la troisième étape consiste en l'envoi effectif du paquet courant et la mise à jours des valeurs de contrôle intermédiaires (nombre de paquets déjà envoyés, nombre de cycle déjà écoulés). La FIGURE 3.13 montre

l'architecture globale d'un générateur de trafic initiateur et la FIGURE 3.14 celle de la FSM de contrôle dans sa globalité.

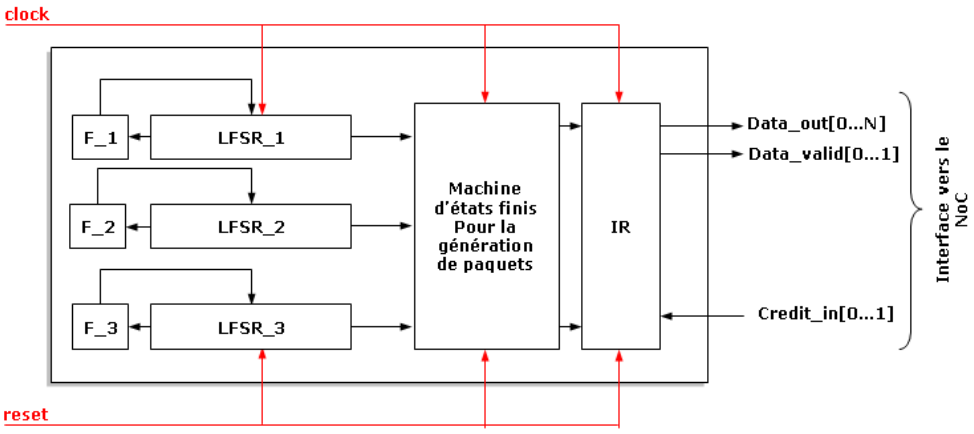


FIGURE 3.13 - Microarchitecture du générateur de trafic initiateur

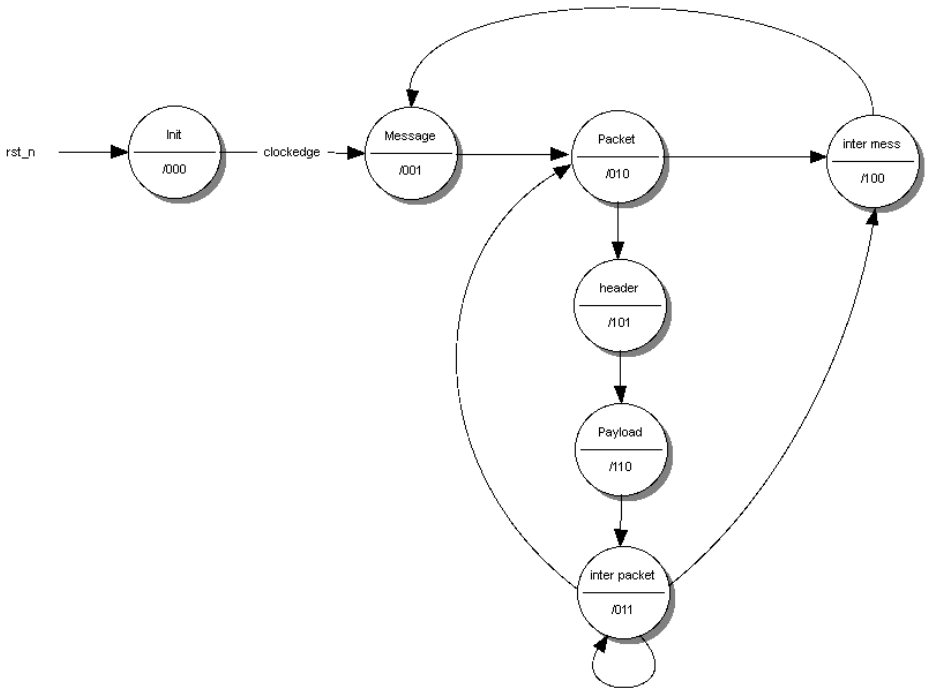


FIGURE 3.14 - Machine d'états finis pour la génération de messages

iii. Génération du plan d'adressage

Dans le paragraphe précédent nous nous sommes intéressés uniquement à la façon de générer les messages. Cependant les adresses de destination des messages tiennent une place tout aussi importante sur les performances mesurées. Un message est généralement adressé à un nœud unique bien déterminé sur le réseau (le support du multicast et du broadcaste étant extrêmement rares). La question à laquelle nous tentons de répondre ici est de savoir comment générer un plan d'adressage global réaliste en partant de la description du modèle des besoins (graphe des tâches).

Après mappage du graphe des tâches sur le réseau intégré, il possible de savoir quel sont les nœuds qui communiquent avec quels autres nœuds du réseau, et même d'avoir une estimation de la fréquence et du volume de ces communications. Afin d'exploiter ces informations pour la génération du plan d'adressage, nous avons utilisé une table de correspondance donnant pour chaque adresse du réseau une probabilité. Au niveau d'un nœud *j*, une probabilité d'adressage P_i exprime la probabilité pour que le message courant soit adressé au nœud *i*. Pour des raisons de simplicité la probabilité P_i exprime à la fois le volume et la fréquence des communications.

TAB 3.1 Table de correspondance pour la génération d'adresses

Nœud	*Probabilité*
j	*0*
…	…
i	P_i

Tout comme pour les autres paramètres du trafic, nous avons utilisé comme générateurs de nombre pseudo aléatoires un LFSR. Le mécanisme de base étant qu'un paquet est adressé à un nœud *i* si la valeur du registre LFSR à l'instant *t* est comprise dans l'intervalle

$$] V_i, V_{i+1}] \quad avec VMAX(LFSR)/(V_{i+1} - V_i) = P_i .$$

Cette approche de génération d'adresses est une approche efficace d'un point de vue de l'implémentation matérielle, et garantit que le trafic généré reproduit des conditions très proches de celles de l'application ciblée. Cela a pour effet d'augmenter la précision des mesures de performances.

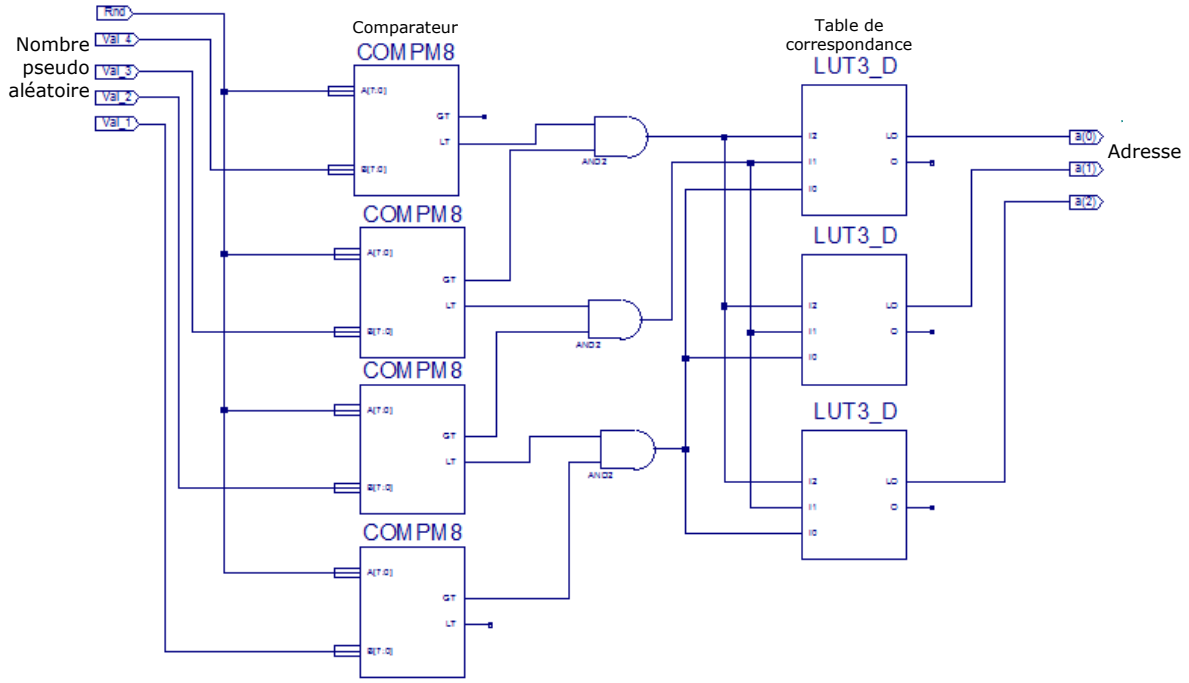


FIGURE 3.15 - Détail du générateur d'adresses

B. Les générateurs de trafic cibles

Contrairement à un générateur de trafic initiateur, un générateur de trafic cible ne peut initier de transferts sur le réseau de lui-même, mais seulement en réponse à un message reçu. Typiquement ces composants sont utiles pour modéliser le comportement d'une mémoire. Cependant ils peuvent tout aussi bien servir pour la modélisation d'autres composants tels que des accélérateurs matériels.

Pour le développement des générateurs passifs, nous avons pris comme modèle une mémoire RAM, cependant il nous a fallut trouver un juste milieu entre un comportement très spécialisé et donc très peu flexible, et un comportement très générique avec ce que cela implique comme ressources additionnelles. La question centrale était de choisir la longueur des messages réponses, sachant que l'adresse destination est celle de la requête reçue, et que

le contenu des messages n'est pas significatif. Partant du principe qu'un processeur adresse généralement une mémoire en octet, mot (16 bits), double mot (32 bits) et en quadruple mot (64 bits) , et éventuellement en 128 bits, la longueur d'un message réponse prend sa valeur dans cet intervalle en prenant en compte la largeur du flit. Par exemple si le réseau est configuré pour une longueur de flit de 32 bits, la longueur des messages réponse en flits variera de façon aléatoire dans l'intervalle (1, 2, 4). Enfin, nous noterons qu'ici aussi nous avons fait usage d'un registre LFSR, pour le calcul de la longueur des messages à envoyés.

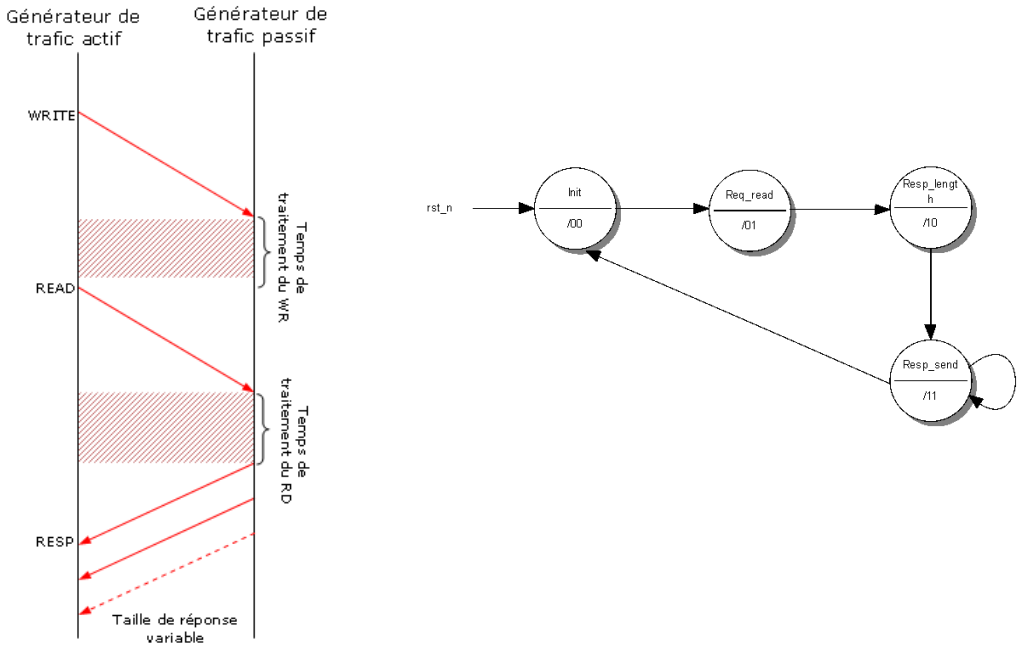


FIGURE 3.16 - Fonctionnement et machine à état finis d'un générateur de trafic cible

3.8 Les moniteurs de performances

Nous allons aborder dans cette section l'implémentation des composants moniteurs. Les moniteurs de performances vont permettre de collecter et de mesurer certaines métriques de performances à des points bien précis sur le NoC. Cependant, avant d'aborder plus en détails l'implémentation matérielle de tels composants, il est important de souligner la difficulté de mesurer des performances en temps-réel sur un circuit électronique. En effet, contrairement à

une approche logicielle, l'implémentation matérielle limite drastiquement l'observabilité et la contrôlabilité du circuit. Tenter d'augmenter l'un ou l'autre de ces paramètres se traduit par une augmentation des ressources déjà très limitées sur un circuit FPGA. Nous avons principalement développé trois types de moniteurs de performances, des moniteurs de latences point-à-point, des moniteurs de charge et des moniteurs de congestion.

A. Les moniteurs de latence

La latence est définit comme étant le nombre de cycles écoulés depuis la production (et non l'injection dans le réseau) de l'entête d'un paquet jusqu'à ce que le dernier flit du paquet ait été reçu par le nœud destinataire. Afin d'être en mesure de calculer la latence, il est tout d'abord nécessaire de définir une base de temps commune à tout le réseau. Sur un réseau complètement synchrone le signal d'horloge peut être utilisé à cette fin, cela devient plus difficile sur des réseaux multi-synchrones ou asynchrones. A chaque fois qu'un paquet est produit et avant qu'il ne soit envoyé sur le NoC il est estampillé par la valeur d'un compteur de cycles (sur 64 bits). Lors du processus de prototypage les données utiles (Payload) dans le corps du paquet peuvent prendre des valeurs quelconques, nous les utilisons donc pour contenir la valeur de l'estampille temporelle. Une fois un paquet reçu par un récepteur de trafic il est facile de calculer la latence, qui sera égale à la différence entre la valeur du compteur de cycles local et l'estampille du paquet (sous l'hypothèse synchrone).

Cependant, afin d'être en mesure d'exploiter les informations de latence il est nécessaire de trouver une solution de sauvegarde très compacte. De plus, comme il n'est pas possible de calculer facilement une moyenne de façon matérielle, il est nécessaire de trouver une représentation adéquate. La solution que nous proposons utilise une table de correspondance au niveau de chaque nœud, une entrée de la table est un triplet dont la structure est donnée comme suit :

(Adresse source, Nombre de paquets reçus, Latence cumulée)

Le calcul des latences moyennes point à point globales pourra ensuite être fait de façon logicielle une fois toutes les valeurs des tables collectées.

B. Les moniteurs de charge

La charge du réseau exprime le rapport de trafic effectivement accepté par le réseau par rapport au trafic total. Pour le calcul de la charge acceptée, nous enregistrons le nombre total de tentatives d'émission au niveau de chaque générateur de trafic ainsi que le nombre total de tentatives réussies. Le rapport pourra ensuite être calculé de façon logicielle tout comme pour les latences moyennes.

C. Les moniteurs de congestion

Les moniteurs de congestion fonctionnent sur le même principe que les moniteurs de charge, à la différence près que la valeur de la congestion est calculée de façon matérielle. Ces moniteurs de congestion vont servir à la fois pour la mesure de performances temps-réel et pour l'implémentation de techniques d'évitement de la congestion. Un moniteur de congestion à la faculté de générer une notification dès que la valeur de congestion d'un port dépasse un certain seuil fixé par le concepteur.

Le principe de fonctionnement du moniteur de congestion est illustré dans la FIGURE 3.17. La notification peut être utilisée soit pour déterminer de façon précise les conditions de trafic (charge) qui conduisent à un taux de congestion donné, soit pour réguler le trafic à l'entrée du réseau afin de réduire le taux de congestion.

$$Seuil = V_1 / V_2$$

$$Cong = '1' \quad si \quad Cong \geq Seuil$$

3.9 Les récepteurs de trafic

Dans la plupart des systèmes à base de NoC, un paquet est retiré du réseau dès qu'il arrive à destination. Le scénario typique étant que le message est stocké temporairement dans une file d'attente matérielle avant d'être traité. Que ce soit une lecture, une écriture ou encore un message (dans le cas d'un système à mémoire distribuée), il est toujours possible de dimensionner la taille de la file d'entrée pour retirer immédiatement de paquet du réseau.

Néanmoins, dans certains rares cas des délais avant le retrait des paquets devrons être considérés pour plus de précision.

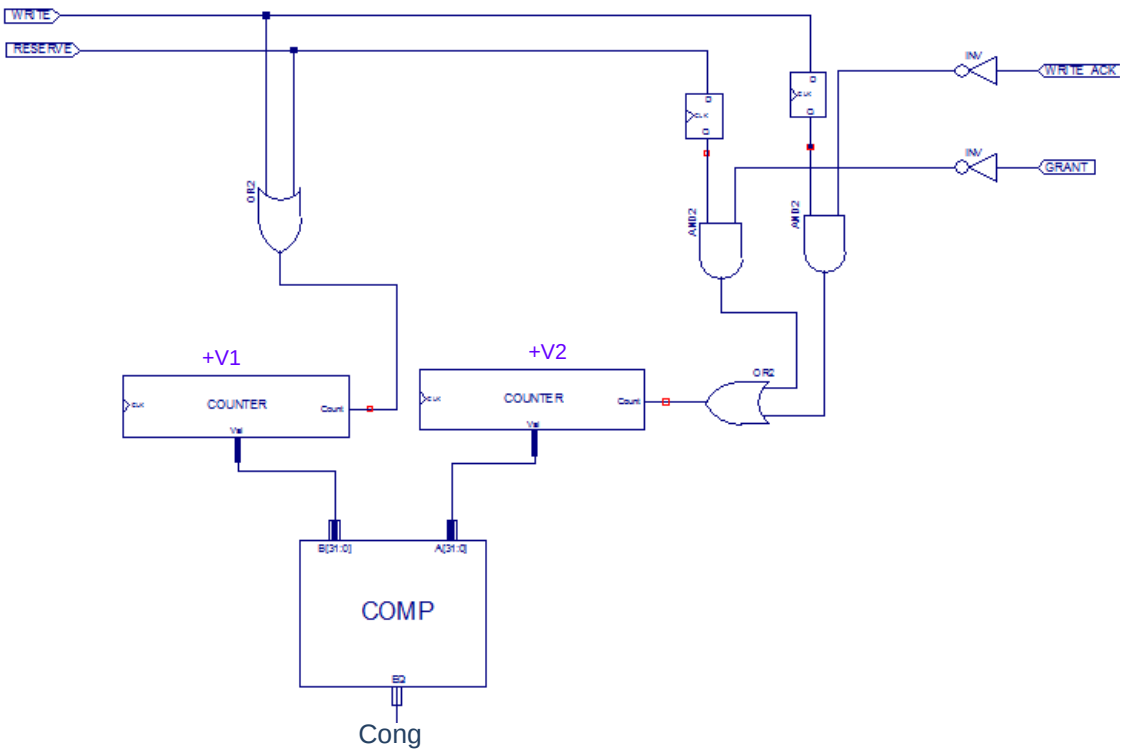


FIGURE 3.17 - Microarchitecture du moniteur de congestion

3.10 Outils et techniques d'optimisation du prototypage des NoCs sur FPGA

On peut trouver dans la littérature différentes techniques d'optimisation de circuits pour une implémentation FPGA. Un certain nombre de techniques [86] [87] on été proposées pour les NoCs, néanmoins nous nous intéresserons dans le cas-présent uniquement aux techniques ne demandant aucune modification au niveau architectural du NoC. Le but essentiel recherché ici est une implémentation optimisée de réseaux intégrés sur une plateforme FPGA. Comme nous l'avons indiqué précédemment cela permet d'une part qu'un maximum de ressources puisse être utilisé par la logique utilisateur (générateurs de trafic, mémoires, Processeurs, IP), et d'autre part que le NoC puisse fonctionner sous des conditions de fréquences optimales.

Parmi les techniques les plus couramment utilisées pour les réseaux intégrés, on peut citer l'exploitation de la structure régulière du NoC afin d'optimiser au mieux le placement FPGA. Des directives spéciales permettent d'indiquer à l'outil de placement routage l'emplacement relatif de chacun des routeurs (et aussi les corps) l'un par rapport à l'autre. Cela afin d'avoir des interconnexions aussi courtes que possible. Le raccourcissement des interconnexions améliore sensiblement la fréquence de fonctionnement maximale sur FPGA et permet de réduire le nombre de ressources de routage. A titre d'exemple la directive RLOC « *Relative Location* » utilisable avec les outils Xilinx permet de spécifier un placement relatif des blocs de logique.

Sur un autre plan, le synthétiseur peu inférer certaines primitives automatiquement, une primitive est une fonction logique de base supportée nativement par la structure FPGA. Plus particulièrement il est intéressant que les buffers internes des routeurs utilisent les Block RAM [88] [89] FPGA, quand on sait que les buffers sont responsables de plus de 70% de la surface totale d'un réseau intégré l'importance d'une telle optimisation devient évident.

3.11 Conclusion

Dans ce chapitre nous avons présenté en détail le processus de prototypage FPGA d'un réseau intégré. Nous avons situé le prototypage FPGA par rapport aux autres solutions possibles, nous avons aussi souligné les avantages et les inconvénients de cette approche. Pour résumer, le prototypage sur plateformes reconfigurables constitue un bon compromis Observabilité/Cout/Vitesse. La principale limitation en est le manque de flexibilité. La deuxième contribution de ce travail est de remédier à cette limitation.

Dans la seconde partie de ce chapitre, nous avons présenté les deux scénarios de test pouvant être appliqués au NoCs. Bien que chacune des approches ait ses avantages et limitations et qu'elle soit plus ou moins adaptée à une situation donnée, l'approche de test distribuée assure un processus de prototypage rapide avec un maximum de flexibilité.

Dans la dernière partie de ce chapitre, nous avons proposé des architectures de générateurs et moniteurs de trafic. Ces composants hautement configurables sont très utiles pour la validation rapide des réseaux intégrés. Nous nous sommes attachés à proposer des

architectures minimales pour ces composants afin de maximiser les ressources disponibles pour l'implémentation du réseau lui-même.

Le chapitre suivant sera consacré au prototypage multi-FPGA : cette solution vise à augmenter la flexibilité et permettre de gérer des réseaux de très grandes tailles.

Chapitre 4 : Emulation Multi-FPGA des réseaux-sur-puce

Sommaire

4.1 Introduction

L'émulation multi-FPGA a pour principal but d'augmenter la capacité d'intégration des plateformes de prototypage existantes, et cela de façon flexible. La flexibilité est définie ici comme étant la capacité à s'adapter à divers systèmes à base de NoCs, quelle que soit leur taille. L'idée principale est donc d'allier les possibilités de reconfigurabilité des plateformes FPGA à leurs possibilités d'interconnexions afin d'augmenter les capacités d'intégration.

Les techniques d'émulation multi-FPGA ont été utilisées depuis déjà longtemps, ainsi plusieurs plateformes d'émulation multi-FPGA de systèmes ASIC sont actuellement disponibles. Nous citerons en particulier la série de plateformes HAPS (« ***H****igh-performances* ***A****SIC* ***P****rototyping* ***S****ystem* ») de chez HARDI [90] et les plateformes Zebu de EVE (eve-team.com). Les émulateurs logiques, qui se distinguent pourtant des plateformes de prototypages, utilisent eux aussi des architecture multi-FPGA. Les systèmes multi-FPGA existants ont tous pour points communs d'une part la généricité de leurs architectures, et d'autre part la nécessité de se conformer à une précision Cycle/Bit (CABA : « ***C****ycle* ***A****ccurate and* ***B****it* ***A****ccurate* »). Les concepteurs de ces architectures ne font aucune supposition quand à leur utilisation finale, c'est ce qui rend ses architectures très généralistes. Quand au niveau de précision CABA, c'est simplement le seul niveau de précision possible (et admissible) jusqu'à présent pour les systèmes d'émulation et de prototypage de bas niveau.

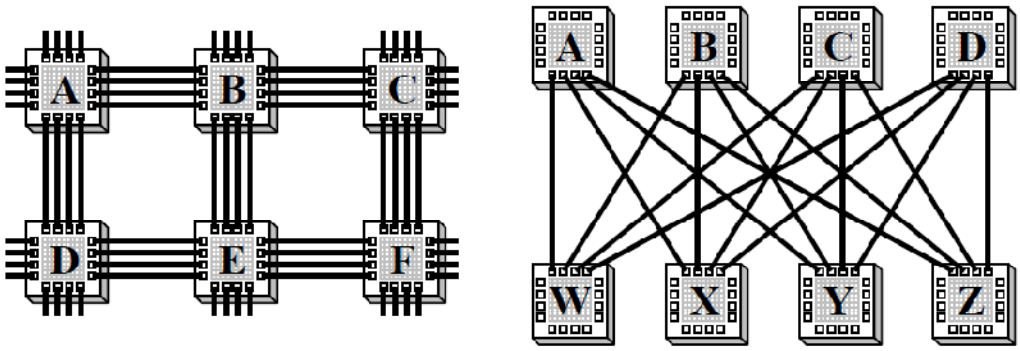
L'approche d'émulation multi-FPGA que nous proposons part du constat suivant : bien que les réseaux intégrés sont hyper sensibles aux variations des conditions de trafic, il subsiste des zones de fonctionnement où le comportement est relativement stable et prévisible, en particulier sous des conditions de trafic non congestionné. Il devient des lors possible d'admettre dans ces zones de fonctionnement certaines imprécisions dans l'estimation des performances (causées par exemple par la plateforme d'émulation elle même).

Dans ce chapitre nous faisons une proposition d'une plateforme Multi-FPGA d'émulation de NoC se basant sur le principe énoncé précédemment. Nous commencerons par introduire le principe et les problématiques liées au prototypage multi-FPGA en général, avant de détailler la plateforme et le flot de conception que nous proposons.

4.2 Principe

Le principe du prototypage multi-FPGA reste le même et ce la quel que soit l'usage projeté (Emulation, Prototypage, Calculateur parallèle). Il repose sur l'assemblage de plusieurs plateformes afin d'en construire une autre avec des capacités d'intégration plus importantes. Hormis leur utilisation, les architectures multi-FPGA se différencient principalement par la topologie d'interconnexion et la technologie des interconnexions.

Tout comme pour les NoCs, la topologie (FIGURE 4.1) définie les interconnexions entre les différents nœuds FPGA, et peut être de type Grille, Crossbar, réseau hiérarchique…etc. Les topologies les plus simples sont généralement privilégiées car moins encombrantes en termes d'interconnexions. Cependant il est tout à fait possible d'avoir un système multi-FPGA avec une topologie qui serait impossible pour un NoC (typiquement les topologies en 3D).



(a) (b)

FIGURE 4.1 - Exemple de topologies multi-FPGA (a) Grille et (b) Crossbar

Pour la technologie d'interconnexions, on peut avoir principalement deux types : des interconnexions séries et des interconnexions parallèles. Les interconnexions de type série présentent un certain nombre d'avantages par rapport aux interconnexions parallèles :

- Un encombrement moindre du fait du nombre de fils réduit
- Une sensibilité bien plus faible aux erreurs que les interconnexions parallèles (cross-talk, phénomène de biaisage de l'horloge…etc.)

L'encombrement des interconnexions joue un rôle déterminant sur la flexibilité de la plateforme : moins les interconnexions sont encombrantes plus grand sera le choix des

topologies réalisables, ou encore la taille maximale de la plateforme. Concernant les vitesses de transmission, celles-ci sont généralement plus élevées pour une transmission série car les phénomènes limitant la vitesse de transmission ont un impact bien moindre en mode série. Les interconnexions (séries ou parallèles) peuvent être soit intégrées au niveau PCB, dans ce cas-ci on est en présence d'une plateforme complètement intégrée sous forme d'une seule carte multi-FPGA [90]. Ou encore être des interconnexions entre plusieurs cartes, cette solution en plus de permettre des solutions complètement ad-hoc, offre une flexibilité bien meilleure.

Même avec des interconnexions intégrées sur le PCB offrant un volume de connectivité globalement plus élevé, il peut s'avérer nécessaire de disposer d'un volume encore plus important. Pour cela, une technique très largement employées consiste à multiplexer plusieurs *« fils »* intra-FPGA d'un système donné sur un seul lien inter-FPGA [91] [92]. Cette technique connue sous le nom de *« fils virtuels »* permet d'outrepasser les limitations d'interconnexions mais impose de recourir à des mécanismes de synchronisation induisant un surcoût non négligeable sur les vitesses d'émulation.

Sur une architecture multi-FPGA le circuit devant être émulé doit au préalable être partitionné en plusieurs sous-circuits chacun affecté à un circuit FPGA. Il est primordial que l'algorithme de partitionnement minimise (en termes de volume et de nombre) les interconnexions inter sous-systèmes pour des raisons évidentes de limitation de ces dernières. Ici le volume des interconnexions peut être exprimé par plusieurs métriques, ces métriques pouvant aussi bien refléter le nombre de fils physiques entre sous-système que la quantité de données entre sous-systèmes (débit).

Un circuit électronique est généralement organisé de façon hiérarchique : plusieurs modules fonctionnels sont combinés dans un module de rang plus élevé responsable d'une fonction plus complexe. Cette hiérarchisation permet de définir plusieurs niveaux de partitionnement dont les principaux sont : le niveau porte logique, le niveau fonction de base et le niveau module fonctionnel. Plus le niveau de partitionnement est proche des niveaux bas de la spécification du circuit plus les besoins en interconnexions sont élevés. Un partitionnement à un très bas niveau ajoute des contraintes additionnelles sur le processus d'émulation du aux besoins accrus en synchronisation.

4.3 Emulation multi-FPGA pour réseaux-sur-puce

4.3.1. Présentation de la technique

L'idée de base derrière l'approche d'émulation que nous proposons repose sur le constat qu'une précision absolue n'est pas toujours nécessaires lors de l'émulation d'un système MPSoC (et par extension les systèmes à base de NoCs). Alors que toutes les plateformes d'émulation actuellement existantes partent du principe de l'aliénation de la précision absolue, il est intéressant de noter que pour certains systèmes, et notamment les systèmes a base de NoCs, il est possible d'admettre que la plate-forme d'émulation introduise certaines variations locales dans le comportement du NoC sans que cela n'affecte de façon significative le comportement globale du système. Dans ce cas-ci les imprécisions sont principalement dues à deux causes :

- L'indépendance complète des sous systèmes FPGA (signaux d'horloge distincts, absence de synchronisation globale)
- Le mode d'interconnexion inter-FPGA série qui introduit des délais supplémentaires.

Dans le cas général, le comportement d'un NoC peut être influencé de deux façons : soit une modification (dégradation ou diminution) des performances réelles de façon linéaire, soit une modification chaotique (très complexe) des performances. Dans le premier cas une compréhension exacte des mécanismes du NoC et de la plateforme permet de formaliser les interactions mises en cause et par conséquent de déduire des performances réelles. Alors que dans le deuxième cas aucune méthode ne permet d'adopter la même démarche. Une autre caractéristique importante de notre approche réside dans le fait que c'est le système à émuler qui fixe l'architecture de la plateforme d'émulation et non l'inverse (approches classiques), ce qui justifie le besoins de flexibilité accrue.

Dans ce qui suit nous détaillerons les différentes étapes nécessaires à l'émulation d'un NoC sur notre plateforme. Avant cela nous donnons une définition formelle de la linéarité du comportement d'un NoC.

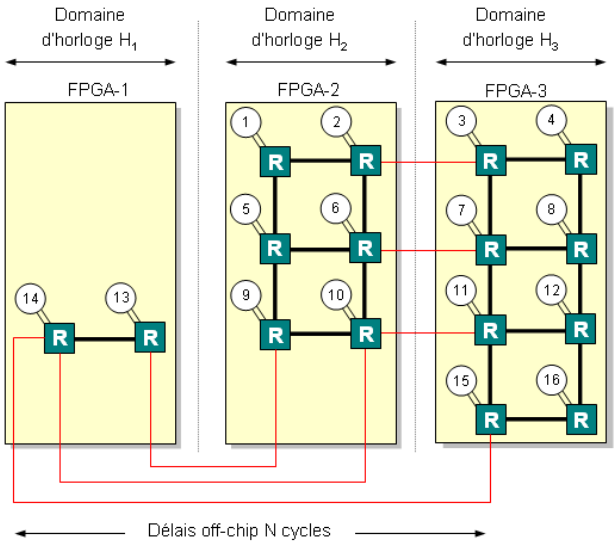


FIGURE 4.2 - Exemple d'une configuration multi-FPGA pour l'émulation d'un NoC

4.3.2. Caractérisation de la composante linéaire du comportement d'un NoC

Classiquement, un NoC présente deux zones de fonctionnement : une zone ou les performances temporelles globales (latence, charge acceptée, débit) évoluent de façon relativement linéaire par rapport à la charge injectée aux entrées du réseau, et une zone de fonctionnement saturé ou ces mêmes paramètres évoluent de façon non linéaire. La frontière entre les deux zones de fonctionnement matérialisée par le point de saturation dépend des caractéristiques physiques du NoC (distribution de l'espace de bufférisations, politique d'ordonnancement...etc.). Il est intéressant d'essayer de quantifier précisément l'impact de chacun des paramètres architecturaux d'un NoC sur sont comportement globale (performances en sortie) et l'impact d'une implémentation multi-FPGA sur ces mêmes paramètres. Nous supposerons dans ce qui suit que les métriques de performances en sorties à savoir la charge acceptée et la latence moyenne sont mesurées en fonction de la charge injectée en entrée (paquet/cycle). Il est clair qu'ici il faut considérer uniquement les paramètres pouvant être modifiés par la plateforme, car dans le cas présent nous essayons d'identifier les relations possibles entre la plateforme d'émulation et les performances temporelles du NoC. À titre d'exemple la topologie du réseau ou encore la politique d'ordonnancement ne sont pas

modifiés bien que le réseau soit partitionnée en plusieurs sous réseaux. Le tableau suivant résume les paramètres du NoCs pouvant être affectés par la plateforme d'émulation

TAB 4.1 Impact de l'émulation multi-FPGA sur les principaux paramètres architecturaux du NoC

Topologie	Inchangée
Politique d'ordonnancement	Inchangée
Politique de bufférisation	Inchangé
Fréquence de fonctionnement	Inchangé mais un décalage de phase inconnu existe entre les sous réseaux
Distribution de l'espace de bufférisation	Peut être altérée de façon indirecte
Caractéristiques temporelles des liens (latence, débit)	Inchangée sauf pour les liens inter-FPGA

L'examen des paramètres pouvant être impactés lors du passage à une implémentation multi-FPGA montre que seules les distributions de l'espace de bufférisation ainsi que les délais sur certains liens peuvent être touchés. Le déphasage de fréquence entre les différents sous-réseaux peut être considéré comme étant négligeable car son effet est infime par rapport à celui des délais inter-FPGA. Concernant la distribution de l'espace de bufférisation, il est important de bien noter que seul le point de saturation est impacté et non pas les latences moyennes ou instantanées pouvant être observées sur ce réseau ; cela est particulièrement vrai dans la zone de linéarité. En pratique, la transition vers le régime congestionné (point de saturation) peut être soit retardée soit avancée par rapport à sa valeur référence (correspondant à la distribution d'espace de bufférisation réelle) selon que l'espace de bufférisation diminue ou augmente. Les variations des délais de transport sur les liens ont quand à eux un impact plus direct à la fois sur les latences moyennes et sur le point de saturation ; les deux phénomènes vont être abordés plus en détails dans les paragraphes suivants.

4.3.3. Flot de synthèse d'émulateur multi-FPGA

Le flot global de synthèse d'émulateur multi-FPGA est tel que présenté dans la FIGURE 4.3. Comme nous l'avons vu précédemment, le principe général est basé sur un

partitionnement préalable du NoC en plusieurs sous réseaux où chacun est affecté à une plateforme FPGA. Comme la phase de partitionnement prends en compte le modèle de l'application visée par l'émulation; deux étapes additionnelles sont donc nécessaires : une étape de mappage et une étape d'estimation de la précision. Nous détaillerons dans ce qui suit chacune de ces étapes. Par rapport aux trois phases données en FIGURE 3.1, ces étapes se situent dans la phase de Validation.

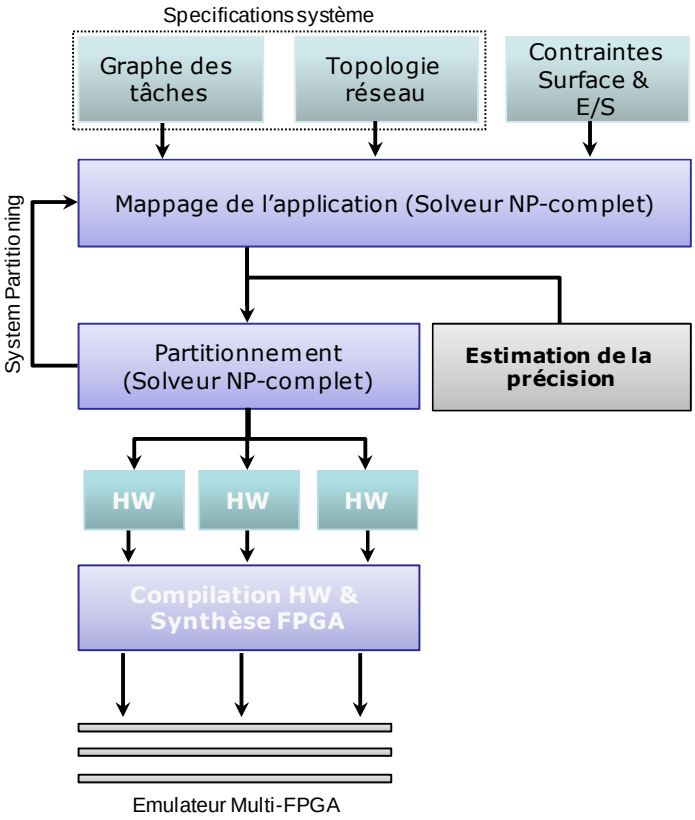


FIGURE 4.3 - Flot de synthèse d'émulateur multi-FPGA pour NoCs

4.3.3.1. Mappage de l'application logiciel sur le NoC

Le mappage de l'application logiciel sur le NoC vise à la prise en considération des caractéristiques de communication de l'application logicielle afin de maximiser la précision de la plateforme. Il peut légitimement sembler que cela puisse limiter le champ d'application de l'émulateur ainsi construit. Cependant il faut considérer aussi que les plateformes multi-

FPGA sont extrêmement flexibles et facilement configurables. De plus la représentation en graphe des tâches est une représentation assez générique permettant d'abstraire un large éventail d'applications par une seule représentation.

Le problème du mappage est tel que définit dans le paragraphe 3.5.2 dans le chapitre 3, c'est un problème NP-Complet où la solution est une fonction de correspondance entre les nœuds du graphe des taches et les nœuds du NoC. La solution obtenue est aussi représentée par un graphe orienté reprenant la structure du NoC, mais où les liens sont décorés par les besoins en communication. Il est à noter qu'il est tout à fait possible d'utiliser le graphe de tâche aux coûts unitaires afin de construire une plateforme complètement neutre par rapport à l'application logicielle.

Pour notre implémentation nous avons utilisé le solveur Open Source [93] que nous avons adapté. Ce solveur part d'une description textuelle du graphe des tâches (FIGURE 4.4) et du NoC et donne en sortie le mappage optimal.

```
16, // Nombre de nœuds dans le NoC (Grille 4x4)

// Graphe des tâches T[i,j] = comm entre les nœuds i et j
0  2  2  2
3  0  0  0
0  1  0  0
13 14 15 16,
// Représentation du NoC R[i,j] = distance entre les nœuds i et j
0  1  2  3
1  0  1  2
2  1  0  1
3  2  1  0
```

FIGURE 4.4 - Exemple de fichier de description de l'application et du NoC

4.3.3.2. Partitionnement de réseaux sur puce en vue de l'émulation multi-FPGA

La phase de partitionnement consiste à trouver un ensemble de sous-réseaux partant du réseau initial en respectant un certain nombre de contraintes. Les contraintes sont principalement destinées à optimiser la précision de la plateforme et à s'assurer que les

ressources FPGA sont suffisantes pour un partitionnement donné. Le solveur de partitionnement prend en entrée le graphe issu de la phase de mappage. Pour rappel ce graphe en plus de refléter la topologie du NoC contient des informations sur les besoins en communications (FIGURE 4.5).

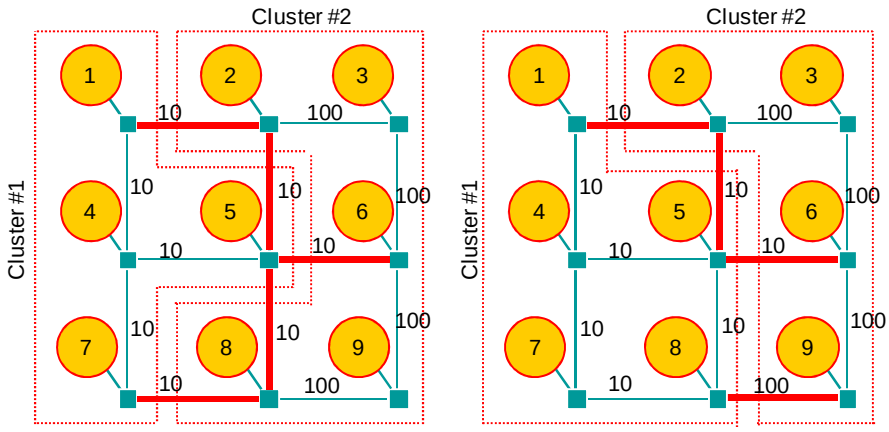


FIGURE 4.5 - Exemples de partitionnements de NoC avec informations sur les communications

Deux contraintes doivent principalement être considérées durant la phase de partitionnement :

1. Chaque sous réseau doit pouvoir être implémenté sur une plateforme FPGA, i.e que les ressources nécessaires à son implémentation ne doivent pas dépasser le maximum de ressources disponibles sur une plateforme.
2. Le volume total de communications inter sous-réseaux doit être minimisé.

Une définition formelle du problème de partitionnement de NoC sous ces contraintes est donnée ci-après. Nous noterons que dans cette formulation le volume des communications réfère soit à la charge de trafic soit au nombre de connexions total inter-clusters. La première notion est à utiliser lorsque des informations relatives au trafic sont disponibles, sinon c'est la deuxième notion qui doit être utilisée et dans ce cas-là on considère uniquement le nombre de liens.

Définition i : *Etant donné un graphe G(V,E) orienté et valué, où V représente l'ensemble des nœuds, et E l'ensemble des liens inter-nœuds. $W = \{w_1,w_2,\ldots w_n\}$ les volumes de communication associés a chacun des liens. $Z = \{z_1,z_2,\ldots,z_m\}$ les coût en surface (ressources matérielles) associés au nœuds du réseau. Le problème de partitionnement de réseau revient à trouver K partitions P_i $(0<i\leq K)$ de G, tel que :*

1. *Le nombre totale de lien inter partitions soit minimal (respectivement le volume total de trafic)*
2. *Pour une partition P_i : $\sum z_t .D_t$ (D_t =1 si $v_t \in P_i$)$\leq$ Ressources disponible dans la plateforme associée*

Essayer de minimiser le nombre de liens (ou volume de trafic) inter partitions trouve sa justification dans les latences accrues de tels liens, phénomène qui est la cause principale de la perte de précision. En effet, en l'état actuel il n'est pas possible d'assurer des interconnexions inter-FPGA avec des performances physiques et temporelles comparables à celles des liens on-chip. Par la suite nous introduirons plus en détails cette notion de précision de l'émulation, notion qui sera ensuite utilisée pour caractériser et mesurer les performances de notre plateforme d'émulation multi-FPGA. D'autre part il est clair que selon la topologie du NoC considéré, il sera plus ou moins facile d'arriver à trouver un partitionnement qui satisfasse les contraintes précédemment énoncées. Cependant, les NoCs étant par définition des structures régulières et simples, le partitionnement s'en trouve simplifié d'autant.

Enfin, le problème de partitionnement de NoC pouvant être directement ramené au problème de partitionnement de graphes, dans notre implémentation nous avons fait usage d'un solveur déjà existant METIS [94]. Ce solveur présente la particularité d'être très bien adapté au graphe présentant un grand nombre de nœuds, ce qui dans notre cas permettra de traiter des réseaux de très grandes tailles.

4.3.3.3. Estimation de la précision

La précision de l'émulation peut être définie comme étant la différence entre le comportement du NoC respectivement lors d'une émulation mono et multi-FPGA. Le comportement du NoC est caractérisé par l'ensemble des métriques de performances (rapport

de charge acceptée, latences moyennes, latences point-à-point), d'autres métriques pouvant être considérées. Dans le cas présent, la difficulté pour mesurer la précision de la plateforme d'émulation vient de plusieurs facteurs :

1. Il n'est pas imaginable de faire d'un coté une implémentation mono-FPGA et de l'autre une implémentation multi-FPGA et en déduire ensuite la précision. Les réseaux considérés étant par définition trop larges pour un seul FPGA ;
2. Les conditions de trafic pouvant être très variables, il devient donc très difficile de donner une valeur exacte à la précision, valeur qui en dépend très fortement.

Le problème de mesure de la précision peut être abordé selon deux points de vue : le premier en considérant que les étapes de partitionnement et de mappage sont suffisamment fiables pour assurer la précision souhaitée (dans un certain intervalle de tolérance), ou bien en identifiant les scénarios de trafic les plus problématiques, en faisant des simulations et en estimant à chaque fois la précision de la plateforme.

Dans toute l'étude qui est faite dans ce chapitre nous avons considéré un seul modèle de trafic, modèle dont nous avons fait varier les paramètres selon une loi de distribution uniforme. Il n'est pas nécessaire d'avoir recours à des distributions de probabilités plus avancées car comme nous le montrerons plus tard la précision dépend d'autres facteurs.

Afin d'être en mesure de faire des simulations précises, nous avons développé des modèles SystemC précis au cycle près, pour les interfaces et liens inter-FPGA. Pour un NoC donné il faudrait donc disposer d'un modèle de simulation SystemC (ou VHDL). Il devient dès lors possible de comparer les performances mono et multi-FPGA et en déduire le taux de précision sous des conditions de trafic données.

4.4 Expérimentations

Dans cette partie du chapitre nous allons considérer une étude de cas réelle qui a été menée, ceci afin de montrer les étapes à suivre lors d'une implémentation d'un émulateur multi-FPGA. Nous allons commencer par une présentation rapide du réseau ainsi que la plateforme FPGA ayant été considérés lors de ces expérimentations. Nous exposerons ensuite les conditions de test ainsi que la précision ayant été observée.

4.4.1 La carte XUPV2PRO

La carte XUPV2Pro est une plateforme de prototypage FPGA conçue autour de la puce FPGA Virtex II Pro de Xilinx [95], en plus de cela elle inclut plusieurs périphériques tel que illustré dans la FIGURE 4.6. Dans le cas présent deux éléments de la plateforme sont très importants : La puce FPGA Virtex-II-Pro [96] et les connecteurs série RocketIO [97].

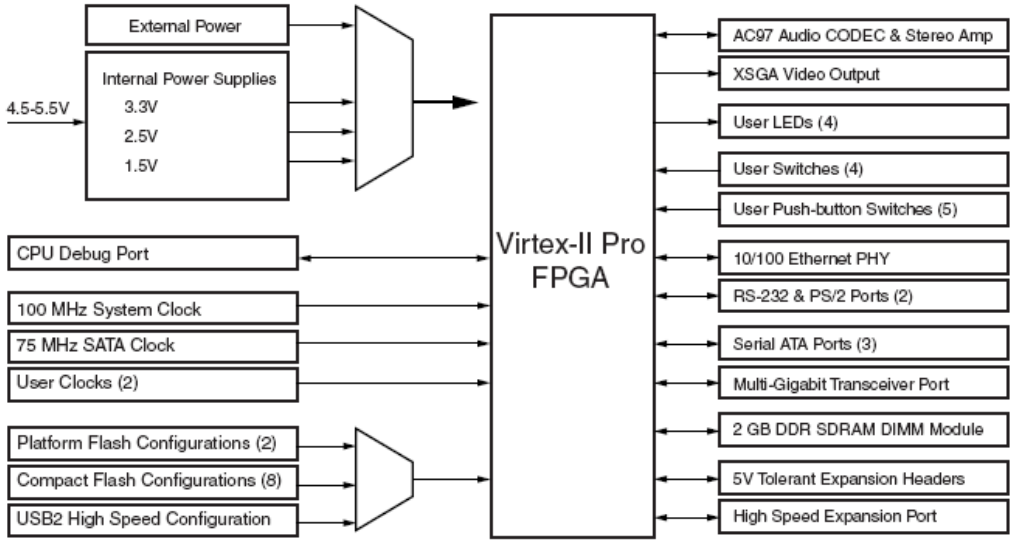


FIGURE 4.6 - Diagramme en bloc de plateforme FPGA XUP

En plus de la logique reconfigurable, la puce VirtexII-Pro XC2VP30 inclut deux processeurs PowerPC405 [98], ces processeurs peuvent être utilisés aussi bien pour la génération de trafic sur le NoC que pour le contrôle du processus d'émulation (voir chapitre 3). En termes de capacités d'intégration, il est difficile de donner une estimation précise du nombre de portes ASIC équivalent (≈ 1,5 M) car cela dépend grandement du système considéré ainsi que des outils utilisés, cependant à titre d'illustration il nous a été possible d'émuler des systèmes contenant jusqu'à quatre processeurs µBlaze [99] avec toute la hiérarchie mémoire et les périphériques d'E/S nécessaires. Concernant les capacités d'intégration relatives aux NoCs il est tout aussi difficile de donner une estimation car la variété des topologies et des microarchitectures de routeurs rendent cela très difficiles, néanmoins nous donnons dans ce qui suit les ressources disponibles sur la XC2VP30 tel

qu'énoncé par Xilinx et les ressources nécessaires à l'implémentation d'un seul routeur (FIGURE 4.7).

TAB 4.2 Ressources FPGA de la puce XC2VP30

Ressources	XC2VP30
Slices	13969
Distributed RAM	428 Kb
Multiplier Blocks	136
Block RAMs	2448 Kb
DCMs (Digital clock manager)	8
PowerPC RISC Cores	2
Multi-Gigabit Transceivers	8

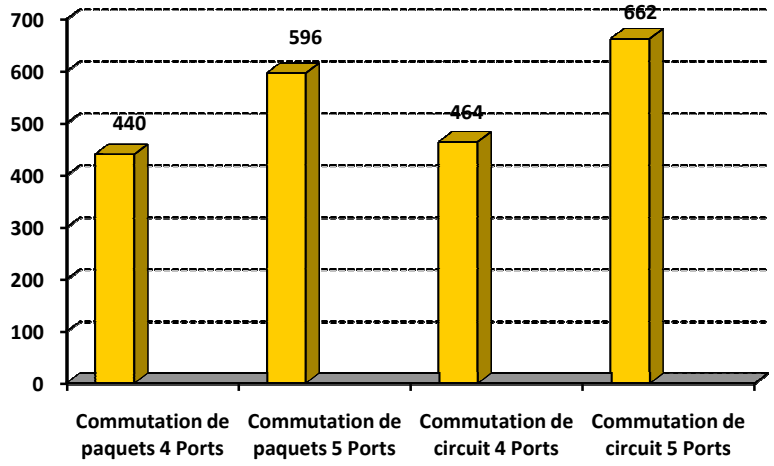


FIGURE 4.7 - Exemple d'utilisation de ressources (Slices) FPGA pour plusieurs configurations d'un routeur de NoC

Le dernier point important concernant la plateforme utilisée est relatif aux capacités d'interconnexions série. Comme indiqué dans le tableau 4.2 il est possible d'utiliser simultanément jusqu'à 8 interconnexions série bidirectionnelles (RocketIO) dont le débit maximal peut atteindre 3.125 Gb/s. Ce nombre élevé d'interconnexions haute vitesse rend possible la construction de plateformes multi-FPGA très variées de part leurs topologies et leurs schéma d'interconnexions.

4.4.2 Conception des interfaces inter FPGA

Le rôle des interfaces inter-FPGA consiste principalement à faire une adaptation du protocole on-chip utilisé en interne par le NoC au protocole off-chip utilisé lui par les interfaces série. La notion de protocole englobe ici tous les mécanismes liés aux échanges de données à savoir : transferts de paquet et contrôle de flux. Une interface inter-FPGA doit assurer la communication transparente entre deux routeurs situés sur deux cartes FPGA distinctes, pour cela elle dispose de deux ports de communication d'un coté un port compatible NoC, et de l'autre un port compatible au lien série utilisé (RocketIO dans notre cas) cf FIGURE 4.8.

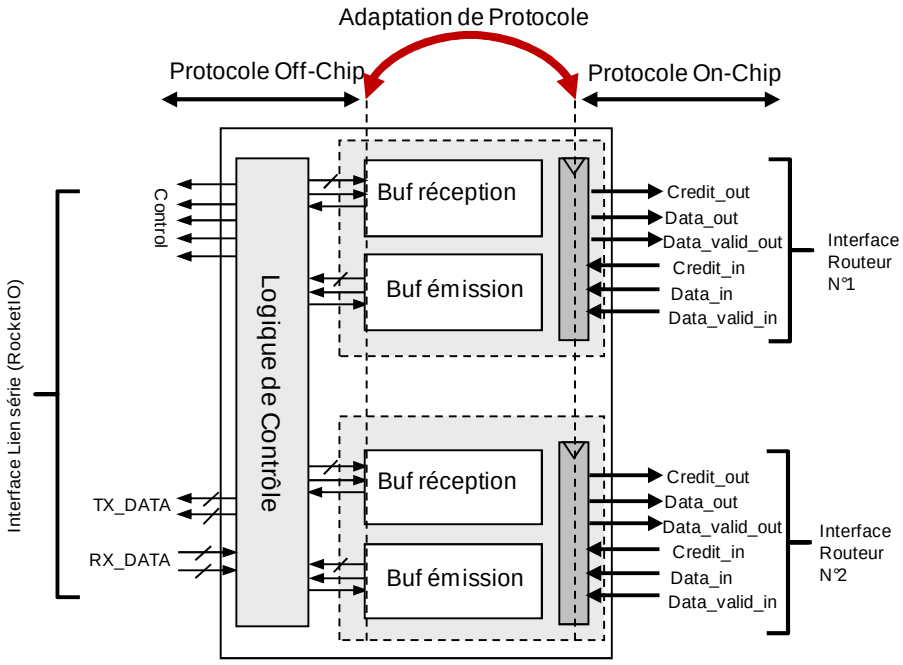


FIGURE 4.8 - Détail de l'architecture interne des interfaces inter-FPGA

Comme on peut le voir sur la description d'architecture donnée ici, nous avons opté pour une interface à deux ports on-chip (compatible NoC), c'est-à-dire qu'un seul lien off-chip peut être partagé par deux liens on-chip. Cette organisation permet d'augmenter virtuellement le nombre de liens séries disponibles en induisant cependant une perte de précision liée à la charge de trafic sur le NoC.

Il a été aussi nécessaire d'introduire des buffers intermédiaires dans les deux sens Emission/Réception. Ces buffers s'avèrent être indispensables afin de garantir des transferts sans pertes de paquets. En effet, pour certaines configurations des différences significatives peuvent exister entre les vitesses de communication on-chip et off-chip d'où la nécessité de sauvegarder les paquets temporairement avant de les envoyer sur les liens off-chip. La logique de contrôle quant à elle est une machine à états qui assure en même temps l'arbitrage entre les deux interfaces de routeurs et l'adaptation au protocole série. Les interfaces série utilisées ici sont les interfaces RocketIO, ces interfaces peuvent être configurées pour fonctionner sous plusieurs modes chacun avec une vitesse qui lui est propre :

- Mode *Fibre Channel* à **1.06-2.12 Gb/s**
- Mode *Gbit Ethernet* à **1.25 Gb/s**
- Mode *PCI Express* à **2.5 Gb/s**
- Mode *XAUI* (10-Gbit Ethernet) à **3.125 Gb/s**
- Mode *XAUI* (10-Gbit Fibre Channel) à **3.1875 Gb/s**
- Mode *Infiniband* à **2.5 Gb/s**
- Mode *Aurora* à **0.6-3.125 Gb/s**
- Mode *utilisateur* à **0.6-3.125 Gb/s**

Le mode *Aurora* [100] est un mode particulièrement intéressant d'une par ce qu'il permet d'atteindre le débit maximal possible par un RocketIO (3.125Gb/S) sur le XC2VP30, d'autre part parce que Xilinx fournit une IP de pilotage des RocketIO en ce mode particulièrement optimisée et simple d'utilisation. Nous avons donc retenu ce mode de fonctionnement pour notre implémentation des interfaces inter-FPGA. Cela ne limite en rien la généricité de l'approche que nous proposons. L'IP Aurora vient s'intercaler entre l'interface décrite dans la FIGURE 4.8 et le RocketIO. La logique de contrôle s'en trouve grandement simplifié car au lieu de piloter directement le RocketIO elle envoi de simples commandes à l'IP Aurora pour l'envoi et la réception de paquets.

Concernant les latences de transfert il faut savoir que l'IP Aurora supporte deux modes de fonctionnement : Un mode appelé *« Framing »* où les échanges se font par trames de tailles variables, et un mode plus simple appelé *« Streaming »*. Le Mode *Streaming* bien que

n'offrant pas toutes les possibilités du mode *Framing* en termes de gestion et encapsulation des données répond très bien à nos besoins, de plus il offre de meilleures performances temporelles.

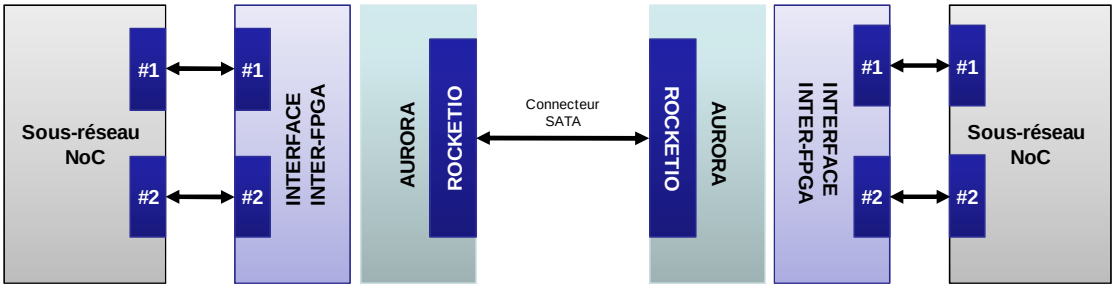


FIGURE 4.9 - Utilisation de l'IP Aurora avec les interfaces inter-FPGA

En mode *Streaming* les communications (FIGURE 4.10) entre les interfaces inter-FPGA et Aurora se font au travers de quatre services donnés comme suit :

a. Initialisation

Ce service réfère à l'état des liens série, il est très important afin d'être en mesure de déterminer si tous les liens sont prêts afin de commencer le processus d'émulation. Le Signal **CHANEL_UP** indique que le lien série concerné à été correctement initialisé quand il est dans l'état haut.

b. Transfert de données

En mode streaming les données sont envoyées sur le bus **TX_D[N :0]** et sont validées par le signal **TX_SRC_RDY_N** . En réception les signaux **RX_D[N :0]** et **RX_SRC_RDY_N** doivent être utilisés.

c. Gestion des erreurs

Bien que les RocketIO offrent des mécanismes de contrôle et de corrections des erreurs CRC, les rapports de tests d'intégrité [101] montrent que ce type de mécanismes reste superflu même en utilisant le débit maximal. Cet aspect peut donc être ignoré dans le cadre du prototypage qui nous intéresse.

d. Contrôle de flux

En mode streaming le signal **TX_DST_RDY_N** indique s'il est possible d'émettre, cependant l'état du récepteur (IP Aurora) est seulement lié à l'état de ses buffers internes (buffers élastiques utilisés pour l'adaptation en fréquence par les RocketIO), autrement dit c'est un contrôle de flux de niveau physique. Dans notre implémentation et afin de garantir des transferts sans pertes avec un minimum de logique matérielle nous avons surdimensionné les buffers des interfaces inter-FPGA par rapport à ceux utilisés en interne dans le NoC, cela afin de garantir que ces buffers ne débordent jamais quelles que soit les conditions de trafic. Un débordement de buffer (pouvant se produire avec une probabilité proche du zéro mais non nulle) qu'il soit en réception ou en émission entraine un arrêt immédiat du processus d'émulation et est signalé à l'utilisateur (voyant sur la carte FPGA). Durant toutes les expérimentations menées ce cas de figure ne s'est jamais présenté.

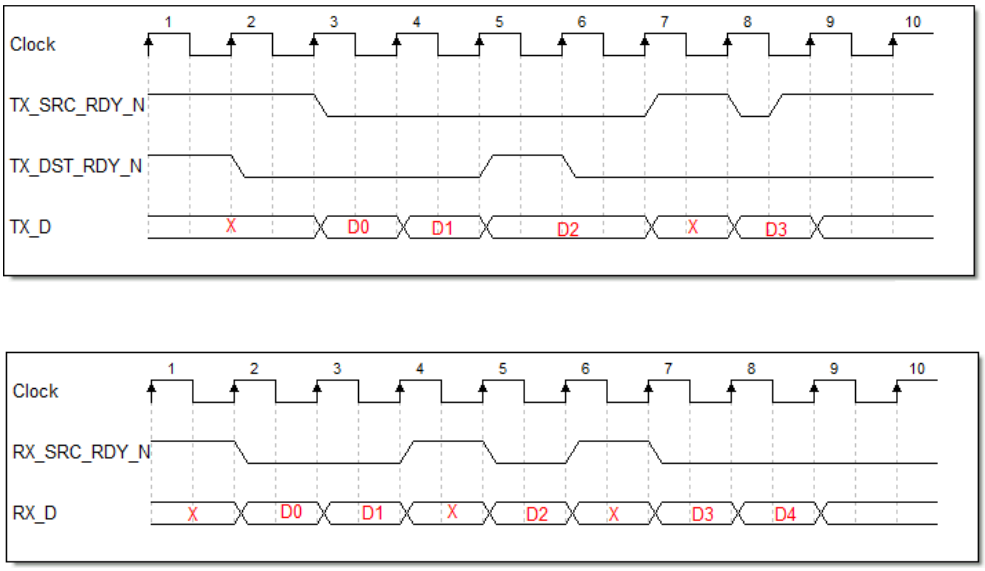


FIGURE 4.10 - Chronogrammes temporelles d'envoi et de réception de données avec l'IP Aurora

TAB 4.3 Ressources FPGA pour un adaptateur de protocole

Ressources	**Utilisation**	**% des ressources totales**
Slices	890	6%
Slice Flip Flops	465	1%
LUTs 4 entrées	1400	5%

4.4.3 Modèle de simulation SystemC de la plateforme

L'estimation de la précision de la plateforme d'émulation multi-FPGA nécessite de faire appel à des simulations logicielles. En effet, par définition le recours à une plateforme multi-FPGA est nécessaire si le réseau-sur-puce ne peut être émulé sur une seule carte FPGA, or justement une estimation des performances réelles (sur une seule carte donc) est nécessaire afin de mesurer la précision de la plateforme multi-FPGA. Toutefois le recours à une simulation n'est nécessaire que si l'on désire caractériser de façon très fine la précision. Des modèles de simulation CABA en SystemC [102] ont été donc développés pour les interfaces inter-FPGA ainsi qu'un modèle générique de routeur réseau à commutation de paquets.

Le modèle des interfaces inter-FPGA reprend l'architecture donnée dans le paragraphe précédent. Les tailles respectives des buffers internes ainsi que le rapport entre les fréquences d'horloge (fonctionnement interne/sérialiseur) sont paramétrables.

L'architecture du routeur est telle que montré dans la FIGURE 4.11. Bien qu'elle reprenne l'architecture du routeur Hermes [38][103], cette modélisation peut être facilement adaptée à une autre architecture à commutation de paquets, notamment les mécanismes de bufférisation (ici on utilise le Wormhole) et d'arbitrage (ici Round-Robin) qui sont représentés dans le modèle de simulation chacun par un module distinct. Enfin, des modèles de générateurs et de récepteurs de trafic ont été développés, ces modèles sont identiques aux modèle VHDL décrits dans le chapitre précédent et utilisent le protocole des interfaces réseau du NoC Hermes.

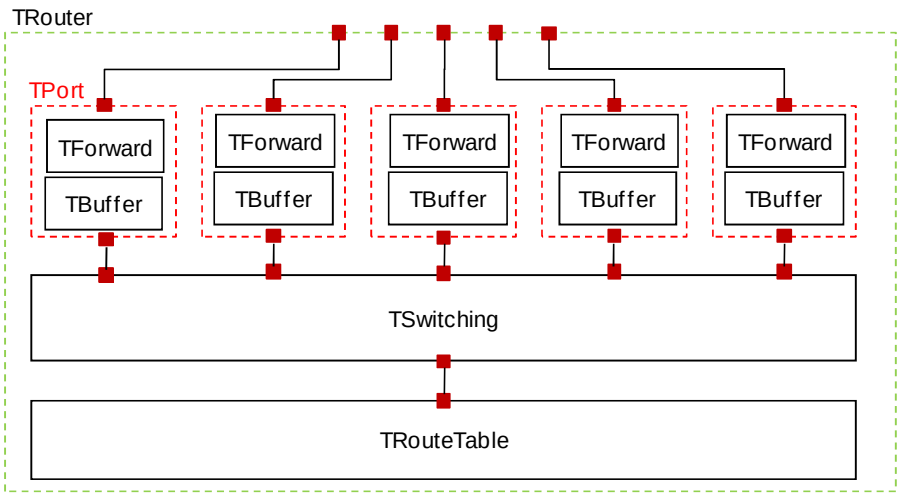


FIGURE 4.11 - Architecture SystemC du routeur générique

4.4.4 Interface lien du réseau STNoC sur carte XUP Virtex2Pro

Ces expérimentations ont été menées dans le cadre d'une collaboration avec le laboratoire AST STMicroelectronics de Grenoble. Elles avaient pour but l'étude de l'intégration des liens séries dans le réseau STNoC [28]. Seul un modèle de l'interface *« Link »* du réseau nous a été fourni (cf FIGURE 2.12 (c)), ce qui nous a permis de faire uniquement une validation fonctionnelle du réseau sur la plateforme.

L'exécution de l'interface link sur la plateforme a requis à deux cartes FPGA XUPV2PRO, ainsi que des adaptations mineurs sur les interfaces inter-FPGA afin d'assurer une compatibilité avec les liens STNoC. Les interfaces inter-FPGA de part et d'autre de la plateforme ont été instrumentées avec des moniteurs Chipscope Pro [73] ainsi que des moniteurs de protocoles que nous avons développés. Les moniteurs (analyseurs logiques) Chipscope servent à enregistrer et puis visualiser les requêtes aux entrés des interfaces. Les moniteurs de protocoles développés ici permettent de vérifier en temps réel si la séquence des paquets reçus suit bien le modèle attendu, cela afin de détecter d'éventuelles erreurs de séquencement qui pourraient être dues soit à une violation de protocole ou à une perte de paquets sur les liens série.

La FIGURE 4.12 montre l'architecture de l'émulateur double FPGA pour l'interface lien du STNoC. Les adaptateurs de protocoles utilisent ici aussi l'IP Aurora [100] déjà présentée, et permettent de connecter chacun deux générateurs de trafic. Les analyseurs logiques sont greffés à la paire TG_1 TR_1. Dans cette configuration chaque générateur de trafic TG_*i* adresse son trafic au récepteur TR_*i* du même rang *i* situé sur l'autre carte FPGA.

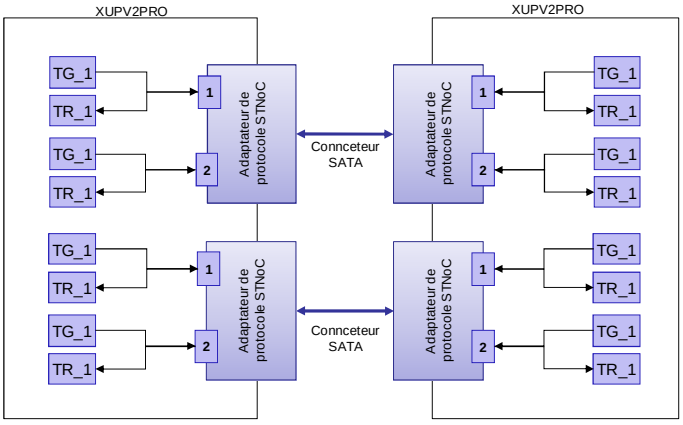


FIGURE 4.12 - Emulateur double-FPGA pour l'interface lien du STNoC

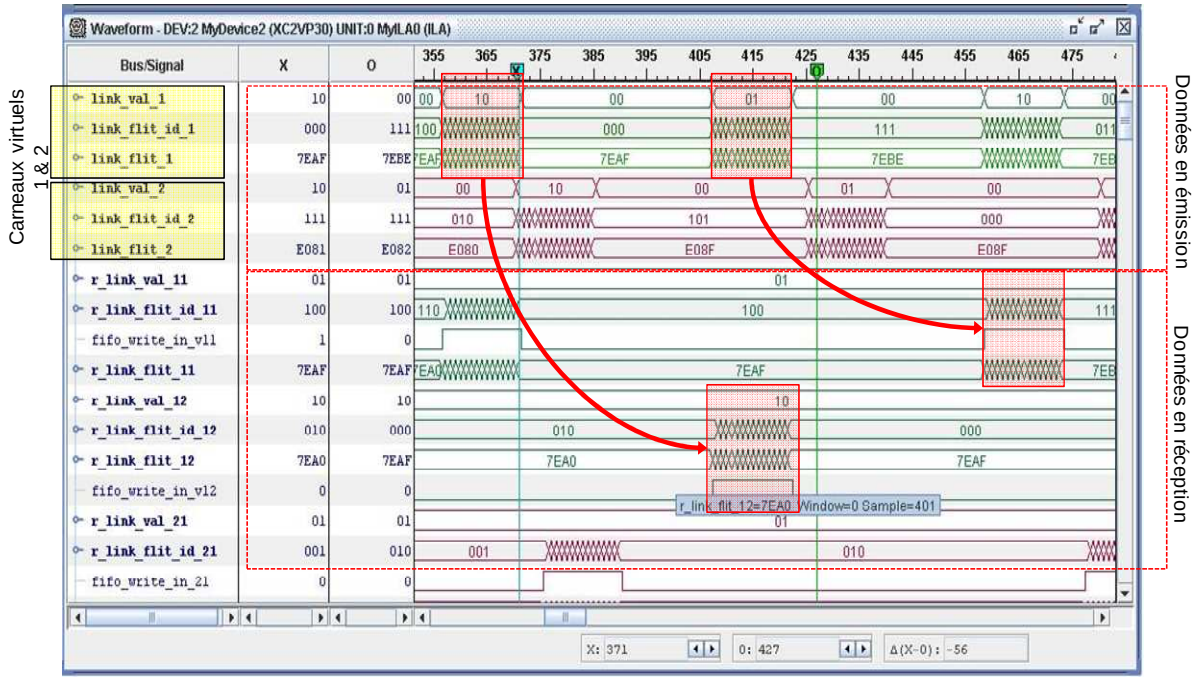


FIGURE 4.13 - Trace « *in-circuit* » pour l'émulation de l'interface lien du STNoC

La FIGURE 4.13 montre une trace d'exécution sur la plateforme multi-FPGA, il est à noter ici que les donnée en émission sont celles du générateur de trafic local et non pas celles du générateurs distant. Chipscope ne permettant pas de se connecter à deux chaines JTAG différentes en même temps. Cependant les générateurs fonctionnant de façon identique et symétrique d'une part et l'autre les données restent représentatives.

4.4.5 Le réseau Hermes sur cartes XUP Virtex2Pro

4.4.5.1 Le réseau Hermes

Les caractéristiques architecturales de ce réseau ont déjà été présentées dans le chapitre II, nous nous contenterons ici de présenter l'instance ayant été utilisée lors des expérimentations (FIGURE 4.14).

Nous avons considéré une instance carrée ayant 16 (4x4) routeurs, où chacun des routeurs est connecté à un générateur de trafic. Nous rappellerons ici que le but de ces expérimentations est de donner une estimation de la précision de l'émulation multi-FPGA dans le cas général, ce qui explique le recours à des générateurs de trafic stochastiques. Cette

taille de réseau à été spécifiquement choisie car elle est suffisamment grande pour que les résultats expérimentaux soient significatifs, et d'autre part les cartes FPGA à notre disposition permettent de construire l'émulateur approprié.

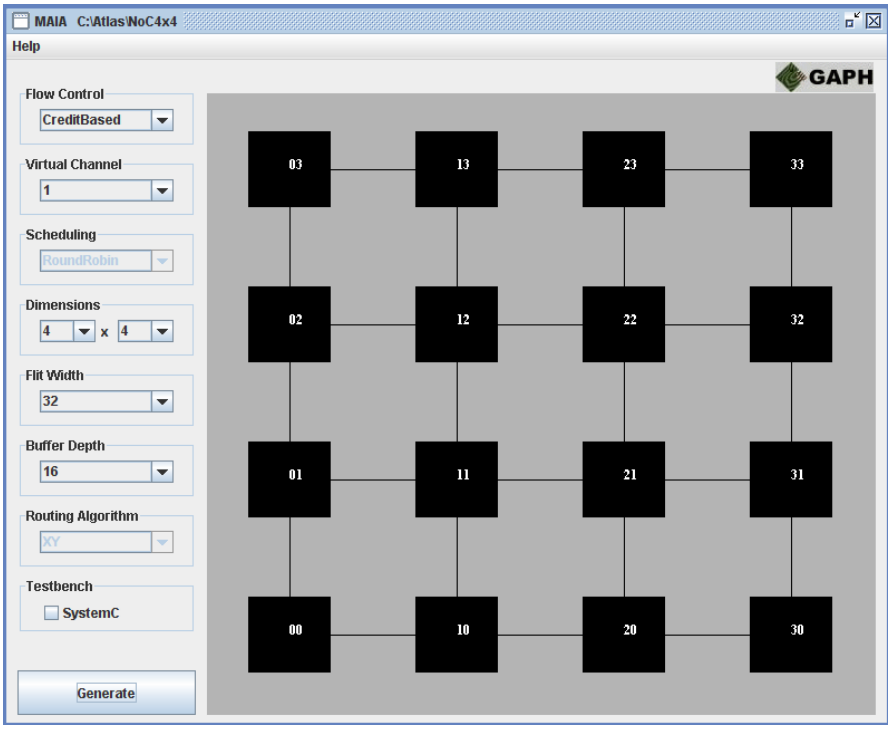


FIGURE 4.14 - NoC Hermes 4X4 routeurs

4.4.5.2 Conditions de trafic et partitionnement du réseau

Le but principal de ces expérimentations est double. D'une part faire une démonstration de l'approche que nous proposons et d'autre part essayer d'avoir une idée sur les déperditions de précision pouvant être occasionnées. Partant de cela, le modèle de trafic ayant été retenu est un modèle uniforme *«All-to-All »*. Un trafic uniforme permet d'avoir une bonne idée sur les imprécisions d'émulation, car les interconnexions off-chip (obtenues après partitionnement et principales cause des imprécisions) sont particulièrement stressées et supportent une grande partie du trafic total. Le flot de synthèse essaie justement d'éviter de telles conditions extrêmes de trafic. Un trafic de distribution uniforme donne un contrôle plus fin sur le trafic généré et permet ainsi de lier plus facilement telle condition de trafic à tel état du réseau.

Les générateurs de trafic utilisés ont leurs tables d'adressage initialisées selon le modèle de trafic uniforme ; pour un générateur k toutes les entrées de la table d'adressage sont initialisées avec la même valeur excepté l'entrée k qui est mise à zéro. Les tailles de paquets, intervalle inter-paquets sont eux de valeurs fixes (16 flits de 32 bits). Les conditions de trafic uniformes sur l'instance de réseau considéré, donnent une topologie où tous les liens supportent exactement la même charge de trafics. Dans un premier temps la plateforme est construite avec seulement deux cartes FPGA ; le partitionnement donne alors deux sous partitions équivalentes comme illustré sur la FIGURE 4.15.

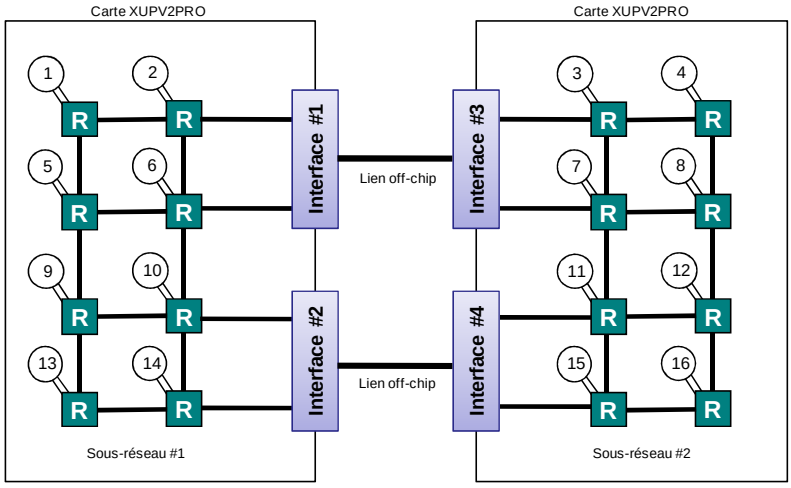


FIGURE 4.15 - Emulateur double-FPGA pour NoC Hermes

4.4.5.3 Estimation de la précision

Nous donnons dans la FIGURE 4.16 les résultats de latences obtenus pour une implémentation mono-FPGA et une implémentation multi-FPGA. Ces résultats ont été obtenus par simulation de toute la plateforme en utilisant la modélisation SystemC déjà présentée dans ce chapitre. Les moniteurs de performances développés dans le cadre de ce travail ne permettent pas de calculer de façon simple les latences moyennes ainsi que les charges sur tous le réseau. Cependant ils permettent de vérifier si des l'intégrité du flot de paquets reçus contre d'éventuelles erreurs.

Une analyse préliminaire de ces résultats montre deux phénomènes très intéressants. Le premier est la relative stabilité de la latence multi-FPGA (biaisée) par rapport à la latence mono-FPGA dans la zone avant la saturation du réseau (FIGURE 4.17). Le second phénomène de moindre importance concerne la différence entre latence réelle et latence émulée qui indique un degré de précision qui reste acceptable même sans l'application de techniques de corrections (erreurs d'imprécision maximale ≤ 30% FIGURE 4.18). La stabilité dans la zone de linéarité est importante à plus d'un titre car ce comportement rends possible l'application de techniques de corrections de la latence, et permet de prédire avec une marge très réduite les performances réelles du NoC. Concernant la charge acceptée par le réseau (FIGURE 4.19 et FIGURE 4.18), nous notons que l'erreur reste dans un intervalle lui aussi très réduit ±5%, cela se traduirait pour une application émulée sur notre plateforme, par un comportement quasi identique à celui d'une émulation mono-FPGA. Ces résultats expérimentaux montrent que dans sa zone de linéarité le NoC peut supporter certaines modifications introduites par la plateforme d'émulation sans que son comportement ne change de façon radicale ce qui rend possible la mesure de performance avec une précisons relativement bonne.

Il peut paraitre excessif d'affirmer qu'une précision de l'ordre de 30% soit acceptable, cependant cela est dû aux conditions sous lesquelles les tests ont été réalisés. En effet, un trafic uniforme est un trafic soutenu tout au long du processus d'émulation ce qui cause un phénomène d'accumulation des erreurs. Cela contraste avec ce qu'on peut rencontrer dans la majorité des applications réelles où des phases de trafic soutenu alternent avec d'autres phases de trafic plus léger. Partant de cette observation les valeurs données correspondent à des valeurs d'erreurs maximales difficilement atteignables avec des applications réelles. Un autre paramètre à prendre en compte pour l'interprétation de ces résultats consiste en la quantité de trafic passant sur les liens off-chip, dans le cas présent, c'est presque 50% du trafic global qui passe par les liens off-chip. Pour une application réelle la proportion de trafic off-chip sera beaucoup moins importante.

Le scénario de test ici à été choisi pour donner une bonne estimation des erreurs d'imprécision maximales pouvant être introduites par notre plateforme. Cependant la précision peut s'avérer être insuffisante pour certaines applications. Dès lors le recours à des

méthodes de correction des performances devient nécessaire. Dans le paragraphe suivant nous exposerons quelques techniques pouvant être utilisées.

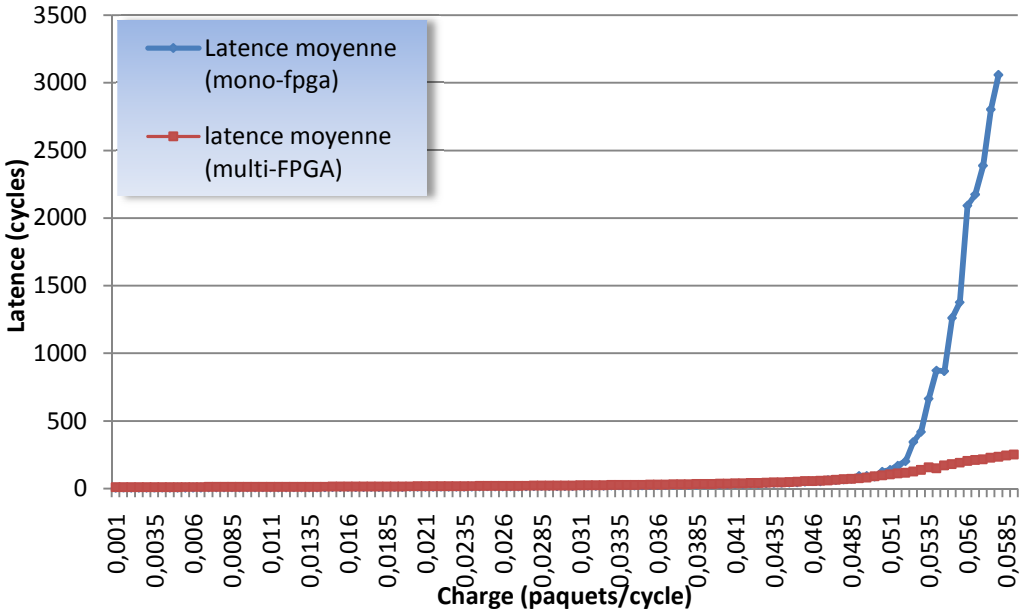


FIGURE 4.16 - Evolution globale de la latence mesurée en émulation mono et multi-FPGA

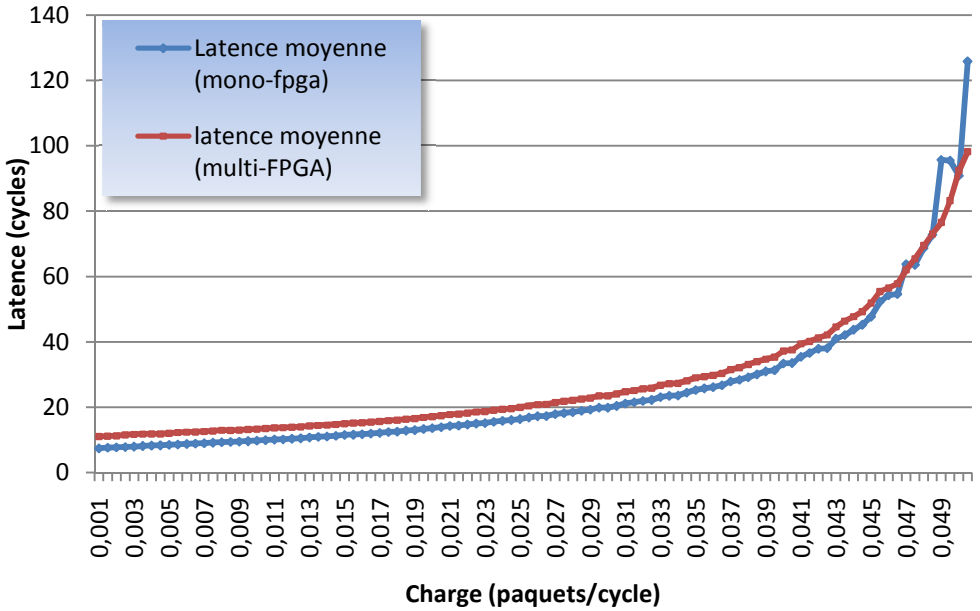


FIGURE 4.17 - Détail sur la zone de fonctionnement non congestionné

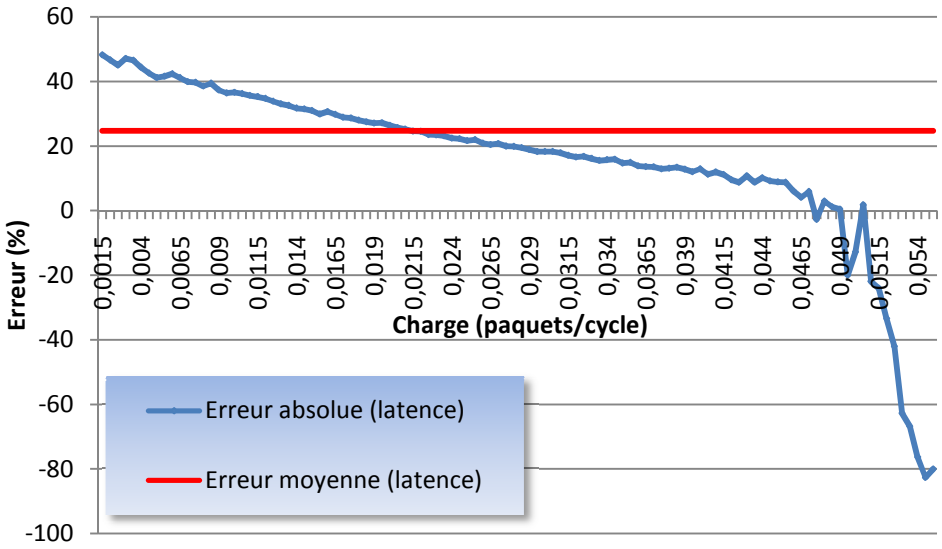


FIGURE 4.18 - Estimation de l'erreur d'émulation (latence)

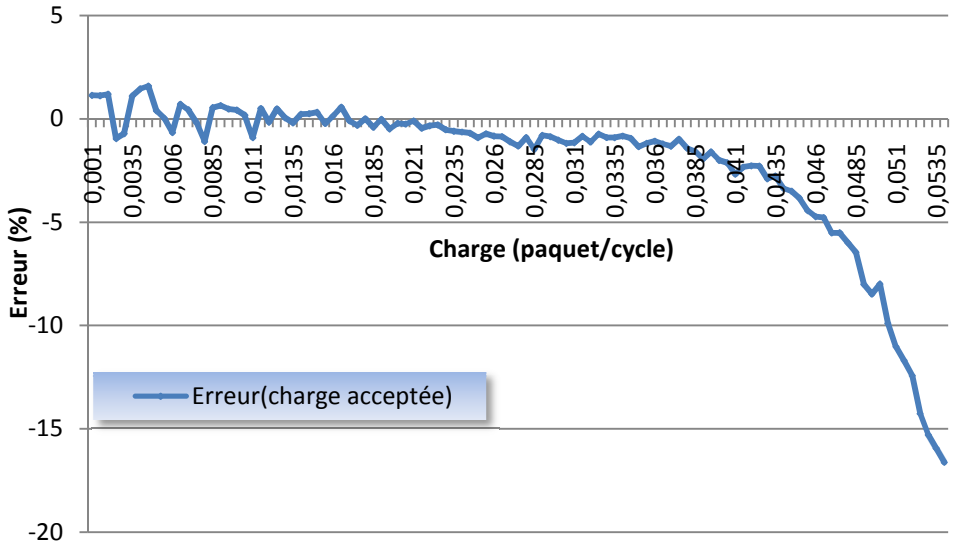


FIGURE 4.19 - Estimation de l'erreur d'émulation (charge acceptée)

4.5 Techniques de correction des mesures de performances

La correction de la latence peut se faire de deux manières différentes : soit faire une émulation multi-FPGA et ensuite appliquer des techniques analytiques pour corriger les

latences mesurées, ou bien en agissant sur la plateforme multi-FPGA bien avant le processus d'émulation afin de garantir un minimum d'imprécisions.

4.5.1 Technique analytique de correction de la latence

Cette technique sert principalement à corriger la latence point à point calculée après émulation multi-FPGA. L'idée de base est assez simple car consistant à déduire le nombre de cycles nécessaires à une transmission d'un flits (ou paquet) sur une interface off-chip de la valeur ayant été mesurée. Bien sûr seuls les liens passant effectivement sur une interface off-chip sont concernés. Cette technique reste valide dans la zone de fonctionnement linéaire du NoC.

$$Latence\ réelle\ = latence\ mesurée - K_{off\text{-}chip} \qquad (1)$$

K étant le nombre de cycles nécessaires à une transmission inter-FPGA

Dans le cas ou plusieurs liens on-chip partagent la même interface série, il devient nécessaire de prendre en compte le temps d'arbitrage maximal pouvant être observé, le nombre de cycles maximal à soustraire K peut être estimé en

$$K = K_{off\text{-}chip} * N \qquad (2)$$

N étant le nombre de lien on-chip partageant le lien série

Pour que cette méthode soit applicable un certain nombre de conditions doivent être vérifiées principalement pour garantir que le réseau émulé reste dans sa zone de linéarité. Cependant, même avec cette méthode les mesures ne peuvent être précises à 100%. Dans ce cas là une autre alternative reste possible, néanmoins cette nouvelle solution nécessite que des modifications matérielles soit opérées sur la plateforme d'émulation.

4.5.2 Techniques matérielles pour la réduction des imprécisions

Il est possible de s'attaquer au problème des imprécisions plus en amont lors du flot de conception de la plateforme d'émulation. La principale cause des imprécisions étant due aux délais des interconnexions série, il existe une seule solution pour pallier cela. Cette solution se décline en deux variantes équivalentes, la première consistant à augmenter la fréquence de fonctionnement des liens série et la deuxième consistant à diminuer la fréquence du système émulé. Dans les deux déclinaisons de la solution l'idée consiste à réduire le rapport entre le nombre de cycles nécessaires au transfert d'un flit entre deux cartes FPGA et le nombre de cycle pour la même opération en intra-FPGA.

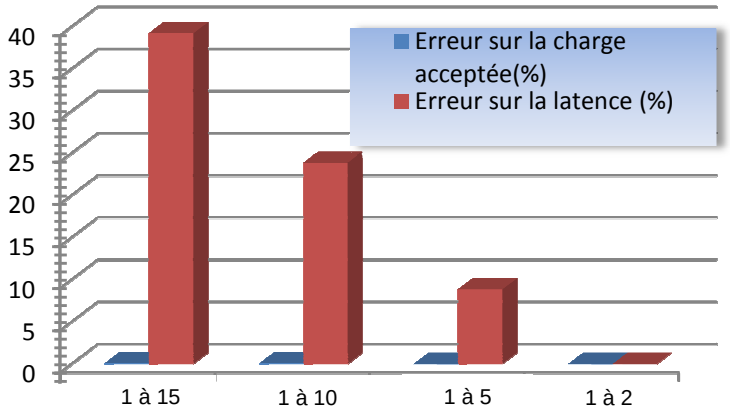


FIGURE 4.20 - Relation entre le rapport vitesse Emulation/Vitesse transfert et la précision

La FIGURE 4.20 illustre bien l'accroissement de la précision à mesure que les temps de communications inter-FPGA diminuent par rapport à la vitesse de l'émulation. Cependant, il est impératif de noter ici que le scénario de test utilisé amplifie les effets des latences inter-FPGA. Sous des conditions de trafic réelles des rapports 1 à 10 et 1 à 5 sont généralement largement suffisant, car après partitionnement le trafic inter-FPGA est minimal.

4.6 Discussion sur la précision de l'émulation

Le manque de benchmarks pour réseaux NoCs rends très difficile l'estimation d'intervalles de fonctionnements pour notre plateforme ou les déviations de performances seraient bien délimitées. Mis à part les benchmarks OCP-IP [104] non librement accessibles, il n'existe pratiquement pas de benchmarks spécifiques aux NoC qui seraient adaptés à nos besoins actuels. Dans ce paragraphe nous essayerons d'identifier les scénarios qui seraient les plus problématiques à émuler sur notre plateforme, pour cela nous commençons par une classification des architectures matérielles à base de NoC ainsi qu'une classification des modèles de communication.

Une étude des modèles d'architectures à base de NoC dégage trois familles d'architectures selon le mode d'adressage mémoire utilisé : des architectures avec un modèle d'adressage global et des architectures avec modèle d'adressage local. La troisième famille d'architectures combine les deux modes selon une organisation en clusters. Un modèle d'adressage global donne une vue globale sur la mémoire alors qu'un modèle d'adressage local ne permet d'adresser que les mémoires locales à un nœud donné. Dans ce second mode d'adressage, les accès aux mémoires distantes ne peuvent se faire qu'à travers un système de passage de messages. D'un point de vue logiciel, on a généralement à faire à une organisation en pipeline où une tâche produit, par des traitements locaux, un flux de données passé à une autre tâche en aval pour d'autres traitements. Modes d'adressages et modèles de traitements combinés nous permettent d'identifier deux scénarios de communications : un scénario par passage de messages et un scénario requête/réponse.

Dans le premier cas de figure où toutes les communications se font par passage de messages, l'impact d'un délai additionnel sur les canaux de communications n'est pas cumulable quelque soit le volume et la distribution des communications grâce à l'effet pipeline des interfaces série. Cet effet donne la garantie que l'erreur cumulée est fonction des fréquences des événements de communication et non pas fonction du volume instantanés des transferts. Dans le second cas ou les communications se font par un système de requête/réponse, l'effet pipeline garantit le même comportement mais cette fois ci le nombre de liens off-chip parcouru joue un rôle plus important, mais les erreurs cumulées restent

complètement linéaires par rapport à la fréquence des événements de communications et le nombre de liens off-chip parcouru.

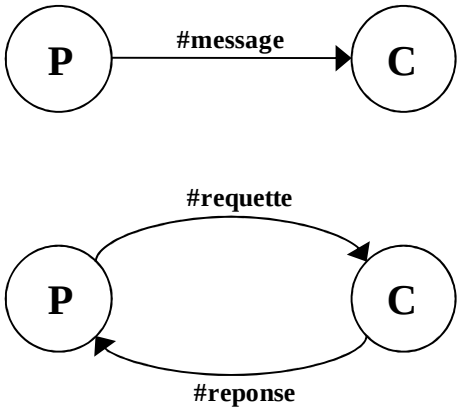


FIGURE 4.21 - Modes de communications sur les systèmes à base de NoC

Concrètement cela se traduit par une stabilité de l'erreur pouvant être observée, quelle que soit la durée du processus d'émulation. Sur le graphique donné en FIGURE 4.22 nous donnons les résultats de l'erreur estimée obtenue en variant la durée du processus d'émulation. Comme on peut le voir aucun phénomène d'accumulation de l'erreur n'est observé.

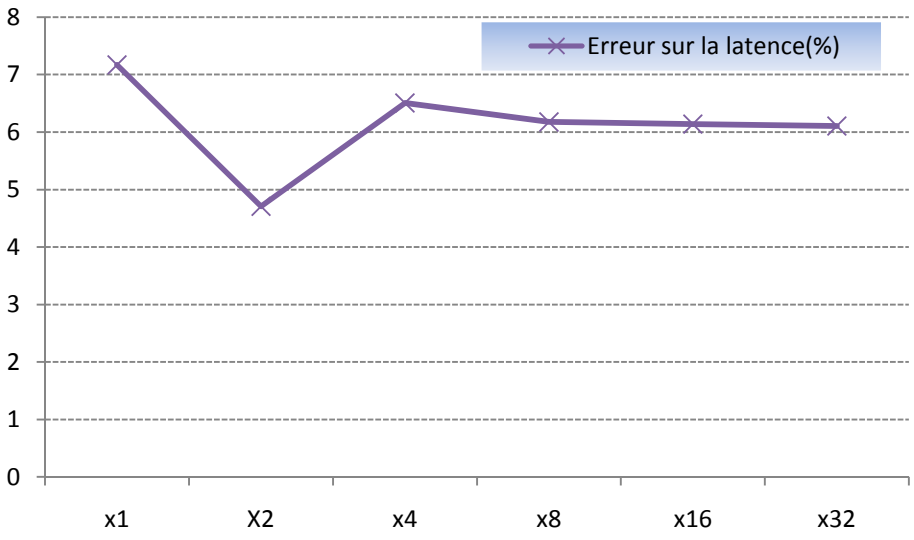


FIGURE 4.22 - Phénomène de non accumulation de l'imprécision

Le modèle de trafic utilisé ici est un modèle reprenant un modèle de communication par passage de message. Le cas où l'application serait sous un modèle requête/réponse (ex : un trafic de lignes de cache d'un nœud mémoire vers un nœud de calcul) une solution pour pallier au phénomène d'accumulation de l'erreur tout au long du processus d'émulation serait de mettre les liens concernés sur un même FPGA. Ou encore de recourir à une diminution de la latence off-chip par l'une des techniques introduite précédemment dans ce chapitre.

4.7 Comparaison avec les techniques de prototypage Multi-FPGA existantes

Selon nos connaissances il n'existe actuellement aucune solution de prototypage et d'émulation conçue spécifiquement pour les NoCs. Toutefois nous pouvons faire une comparaison de quelques techniques avec notre approche d'émulation dans le contexte des NoCs et en se basant sur les trois caractéristiques les plus importantes qui sont : La scalabilité, la vitesse d'émulation et la précision.

A. Scalabilité

Les cartes de prototypage et les émulateurs multi-FPGA existants abordent généralement cet aspect d'un point de vue capacité d'intégration et non pas du point de vue de l'évolutivité. Ainsi les plateformes multi-FPGA tel que [105][90] intègrent plusieurs circuits FPGA sur un même PCB, bien que la capacité d'intégration en soit augmentée en conséquence il n'est pas possible d'augmenter les capacités de telles plateformes dans le cas toujours possible d'un système très grand. Se pose aussi le problème de l'utilisation efficace des ressources de telles plateformes. En effet un nombre insuffisant des signaux d'E/S FPGA peut conduire à une sous utilisations des ressources. Alors que le style d'interconnexions de notre plateforme réduit à l'extrême un tel problème. Un autre point important relatif à la scalabilité concerne le coût, qui sera fonction des besoins de l'utilisateur et non plus fixé par la plateforme elle-même.

B. La vitesse

Sur ce point il faut faire une distinction entre les plateformes destinées à d'émulation de circuit ASIC et celles destinées au prototypage. Alors que les vitesses des émulateurs sont beaucoup moins élevées de celles des plateformes de prototypage, elles offrent beaucoup plus d'observabilité. À cet égard, et concernant la vitesse, notre plateforme est à rapprocher des plateformes de prototypage. De façon générale un système de prototypage multi-FPGA aura une fréquence de fonctionnement beaucoup moins élevée qu'un système mono-FPGA (à technologie équivalent). Par exemple et en reprenant la plateforme [106] basée sur des circuits Virtex 5 [107] a une vitesse maximale de 100 Mhz au meilleure des cas, alors que les circuits Virtex 5 peuvent aller jusqu'à 550Mhz. Dans le cas de notre plateforme la vitesse maximale est beaucoup moins impactée car dépendant seulement du degré de précision souhaité.

C. La précision

Ici on distingue deux nuances de cet aspect, la précision fonctionnelle qui est dans notre plateforme assurée à 100%, par contre la précision des performances temporelles du NoC peut être altérée. Cette notion de précision des performances temporelles n'est pas vraiment présente dans les systèmes d'émulation et de prototypage et il n'est donc pas possible de faire une comparaison.

4.8 Conclusion

Dans ce chapitre nous avons présenté une approche de prototypage multi-FPGA pour réseaux NoCs ainsi que quelques résultats d'expérimentation. Les expérimentations ayant été conduites l'ont été dans le principal but de valider de façon globale l'approche proposée, ainsi que de donner une estimation des imprécisions inhérentes à une telle approche. L'analyse des résultats obtenus confirme bien l'utilité de notre méthode d'émulation multi-FPGA. Bien qu'il soit difficile de généraliser les résultats obtenus au vu de la diversité des applications pouvant exister, il est clair que pour une majorité de cas réels la précision demeure suffisante. Dans les rares cas où la précision n'est pas suffisante nous avons proposé deux méthodes de correction

des imprécisions, une méthode analytique simple et une méthode matérielle plus conséquente mais assurant plus de précision.

Au delà du contexte présent de l'émulation de réseau sur puce, le concept même *« d'émulation matérielle imprécise »* est le concept de *« quantification de la précision »* sont très intéressants. En effet, ce concept bien que déjà très courant dans les approches basé sur la simulation est nouveau dans le domaine de l'émulation matérielle qui comme ont le disait au début du chapitre sous-entend toujours une précision CABA *« Cycle Accurate Bit Accurate»*.

Dans le chapitre suivant nous essayerons de montrer comment le flot de synthèse d'émulateur multi-FPGA peut être adapté à la synthèse de réseaux-sur-puce empilées (*System-in-Package*), en effet plusieurs similitudes existent entre la technologie d'émulation multi-FPGA et les systèmes multi-Puces.

Chapitre 5 : Application aux réseaux empaquetés

Sommaire

5.1 Introduction

Dans ce chapitre nous allons présenter une méthode de synthèse de réseaux empaquetés grandement inspirée de notre méthode de synthèse d'émulateurs multi-FPGA. La méthodologie que nous présenterons dans ce chapitre à pour but d'aider le concepteur à construire un réseau-sur-puce adapté aux nouvelles technologies d'intégrations consistant à empiler plusieurs puces de silicium et les rassembler dans un seul paquetage [10].

Nous allons commencer par introduire la technologie des SiP *« System-In-Package »* et nous présenterons ensuite les problématiques introduites par cette nouvelle technologie d'assemblage principalement en ce qui concerne les réseaux-sur-puce. Après l'étude de l'adaptabilité des NoCs à la technologie des SiP nous proposerons un flot de synthèse ainsi qu'une nouvelle architecture de NoC multi-synchrone que nous avons appelé MS-NoC *«Multi-Synchronous Network-On-Chip»* adaptée à cette technologie. Nous finirons ce chapitre par une étude de performances du NoC proposé sous diverses configurations.

5.2 Présentation de la technologie des systèmes empaquetés «SiP»

Les technologies d'intégration *SiP* sont la réponse technologique à la complexité croissante des applications embarquées, nécessitant de plus en plus souvent le recours à des systèmes hétérogènes non seulement par leurs fonctionnalités (plusieurs familles de processeurs, plusieurs types de mémoires) mais aussi par leurs technologies de conception même. A titre d'exemple il devient courant de combiner un système analogique avec un système numérique, les applications de communication ont typiquement recours à ce genre de combinaisons (dispositifs de communication 3G, 4G). En plus des grandes possibilités offertes par la technologie des SiP au regard des fonctionnalités ; elle présente plusieurs avantages dont :

- Un meilleur rendement et un meilleur coût, du fait que la réutilisation de composants existants se fait de façon beaucoup plus simple que si le composant devait être intégré sur la même puce avec d'autres composants.
- Un rapport taux intégration/surface très élevé
- Un meilleur profile énergétique que les systèmes mono-puce classiques

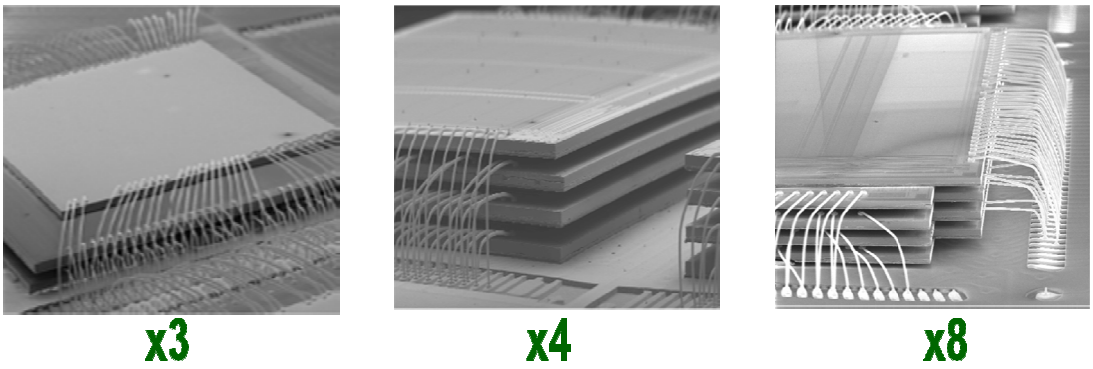


FIGURE 5.1. - Exemples de systèmes SiP à 3,4 et 8 puces empilées

Sur la FIGURE 5.1 sont illustrés quelques exemples de SiP comprenant respectivement 3, 4 et 8 puces (communément appelés *« Die»*). Sur les exemples donnés ici les die sont assemblés de manière verticale et les interconnexions inter-die sont assurées par des liens en bords de die. Cependant, il faut savoir que l'industrie des SiP définie plusieurs standard d'assemblage et d'interconnexions. L'assemblage des die peut être soit sous forme d'un empilement vertical, horizontal ou bien mixte où les deux déclinaisons sont utilisées. Quand aux technologies d'interconnexions il en existe principalement trois:

– *« Wire-bonding » :* Les interconnexions sont sous forme de micro-fils en or ou en aluminium (250 à 400 µm) reliant les die entre-eux. On constatera ici que la densité d'interconnexion peut être très limitée par rapport aux besoins réels, car les interconnexions ne peuvent se faire qu'on bord de die.
– *« Flip-Chip » :* Cette technique consiste à utiliser des « microbilles » conductrices reparties sur toute la surface du die. Les interconnexions se font alors par plaquage entre deux surfaces. Cette technique présente plusieurs avantages par rapport au *« Wire-bonding »* le principal étant une densité d'interconnexions plus élevée car toute la surface du die peut être utilisée et seulement les bords. En plus de cela, les interconnexions étant beaucoup plus courtes les caractéristiques électriques s'en trouvent grandement améliorées (capacités résiduelles, temps de propagations).
– *« Through-silicon vertical vias »* Ce type d'interconnexions passe directement à travers les dies ce qui permet la construction d'un vrai schéma d'interconnexion en 3 dimensions. Des interconnexions inter-die directes assurent de meilleures performances

électriques (bilan énergétique) et temporelles (fréquences) que le mode « *Flip Chip* », en plus de cela la densité d'interconnexion est beaucoup plus élevée.

Un des principaux critères de classification des technologies d'interconnexions est bien la densité de ces dernières (en plus des caractéristiques électriques). Bien que la technologie des vias verticaux puisse offrir une densité d'interconnexion très élevée elle reste encore une technologie jeune et peu utilisée. Le *« Wire-Bonding »* et le *« Flip-Chip »* quand à eux sont limités de ce coté, le problème qui se pose alors consiste à optimiser un système donné pour une implémentation SiP en considérant ces limitations. Dans ce qui suit nous introduirons le concept de « *NiP : networks-In-package »* concept découlant de l'adaptation des systèmes MPSoC construits autour de NoCs à la technologie multi-die SiP.

5.3 Les « NiP : Networks-In-package » ou réseaux empaquetés

Les NiP constituent l'évolution naturelle des systèmes NoC vers la technologie SiP (FIGURE 5.2) [9] [108]. Un NiP est un réseau sur puce dont l'architecture est distribuée sur plusieurs die. Avant de donner plus de détails sur l'architecture d'un NiP il est important de savoir que l'adaptation d'un système NoC à une technologie SiP peut se faire à plusieurs niveaux, (le problème se pose de la même façon que pour l'émulation multi-FPGA) respectivement : niveau porte logique, niveau module, niveau unité fonctionnelle. Par niveau d'adaptation nous entendons niveaux de partitionnement : autrement dit à quel niveau le système considéré peut être partitionné afin de pouvoir être adapté à une architecture multi-puces ? Pour répondre à cette question il faut prendre en considération la contrainte principale d'un système multi-puces qui consiste en la limitation des interconnexions inter-puces. Il devient évident que les deux niveaux d'adaptation, à savoir le niveau porte et niveau module, nécessitent un volume trop élevé d'interconnexion alors que le niveau « unité fonctionnelle » est le mieux adapté.

Dans ce qui suit nous proposerons un flot de synthèse et d'exploration de réseaux NiP ainsi que quelques résultats expérimentaux. Nous commencerons par l'introduction du flot de synthèse global.

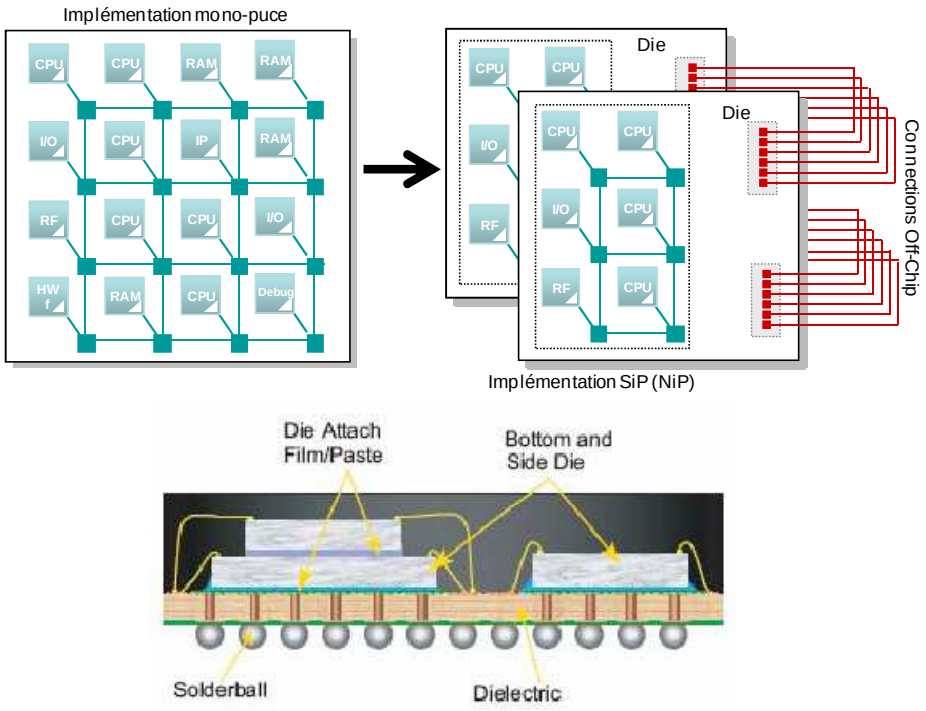


FIGURE 5.2. - Implémentation multi-die d'un réseau sur puce

5.4 Flot de synthèse de réseaux NiP

Sur la FIGURE 5.3 est donné le flot global de synthèse de NiP que nous proposons. Comme on peut le voir le flot prend comme entrée une description de NoC existant et cela bien qu'il soit possible d'imaginer de nouvelles architectures de NoCs en 3D. Dans ce qui suit nous détaillerons chacune des étapes du flot.

5.4.1 Partitionnement du NoC

La toute première étape dans le processus de synthèse consiste à partitionner une topologie de NoC existant sur l'architecture SiP ciblée. Tout comme pour la synthèse d'émulateurs multi-FPGA, le problème de partitionnement est crucial dans le cas présent.

En effet, en plus des limitations en volume des interconnexions off-chip, celles-ci ont des spécifications physiques et temporelles complètement différentes de celles des liens on-chip (inter-die). D'une part les temps de propagation peuvent être beaucoup plus grands et d'autre part la conjonction d'un certain nombre de phénomènes (cross-talk, capacités parasite) augmente sensiblement la dissipation énergétique par flit émis. Le problème de

partitionnement peut donc être vu sous plusieurs angles : soit on effectue un partitionnement guidé par le nombre d'interconnexions pour optimiser les performances temporelles soit on considère le critère de minimisation de l'énergie et on fait le partitionnement en conséquence. La minimisation de l'énergie passe par une connaissance de l'activité dynamique sur chaque lien, activité pouvant être très bien approximée par le volume de communications. La définition formelle du problème du partitionnement est identique à celle donnée au chapitre 5.

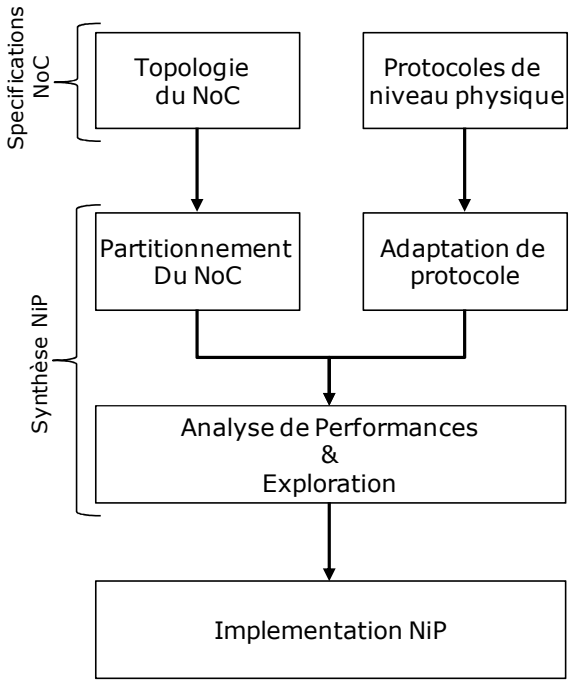


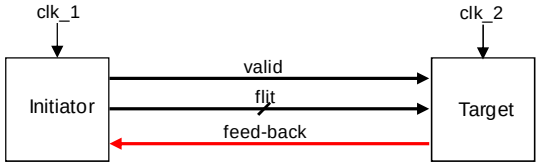
FIGURE 5.3. - Flot de synthèse proposé pour les réseaux NiP

5.4.2 Adaptation du protocole physique

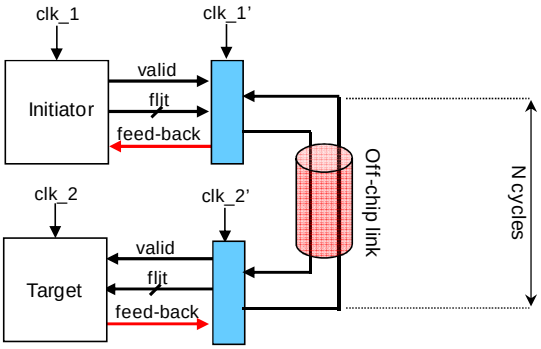
L'adaptation du protocole utilisé en interne par le NoC est nécessaire pour chaque routeur ayant au moins une connexion off-chip. Cette adaptation consiste en la sérialisation des liens inter-die du fait de la limitation des interconnexions. Une autre analogie avec les émulateurs multi-FPGA consiste donc en la sérialisation de toutes les connexions off-chip. La sérialisation permet d'utiliser au mieux le volume d'interconnexion disponible. Des modules SER/DES doivent être utilisés sur chaque liaison off-chip. Il se pose alors le problème de

choisir le schéma de sérialisation et la fréquence de sérialisation adéquats. Idéalement, il faut trouver une distribution optimale du volume de liens off-chip disponibles et des fréquences de sérialisation qui garantissent à la fois une dissipation d'énergie minimale et une latence moyenne minimale (proche de 1 cycle). La consommation étant proportionnelle à la fréquence de sérialisation et la latence lui étant inversement proportionnelle. Dans la suite de ce chapitre nous proposerons une méthode formelle permettant de trouver le schéma de sérialisation optimal partant des contraintes d'interconnexions et de consommation.

Le deuxième volet sur l'adaptation du protocole physique concerne l'adaptation des protocoles de contrôle de flux. Le recours à la sérialisation induisant forcément des latences plus importantes, il devient indispensable d'adapter les protocoles de contrôle de flux afin qu'ils garantissent qu'aucun dépassement de buffer n'ait lieux même avec des latences différentes de 1 cycle (assimilées aux liens on-chip) et cela avec des garanties de débits maximales. Pour ce faire nous avons fait une étude comparative de quelques protocoles de contrôle de flux les plus couramment utilisés dans les NoCs sous des conditions de latences non optimale (>1 cycle). Le tableau 5.1 résume les performances des différents protocoles en supposant que la latence des liens off-chip est égale à N cycles.



Protocol de contrôle de flux on-chip



Protocol de contrôle de flux off-chip

FIGURE 5.4. - Contrôle de flux on-Chip/Off-chip

TAB 5.1 Comparatif des performances off-chip des protocoles de contrôle de flux

	***Hand-Shake*[55]**	***START/STOP*[109]**	***Protocol à Credit*[110]**
On-Chip (1 cycle)	1/2 flit/cycle	1 flit/cycle	1 flit/cycle
Off-chip (N cycles)	1/2N flit/cycle	1/N flit/cycle Avec augmentation de la taille des buffers (+ N)	1/N flit/cycle

Il apparait clairement que les protocoles à crédit offrent les meilleures performances avec des latences de liens non égales à 1 cycle sans qu'aucune modification physique ne soit nécessaire. Le protocole START/STOP peut donner des performances similaires mais nécessite que les taille des buffers soient adaptées (augmentées) en fonction des nouvelles latences off-chip.

5.4.3 Analyse de performances et exploration

Comme pour tout système électronique complexe il est nécessaire de procéder à une étape d'analyse des performances, ceci afin de vérifier l'adéquation du NiP aux besoins et contraintes de l'application. Dans notre approche de synthèse de NiP nous proposons une méthode d'exploration des performances prenant en compte deux métriques très importantes relatives aux NoCs qui sont la latence et l'énergie dissipée. Le problème d'exploration des performances peut être posé comme suit : partant d'un partitionnement de NoC sur plusieurs die, et connaissant une estimation de l'activité dynamique sur chaque lien inter-die et le volume maximal des interconnexions inter-die. Il faut trouver pour chaque lien inter-die un schéma de sérialisation (N→K) qui donne le meilleur compromis (latence/énergie).

Ce problème est un problème d'optimisation multi-objectif et pour ce faire nous proposons une formulation linéaire à ce problème. Cette formulation permet de trouver l'ensemble des solutions répondant aux critères de performances énoncés précédemment.

Pour des raisons de minimisation des ressources matérielles, nous avons considéré ici que la sérialisation utilise un protocole synchrone, autrement dit que l'émetteur et le récepteur

utilisent des signaux d'horloge de fréquences et de phases identiques. Cela rend très facile la fréquence de ces modules: une fréquence trop basse augmente la latence et une fréquence trop élevée augmente la dissipation d'énergie, la formulation que nous donnons permet de trouver le meilleure compromis en fonction des contrainte utilisateurs.

K *: Le nombre total de connexions off-chips du NoC*
F *: La taille unitaire du flit (bits)*
W *: Le volume total des interconnexions off-chip disponibles*
$\boldsymbol{T_i}$ *: Activité dynamique sur le lien i*
$\boldsymbol{Lt(x_i)}$ *: Latence de transfert d'un flit selon le schéma x_i (cycles)*
$\boldsymbol{Pw(x_i)}$ *: Puissance de transfert d'un flit selon le schéma x_i (mW)*

$$1 \le x_i \le F \;;1 \le i \le K$$

$$\begin{cases} Minimize \quad \left(\alpha \times \sum_{i=0}^{i<K} Pw\left(x_i\right) + \beta \times \sum_{i=0}^{i<K} T_i \times Lt\left(x_i\right) \right) \\ \sum_{i=0}^{i<K} x_i \le W \\ Pw\left(x_i\right) = f_i \times F / x_i \\ Lt\left(x_i\right) = F / \left(x_i \times f_i\right) \end{cases}$$

L'ensemble des x_i constitue l'ensemble des schémas de sérialisation devant être trouvés, et les f_i l'ensemble des fréquences qui y sont associées. α et β sont des poids de pondération associés à la fonction objective selon que l'on veuille donner plus d'importance à la minimisation de la latence ou de l'énergie. L'expression de la fonction objective est formée par deux termes, le premier correspond à la somme de la puissance électrique nécessaire pour les schémas de sérialisation choisis. Le second terme dans la fonction objective correspond à la latence de tous les liens off-chip. Dans ce qui suit nous donnons une application directe de cette formalisation, mais avant nous présenterons l'architecture du NoC avec laquelle nous avons fait les explorations.

5.5 Réseau sur puce multi-synchrone *«MS-NoC»*

La conception du réseau MS-NoC répond principalement au besoin de disposer d'une architecture de NoC pouvant être rapidement convertie en réseau NiP. Partant des spécificités que nous avons pu observer sur de telles architectures, nous avons fait la conception de ce réseau.

Le routeur de base du réseau MS-NoC (FIGURE 5.5) permet de construire des topologies en grilles N*x*M avec un politique de routage *« X puis Y »*. L'architecture de MS-NoC est complètement modulaire, cela présente l'avantage de faciliter grandement l'ajout de nouvelles fonctionnalités (nouvelle politique d'ordonnancement, nouvelle topologie). En plus, et plus généralement, la modularité de la conception permet d'optimiser les processus de synthèse matérielle. Comme nous l'avons montré précédemment dans ce chapitre, le contrôle de flux par crédits donne les meilleures performances en termes de débit maximal sous des conditions de latences non idéales. Le réseau MS-NoC supporte donc nativement ce mécanisme de contrôle de flux.

Le recours au paradigme multi-synchrone, comme nous l'avons fait, répond au besoin d'avoir une séparation stricte entre la logique interne du routeur et la logique responsable du transport des données. Il devient dès lors très facile de choisir la fréquence optimale de fonctionnement des liens inter-routeurs (en fonction des résultats de l'exploration), particulièrement des liens off-chip. L'implémentation multi-synchrone de MS-NoC est basée sur l'utilisation de FIFOs bi-synchrones, c'est-à-dire des FIFOs utilisant deux domaines d'horloges : un domaine pour l'écriture et un domaine pour la lecture. Ces FIFOs permettent aussi de définir une taille pour l'unité de lecture qui est différente de celle de l'unité d'écriture ce qui simplifie l'implémentation des mécanismes de sérialisation et désérialisation [111].

5.6 Mesure de performances est exploration « MS-NoC »

D'un point de vue « flot de conception » il est très intéressant d'essayer de quantifier les différences de performances entre un NoC et son implémentation NiP. Cette importance traduit en fait les besoins accrus de diminution des temps de conception qui motivent le recours à la réutilisation de composants déjà existants avec des performances pré-caractérisées.

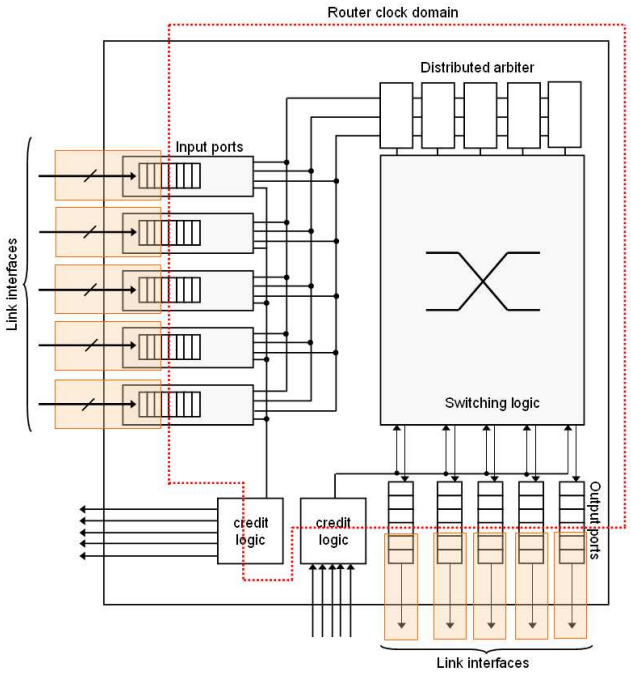


FIGURE 5.5. - Microarchitecture du routeur MS-NoC

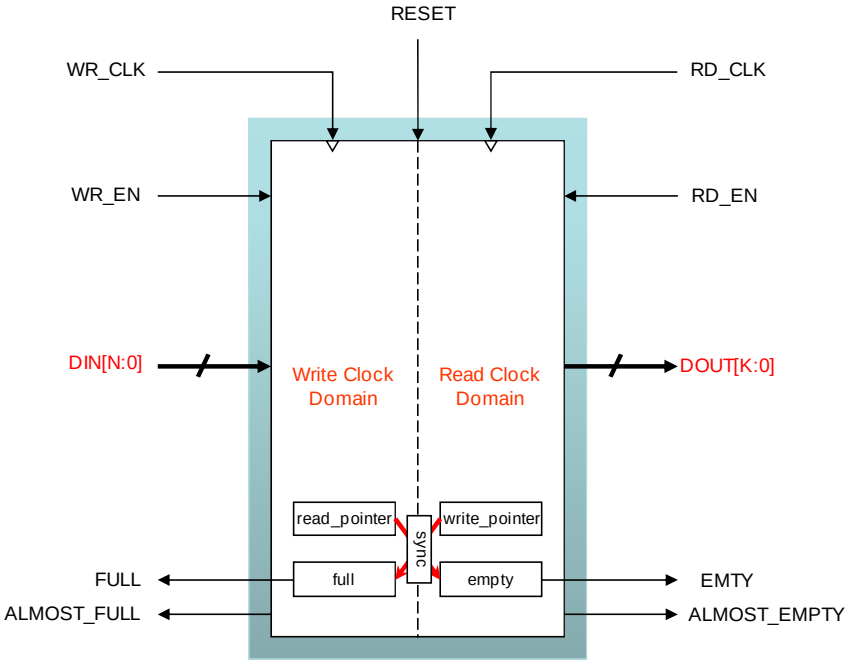


FIGURE 5.6. - Microarchitecture d'une FIFO bi-synchrone

Pour cette première phase d'étude de performances nous nous sommes donc focalisés sur l'identification des déviations de performances NoC→NiP. Pour se faire nous avons considérés une instance en grille 4*x*4 de MS-NoC et son équivalent NiP sur deux dies. La taille unitaire du flit étant de 64 bits, nous avons considéré trois schémas de sérialisation possibles utilisables sur les liens off-chip (64→32, 64→16, 64→8) sans aucune augmentation de fréquence sur les liens off-chip (i.e. utilisation du même signal d'horloge que celui utilisé en interne). La FIGURE 5.7 montre l'évolution de la latence du NoC original ainsi que celle des trois configurations de NiP qui y sont issues. Ces résultats ont été obtenus par une simulation CABA du modèle MS-NoC avec un modèle de trafic uniforme *« All-to-All »*.

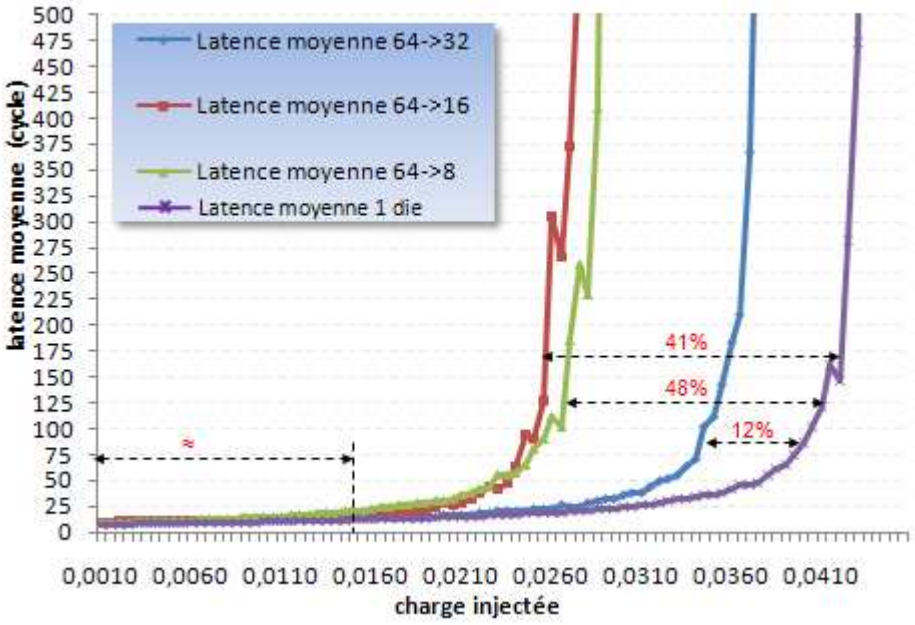


FIGURE 5.7. - Evolution de la latence en fonction de la charge injectée

La première observation est relative au fonctionnement en mode non saturé des trois configurations de NiP. On remarque en effet que la déviation des performances reste relativement limitée de l'ordre de 10% en moyenne. Pour rappel le trafic uniforme utilisé lors des test peut être assimilé à une trafic stressant. Cette deviation limitée permet dans une certaine mesure la réutilisation de rapports de performances de NoC déjà existants en y associant des intervalles d'erreurs. La deuxième observation concerne la relation entre le

schéma de serialisation choisi et le point de saturation du NoC. D'après les résultats obtenus et comme on pouvaient s'y attendre plus le shéma de sérialisation s'éloigne de la valeur d'horigine de la taille du flit plus le point de saturation diminue. Selon les contraintes de l'application visée en termes de délais de transport, on peut choisir le schéma de serialisation adéquat. Cependant les limitations en volume total d'interconnexions inter-die peuvent ne pas permettre de chosir le schéma de serialisation desiré, dans ce cas le formailisme d'exploration que nous avons donné permet de trouver la meilleure distribution de schémas de serialisation.

Sur les figures 5.8 et 5.9 nous donnons les résultats d'exploration obtenus sous un trafic uniforme *« all-to-all »*. Trois valeurs de W (nombre maximal d'interconnexions inter-die) ont été considerées ansi que trois valeurs pour α et β.

TAB 5.2 Espace de recherche pour l'exploration de NiP

Taille du flit	64 bits				
Nombre d'interconnexions (W)	512		256		128
Fréquences (Mhz)	6,75	12,5	25	50	100
Sérialisation (x_i)	64	32	16	8	4

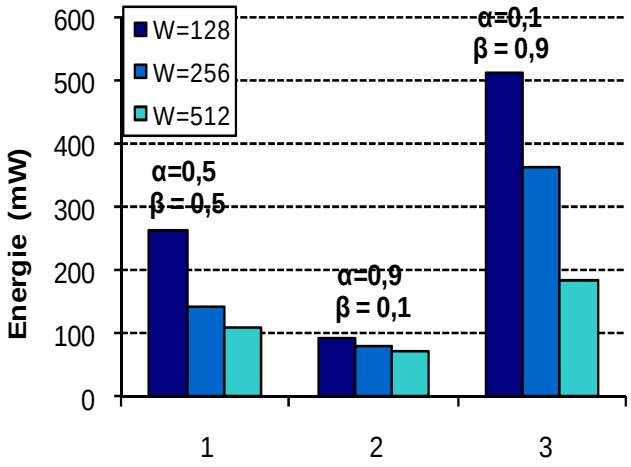


FIGURE 5.8. - Meilleures configurations en termes d'énergie dissipée

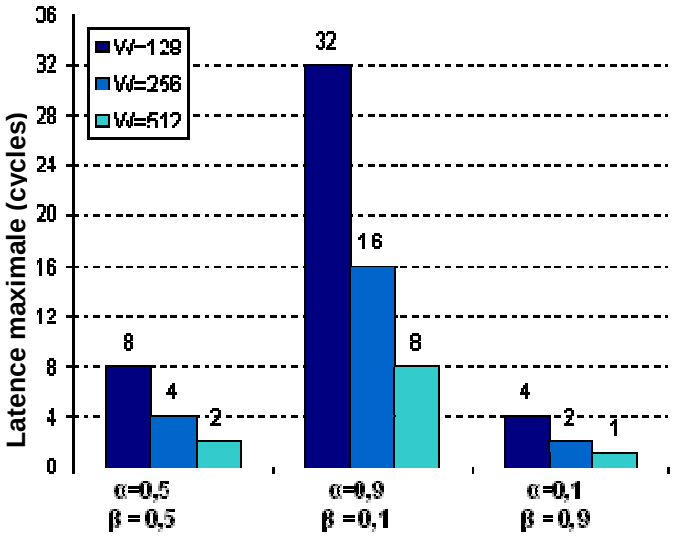


FIGURE 5.9. - Meilleures configurations en termes de latence off-chip

Les résultats obtenus montrent bien que plus les capacités d'interconnexions sont élevées, meilleur sera le bilan énergétique ainsi que les latences inter-die. Cependant même dans les cas ou des limitations existent le meilleurs compromis peut être trouvé en choisissant les valeurs adéquates de α et β.

5.7 Conclusion

Ce chapitre aborde le concept de NiP, plus particulièrement une méthode de synthèse et d'exploration de performances à partir de réseaux NoCs existants. Un flot de synthèse approprié à été proposé afin de faciliter la conception de telles systèmes ; ce flot est grandement inspiré par le flot de synthèse d'émulateurs multi-FPGA présenté auparavant. L'étude des besoins et caractéristiques des NiP nous a amené à proposer une architecture de NoC pouvant être facilement convertie en NiP. Les principales caractéristiques du réseau que nous proposons peuvent être résumées ainsi :

– Une architecture complètement modulaire avec une séparation stricte entre logique interne et logique de transport.
– Un support natif du paradigme multi-synchrone très utile dans le contexte des NiP

– Des interfaces de liens pouvant aussi bien fonctionner en mode parallèle qu'en mode série.

Nous avons aussi présenté quelques résultats d'expérimentation afin de donner une idée sur la déviation pouvant exister entre les performances d'un NoC et son implémentation en NiP, cela afin d'éviter d'avoir recours à des phases de test et de mesure souvent très longues. Finalement, nous avons démontré l'utilité de notre méthode formelle d'exploration de performances des NiP, méthode permettant de trouver pour une configuration et un ensemble de contraintes donnée la meilleure implémentation.

Chapitre 6 : Conclusions & Perspectives

6.1 Conclusions

Les travaux menés durant cette thèse nous ont permis d'aborder la problématique de validation rapide de systèmes à base de réseaux-sur-puce. Ainsi une approche d'émulation multi-FPGA a été proposée et un flot partant de la description du NoC jusqu'à l'implémentation finale à été proposée. Dans un second volet nous nous sommes intéressés à la synthèse de réseaux empaquetés (NiP) en nous inspirant du flot de synthèse d'émulateurs multi-FPGA. Un nouveau concept d'émulation non précise au cycle près à été introduit. Ce concept permet de libérer le concepteur d'un certain nombre de contraintes, contraintes qui obligeaient auparavant le recours à des techniques très lourdes :

- *Des émulateurs industriels au cout financier très élevé*
- *Réduction drastique des vitesses d'émulation*
- *Le maintien d'une synchronisation globale sur la plateforme*

L'approche d'émulation multi-FPGA étudiée ici tend à résoudre ces trois problématiques au coût d'une perte de précisons. En effet, le coût de construction d'une plateforme multi-FPGA telle que la nôtre est insignifiant comparé à celui d'un émulateur industriel (rapport $\approx$ 1 à 1000). De plus, pour la plupart des systèmes à base de NoCs l'émulation peut être conduite aux vitesses maximales sans que pour autant la perte de précision ne soit prohibitive.

Les expérimentations qui ont été menées au cours de cette thèse ont été menées sous des conditions de trafic extrêmes justement pour identifier les domaines de tolérance de notre approche. Les tests ont bien montré que la déviation pouvait être très limitée sous un trafic relativement soutenu. Par ailleurs, la déviation étant toujours linéaire et non pas aléatoire,

cette approche peut tout à fait être utilisée pour déterminer si une configuration de NoC est adaptée à des conditions de trafic données.

Au vu des particularités de l'émulation multi-FPGA et afin de limiter le phénomène de déviation des performances, nous avons proposé un flot de synthèse d'émulateurs multi-FPGA. Ce flot a pour but de construire de façon automatique une plateforme adaptée à un système MPSoC donné. Ainsi les étapes de prise en compte des conditions de trafic (mappage) et de partitionnement du NoC sont automatisées. Comme préalable, nous avons introduit dans le chapitre III un flot de prototypage FPGA pour réseaux NoCs. Une architecture générique de composants générateurs et récepteurs de trafic à été présentée. Ces deux éléments permettent la mise en place de flots de prototypage rapides sans avoir recours à une intégration complète du système. Afin de permettre une mesure temps réelles des performances du NoC, des moniteurs de latences et des moniteurs de charge ont été introduits. Ces composants ont été développés avec le souci de préserver au maximum les ressources FPGA disponibles. En plus du monitoring en temps réel des performances du NoC, les moniteurs de performance ont été utilisées pour la détection et l'évitement de l'état de congestion sur un NoC.

Dans le dernier chapitre de ce mémoire nous avons introduit une méthode de synthèse de réseaux NiP. Au vu des nombreuses similitudes qui existent entre ce type de réseaux et les réseaux émulés sur plusieurs FPGA, cette méthode est largement inspirée du flot de synthèse d'émulateurs multi-FPGA. Les étapes de mappage et de partitionnement sont identiques. Mais contrairement au prototypage FPGA en général, lors de la synthèse d'architectures ASIC le paramètre consommation d'énergie tient une place primordiale. Pour cela, le flot de synthèse que nous avons proposé tient compte de ce paramètre tout comme nous avons tenu compte de la latence. L'optimisation de ces deux paramètres sur les liens off-chip pouvant être contradictoire nous avons donc proposé une méthode formelle d'exploration permettant de trouver directement une solution offrant le meilleure compromis (latence maximale/énergie).

Pour finir, les deux volets de ce travail de thèse nous ont permis d'appréhender de nouvelles problématiques, problématiques auxquelles nous avons essayé de proposer des solutions qui soient les mieux adaptées. Néanmoins nous citerons ici quelques améliorations possibles sous forme de perspectives.

6.2 Perspectives

Étant tout à fait conscient que les niveaux de précision pouvant être atteints aux vitesses de fonctionnement les plus élevées peuvent ne pas être suffisantes pour un certain nombre d'applications, il convient de réfléchir avant toute chose à une façon d'améliorer la précisions à ces fréquences. Une première solution pourrait être de disposer de technologies de liens séries à très haute vitesse. Nous avons montré durant ce travail de thèse qu'une solution équivalente consistant à diminuer la fréquence d'émulation donnait de très bons résultats de précision. Une autre solution possible consisterait en le couplage de plusieurs liens physiques en un seul lien logique offrant un débit cumulé plus élevé ou encore le recours à des liens de transmission optiques. Ces solutions partagent un point commun qui consiste en la diminution de la latence off-chip apparente, mais au prix de nouvelles contraintes sur les plateformes de prototypage ce qui constitue, dans une certaine mesure, une perte de généricité de l'approche.

Une autre piste d'évolution qui nous semble ouvrir beaucoup plus de pistes serait de recourir à un mécanisme de synchronisation relaxé global qui permettrait de diminuer les erreurs sans pour autant pénaliser la vitesse d'émulation. Cela reviendrait à avoir un processus d'émulation à plusieurs vitesses, un mode de fonctionnement à vitesse maximale lorsque les conditions de trafics inter-FPGA sont relativement faibles et un mode de fonctionnement à vitesse réduite dès que le trafic off-chip atteint un certain seuil. Cette solution sous-entend la présence d'un mécanisme de signalisation très rapide, soit en donnant une priorité élevée au message de signalisation inter-FPGA soit en utilisant des canaux physiques dédiés.

Concernant le second volet de ce travail de thèse sur les réseaux NiP, les perspectives sont multiples. Sur le plan de l'architecture même des NoCs il serait intéressant de travailler sur de nouvelles topologies et algorithmes de routages qui seraient toujours des topologies en deux dimensions mais avec des extensions sur la troisième dimension. Bien sûre ces topologies ne seraient plus régulières mais elles sont plus adaptées à la technologie SiP et offriraient ainsi de meilleures performances. Une autre piste intéressante, sur un autre plan, concerne l'étude d'algorithmes de placement en 3D des sous réseaux les uns par rapport aux autres permettrait d'optimiser les longueurs des liens avec tous ce que cela implique comme diminution de la latence et de l'énergie.

Liste des Publications

Publication internationales

- Kouadri A., Senouci B., Pétrot F., *"Large Scale On-Chip Networks : An Accurate Multi-FPGA Emulation Platform", Digital System Design Architectures, Methods and Tools, 2008. DSD '08. 11th EUROMICRO Conference on*, pp.3-9, 3-5 Sept. 2008.
- Senouci B., Kouadri A., Rousseau F. , Petrot F., *"Multi-CPU/FPGA Platform Based Heterogeneous Multiprocessor Prototyping: New Challenges for Embedded Software Designers", Rapid System Prototyping, 2008. RSP '08. The 19th IEEE/IFIP International Symposium on* , vol., no., pp.41-47, 2-5 June 2008.
- Kouadri A., Senouci B., Petrot F., *"Networks-In-Package: Performances management and design methodology", VLSI Design, Automation and Test, 2008. VLSI-DAT 2008. IEEE International Symposium on* , pp.140-143, 23-25 April 2008.
- Kouadri A., Senouci B., Petrot F., *"Scalable Multi-FPGA Platform for Networks-On-Chip Emulation", Application -specific Systems, Architectures and Processors, 2007. ASAP. IEEE International Conf. on* , pp.54-60, 9-11 July 2007.

Colloques nationaux

- Pétrot F., Kouadri A., *"From Networks-On-Chip Emulation to Networks-In-Package Synthesis"*, French Winter School on Heterogeneous Embedded Systems Design, Montebello, Québec, Canada, 7-9 January 2008.
- Kouadri A., *Pétrot F., "Flexible Architectures for HW/SW Interfaces" French Winter School on Heterogeneous Embedded Systems Design, Villard-de-Lans, France, January 10 – 12, 2007.*

Références Bibliographiques

[1] *De Micheli G., Benini L., "Networks on chip: a new paradigm for systems on chip design", Design, Automation and Test in Europe Conference and Exhibition, 2002. Proceedings, vol., no., pp.418-419, 2002.*

[2] *Baghdadi A., Zergainoh N-E., Cesario W., Roudier T., Jerraya A. A., "Design Space Exploration for Hardware/Software Codesign of Multiprocessor Systems", 11th IEEE International Workshop on Rapid System Prototyping, 2000.*

[3] *Lahiri K., Raghunathan A., Dey S., "Design space exploration for optimizing on-chip communication architectures", Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on , vol.23, no.6, pp. 952-961, June 2004.*

[4] *Goossens K., "Formal methods for networks on chips", Application of Concurrency to System Design, 2005.Fifth International Conference on, pp. 188-189, 7-9 June 2005.*

[5] *Marculescu R., Jingcao H,. Ogras U.Y. , "Key research problems in NoC design: a holistic perspective", Hardware/Software Codesign and System Synthesis, 2005. Third IEEE/ACM/IFIP International Conference on, pp.69-74, Sept. 2005.*

[6] *Marti JB., "Large Tandem Queueing Networks with Blocking", Queueing Systems: Theory and Applications, v.41 n.1-2, pp.45-72, June 2002.*

[7] *OSCI TLM 2.0 Language Reference Manual at www.systemc.org*

[8] *Blampey A., "Interopérabilité en Emulation et Prototypage Matériel", Thesis, TIMA laboratory, 2006.*

[9] *Kouadri A., , Senouci, B., Pétrot, F., "Networks-In-Package: Performances management and design methodology", VLSI Design, Automation and Test, 2008. IEEE International Symposium on, pp.140-143, 23-25 April 2008.*

[10] *Lim S.K. ,"Physical design for 3D system on package", Design & Test of Computers, IEEE, Vol. 22, pp. 532- 539. 0740-7475. 2005.*

[11] *Sherwani N., Yu Q. and S. Badida, "Introduction To Multichip Modules", Ed. John Wiley & Sons LTD, 1995.*

[12] *Arteris white paper. "A Comparison of Network-on-Chip and Bussses", Design and Reuse, http://www.us.design-reuse.com/articles/article10496.html, 2005.*

[13] *ARM limited, "AMBA Specification", www.arm.com, 1999.*

[14] *International Business Machines Corporation, "The CoreConnect™ Bus Architecture",. http://www-01.ibm.com, 1999.*

[15] *Zarlink Semiconductor, "ST-BUS Generic Device Specification", http://www.eetindia.co.in, 2003.*

[16] *Eui Bong J., Han Wook C., Neungsoo P., Yong Ho S., "SONA: An On-Chip Network for Scalable Interconnection of AMBA-Based IPs", Computational Science, International Conference on, pp. 244-251, 2006.*

[17] *"Reducing the dynamic power and leakage power of a high performance SoC" http://www.cadence.com/rl/Resources/conference_papers/lptp_cdnlivesv2006_tobing_reducingpower.pdf*

[18] *GaoMing D.; DuoLi Z.; YongSheng Y.; Liang M.; LuoFeng Geng; YuKung Song; MingLun Gao, "FPGA prototype design of Network on Chips", Anti-counterfeiting, Security and Identification 2008. 2nd International Conference on, pp.348-351, 20-23 Aug. 2008.*

[19] *Hur JY., Stevanov T., Wong S., Vassiliadis S., "Systematic Customization of On-Chip Crossbar Interconnects", Springer Berlin / Heidelberg, 2007. pp. 61-72. Vol. 4419/2007. 78-3-540-71430-9, 2007.*

[20] *Murali S., Benini L., De Micheli G., "An Application-Specific Design Methodology for On-Chip Crossbar Generation", Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on , vol.26, no.7, pp.1283-1296, July 2007.*

[21] *Ivanov A., De Micheli G., "Guest Editors' Introduction: The Network-on-Chip Paradigm in Practice and Research", IEEE Design and Test of Computers, vol. 22, no. 5, pp. 399-403, Sep./Oct. 2005.*

[22] *Dally W.J., Towles B., "Route packets, not wires: on-chip interconnection networks" Design Automation Conference, 2001. Proceedings, pp. 684-689, 2001.*

[23] *Donno M., Ivaldi A., Benini L., Macii E., "Clock-Tree Power Optimization based on RTL Clock-Gating", dac, pp.622, 40th Design Automation Conference, 2003.*

[24] *Srinivasan, K., Chatha, K.S., Konjevod, G., "Application Specific Network-on-Chip Design with Guaranteed Quality Approximation Algorithms", Design Automation Conference, Asia and South Pacific, pp.184-190, 23-26 Jan. 2007.*

[25] *Pereira Frantz A., Carro L., Cota E., Lima Kastensmidt F., "Evaluating SEU and Crosstalk Effects in Network-on-Chip Routers", pp.191-192, 12th IEEE International On-Line Testing Symposium, 2006.*

[26] *Yaoting Yang J., Yulu Y., Ming H., Mei Y., Yingtao J., "Multi-path Routing for Mesh/Torus-Based NoCs", Information Technology, Fourth International Conference on, pp.734-742, 2-4 April 2007.*

[27] *Guerrier P., Greiner A., "A generic architecture for on-chip packet-switched interconnections", Design, Automation and Test in Europe Conference and Exhibition, Proceedings, pp.250-256, 2000.*

[28] *Coppola M., Locatelli R., Maruccia G.; Pieralisi L., Scandurra A., "Spidergon: a novel on-chip communication network", System-on-Chip. Proceedings. International Symposium on, pp. 15-, 16-18 Nov. 2004.*

[29] *Feero B.S, Friedman E.G,"3-D Topologies for Networks-on-Chip", Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, Vol. 15, pp. 1081-1090. 2007.*

[30] *Feero B. and Pande P.P, "Performance Evaluation for Three-Dimensional Networks-On-Chip", IEEE Computer Society Annual Symposium on , vol., no., pp.305-310, 9-11 March 2007.*

[31] *Kuei-Chung C., Jih-Sheng S., Tien-Fu C., "Evaluation and Design Trade-Offs Between Circuit-Switched and Packet-Switched NOCs for Application-Specific SOCs", Design Automation Conference Proceedings, San Francisco, California, USA, July 24.28, 2006.*

[32] *Pullini A., Angiolini F., Bertozzi, D., Benini L., "Fault Tolerance Overhead in Network-on-Chip Flow Control Schemes", Integrated Circuits and Systems Design, 18th Symposium on, pp.224-229, 4-7 Sept. 2005.*

[33] *Palesi M., Holsmark R., Kumar, S., Catania V., "Application Specific Routing Algorithms for Networks on Chip", Parallel and Distributed Systems, IEEE Transactions on ,vol.20, no.3, pp.316-330, March 2009.*

[34] *Barati, H., Movaghar A., Barati, A., Mazreah A.A., "Routing Algorithms Study and Comparing in Interconnection Networks", Information and Communication Technologies: From Theory to Applications, 2008. ICTTA 2008. 3rd International Conference on, pp.1-5, 7-11 April 2008.*

[35] *Di Ianni M., "Wormhole Deadlock Prediction", Proceedings of the Third International Euro-Par Conference on Parallel Processing, p.188-195, August 26-29, 1997.*

[36] *Holsmark R., Palesi M., Kumar S., "Deadlock free routing algorithms for mesh topology NoC systems with regions", 9th EUROMICRO Conference on Digital System Design, In Proceedings, September 2006.*

[37] *Ascia G., Catania V., Palesi M., Patti D., "Neighbors-on-Path: A new selection strategy for on-chip networks", In Fourth IEEE Workshop on Embedded Systems for Real Time Multimedia, pages 79–84, Seoul, Korea, October 2006.*

[38] *Moraes F., Calazans N., Mello A., Moller L., Ost L., "HERMES: an infrastructure for low area overhead packet-switching networks on chip", Integration, the VLSI Journal, Volume 38, Issue 1, Pages 69-93, October 2004.*

[39] *Vivet P., Lattard D., Clermidy F., Beigne E., Bernard C., Durand Y., Durupt J., Varreau D., "FAUST, an Asynchronous Network-on-Chip based Architecture for Telecom Applications",http://www.date-conference.com/conference/2007/prog/Sessions/Session4S48.pdf, 2007.*

[40] *Hu J. and Marculescu R., "Smart routing for networks-on-chip", Technical report, ECE Department, Carnegie Mellon University, Available at http://www.ece.cmu.edu/_sld/pubs, March 2004.*

[41] *Borrione D., Helmy A., Pierre L,. Schmaltz J., "A Generic Model for Formally Verifying NoC Communication Architectures: A Case Study", First International Symposium on Networks-on-Chip, In Proceedings, pp.127-136, 2007.*

[42] *Se-Joong L., Kangmin L., Hoi-Jun Y., "Analysis and implementation of practical, cost-effective networks on chips", Design & Test of Computers, IEEE , vol.22, no.5, pp. 422-433, Sept.-Oct. 2005.*

[43] *Murali S., Meloni P., Angiolini F., Atienza D., Carta S., Benini L., De Micheli G., Raffo, L., "Designing Application-Specific Networks on Chips with Floorplan Information", Computer-Aided Design, IEEE/ACM International Conference on, pp.355-362, 5-9 Nov. 2006.*

[44] *Grosse1 P., Durand1 L., Feautrier P., "Power Modeling of a NoC Based Design for High Speed Telecommunication Systems", Springer Berlin / Heidelberg, Integrated Circuit and System Design. Power and Timing Modeling, Optimization and Simulation, pp 157-168, 2006.*

[45] *Rodriguez, S., Jacob, B., "Energy/Power Breakdown of Pipelined Nanometer Caches (90nm/65nm/45nm/32nm)", Low Power Electronics and Design, Proceedings of the 2006 International Symposium on, pp.25-30, 4-6 Oct. 2006.*

[46] *Mullins R., "Minimizing Dynamic Power Consumption in On-Chip Networks", System-on-Chip, International Symposium on, pp.1-4, 13-16 Nov. 2006.*

[47] *Synopsys PrimePower http://www.bitpipe.com/detail/PROD/1017065073_132.html*

[48] *Chan J., Parameswaran S., "NoCEE: energy macro-model extraction methodology for network on chip routers", Computer-Aided Design, IEEE/ACM International Conference on, pp. 254-259, 6-10 Nov. 2005.*

[49] *Amde M., Felicijan T., Efthymiou A., Edwards D., Lavagno L., "Asynchronous on-chip networks", Computers and Digital Techniques, IEE Proceedings , vol.152, no.2, pp. 273-283, Mar 2005.*

[50] *Werner T., Akella V., "Asynchronous processor survey", Computer, vol.30, no.11, pp.67-76, Nov 1997.*

[51] *Beigne E., Clermidy E., Vivet P., Clouard A., Renaudin M., "An asynchronous NOC architecture providing low latency service and its multi-level design framework", Asynchronous Circuits and Systems, 11th IEEE International Symposium on, In Proceedings, pp. 54-63, 14-16 March 2005.*

[52] *Sheibanyrad A., "Implémentation Asynchrone d'un Réseau-sur-Puce Distribué", Thesis LIP6 Laboratory, 2008*

[53] *Salminen E., Kulmala A., Hamalainen T.D., "On network-on-chip comparison", Digital System Design Architectures, Methods and Tools, 10th Euromicro Conference on, pp.503-510, 29-31 Aug. 2007.*

[54] *Cidon I. and Keidar I., "Zooming in on Network-on-Chip Architectures", CCIT research report, Technion University, Haifa, Israel, December 2005.*

[55] *Bjerregaard T., Mahadevan S., "A Survey of Research and Practices of Network-on-Chip", ACM Computing Surveys, 38(1), pp. 1-51, 2006.*

[56] *Delorme J., "Méthodologie de Modélisation et d'Exploration d'Architecture de Réseaux sur Puce Appliquée aux Telecommunications", Thesis, IETR Laboratory, 2007.*

[57] *ARTERIS- The Network-on-Chip Company, http://www.arteris.com/*

[58] *Li X., Hammami O., "NOCDEX: Network on Chip Design Space Exploration Through Direct Execution and Options Selection Through Principal Component Analysis", Industrial Embedded Systems, International Symposium on , pp.1-4, 18-20 Oct. 2006.*

[59] *http://www.design-reuse.com/news/9888/arteris-products-building-networks-chip-noc.html*

[60] *Ciordas C., Hansson A., Goossens K., Basten T., "A Monitoring-Aware Network-on-Chip Design Flow", Digital System Design: Architectures, Methods and Tools, 9th EUROMICRO Conference on, pp.97-106, 2006.*

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

[62] *Miro Panades I., Greiner A., Sheibanyrad A., "A Low Cost Network-on-Chip with Guaranteed Service Well Suited to the GALS Approach", Nano-Networks and Workshops, 1st International Conference on, pp.1-5, Sept. 2006.*

[63] *Bolotin E., Cidon I., Ginosar R., Kolodny A., "QNoC: QoS architecture and design process for network on chip", Journal of Systems Architecture, Volume 50, Issues 2-3, Special issue on networks on chip, Pages 105-128, February 2004.*

[64] *Bertozzi D., Benini L., "Xpipes: a network-on-chip architecture for gigascale systems-on-chip", Circuits and Systems Magazine, IEEE , vol.4, no.2, pp. 18-31, 2004.*

[65] *Evain S., Diguet J-P., Houzet D., "μspider: a CAD tool for efficient NoC design", Norchip Conference, pp. 218-221, 8-9 Nov. 2004.*

[66] *Jayasimha D. N., Zafar B., Hoskote Y., "On-Chip Interconnection Networks: Why They are Different and How to Compare Them", http://blogs.intel.com/research/terascale/ODI_why-different.pdf*

[67] *Marescaux T., Rångevall A., Nollet V., Bartic A., Corporaal H., "Distributed congestion control for packet switched networks on chip". Parallel Computing Conference , In Proceedings, Malagà, Spain, September, 2005.*

[68] *Gerstlauer A., Gajski D.D., Domer R., Shin D., "Automatic network generation for system-on-chip communication design", Hardware/Software Codesign and System Synthesis, Third IEEE/ACM/IFIP International Conference on ,pp.255-260, Sept. 2005.*

[69] *Moadeli M., Shahrabi A., Vanderbauwhede W., Ould-Khaoua M., "An Analytical Performance Model for the Spidergon NoC", Advanced Information Networking and Applications, 21st International Conference on , pp.1014-1021, 21-23 May 2007.*

[70] *Kambiz S., Kahng A., Li B., Peh L., "ORION 2.0: A Fast and Accurate NoC Power and Area Model for Early-Stage Design Space Exploration", Poster, GSRC Annual Symposium, 29, September, 2008.*

[71] *Ogras U.Y., Marcillescu R., Hyung Gyu L., Choudhary P., Marculescu D., Kaufman, M., Nelson, P., "Challenges and Promising Results in NoC Prototyping Using FPGAs", Micro, IEEE , vol.27, no.5, pp.86-95, Sept.-Oct. 2007.*

[72] *Hyung Gyu L., Ogras U.Y., Marculescu R., Chang N., "Design space exploration and prototyping for on-chip multimedia applications", Design Automation Conference, 43rd ACM/IEEE , vol., no., pp.137-142. 2006.*

[73] *Xilinx Chipscope Pro, http://www.xilinx.com/ise/optional_prod/cspro.htm*

[74] *Synplicity Identify, http://www.synplicity.com/products/identify/index.html*

[75] *Goossens K., Radulescu, A., Hansson A., "A unified approach to constrained mapping and routing on network-on-chip architectures", Hardware/Software Codesign and System Synthesis, Third IEEE/ACM/IFIP International Conference on, pp.75-80, Sept. 2005.*

[76] *Hansson A., Wiggers M., Moonen A., Goossens K., Bekooij M., "Applying Dataflow Analysis to Dimension Buffers for Guaranteed Performance in Networks on Chip", Networks-on-Chip, Second ACM/IEEE International Symposium on, pp.211-212, 7-10 April 2008.*

[77] *Murali S., Coenen M., Radulescu A., Goossens K., De Micheli, G., "A Methodology for Mapping Multiple Use-Cases onto Networks on Chips", Design Automation and Test in Europe, In Proceedings , vol.1, pp.1-6, 6-10 March 2006.*

[78] *Ascia G., Catania V., Palesi M., "Multi-objective mapping for mesh-based NoC architectures", Hardware/Software Codesign and System Synthesis, International Conference on, pp. 182-187, 8-10 Sept. 2004.*

[79] *Ciordas C., Goossens K., Radulescu A., Basten T., "NoC monitoring: impact on the design flow", Circuits and Systems, IEEE International Symposium on ,In Proceedings, pp.4 pp.-1984, 2006.*

[80] *Becker J.E., Bieser, C., Becker J., Mueller-Glase K.-D., "Evaluation of a Packet Switching Algorithm for Network on Chip Topologies using a Xilinx Virtex-II FPGA based Rapid Prototyping System", Industrial Electronics, IEEE International Symposium on , vol.4, pp.3184-3189, 9-13 July 2006.*

[81] *Soteriou V., Hangsheng W., Peh L., "A Statistical Traffic Model for On-Chip Interconnection Networks", Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 14th IEEE International Symposium on, pp. 104-116, 11-14 Sept. 2006.*

[82] *Scherrer A., Fraboulet A. and Risset T., "Long-Range Dependence and On-chip Processor Traffic", Microprocessors and Microsystems, pp72-80, 2009.*

[83] *"Linear Feedback Shift Register", Wikipedia, http://en.wikipedia.org/wiki/Linear_feedback_shift_register*

[84] *"LFSRReference", http://www.newwaveinstruments.com/resources/articles/m_sequence_linear_feedback_shift_register_lfsr.htm*

[85] *Genko N. , Atienza D., De Micheli G., Mendias J.M., Hermida R., Catthoor F., "A complete network-on-chip emulation framework", Design, Automation and Test in Europe, In Proceedings, pp. 246-251 Vol. 1, 7-11 March 2005.*

[86] *Sethuraman B., Vemuri R., "optiMap: a tool for automated generation of NoC architectures using multi-port routers for FPGAs", Design Automation and Test in Europe, In Proceedings , vol.1, pp 6-10 March 2006.*

[87] *Gindin R., Cidon I., Keidar I., "NoC-Based FPGA: Architecture and Routing", Networks-on-Chip, First International Symposium on, pp.253-264, 7-9 May 2007.*

[88] *Xilinx "Using Block RAM in Spartan-3 Generation FPGAs" http://www.xilinx.com/support/documentation/application_notes/xapp463.pdf*

[89] *"FIFOs Using Virtex-II Block RAM" http://www.xilinx.com/support/documentation/application_notes/xapp258.pdf*

[90] *Synplicity High-performances ASIC Prototyping System, "HAPS", http://www.synplicity.com/products/haps/*

[91] *Babb J., Tessier R., Dahl M., Hanono S.Z., Hoki D.M., Agarwal A., "Logic emulation with virtual wires", Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on , vol.16, no.6, pp.609-626, Jun 1997.*

[92] *Rosemary M. and Simon W., "FPGAs with time-division multiplexed wiring: an architectural exploration and area analysis", The ACM/SIGDA international symposium on Field programmable gate arrays ,In Proceedings , pp. 285-285, 2009.*

[93] *"Robust Taboo Search (TS) for the Quadratic Assignment Problem", http://mistic.heig-vd.ch/taillard/codes.dir/tabou_qap.cpp*

[94] *"METIS - Family of Multilevel Partitioning Algorithms" http://glaros.dtc.umn.edu/gkhome/views/metis*

[95] *Xilinx www.xilinx.com*

[96] *"Xilinx Virtex II Pro datasheet", http://www.xilinx.com/support/documentation/virtex-ii_pro_data_sheets.htm*

[97] *"RocketIO™ Transceiver User Guide" http://www.xilinx.com/support/documentation/user_guides/ug024.pdf*

[98] *"PowerPC 405 Embedded Cores", http://www-01.ibm.com/chips/techlib/techlib.nsf/products/PowerPC_405_Embedded_Cores*

[99] *"MicroBlaze Processor", http://www.xilinx.com/products/design_resources/proc_central/microblaze.htm*

[100] *"Aurora Link-layer Protocol", http://www.xilinx.com/products/design_resources/conn_central/grouping/aurora.htm*

[101] *"RocketIO™ Transceiver Characterization Report for Virtex-II Pro™ X FPGA" www.xilinx.com*

[102] *Fraboulet A.,Risset T., Scherrer A., "Cycle Accurate Simulation Model Generation for SoC Prototyping"», Research Report No 2004-18, http://www.ens-lyon.fr/LIP/Pub/Rapports/RR/RR2004/RR2004-18.pdf*

[103] *Riso S., Sassatelli G., Torres L. , Robert M., Moraes F.G, "Réseau d'Interconnexion pour les Systèmes sur Puce : le Réseau HERMES", Signal Circuit and Systems ,IEEE International Conference on, Monastir, Tunisie, pp. 634-638, 2004.*

[104] *"On-Chip Protocol" http://www.ocpip.org/home*

[105] *http://www.dinigroup.com*

[106] *Hitechglobal http://www.hitechglobal.com/Boards/Quad-V5LX330.htm*

[107] *Virtex 5 datasheet http://www.xilinx.com/support/documentation/virtex-5.htm*

[108] *Murali S., Seiculescu C., Benini L., De Micheli G., "Synthesis of networks on chips for 3D systems on chips", Design Automation Conference,. Asia and South Pacific, In Proceedings, pp.242-247, 19-22 Jan. 2009.*

[109] *"Flow control", Wikipedia, http://en.wikipedia.org/wiki/Flow_control*

[110] *Seongmin N., Daehyun K., Vu-Duc N., Hae-Wook C., "Performance and Complexity Analysis of Credit-Based End-to-End Flow Control in Network-on-Chip", Springer Berlin / Heidelberg , Parallel and Distributed Processing and Applications, In Procedings, pp 268-277, 2007.*

[111] *Miro Panades. I., Greiner, A., "Bi-Synchronous FIFO for Synchronous Circuit Communication Well Suited for Network-on-Chip in GALS Architectures", Networks-on-Chip, First International Symposium on, pp.83-94, 7-9 May 2007.*

Architectures Flexibles pour la Validation et L'exploration de Réseaux-sur-Puce

Abstract:

For A multiprocessor system-on-chip (MPSOC), the communication backbone is a central component of prime importance. This is due to the importance of the communications on such distributed systems. Now that networks-on-chip (NoCs) are admitted to be the solution which theoretically best solves the problem of on-chip communications, an important problem which rises consists in providing the designer with fast validation techniques able to tackle such complexes systems. Indeed, despite their regular architectures networks-in-chip internal interactions are difficult to formalize. On the other side, classical validation approaches are far from being suited for NoC-based systems due to their lack of flexibility and scalability. This thesis introduces a new concept in the field of hardware validation of networkson- chip; we have called this new concept "Inaccurate Hardware Emulation" in contrast with most hardware emulation approaches which assume a "cycle accurate bit accurate" precision. Our approach inherits from all advantages of hardware prototyping on reconfigurable devices and adds new scalability features. Study conducted during this thesis showed that under the non-congested regime a NoC may admit a number of alterations on its characteristics (introduced by the emulation platform) without adopting a completely different behavior. The multi-FPGA emulation technique proposed in this thesis is highly flexible since it relies on serial inter-FPGA interconnections. Serial interconnections are less sensitive to noises than parallel style of interconnections, and allow then for higher transfer rates. On the other hand, our emulation approaches does not poses any constraint on the emulation speed. If we consider the fact that serial interconnection schemes may introduce additional delays and the high speeds of the emulation process, performance of the NoC being emulated on the multi-FPGA emulator may deviate from the original NoC. We have studied this phenomenon and we have proposed various solutions for it.
Key words: network-on-chip, FPGA, reconfigurable-platform, prototyping.

Resume :

L'infrastructure de communication pour un système multiprocesseur mono-puce (MPSoC) est un organe central et de première importance. Cette importance s'explique par la place importante que tiennent les communications dans de tels systèmes distribués. Alors qu'il est maintenant admis que les réseaux -sur-puce (NoCs) constituent une solution théoriquement idéale, il se pose le problème de la validation de telles architectures complexes. En effet, malgré la régularité de leurs architectures, les réseaux-sur-puce restent des systèmes dont les interactions internes sont très difficiles à appréhender. Par ailleurs, les approches de validation classiquement employées sont très mal adaptées aux systèmes à base de NoC car très peu flexibles et très peu scalables. Cette thèse introduit un nouveau concept dans la validation matérielle des réseauxsur- puce, ce concept que nous avons appelé « émulation imprécise » contraste avec les approches d'émulation matérielles classiques qui sous-entendent toutes une précision au « cycle près, bit près ». Notre approche hérite de tous les avantages liés au prototypage matériel sur les plateformes reconfigurables et y ajoute un degré de flexibilité très élevé. En effet, l'étude menée au cours de ce travail sur le comportement des réseaux -sur-puce à commutation de paquets en régime non congestionné montre que, sous certaines conditions, des modifications des caractéristiques du NoC (introduites par la plateforme d'émulation elle même) peuvent être tolérées sans que pour autant le comportement du réseau ne change de façon radicale. La technique d'émulation multi-FPGA étudiée dans cette thèse est une technique très flexible car basée sur un mode d'interconnexions inter-FPGA série. Les interconnexions séries sont beaucoup moins sensibles aux phénomènes de parasitage que les interconnexions parallèles et par conséquent les vitesses de transferts sont beaucoup plus élevées. D'autre part la technique d'émulation que nous proposons ne pose aucune condition sur la vitesse du processus d'émulation lui-même. Considérant les délais additionnels induits pas les liaisons séries et les vitesses d'émulation très élevées, un phénomène de déviation des performances peut être observé d'où l'imprécision de l'émulation. Ce phénomène a été étudié dans le cadre de cette thèse et nous avons proposé plusieurs solutions afin d'y remédier.Mots cles : MEMS RF, interrupteur, modelisation, modele statistique, test, evaluation, regression lineaire.

Mots-clés : FPGA de type SRAM, plateformes reconfigurables, prototypage.

ISBN : 978-2-84813-135-1