N° d'ordre :

REPUBLIQUE TUNISIENNE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE DE MONASTIR
FACULTE DES SCIENCES DE MONASTIR
UNIVERSITE DES SCIENCES ET TECHNOLOGIES DE LILLE

THESE

Présentée pour obtenir le titre de Docteur en Physique

(Option Electronique) et Informatique

Par :

Majdi ELHAJJI

Co-Design de l'application H264 et implantation sur un NoC-GALS

Soutenue le 05 /07 /2012, devant le jury composé de :

Pr. Ridha BOUALLEGUE	SUPCOM,Tunis	*Président et Rapporteur*
Pr. Dominique HOUZET	Grenoble INP	*Rapporteur*
Pr. Jean-luc DEKEYSER	Université de Lille1	*Directeur de thèse*
Pr. Rached TOURKI	Faculté des Sciences de Monastir	*Directeur de thèse*
Pr. Virginie FRESSE	Université Saint-Etienne	*Examinateur*
Mc. Abdelkrim ZITOUNI	Faculté des Sciences de Monastir	*Examinateur*

A.U : 2011-2012

N° d'ordre :

REPUBLIQUE TUNISIENNE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE DE MONASTIR
FACULTE DES SCIENCES DE MONASTIR
UNIVERSITE DES SCIENCES ET TECHNOLOGIES DE LILLE

THESE

Présentée pour obtenir le titre de Docteur en Physique

(Option Electronique) et Informatique

Par :

Majdi ELHAJJI

Co-Design de l'application H264 et implantation sur un NoC-GALS

Soutenue le 05 /07 /2012, devant le jury composé de :

Pr. Ridha BOUALLEGUE	SUPCOM,Tunis	*Président et Rapporteur*
Pr. Dominique HOUZET	Grenoble INP	*Rapporteur*
Pr. Jean-luc DEKEYSER	Université de Lille1	*Directeur de thèse*
Pr. Rached TOURKI	Faculté des Sciences de Monastir	*Directeur de thèse*
Pr. Virginie FRESSE	Université Saint-Etienne	*Examinateur*
Mc. Abdelkrim ZITOUNI	Faculté des Sciences de Monastir	*Examinateur*

A.U : 2011-2012

Remerciements

Cette thèse est le fruit de la patience et la générosité de mes directeurs de recherche, Monsieur Rached Tourki, Professeur à la faculté des sciences de Monastir, et Monsieur Jean-Luc Dekeyser, Professeur à l'Université de Lille 1, que je veux vivement remercier. Leurs conseils précieux et leurs encouragements m'ont permis de surmonter toutes les difficultés du parcours. Qu'ils trouvent ici l'expression de ma reconnaissance sincère.

Je tiens à présenter ma vive gratitude à mes deux co-directeurs M. Samy Meftali, Maître de conférences à l'Université de Lille 1, et M.Abdelkrim Zitouni mon encadrant depuis mon projet de fin d'études, Maître de conférences à la Faculté des sciences de Monastir, qui ont contribué activement à la réalisation de mes travaux, par leurs conseils efficaces et leurs remarques pertinentes.

Je remercie également les membres du jury qui ont accepté d'évaluer mon travail de thèse. Merci à M. Ridha Bouallegue, Professeur à SUP'COM de Tunis, pour l'honneur qu'il m'a fait en acceptant de présider le jury et rapporter cette thèse. Je remercie vivement M. Dominique Houzet, Professeur Grenoble-INP et Madame Virginie Fresse, Maître conférence à l'université Jean Monnet Saint-Etienne, pour avoir lu et commenté ce manuscrit et aussi de m'avoir fait l'honneur d'évaluer ce travail avec intérêt.

Un grand merci est adressé à Monsieur Pierre Boulet, professeur à l'université Lille, pour ses conseils précieux qui m'ont permis de mener à bien ce travail.

J'exprime ma gratitude à l'ensemble des membres du laboratoire d'électronique et micro-électronique, de la faculté de science de Monastir, et l'équipe DaRT de L'INRIA de Lille

Table des matières

Listes des Figures

Chapitre 1
Introduction générale

Les systèmes embarqués constituent de plus en plus un aspect essentiel de notre vie sociale et professionnelle. La détection radar, le codage audio vidéo, l'avionique et les systèmes de télécommunication sont des exemples de ces applications qui sont fréquemment utilisables tant dans les transports que dans les domaines militaires, médicaux, etc. Elles font partie d'un vaste ensemble appelé traitement de signal intensif. Ce type d'application est caractérisé par une quantité d'informations à traiter assez importante et un aspect calculatoire répétitif. Suite à ces contraintes, l'implémentation et la conception de ces architectures devient vite incontrôlable en termes d'énergie consommée, de temps de développement, de temps de mise en marché et de taux d'intégration sur silicium.

Parallèlement à la croissance du besoin des systèmes à aspect calculatoire, la technologie des semi-conducteurs ne cesse de s'améliorer. Suivant la loi de Moore, qui est vraie jusqu'à nos jours, le taux d'intégration des transistors, sur la même pièce en silicium, double tous les 18 mois. Cela donne naissance à des systèmes trop complexes à gérer. Les SoC en anglais « ***Systems on Chips*** », sont des systèmes composés de deux parties : une infrastructure de communication et les IPs «***Intellectual property*** » qui décrivent une fonctionnalité ou la tâche d'une certaine application. Un SoC peut contenir des mémoires, des unités de calcul et des périphériques d'entrée/sortie etc. Depuis quelques années, les concepteurs de SoC utilisent les bus comme mediums de communication mais suite à l'évolution des systèmes, les bus ont montré des limites en tant que solutions pour la communication en termes de débit, de bande passante, de problèmes physiques et de parallélisme au niveau communication, etc. Benini et Demechili[1] ont proposé dans des travaux de recherche les « Network on Chips » comme solution pour remédier à ces problèmes.

Les interconnexions physiques sont étudiées depuis plusieurs décennies, le bus est un moyen de communication qui implique des facteurs limitant les performances des SoCs : longueur des interconnexions, consommation d'énergie, problèmes physiques reliés à la perturbation électrique, interférences électromagnétiques. Donc la transmission des informations sera non déterministe et non certaine. Ces problèmes ont poussé les chercheurs et les concepteurs à proposer de nouvelles architectures d'interconnexions. Cependant, nous avons vu des propositions de nouvelles architectures d'interconnexion pour les systèmes sur puce. En particulier, ***"Tewksburyetall"*** [2] ont proposé de remplacer les bus par des réseaux sur puce à base de routeurs.

Les NoCs sont capables de proposer des solutions pertinentes pour résoudre les problèmes reliés à l'intégration complexes des SoCs. Ces infrastructures d'interconnexion devront satisfaire à de nombreuses contraintes de consommation d'énergie, de surface de silicium, de qualité de services, de coût de développement en termes de temps et d'argent, et de synchronisation, en tenant compte des caractéristiques de l'application à implémenter sur l'architecture.

Les avantages des SoCs ne vont pas sans quelques inconvénients majeurs tels que la complexité de conception, le risque d'avoir un produit final qui ne corresponde pas à la fonctionnalité souhaitée et le délai de production qui peut ralentir le temps de leur mise sur le marché. En plus de ces problèmes, il y a aussi la demande croissante en puissance de calcul suivant l'exigence de l'application.

Face à tous ces problèmes, nous trouvons des axes de recherche qui peuvent offrir des solutions, tels que les accélérateurs matériels et le développement des méthodologies de conception. Ces derniers définissent en quelque sorte une relation entre un SoC et son concepteur.

1. Contexte général

En suivant la loi de Moore, les SoCs sont passés de petits systèmes composés de quelques IPs, à des systèmes intégrant de multiples blocs. Différentes réflexions se sont succédées, dont l'objectif est de réduire le temps de mise sur le marché, d'améliorer les caractéristiques des systèmes comme la réduction du taux de compression, d'offrir une architecture flexible et peu énergivore pour les codeurs vidéo par exemple, tels que la norme H.264 [3]. D'autres ont proposé des architectures parallèles pour augmenter le débit de traitement de données. Il s'agit dans ces architectures de segmenter le traitement des instructions et d'augmenter le nombre de ressources pour exécuter plus d'instructions simultanément.

Corrélativement, les applications de traitement de signal intensif, telles que les standards de compression vidéo, sont de plus en plus sophistiquées. Elles traitent une masse considérable de données d'où la nécessité d'unités puissantes en calcul, que seules les architectures massivement parallèles peuvent fournir, étant donné les contraintes d'exécution des tâches ainsi que le problème de l'énergie consommée. D'autre part, les réseaux sur puce deviennent actuellement le support adéquat pour la communication dans un système.

En effet, la conception des SoCs devient trop complexe. Ils doivent être conçus de manière efficace afin de prouver une certaine fiabilité et de réduire le temps de mise en marché. D'un point de vue industriel et commercial, ces deux facteurs décrivent la qualité et la réussite du produit final. En outre, ces systèmes doivent être développés et vérifiés de manière sûre pour minimiser les coûts.

Nos travaux de thèse s'inscrivent dans le cadre du traitement de signal intensif (TSI). Le TSI est composé de deux parties, une partie qui s'appuie sur la réalisation des calculs sur les données indépendamment de leurs valeurs, donc c'est un traitement régulier appelé traitement de signal systématique. Cette partie est suivie d'une partie de traitement de données intensif (TSI) plus dépendante de la valeur. Un exemple d'une application considérée comme un système à traitement de signal systématique est le H.264.

Pour répondre à ces besoins et suivre l'évolution incessante des technologies des semi-conducteurs, différents partenariats académiques multiplient leurs efforts et rassemblent différentes compétences dans le but de faciliter le développement des systèmes sur puce. Le projet Euro-med 3+3 est un projet de recherche académique, entre l'équipe « synthèse de communication » du laboratoire d'électronique

et micro-électronique de la Faculté des sciences de Monastir et l'équipe DaRT de l'Institut national de recherche en informatique et automatique de Lille. Ce projet s'articule autour de la modélisation et de la validation des architectures réseaux sur puce(MVAR) et leur application aux systèmes multimédia. Les objectifs de cette thèse font partie du cahier des charges de ce projet. Un des principaux objectifs est le Co-design de l'application H.264 et son implémentation sur un NoC qui peut être de type GALS ***(Globalement asynchrone Localement synchrone)*** en suivant le flot de conception de l'environnement Gaspard [4] développé au sein de l'équipe DaRT. Donc il s'agit d'un échange entre des informaticiens et des électroniciens pour proposer des méthodologies permettant de passer d'un niveau d'abstraction bas, qui est le niveau RTL ***(Registre Transfert Level)***, vers un niveau plus élevé qui est la modélisation en utilisant le profil MARTE ***(Modeling and Analysis of Real-Time and Embedded systems)*** [5]. Il faut tenir compte des contraintes réelles du SoC, afin de générer un code VHDL **(*VHSIC hardware description language)*** qui doit être synthétisable sur un FPGA ***(Filed-Programmable Gate Array)*** ou un ASIC ***(Application-Specific Integrated Circuits)***.

2. Problématique

L'augmentation de la complexité de conceptions des SoC fait l'objet d'une réflexion pour développer des outils de conception puissants. Nous avons donc besoin d'outils pour la conception des systèmes sur puce d'une manière plus rapide, qui permette la réduction du temps de mise sur le marché d'un nouveau produit, et d'une manière plus sûre en termes de performances. Dans ce cadre, nous avons identifié trois principaux facteurs qui définissent la problématique de cette thèse :

La méthodologie de conceptions des SoCs au niveau système.

Dans la conception des SoCs, les problèmes de tests, de programmation sûre et de réutilisation de ce qui est déjà existant sont devenus des obstacles pour la mise en œuvre d'un nouveau produit. L'ingénierie dirigée par les modèles (IDM) est considérée comme une approche de développement des SoCs, permettant d'unifier les méthodologies de conception technologique d'un système homogène en se basant sur la notion de modèle. D'après Havre [5] tout système peut être vu comme étant un modèle. En effet, afin d'identifier les différentes parties et de décrire le fonctionnement d'un système, il est possible d'utiliser les modèles permettant l'expression et la compréhension des principales tâches des systèmes étudiés. Le but de cette façon de voir est d'avantager la séparation des différents aspects du système. En effet, cette séparation mène à une augmentation de la réutilisation et facilite l'échange entre les chercheurs, ce qui influe directement sur la productivité. L'IDM présente une approche pour passer d'un niveau d'abstraction élevé, en masquant les détails qui ne sont pas pertinents pour le système, à une phase de génération de code indépendante de la technologie cible. Les concepts de l'IDM sont utilisés dans cette thèse pour résoudre les problèmes liés à la complexité de conception du système, plus précisément pour les NoCs.

Estimation architecturale et amélioration des performances au niveau RTL

L'intégration à très grande échelle ou VLSI en anglais ***« Very Large-Scale Integration »*** est une technologie de conception des SoCs dont la densité permet l'intégration d'un nombre élevé de

transistors. Les accélérateurs matériels sont décrits pour la VLSI. Ce sont des circuits dédiés pour réaliser une fonctionnalité bien déterminée dans différents domaines. Ces méthodes offrent un espace où on peut décrire le parallélisme des tâches ou des données de ce qu'on veut concevoir. Par conséquent, elles fournissent un support d'exécution optimal pour le traitement de tâches répétitives et régulières. Cependant, et lors de la phase de conception des accélérateurs, les chercheurs peuvent rencontrer plusieurs problèmes. Le coût de conception élevé, les risques d'erreurs dus à l'intervention humaine et le délai de production : sont des obstacles qu'on peut rencontrer dans le processus de conception lorsque cet accélérateur est conçu au niveau RTL dans un langage HDL ***« Hardware Description Langage »***. La complexité incessante des accélérateurs et les contraintes de temps de mise sur le marché ont encouragé le développement des environnements censés faciliter leur conception. L'automatisation des phases de conception s'avère efficace pour aider les concepteurs à la description des accélérateurs à haute performances. L'objectif majeur de ces outils est donc la génération automatique d'un code HDL à partir d'une description unifiée abstraite donc sans détails d'implémentation. On définit alors la synthèse de haut niveau. Mais ces niveaux d'abstraction ne sont pas suffisamment élevés limitant les utilisations du code généré. De plus, ces outils n'exploitent pas suffisamment le parallélisme potentiel qui peut exister dans l'application : c'est dû à l'inadaptation des langages de description. C'est pourquoi les industriels et les chercheurs se sont attachés à développer des environnements capables de tenir compte des contraintes des systèmes aux niveaux les plus élevés de la conception. Ainsi, la synthèse de haut niveau devient une phase primordiale dans la méthodologie de conception conjointe logiciels/matériels ***(Co-design)*** dans les systèmes électroniques.

Co-design application architecture et synthèse de communication

Dans l'ancienne méthodologie de conception de systèmes, les conceptions des parties logicielles et matérielles sont développées séparément. Ces phases demeurent indépendantes et ont peu d'interaction, ce qui restreint les possibilités d'échange, le partage des idées et la possibilité d'exploration de différente vision. Avec l'évolution de la technologie et l'apparition des plateformes d'exécution telles que les ASIC et les FPGA, la conception en interaction devient une nécessité. L'implémentation en logiciel ou en matériel peut être faite en fonction des besoins. Deux problèmes sont abordés dans le Co-design, d'une part la spécification du système, d'autre part le partage matériel/logiciel. La synthèse de communication est un sujet qui a fait l'objet de nombreux travaux [6] qui peuvent être classés en fonction de l'interaction entre les différentes phases de partitionnement, d'ordonnancement, d'allocation, le type d'application traitée et le choix du support cible qui représente l'architecture sur laquelle le système va être exécuté, etc. Le concepteur choisit une architecture matérielle et alloue les constituants fonctionnels sur les unités matérielles de l'architecture cible. Un choix judicieux doit satisfaire plusieurs critères tels que la flexibilité, la réutilisabilité, la sécurité etc. Après avoir choisi l'architecture cible, le problème se limite à un problème d'allocation qui peut se résoudre avec une heuristique basée sur une fonction pondérée dont les coefficients dépendent des critères retenus. Plusieurs méthodologies sont proposées dans la littérature pour manipuler le Co-design. Elles se distinguent essentiellement par : les concepts de modélisation utilisés, les modèles de l'architecture

cible et la méthode d'allocation, d'où la nécessité d'outils d'exécution qui définissent un espace de travail méthodologique pour accomplir toutes ces phases.

L'environnement GASPARD développé au sein de l'équipe DaRT permet la Co-conception des SoCs. Il permet la représentation unifiée de l'application, de l'architecture, d'allocation et du déploiement des IPs, ce qui favorise l'échange des informations et la réutilisation de ce qui est déjà fait pour un nouveau produit. Il se base sur le modèle en « Y » de Gyski[7] et s'appuie sur un cadre méthodologique guidé par L'IDM. Cette présentation commence à partir d'un niveau d'abstraction très élevé, c'est –à-dire indépendamment des détails d'implémentation, jusqu'à arriver à la génération de code. A partir d'une telle méthodologie, différentes plates-formes d'exécution sont ciblées. L'avantage de travailler dans une démarche IDM est l'unification des phases qui interviennent dans le Co-design des SoCs, l'application et l'architecture du SoC peuvent être développées d'une manière indépendante.

3. Contribution

Notre contribution dans cette thèse, concerne donc les différentes phases de Co-design d'un SoCs suivant le flot de GASPARD, à travers l'exemple de co-design de l'application H.264 sur un NoC qui peut être de type GALS. La modélisation en haut niveau, l'optimisation de performance au niveau RTL de l'architecture et de l'application, la synthèse de communication et la génération de code sont effectuées. En effet nos travaux sont :

Une méthodologie de haut niveau pour la modélisation des NoCs.

Nous proposons une méthodologie claire pour la modélisation des différents concepts des NoCs tels que la topologie, les algorithmes de routage et la technique de commutation. Le profil MARTE a été adopté en se basant sur un formalisme permettant de représenter d'une façon compacte des architectures à aspect répétitif. Nous avons commencé par la modélisation de la topologie. Pour ce faire, une méthodologie basée sur l'étude des caractéristiques, telles que la régularité, le degré etc, de la topologie a été proposée. Cette méthodologie est validée pour différentes topologies régulières afin de prouver la possibilité de modéliser les topologies irrégulières ou Globalement Irrégulières Localement Régulières.

En second lieu nous sommes passés à la modélisation des algorithmes de routage. Plusieurs heuristiques d'implémentations sont proposées dans la littérature. Une classification de ces algorithmes a été effectuée en se basant sur un modèle analytique, proposé dans la littérature, afin de modéliser à l'aide des diagrammes d'activité. Après avoir validé cette méthodologie, nous avons proposé une extension : un stéréotype « **hw_Router** » dans le package « hw_media » du profil MARTE.

L'objectif de cette méthodologie est d'aider les concepteurs à la modélisation des NoCs à l'aide du profil MARTE sans être un expert dans le domaine de la modélisation unifiée.

Modélisation à bas niveau d'un NoC

Dans cette partie, on procède à la modélisation au niveau RTL d'une nouvelle topologie, inspirée de la théorie de la littérature, appelée « Digonal Mesh ». L'architecture du routeur qui forme la brique de base de cette topologie est conçue afin de garantir certaines qualités de service telles que la flexibilité, l'extensibilité, une latence réduite et un faible coût de consommation. Cette topologie forme le support de communication de notre système.

Modélisation à haut niveau de l'application.

Le H.264 ou MPEG4-AVC ***« Advanced Video Coding »*** est un standard de codage vidéo développé en coopération par l'IUT-T et MPEG. Il correspond à une application de traitement de signal systématique intensif et régulier. Cette application traite une quantité énorme de données, d'où la nécessité de l'extraction du parallélisme potentiel de tâches. Nous avons modélisé cette application à l'aide du profil MARTE en exploitant les concepts de base du package RSM **« Repetitive Structure Modeling »** en vue d'extraire ce parallélisme afin de répondre à la contrainte temps réel de l'application.

Modélisation à bas niveaux des composantes de l'application.

Le codeur vidéo H.264 est composé par plusieurs unités qui traitent une quantité importante de données, telles que l'estimateur de mouvement, la transformée en cosinus discrète (DCT), la quantification(Q) le codage entropique (CAVLC), et la compensation de mouvement. Ces éléments sont conçus en VHDL dont l'objectif est de fournir la brique de base pour pouvoir exprimer le parallélisme potentiel. Dans le même cadre, des architectures flexibles sont proposées et répondent aux exigences de l'application. Elles offrent un faible coût de consommation en énergie par rapport à d'autres architectures proposées dans la littérature, à l'aide de l'intégration des unités de contrôle distribuées. Egalement, nous proposons deux architectures d'un estimateur de mouvement à taille de bloc fixe et variable pour l'algorithme de recherche complet **« Full Search Block Matching (FSBM)**».

Co-design application architecture suivant le flot de GASPARD et évaluation de performance

Dans cette partie, les phases du Co-design suivant GASPARD sont abordées. Après avoir modélisé l'application et l'architecture en haut niveau à l'aide du profil MARTE, nous sommes passés à la modélisation, en MARTE, de l'association en utilisant le stéréotype **« Distribute »** et le déploiement des IPs, afin de favoriser leur réutilisation dans le flot de Co-design, par la mise en place d'une bibliothèque de composants modélisés en VHDL. L'allocation des IPs sur les routeurs a été faite en décrivant un modèle analytique qui représente une fonction de coût de communication. Finalement, une évaluation de performances a eu lieu.

Le diagramme ci-dessous présente le flot de conception de la thèse.

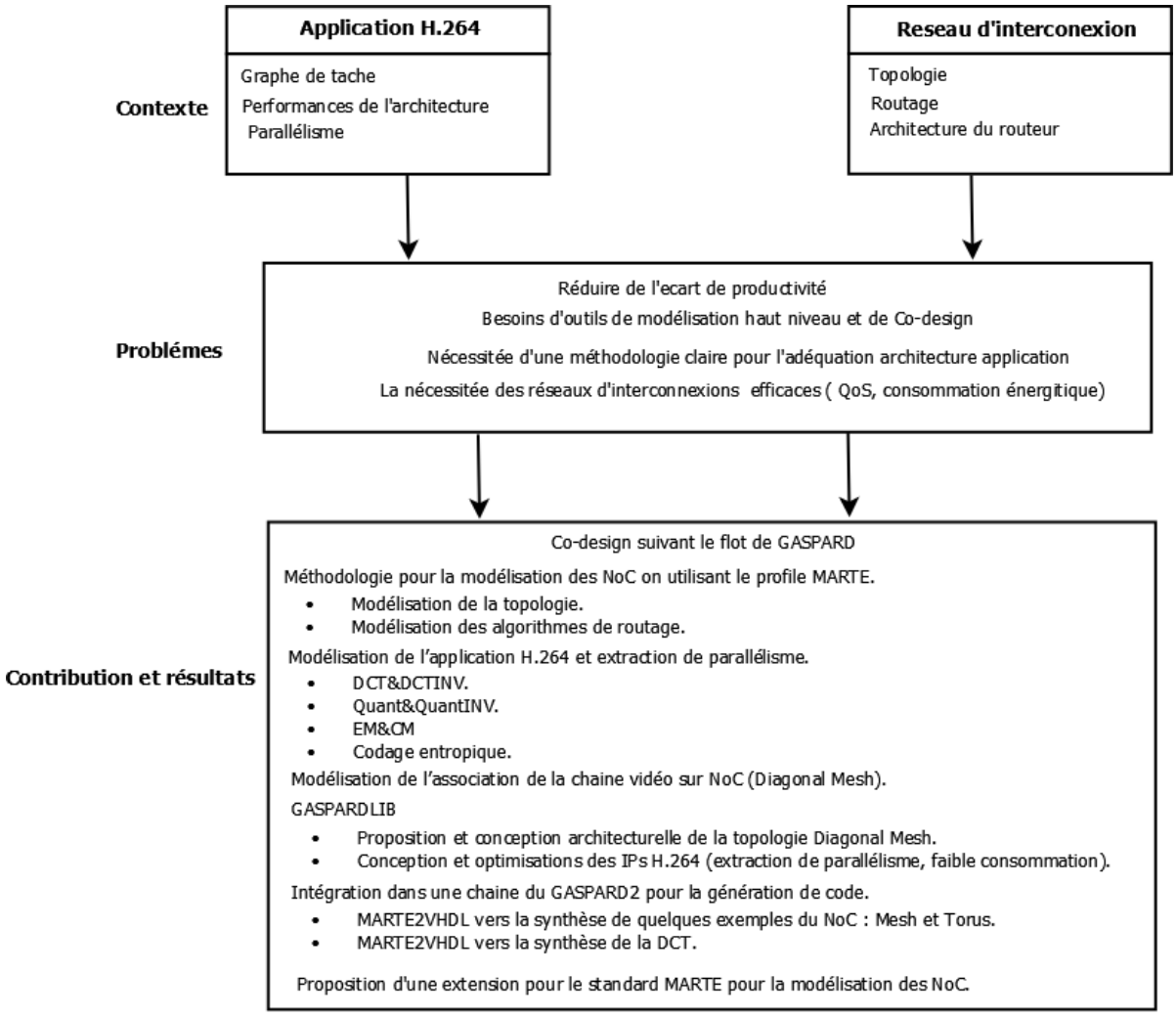

Figure 1. 1 : Flot de conception de la thèse.

4. Plan de la thèse

Le reste de notre manuscrit est organisé selon le plan suivant :

Chapitre 2 : il est consacré à l'état de l'art : dans ce chapitre nous présentons une vue globale sur les thématiques abordées dans cette thèse. A partir de cette étude, nous présentons les motivations de cette thèse et nous positionnons nos travaux pour donner les grandes lignes de nos contributions.

Chapitre 3 : Il porte sur la modélisation des concepts de base des NoCs dans une démarche IDM. Nous présentons une vue globale de la méthodologie proposée. Nous détaillons la nécessité de cette méthodologie, ses caractéristiques et la validation par quelques exemples afin de prouver la nécessité de l'extension du standard.

Chapitre 4 : Dans ce chapitre, nous détaillons l'architecture du routeur conçu pour construire notre NoC. Nous éclaircissons tous les détails et les paramètres architecturaux afin d'implémenter le routeur sur un FPGA et de prouver ses performances par une étude comparative.

Chapitre 5 : il décrit la modélisation et la conception architecturale du codeur H.264 à faible coût de consommation jusqu'à l'association sur un NoC. Nous commençons par la modélisation en haut niveau dont l'objectif est d'extraire le parallélisme potentiel qui peut exister. Nous formulons un modèle analytique pour justifier la phase d'allocation afin de le modéliser en MARTE. Nous détaillons par la suite les architectures de tous les composants qui forment l'encodeur H.264 et nous proposons des architectures optimales en termes de consommation d'énergie consolidée par des résultats expérimentaux.

Chapitre 6 :
Ce chapitre présente la partie expérimentale sur GASPARD. Nous expliquons les différents concepts du paquetage Deployment, le GaspardLib et un exemple d'intégration et génération de code.

Chapitre7 : Nous concluons nos travaux par un bilan et nous détaillons les contributions apportées afin de définir quelques perspectives de cette thèse.

Chapitre2
Etat de l'art

1. Introduction.

Ce chapitre traite tous les points clés pour la modélisation et la Co-conception des systèmes sur puces. Il familiarise le lecteur avec le contexte de nos travaux. L'objectif principal de notre travail est la contribution à la mise au point d'un outil permettant d'éclaircir le flot de Co-conception des systèmes sur puce afin d'aboutir à des performances. Nous présentons alors des outils de conception qui permettent la génération des accélérateurs. Dans la même optique, les systèmes sur puce et la notion de Co-conception sont étudiés. Ensuite nous passons à une introduction au traitement de signal systématique et à la modélisation multidimensionnelle. La modélisation des structures répétitives en MARTE, la Co-conception suivant GASPARD et la modélisation haut niveau des NoCs sont introduits dans la deuxième partie.

2. La Co-conception des systèmes sur puce

Depuis le début des années 2000, les systèmes sur puce ont émergé comme un nouveau paradigme pour les systèmes embarqués. Un SoC peut contenir plusieurs éléments tels que les processeurs, les mémoires, les périphériques d'entrée sortie. La conception des SoCs peut être vue de plusieurs côtés, y compris la modélisation par l'agrégation des éléments fonctionnels, la vérification du système et l'implémentation sur un ASIC ou un FPGA. Suite à ces aspects, la complexité de conception des SoC devient difficile à gérer.

En 1983, Gajski [7] a présenté le modèle en Y pour expliquer les phases de conception des circuits VLSI. Suivant le modèle de Gajski, un système est décrit selon trois vues : structurelle, comportementale et physique. Dans la littérature, plusieurs travaux s'appuient sur ce modèle, dont le but est d'organiser les phases de la Co-conception dans un système sur puce. La figure 2.1 détaille cette organisation. Les trois axes représentent les différents points de vue de description. Les cercles représentent les niveaux d'abstractions. Chaque point de vue possède plusieurs niveaux d'abstraction. Les 3 axes sont définis de la manière suivante :

Le niveau comportemental : décrit la fonctionnalité du système indépendamment de la technologie cible.

Le niveau structurel : dans ce domaine, le système est un assemblage de sous- composants qui sont interconnectés entre eux.

Le niveau physique : c'est le niveau le plus bas dans lequel en décrit le système à l'aide de transistors.

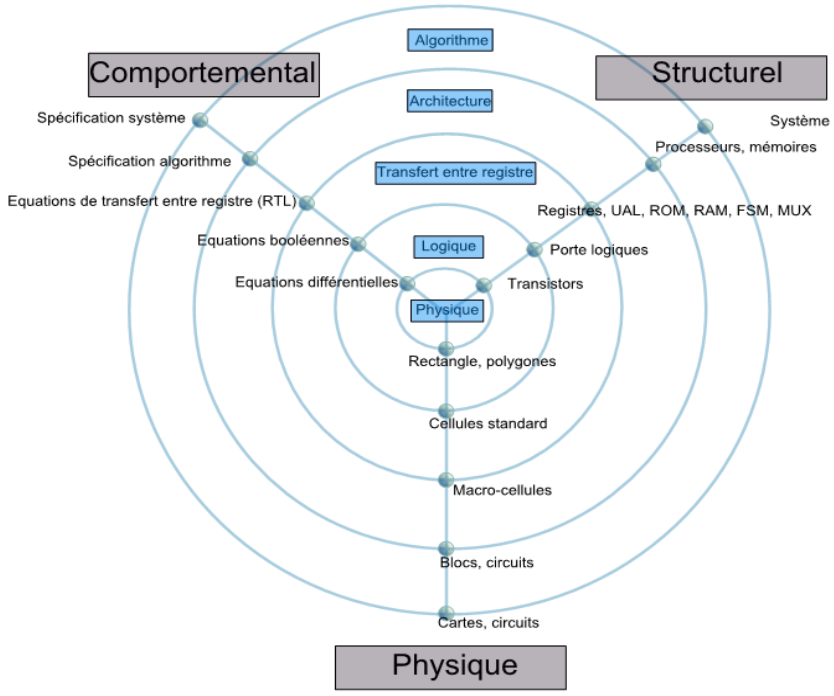

Figure 2. 1: la Co-conception suivant Gajski

Ce modèle en Y, référencé par plusieurs travaux [8, 9, 10, 11], favorise une séparation entre la modélisation de l'architecture, celle de l'application et celle de l'allocation.

2.1. Co-conception conjointe logicielle/matérielle

La conception conjointe logiciel-matériel consiste à choisir une implémentation, pour chaque tâche de l'application, qui peut être matérielle ou logicielle. Une tâche peut être conçue en matériel à l'aide d'une description dans un langage HDL afin de l'implémenter sur un FPGA ou un ASIC, tandis qu'une conception logicielle consiste en une implémentation programmée sur des processeurs. Ce compromis est difficile à gérer, d'autant plus qu'il peut faire intervenir plusieurs facteurs tels que les exigences des utilisateurs, le temps de mise sur le marché, le prix, etc. Le problème de la Co-conception touche plusieurs domaines, à savoir la télécommunication mobile, l'automobile, l'avionique etc.

La conception de systèmes sur puce contenant des modules matériels et logiciels n'est pas un problème nouveau. Les approches traditionnelles consistent à réaliser tout le système en matériel, ce qui conduit à des coûts de fabrications très élevés. De même, un système conçu totalement en logiciel est dédié à des exécutions sur des processeurs, ce qui diminue sa vitesse. D'autres approches proposent une conception commençant par traiter le matériel et passant par la suite au développement logiciel. Les récents travaux ont amené une ré-observation des problèmes de frontières entre le logiciel et le matériel d'où la naissance de la nouvelle vision basée sur la conception mixte matérielle/logicielle. Plusieurs définitions ont été proposées pour la Co-conception :

En 1991, Franke [12] définit la Co-conception comme étant un processus qui combine le matériel et le logiciel dès les premières phases de conception afin d'exploiter la flexibilité du produit.

En 1994 Wolf [13] indique que le matériel et le logiciel doivent être conçus en parallèle afin de garantir le bon fonctionnement du système et ses performances.

En 1995 Stoy [14] ajoute que le but de cette approche est de former une méthodologie unique pour la conception. Cela signifie que la conception mixte est vue comme une unique entité.

En 1997 Edward [15] affirme que le but est de produire un système équilibré entre les composants matériels et logiciels qui fonctionnent ensemble pour réaliser un comportement bien défini.

En 1997 selon De Micheli [16], la Co-conception permet de répondre aux défis rencontrés par les nouveaux systèmes en exploitant la cohérence entre le matériel et le logiciel.

La Co-conception est une discipline qui croit exponentiellement avec les progrès effectués dans les domaines de la compilation logicielle et de l'implémentation matérielle. Le processus de co-conception dans les différents travaux cités précédemment, suit un certain nombre de phases. La figure 2.2 illustre les phases typiques de cette approche.

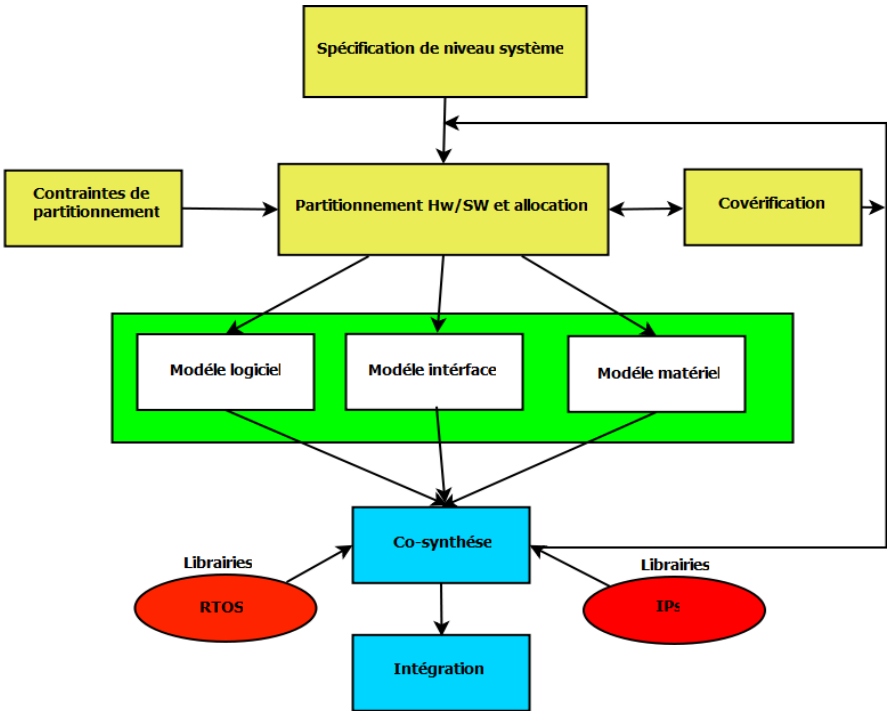

Figure 2. 2 : phases de conception conjointe matérielle/logicielle

Commençant par une phase de spécification mixte décrivant le système à concevoir dans sa totalité, le processus comporte ensuite les phases suivantes :

Le partitionnement consiste à trouver une répartition convenable entre matériel et logiciel pour chaque tâche du système. Il s'agit de chercher le meilleur compromis, en se basant sur des contraintes du système, entre l'implémentation en matériel ou en logiciel pour chaque composant du système. Les contraintes peuvent être de type fonctionnel ou non fonctionnel à savoir le temps d'exécution, l'espace mémoire, le coût de développement, la surface, le degré de parallélisme, la consommation d'énergie, etc. Le système partitionné doit être validé avant d'entamer la phase suivante. Des lignes de retours sont nécessaires tant qu'une solution judicieuse n'a pas été obtenue.

Une fois le partitionnement achevé, la phase de synthèse aura lieu. On doit effectuer un choix des composants pour l'implémentation du matériel et la synthèse du logiciel.

Cette phase de synthèse doit tenir compte de l'aspect communication et de la synchronisation entre les différents éléments qui forment le système. Si on ne tient pas compte de l'aspect communication, un goulot d'étranglement peut se produire. Cette phase est la synthèse de communication.

Le système est simulé afin de prouver que les contraintes spécifiques de l'application sont vérifiées et que le système décrit jusqu'à cette phase présente le fonctionnement souhaité et répond aux exigences de temps réel.

La dernière phase est le prototypage physique du système sur l'architecture cible choisie.

2.2. La Co-conception application/architecture suivant le modèle en Y

De ce point de vue l'application et l'architecture peuvent être conçues d'une manière parallèle et c'est durant la phase l'allocation que les deux peuvent s'entrelacer. La figure 2.3 explique mieux cette nouvelle organisation [17]. De ce fait, la conception de l'application peut se faire indépendamment de l'architecture qui fournit le support de communication. En effet, une grande réduction du temps de conception résulte de la séparation des tâches.

Après avoir terminé la spécification des deux parties (architecture et application), une phase d'allocation aura lieu. Il s'agit habituellement de placer les unités de l'application sur les différents types de processeurs. Dans notre cas, cela va être un placement des IPs de l'application H.264 sur les routeurs qui forment le NoC. Cette phase d'allocation est à la fois topologique (géométrique) et temporelle (ordonnancement). Une fois l'allocation terminée, la description électronique du système est terminée. Ensuite une phase de vérification et de validation est nécessaire pour étudier les performances du système. Pour ce faire, plusieurs paramètres doivent être observés :

- S'assurer du bon fonctionnement de système à l'aide d'un scénario de simulation qui tient compte des entrées exigées par l'application.
- Le temps d'exécution doit répondre à la contrainte temps réel de l'application.
- Des critères physiques sont reliés à la surface en silicium et à la consommation énergétique.

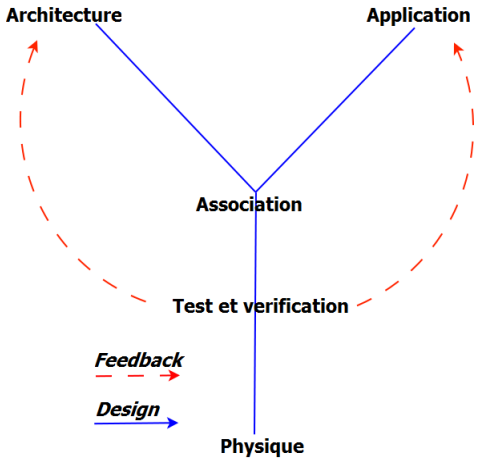

Figure 2. 3: flot de conception des SoCs en modèle de Y

Au cours de cette thèse, nous adoptons ce flot pour la Co-conception de l'application H.264 sur un NoC. Orthogonalement à ces approches, la conception des accélérateurs au niveau RTL a fait naître des outils qui permettent de générer des accélérateurs surtout pour les applications multimédia, gourmandes en termes de calcul. Nous détaillons quelques outils dans les sections qui viennent.

2.3. Conceptions des accélérateurs matériels assistés par des outils

Dans cette section, nous présentons quelques outils pour la génération des accélérateurs matériels dédiés aux applications à calcul intensif. Habituellement, les accélérateurs matériels sont conçus au niveau RTL et écrits dans un langage de description matérielle comme le VHDL ou le verilog. Plusieurs travaux dans la littérature présentent de telles descriptions. Cette conception est toujours associée à des risques d'erreur à cause de l'intervention humaine. De plus, l'exploration de l'architecture est souvent freinée par une structure rigide définie par le concepteur. Le niveau de description (structurel ou comportemental) influe directement sur la validation du bon fonctionnement de chaque composant qui forme notre système, ce qui se reflète sur le temps de conception. Malgré ces problèmes de conception, les accélérateurs matériels restent fréquemment utilisés dans le domaine de traitement de signal intensif. Cependant, les chercheurs ont recours à des outils pour la réalisation des accélérateurs dédiés.

2.3.1. MMAlpha

L'environnement MMAlpha [18] est une interface pour le logiciel Mathematica [19] qui peut compiler des programmes dans le langage ALPHA. C'est un langage à parallélisme de données [20, 21, 22] créé par l'équipe API de Rennes. Ce langage est fondé sur des équations formelles récurrentes [23, 24]. Le langage ALPHA a subi des évolutions pour être utilisé dans le cadre de la synthèse des architectures. Dans le langage ALPHA, on ne tient pas compte de la notion de temps. Il se base sur le calcul, de sorte que les dépendances de données expriment l'ordre dans lequel ces calculs peuvent être réalisés. C'est un langage qui se base sur la manipulation de tableaux dont les variables sont des tableaux de formes variables. La modification apportée au langage alpha donne naissance à ALPHA0 et ALPHARD [25,

26]. Un programme ALPHA0 est généré depuis un programme ALPHA. La génération de l'architecture est reliée à des équations du système ALPHA et interprétée de manière à lui faire correspondre un élément architectural. Un programme ALPHARD est généré depuis un programme ALPHA0.

2.3.2. SDF : Synchronous Data Flow

SDF est un modèle de calcul à flot de données qui permet la description des algorithmes du traitement de données. Cette description se fait à l'aide d'une sémantique basée sur la notion des nœuds et des arcs [27]. Une application est écrite sous la forme d'un graphe orienté dont chaque nœud consomme et produit des données. Williamson en 1998 [28] a dirigé des travaux qui ont permis la génération de code VHDL depuis un modèle de calcul SDF. Cependant, ce modèle de calcul peut manipuler une seule dimension alors que dans le domaine du traitement de signal il y a la gestion de tableaux multidimensionnels. En 2006 Filiba [29] génère du code VHDL, depuis l'environnement PTOLEMY, réalisant des tâches simples telles qu'un additionneur, soustracteur, etc. Par contre, ce n'est pas possible d'implémenter les dépendances de données.

Plusieurs autres outils sont cités dans la littérature permettant la conception et la génération des accélérateurs matériels. L'objectif est de faciliter, d'automatiser cette génération en partant d'une spécification à haut niveau, à l'aide de l'exploitation des modèles de calcul et de faire les transformations nécessaires jusqu'à la génération du code de l'accélérateur.

2.4. Adéquation Algorithme Architecture (AAA)

Cette approche d'Adéquation Algorithme Architecture est basée sur une modélisation graphique pour décrire d'une part l'algorithme (application) et d'autre part l'architecture qui en est le support matériel. La modélisation de l'algorithme permet l'extraction du parallélisme potentiel de l'application. Le graphe décrivant l'algorithme subit des transformations jusqu'à ce qu'il atteigne le graphe modélisant l'architecture. Toutes ces transformations correspondent à une distribution et à un ordonnancement des différents IP sur les processeurs et des communications sur l'infrastructure de communication. Donc, après cette phase d'allocation spatiale et temporelle, un exécutif est généré permettant l'exécution de l'application sur l'architecture. Cette méthodologie est intégrée dans l'outil SynDEx.

2.4.1. SynDEx

SynDEx [30] présente la méthodologie « Adéquation Algorithme Architecture ». C'est un environnement cohérent de co-conception et d'implémentation. Il permet la distribution et l'ordonnancement optimisés d'une représentation graphique correspondant à la fois à un graphe flot de données et à un graphe d'architecture matérielle. L'allocation ou l'adéquation est effectuée en fonction des contraintes fonctionnelles du système telles que le type des processeurs à adopter et le support de communication etc. Afin d'ouvrir le chemin à la génération d'accélérateurs matériels, des travaux ont proposé une extension de SynDEx pour le rendre capable de générer des codes HDL, d'où l'apparition de SynDEx-IC.

2.4.2.SynDEx-IC : environnement pour la génération de code.

C'est un environnement [31, 32] d'aide au prototypage rapide d'application sur des ASICs ou des circuits reconfigurables de type FPGA. Les architectures supportées sont de type « multiprocesseur hétérogène » à savoir les DSP, les ASICs dédiés, les microcontrôleurs. L'objectif est de trouver une adéquation judicieuse qui respecte les contraintes de l'application. Les phases de prototypage sur SynDEx-IC sont illustrées dans la figure 2.4 :

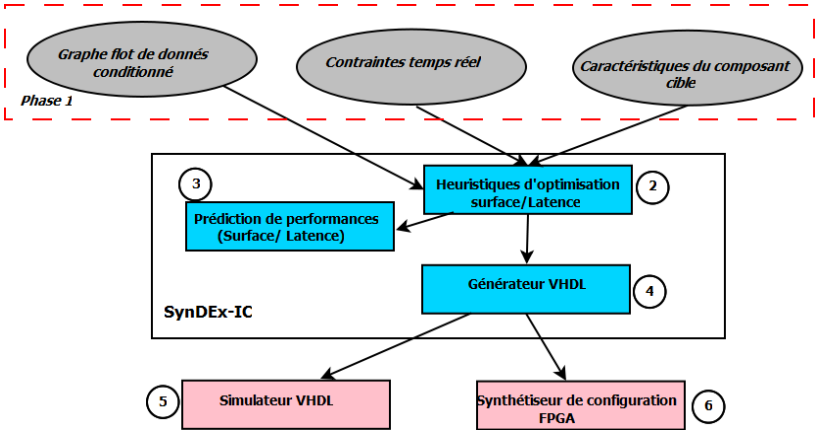

Figure 2. 4: Phase de prototypage sur SynDEx-IC[31]

La phase 1 est la spécification graphique, conditionnée, de l'algorithme à implémenter. Cela permet l'extraction du parallélisme potentiel de l'application et l'exploitation de la régularité des données à traiter, dans l'objectif de faire un traitement répétitif. Le graphe peut être factorisé à l'aide de l'expression d'une répétition sur une tâche. Les tâches factorisées sont décrites par des nœuds de factorisation. Afin de spécifier les caractéristiques du composant cible, ces caractéristiques, correspondant à chaque élément du graphe de l'algorithme, peuvent être de type latence et quantité de ressource à utiliser.

(2) Choix et exécution d'une des heuristiques de SynDEx-IC : deux familles d'heuristiques sont disponibles :

- Les heuristiques de recuit simulé qui exploitent plus de solutions pour avoir une optimisation plus fine mais sont plus lentes.
- Les heuristiques gloutonnes, plus rapides que les premières, elles permettent une implantation qui respecte des contraintes données.

(3) Prédiction de performances : elle permet d'afficher les paramètres correspondant à l'heuristique choisie et les durées d'exécution calculées.

(4) Synthèse automatique du code VHDL : L'environnement SynDEx-IC génère l'ensemble du code VHDL nécessaire à l'exécution de l'algorithme spécifié par des graphes. Il est capable de synthétiser toute la partie de contrôle, des parties répétées du graphe et des parties conditionnées.

(5) Simulation du code généré.

(6) Implémentation sur un FPGA.

Finalement nous remarquons très bien l'importance de la conception des systèmes sur puces. Depuis une vingtaine d'années, plusieurs travaux ont examiné les problématiques reliées à cette phase importante avant de déposer un nouveau produit.

2.5. La méthodologie MCSE

Cette méthodologie (Méthodologie de Conception des Systèmes Electroniques) définit une solution possible basée sur un schéma d'organisation pour la conception d'un système électronique à temps réel. Elle conduit directement à la conception et à la réalisation des systèmes mixtes [33]. La co-conception suivant la méthodologie MCSE est caractérisée par une modélisation fonctionnelle, comportementale et architecturale. Le développement selon MCSE suit 4 étapes comme le montre la figure 2.5.

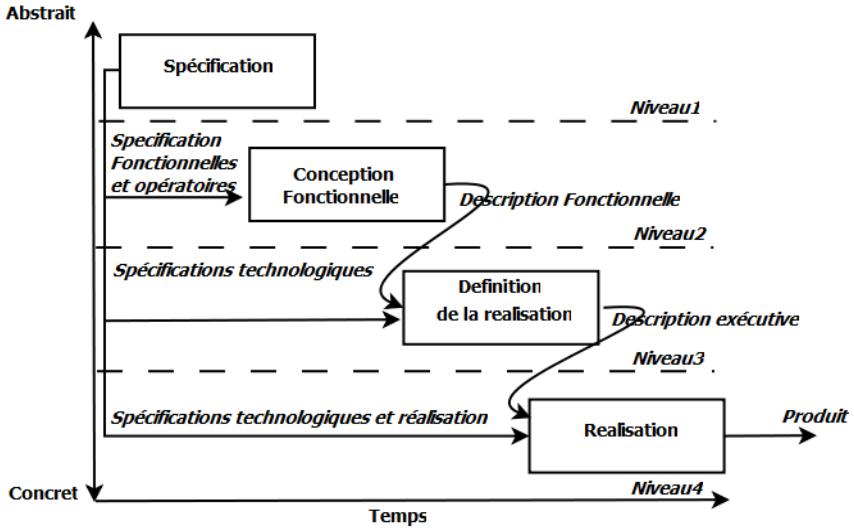

Figure 2. 5: démarche de développement suivant MCSE [33].

- Etape de spécification : elle permet la description complète du système à concevoir en suivant le cahier des charges fourni.
- Etape de conception fonctionnelle : elle a pour objectif de composer le système sous la forme d'un ensemble de composants communicants.
- Etape de définition de la réalisation : elle définit le partitionnement matériel/logiciel et la spécification de la plateforme cible.
- Etape de réalisation : elle consiste à développer le matériel et le logiciel tout en tenant compte des étapes précédentes.

La Co-conception dans cette méthodologie s'intègre comme un sous ensemble de l'étape 3. La figure 2.6 représente les différentes phases de la co-conception dans une démarche MCSE.

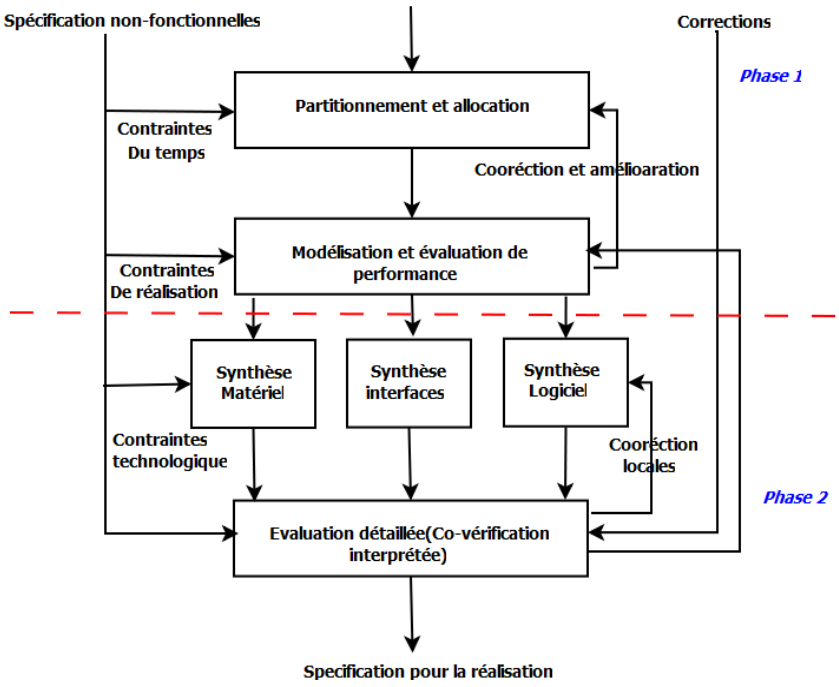

Figure 2. 6: Co-conception dans une démarche MCSE.

Comme la montre la figure2.6, la démarche est décrite par 2 phases :
Phase1 : *partitionnement et allocation, évaluation*

- Permet de décrire la solution fonctionnelle en une partie logicielle et une partie matérielle.
- Spécification de l'architecture matérielle cible et de l'allocation des tâches sur les composants.
- Vérification et évaluation de la solution tout en tenant compte des caractéristiques non fonctionnelles du système.

Phase2 : Synthèse, génération, évaluation.

- Description et synthèse architecturale.
- Définition de la partie logicielle.
- Spécification des interfaces.

Le modèle fourni après une démarche MCSE est écrit dans une forme textuelle qui peut être le fichier d'entrée d'un générateur de code (MCSE-GEN) qui le traduit en un code VHDL. La simulation du code VHDL génère un fichier décrivant l'évolution des fonctions, ce fichier lui-même représente l'entrée pour un outil d'analyse de performances (MCSE-PERF). Le générateur de code (MCSE-GEN) pourra produire automatiquement 60% du code complet. Le reste doit être fait par le concepteur.

2.6. Bilan de la Co-conception

Actuellement, les efforts de la communauté du Co-conception s'intéressent essentiellement au développement de méthodologies claires pour la conception, permettant la réduction du gap entre l'évolution de la technologie des semi-conducteurs et des outils de conception des systèmes

numériques. Suivant la présentation des différentes façons de voir la Co-conception, il s'est avéré que la Co-conception n'est pas une partie isolée de la conception de la totalité du système. Les méthodologies présentées dans cette partie du chapitre se distinguent essentiellement par :

- Les concepts, utilisés durant toute cette phase, de la modélisation jusqu'à la spécification du système au produit final.
- La méthodologie suivie et son outil support.
- La façon de modéliser l'application, l'architecture et le modèle d'exécution.

Dans cette thèse, la méthodologie de Co-conception utilisée est celle basée sur le modèle en Y. Elle représente le flot de travail de l'outil GASPARD que nous allons utiliser pour notre système.

3. Le traitement de signal intensif

Le traitement de signal (TS) est la branche qui s'intéresse à l'étude et au développement des techniques de traitement, d'analyse et d'interprétation des signaux. Le signal est défini comme étant le support de l'information qui peut subir différentes opérations suivant le domaine d'application. Parmi les opérations possibles, on cite la compression, la quantification, le filtrage, la transmission, etc. Dans le traitement de signal, on trouve le sous-ensemble tel que le traitement de signal intensif (TSI) qui est le plus gourmand en termes de calcul. Il se compose d'une partie de traitement de données intensif (TDI) et une partie de traitement de signal systématique (TSS). Ce dernier consiste à réaliser des calculs sur les signaux indépendamment de leur valeur. Par contre, le TDI est caractérisé par un traitement irrégulier à cause de la dépendance à la valeur de données. Le TDI analyse l'information issue du traitement systématique, qui peut être utilisé pour prendre des décisions. On trouve le TSI dans différents champs d'application allant du domaine de traitement vidéo aux communications satellites, aux transformations fréquentielles, etc. Ce type d'applications est souvent embarqué dans un système.

3.1. Modélisation multidimensionnelle dans les SoC

La complexité des applications du traitement de signal est due à la quantité énorme de données à manipuler, à la façon d'accéder aux tableaux intermédiaires et à l'enchaînement des tâches qui forment l'application. La manipulation du parallélisme dans ces applications s'avère importante du fait qu'on a besoin de systèmes avec une grande puissance de calcul. Dans un système sur puce, plusieurs composants sont intégrés tels que les mémoires, les processeurs et l'infrastructure de communication qui peut être de type NoC. Dans la conception des SoC, la modélisation par l'assemblage des unités et l'agrégation des composants fonctionnels permet de simplifier et d'identifier les différents composants du système. La structure des données dans ces SoC est représentée à l'aide de tableaux multidimensionnels. L'image dans une séquence vidéo est manipulée comme étant un tableau à 2 dimensions. Dans la modélisation multidimensionnelle, la définition des modèles de calcul est nécessaire pour l'extraction et l'exploitation du parallélisme. Cela facilite l'accès aux tableaux de données aussi bien que l'ordonnancement de ces applications avec les ressources envisagées. Plusieurs modèles de calcul sont définis pour fournir une manière pertinente permettant la modélisation multidimensionnelle.

3.2. Array-OL

Array-OL (Array Oriented Langage) est un modèle de calcul inventé par Alain Demeure en 1995 [33], développé chez ThALES. Array-OL permet la spécification d'applications de traitement de signal intensif, qui traitent une quantité énorme de données. Il n'exprime pas la fonctionnalité d'une tâche mais s'intéresse à la modélisation des dépendances de données et au parallélisme de tâches ou de données. Le but principal d'Array-OL est de proposer un langage mixte graphique/textuel pour la modélisation des applications multidimensionnelles de traitement de signal intensif.

Les tâches dans une application de traitement de signal intensif consomment et produisent des tableaux. Ces tableaux peuvent être des tableaux contenant les données à traiter : ce sont des tableaux de taille finie. Les tableaux consommés ou produits sont découpés en sous- tableaux de même taille, on les appelle motifs.

A partir des exigences de ce domaine, le formalisme Array-OL est basé sur les principes suivants [34]:

- Le parallélisme de tâches et de données doivent être spécifié dans la spécification.
- Array-OL est un langage qui permet l'expression des dépendances de données afin d'extraire le parallélisme de l'application.
- L'assignation est unique dans Array-OL, cela veut dire qu'aucune donnée n'est écrite deux fois mais elle peut être lue plusieurs fois.
- Les accès aux données sont faits par des sous-tableaux appelés motifs.

Une application Array-OL est définie par l'enchainement des tâches connectées par des ports. Les tâches reçoivent et émettent des données, respectivement, sur leurs ports d'entrées /sorties. Les tâches en Array-ol sont classées en trois types :

(1) Une tâche élémentaire est vue comme une boite noire, elle peut être un composant dans une bibliothèque.
(2) Une tâche composée est décrite par un graphe qui exprime la dépendance et dont les sous-tâches sont connectés par des ports.
(3) Une répétition exprime comment une seule sous- tâche est répétée.

3.2.1. Le parallélisme de tâches.

Le parallélisme de tâches est représenté à l'aide d'un graphe orienté acyclique dans lequel les tâches sont spécifiées par des nœuds et les tableaux par des arcs. Il n'existe aucune relation entre les formes des ports d'entrée et de sortie d'une tâche. Cela veut dire qu'une tâche peut consommer deux tableaux bidimensionnels et produire un tableau à N dimensions. La figure 2.7 illustre un exemple du parallélisme de tâches.

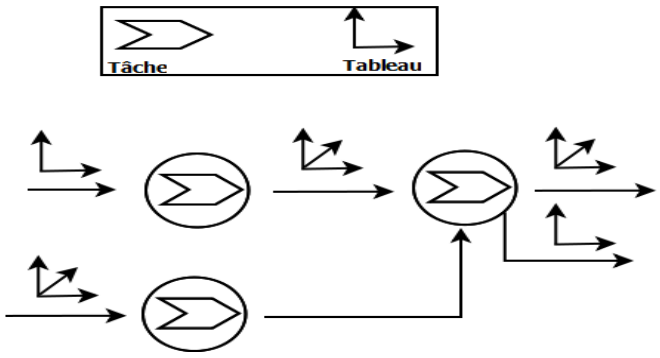

Figure 2. 7: Exemple de parallélisme de tâches.

3.2.2. Le parallélisme de données

La figure 2.8 montre un exemple de parallélisme de données défini par un modèle Array-OL qui exprime la répétition d'une tâche. L'identification des dépendances de données est réalisée par des tableaux d'entrée/sortie. Les tâches répétées dans l'espace de répétition se basent sur la manipulation des tableaux. Chaque répétition peut consommer ou produire, par son port d'entrée/sortie, des sous-tableaux. Tous les éléments des sous-tableaux sont uniformément espacés.

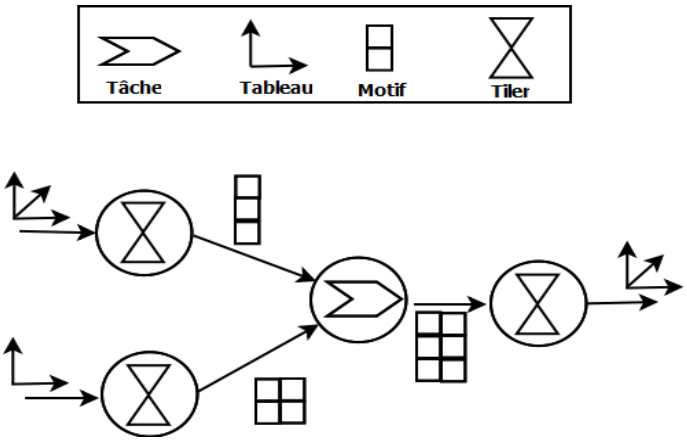

Figure 2. 8: Exemple de parallélisme de données.

Cette description permet une représentation répétitive et compacte du parallélisme de données dans un modèle Array-OL. Les sous-tableaux d'entrée/sortie sont appelées motifs lorsqu'ils sont conformes à un motif de référence, et tuiles lorsqu'ils sont considérés comme les éléments des tableaux de la tâche répétée. Un tiler correspond à la modélisation d'un lien entre les ports d'une tâche répétée, permettant la création de ces motifs. Il est défini par les éléments suivants :

(1) F (Fitting en anglais), qui est une matrice d'ajustage.
(2) P, qui définit la matrice de pavage.
(3) O, c'est un vecteur qui définit l'origine du motif de référence.

Le formalisme Array-OL est conçu spécialement pour les applications à une grande puissance de calcul telles que les applications de traitement de signal intensif, ou aussi pour modéliser les architectures à

structure répétitive. Dans la section suivante, nous étudions l'importance du formalisme d'Array-OL dans la modélisation des répétitions dans un système en utilisant les paquetages fournis par le profil MARTE d'OMG.

4. MARTE: Modeling and Analysis of Real Time and Embedded Systems.

Vu la complexité des systèmes sur puce, une abstraction permet la simplification, la compréhension et la décomposition du système en sous-systèmes faciles à gérer. Dans l'esprit de la modélisation, tout système est considéré comme étant un modèle [35]. Ce modèle décrit certaines caractéristiques du système indépendamment de la technologie et des détails d'implémentation. Donc la modélisation permet de masquer les détails qui ne sont pas pertinents et qui peuvent ralentir le processus de conception. Plusieurs propositions ont permis la manipulation des modèles ; SSADM (Structred Systems Analysis and Design Methodology) [36] UML (unified Modeling Language) [37]. En se basant sur cette approche d'abstraction, les concepteurs peuvent se concentrer sur un domaine particulier à l'aide d'une modélisation visuelle qui améliore la communication entre le système et son concepteur.

MARTE est un profil UML proposé par l'OMG. Son objectif est la modélisation des systèmes embarqués à temps réel. Il permet la modélisation de l'application, l'architecture et l'allocation. Il fournit une extension d'UML dont l'objectif est de pouvoir faire l'analyse des performances, de tenir compte de la plateforme cible et de la modélisation de l'ordonnancement. La figure 2.9 montre l'architecture globale du profil MARTE selon une décomposition en paquetage. Dans cette architecture, la séparation entre la modélisation des concepts de système et l'annotation des modèles permet l'analyse des propriétés du système. Cette séparation est faite par les deux paquetages MARTE design model et MARTE analysis model.

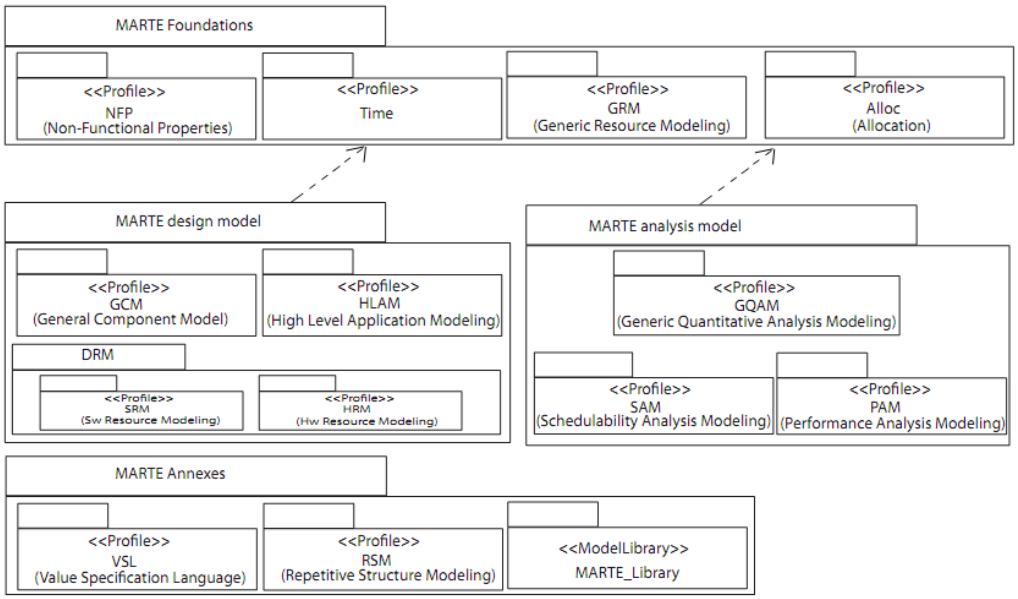

Figure 2. 9: Architecture globale du profil MARTE [38].

Le standard MARTE est composée de quatre paquetages principaux :

Fondations : ce paquetage fournit des notions fondamentales pour les systèmes embarqués. Il définit des modèles permettant la mise œuvre des propriétés non fonctionnelles *(Non-Functional Properties-NFPS)*. Il intègre des notions de temps *(Time)*, de ressources abstraites *(Generic Resource Modeling-GRM)*. Ce paquetage contient les notations utiles pour la modélisation de plateformes d'exécution des systèmes embarqués. Le paquetage *(GCM)* englobe les concepts de base pour la modélisation en utilisant la notion de composant. Finalement il y a le paquetage *(Alloc)* permettant la modélisation de l'allocation.

MARTE design model : contient les concepts reliés à la modélisation générale des composants *(General Component Model- GCM)* et à la modalisation à haut niveau de l'application *(High Level Application Modeling-HLAM)*. Les sous-paquetages de la modélisation de la plateforme logicielle *(Software Resource Modeling- SRM)* et matérielle *(Hardware Resource Modeling-HRM)* sont intégrés dans le paquetage *(Detailed Modeling Resource-DRM)*.

MARTE analysis model : Composé par le paquetage (Generic Quantitative Anaylisis Modeling-CQAM) et ses sous-paquetages *(Performance Analysis Modeling -PAM)* et *(Scheduling Analysis Modeling -SAM)*. Il permet l'analyse des performances des systèmes modélisés.

MARTE annexes : L'annexe de MARTE contient trois sous-paquetatges ; le VSL *(Value Specification Language)*, le RSM *(Repetitive Structure Modeling)* et *MARTE_Library*. Le sous-paquetage RSM décrit les concepts reliés à la modélisation des structures répétitives. Il est inspiré du modèle de calcul Array-OL. La bibliothèque *MARTE_Library* définit les opérations prédéfinies utilisées dans les systèmes embarqués.

Le profil MARTE fournit plusieurs avantages tels que la modélisation à haut niveau d'abstraction, l'analyse de performances et principalement la séparation claire entre les composants logiciels et matériels. Cette séparation favorise la modélisation séparée de l'application et l'architecture, ce qui permet la Co-conception architecture/ application.

Dans la section suivante, nous détaillons le paquetage RSM qui est utilisé dans cette thèse pour la modélisation de l'architecture, l'expression des répétitions et l'extraction du parallélisme potentiel de l'application.

4.1. Modélisation des structures répétitives en MARTE

Le paquetage RSM de MARTE présente la possibilité de l'expression compacte des architectures ayant une structure répétitive telles que les NoC ou les MPSoCs. De plus il fournit les annotations nécessaires pour l'expression du parallélisme potentiel de l'application. Les structures répétitives sont décomposées en sous-composants interconnectés par des motifs réguliers. Les mécanismes fournis sont classés en deux sortes :

(1) La spécification de la forme *(Shape)* d'une répétition par un tableau et la représentation des liens sous une forme multidimensionnelle. Par ailleurs, ce mécanisme facilite la description des liens physiques de la topologie et augmente le pouvoir de description de la structure répétée.

(2) Une spécification des informations topologiques sur des relations entre les éléments répétés pour exprimer la façon dont la topologie est construite.

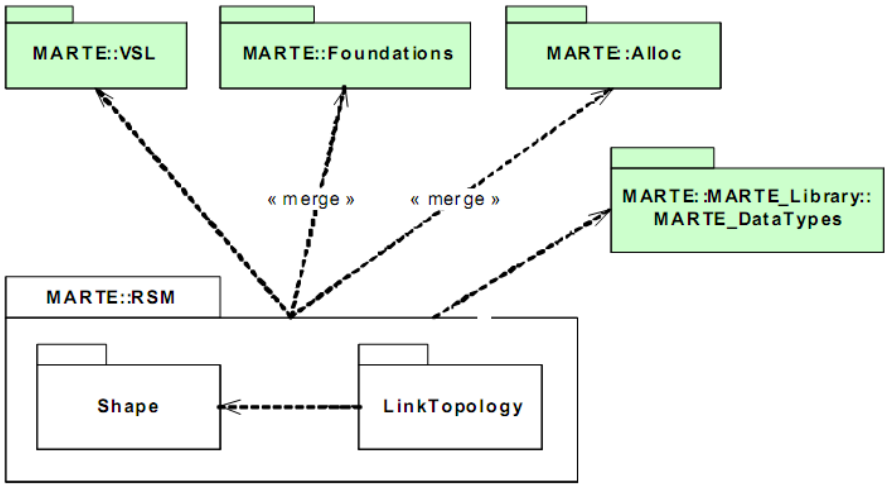

Figure 2. 10: l'architecture globale de paquetage RSM [38].

A partir de ces mécanismes, on peut spécifier les architectures matérielles par l'extraction du parallélisme disponible et la modélisation d'une façon compacte, suivie de l'expression du parallélisme de données ou de tâches pour l'application, afin de fournir un placement régulier de l'application sur l'architecture cible. Tout cela se fait d'une manière indépendante de la technologie cible.

Le paquetage RSM étend la construction de base du profil MARTE par l'ajout des concepts *Shaped*, pour exprimer la multiplicité, et ceux relatifs à l'expression des liens topologiques. La figure 2.10 [38] présente la construction globale de ce paquetage.

L'objectif d'étendre le concept de multiplicité d'UML est de fournir un sémantique capable de définir la multiplicité d'une instance dans une structure répétitive. Le modèle fourni et sa multiplicité peuvent être vus comme une collection mono- dimensionnelle des éléments. Pour spécifier la forme de cette collection, un élément est considéré comme étant un tableau multidimensionnel décrit par le *Shape* associé à sa multiplicité.

Dans le paquetage RSM, il y a aussi le concept *LinkTopology* qui permet de modéliser les liens topologiques par l'introduction de certaines informations nécessaires à la construction de la structure répétée. Il y a 4 types de connecteur pour décrire les liens topologiques : le *Tiler, le Reshape, InterRepetition et Default Link.* Ces 4 types vont être détaillés dans le chapitre qui suit lors de leurs utilisations pour modéliser la topologie des NoC.

4.2. Modélisation de l'allocation en MARTE

Comme c'est indiqué dans le paragraphe qui décrit le standard MARTE, la Co-modélisation application/ architecture est permise. Ces dernières sont modélisées séparément dans un premier temps. Ensuite une étape d'aassociation des IPs de l'application sur l'architecture aura lieu. La manière

d'allouer géographiquement les IPs sur l'architecture est une phase primordiale dans la conception des SoCs. Pour prouver une meilleure gestion d'allocation, toutes les combinaisons possibles devront être étudiées en termes de test et de comparaison. Mais suite à la complexité incessante des SoCs, cette phase d'allocation devient difficile à gérer et l'espace de solution devient incontournable, d'où la nécessité d'aller dans les niveaux d'abstraction afin de trouver un espace de solution convenable. Le profil MARTE offre la possibilité d'abstraire cette phase d'allocation à l'aide du paquetage *Alloc*. Dans MARTE l'allocation peut être vue sous deux angles : une allocation géographique ou spatiale et une allocation temporelle.

(1) Une allocation simple permet le placement d'une tâche fonctionnelle, décrite par un graphe, sur un processeur. Cette façon est spécifiée en MARTE par le traçage d'une dépendance de la tâche vers un processeur. Ce lien décrivant la dépendance est stéréotypé *allocate*. Cette représentation n'est valable qu'entre deux composants ayant la même multiplicité.
(2) Une allocation distribuée consiste à distribuer des répétitions des composants de la tâche fonctionnelle sur des répétitions des composants d'une architecture matérielle. Cette distribution parallèle s'effectue, en MARTE, par le stéréotype *distribute*. Une distribution est construite à l'aide de deux tilers.

Notre contribution dans MARTE est de vérifier, en premier lieu, la pertinence des concepts fournis par le paquetage RSM pour la modélisation des topologies des NoCs et de proposer une méthodologie pour la modélisation de tous les concepts liés au NoC en utilisant le profil MARTE. Dans le domaine de la modélisation en MARTE, une allocation permet d'allouer des tâches, décrites par des graphes, à des unités de calcul, tandis que dans cette thèse il s'agit d'allouer des IPs matérielles, de l'application H.264, sur des répétitions de routeurs eux-mêmes décrits en matériels.

5. GASPARD2 : environnement pour la Co-modélisation basé sur le profil MARTE

GASPARD2 (Graphical Array Specification for Parallel and Distributed Computing) est un environnement de Co-modélisation, basé sur le profil MARTE, qui reprend la représentation en Y de Gajski. Le but de la méthodologie de GASPARD est, à partir d'une modélisation en MARTE, de générer différents codes afin d'entamer la phase d'exploration du SoC en simulant, de vérifier et de synthétiser. GASPARD cible différents langages, tels que le VHDL qui permet la synthèse d'accélérateur matériels, le systemC permettant la simulation, etc. L'avantage de la modélisation en suivant le flot de GASPARD est de faire la modélisation indépendamment du code à produire. Cela veut dire qu'il permet la génération à partir de l'emplacement d'une application sur une architecture. En effet, GASPARD2 collecte un ensemble de développeurs de différents domaines, ayant comme point de départ l'usage de la méthodologie fournie et la modélisation en utilisant le profil MARTE jusqu'à la génération de code. En commençant par la modélisation conjointe en MARTE de l'application, de l'architecture, du placement et le déploiement des éléments de base, le modèle fourni doit passer par plusieurs niveaux d'abstraction jusqu'à la génération du code. GASPARD2 a contribué au développement du MARTE. Le paquetage RSM de MARTE est basé sur le modèle de calcul Array-OL de GASPARD2.

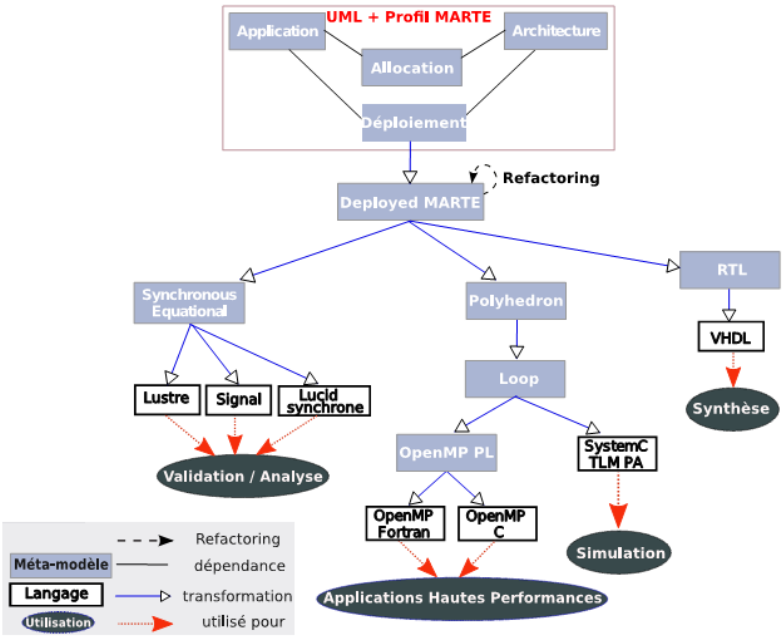

Figure 2. 11: Méthodologie de Co-conception de GASPARD2.

La figure 2.11 représente la méthodologie de Co-conception des SoCs proposée par GASPAR2. Cette méthodologie consiste à modéliser l'application et l'architecture séparément. Le modèle de l'application est ensuite alloué sur l'architecture. Le déploiement en GASPARD2 permet d'attribuer les bouts de code, mis dans la bibliothèque, à chaque composant. Notre contribution dans cette partie consiste à extraire le parallélisme potentiel de l'application H.264, et à modéliser en utilisant le profil MARTE des topologies de NoCs afin de suivre la méthodologie de GASPARD et de montrer la possibilité d'allouer géographiquement des IPs décrits en VHDL sur des routeurs également décrits en VHDL. Donc dans GASPARD2 il y a la modélisation de l'application, de l'architecture et de l'allocation pour arriver à la fin à la génération du code VHDL synthétisable d'un SoC dédié au codage vidéo.

Comme c'est illustré dans la figure 2.11, les principales phases dans la méthodologie de GASPARD sont la modélisation de l'application et de l'architecture indépendamment l'une de l'autre, la modélisation de l'allocation et le déploiement.

5.1. Application

Comme GASPARD2 compile des modèles décrits en MARTE, la modélisation de l'application est spécifiée par l'utilisation du paquetage RSM. Sur GASPARD, on peut exprimer le parallélisme de tâches ou de données par une modélisation compacte de la composante répétée. Deux concepts sont très importants à utiliser lors de la modélisation sur GASPARD : un composant et une instance de composant. Un composant est une boîte noire qui représente la description abstraite d'une fonctionnalité. Ce composant peut contenir des ports d'entrées/sorties qui facilitent le passage des résultats produits et des données consommées. L'ensemble d'instance de composant forme le

composant global, donc un composant peut contenir plusieurs instances mais le sens inverse n'est pas autorisé. Il y a 3 types de composants :

Un composant composé : qui est composé d'instances de composant connectées via des ports.

Un composant répété : contient une instance de composant qui est répétée une ou plusieurs fois.

Un composant élémentaire : composé par une tâche atomique, il ne possède aucune instance de composant.

La notion de composant est utilisée dans l'expression des répétitions de tâches et dans le parallélisme de tâches ou de données. Par conséquent, le stéréotype *Shaped* de RSM, appliqué à une tâche ou à un composant répété indique l'espace de répétition de cette tâche. La façon dont chaque instance de composant consomme des données et produit des résultats est décrite par le concept Tiler de RSM.

5.2. Architecture

Dans cette thèse, il s'agit d'une architecture à base de NoCs, donc l'aspect le plus intéressant pour nous est la modélisation des topologies régulières, irrégulières et hiérarchiques d'une façon compacte. Trouver une méthode compacte de modélisation pour les différentes familles de topologie permet une allocation facile de l'application. Dans l'architecture globale de MARTE il y a le paquetage HRM qui décrit les concepts associés à la modélisation d'une architecture matérielle cible. Il fournit des ressources à différents aspects, par exemple on trouve les unités de calculs (*processsor*), les unités de stockage (*Hw_RAM*) et les unités de communication(*Hw_bus*).

5.3. Association.

Habituellement, une association en GASPARD permet d'allouer des tâches sur des processors. Dans cette thèse, nous proposons l'allocation des composants VHDL sur des composants en VHDL. Comme c'est indiqué précédemment, l'association en GASPARD peut être vue comme une allocation ou une distribution. Le profil MARTE fournit les stéréotypes nécessaires. La figure2.12 montre un exemple de distribution d'une tâche répétée sur une répétition de processeur.

Cette figure montre la distribution de la tâche DCT sur des processeurs répétés, il s'agit de traiter ligne par ligne les pixels d'une matrice 8×8. Donc, pour exploiter le parallélisme de cette application, il est recommandé de distribuer 8 tâches de DCT sur 8 processeurs pour augmenter le temps d'exécution de cette tâche composée.

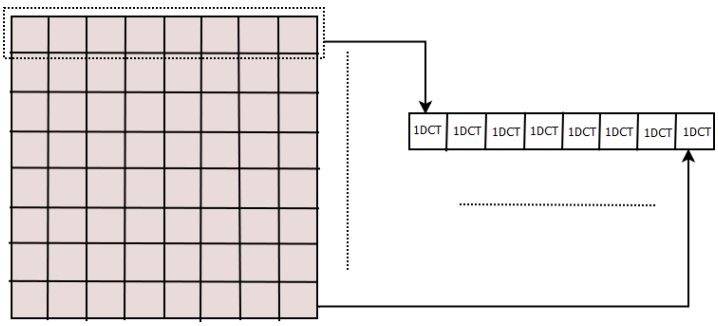

Figure 2. 12: Exemple de placement d'une répétition de tâche sur des répétitions de processeurs.

5.4. Le déploiement

Dans les phases précédentes, la modélisation est faite indépendamment d'une technologie cible. La phase de déploiement consiste à implémenter les modèles de l'application et de l'architecture, dans un langage suivant les objectifs du concepteur afin d'obtenir un code exécutable. En effet, tous les composants élémentaires de l'application et de l'architecture devront être liés à des IPs de la bibliothèque. Cette phase de déploiement dépend des objectifs du concepteur. Suivant la chaîne de GASPARD2 on peut générer :

- Du SystemC si on vise la simulation de comportement d'un SoC à différents niveaux d'abstraction à savoir le TLM PV, le TLM PVT ou le CABA [39].
- Du VHDL si on vise les accélérateurs matériels et la synthèse sur des FPGA ou des ASICs [40]
- Du Lustre pour la vérification formelle du modèle de l'application [41].
- Du OpenMP/Fortran qui permet une exécution concurrente de différentes tâches sur une architecture MPSoC [42].

Les différentes implémentations des IPs sont rassemblées dans un concept appelé *VirtualIP*. La bibliothèque qui regroupe ces IPs est appelée GaspardLIB[39][43].

Dans notre cas, les IPs VHDL de l'application H.264 et les routeurs du NoCs sont placés dans GaspardLib afin de pouvoir générer un SoC complet pour le traitement vidéo.

Bilan de GASPARD2

En résumé, l'environnement GASPARD2 permet :

- La conception des SoCs dans une démarche mixte application/architecture indépendamment l'une de l'autre.
- La modélisation à haut niveau, en utilisant le profil UML/MARTE, de l'application de l'architecture et de l'allocation indépendamment de la technologie cible.
- L'abstraction d'un système dans différents niveaux.
- La génération de code exécutable dans un domaine précis, le code généré peut être synthétisé ou simulé.

6. Les réseaux sur puce.

Les systèmes sur puce SoCs deviennent l'architecture principale d'exécution des applications embarquées. Ils permettent d'intégrer un très grand nombre de fonctionnalités telles que le DSP, les processeurs, les mémoires etc. sur une seule puce. L'évolution incessante entraîne une nécessité grandissante des besoins de communication entre les différents composants qui forment le système. La quantité énorme de données traitées et le nombre important des IPs qui peuvent contenir un SoC, rendent la gestion de l'infrastructure de communication difficile. Cela a influé directement sur les bus de communication traditionnellement utilisés. La bande passante, le débit, la latence, la consommation d'énergie deviennent un goulot d'étranglement dans les SoCs à base de bus. Cependant, avec

l'augmentation du taux d'intégration des transistors sur la même pièce en silicium et la naissance des nanostructures, des nouvelles contraintes auront lieu.

Ayant fait ces observations, différents efforts de recherches ont proposé un nouveau paradigme pour assurer la communication dans les SoCs : le réseau sur puce (Network-On-Chip, NoC).

Le réseau sur puce est inspiré du modèle OSI *(Open Systems Interconnection)* qui décrit la communication dans les réseaux informatiques, d'après les travaux de la littérature ce type d'architecture présente des avantages aux niveaux conception, performance et implémentation.

Nous détaillons dans cette partie, dont l'objectif est de donner au lecteur un préambule à la conception architecturale des NoCs, les problématiques de la communication des SoCs et les concepts généraux liés à ce nouveau paradigme.

6.1. Problématiques des SoCs

Parallèlement à l'évolution des technologies de semi-conducteurs et à l'augmentation du taux d'intégration des transistors, les concepteurs devront faire face à deux tendances : les besoins grandissants des applications en termes de traitement de données et le temps de mise sur le marché d'un nouveau produit.

En effet, les systèmes sur puce deviennent de plus en plus complexes à concevoir et à gérer. Une solution face à ces problèmes est de réduire l'écart de productivité entre l'évolution de la technologie et les environnements de conception des SoCs.

Les technologies d'intégration sur silicium offrent une constante réduction des dimensions des transistors, et par conséquent des circuits intégrés de petite dimension. Le tableau 2.1 fourni par l'ITRS *(International Technology Roadmap for Semiconductors)* montre les futures tendances pour les circuits intégrés.

Tableau2.1 : performances des ASIC et des processeurs d'après ITRS.

Année de production	2005	2006	2007	2009	2012	2016	2020
Nombre maximum de transistors (million)	1928	2430	3061	4859	9718	24488	61707
Taille maximale du circuit (mm^2)	858	858	858	858	858	858	858
Fréquence interne maximale (Mhz)	5204	6783	9285	12369	20065	39683	73122
Niveau de métal	15	15	15	16	16	17	18
Puissance maximale(w)	167	180	189	198	NC	NC	NC

Aujourd'hui, un SoC peut contenir plusieurs fonctionnalités ; la communication, le traitement de données etc. Ensuite, les utilisateurs des SoCs exigent des temps de mise sur le marché de plus en plus courts. Donc un circuit contenant des millions de transistors doit être conçu en un temps plus court pour

assurer la rentabilité. Orthogonalement, les problèmes de productivité apparaissent lorsque les capacités de conception n'arrivent pas à suivre l'évolution des technologies du semi-conducteur. Suite à l'utilisation des IPs, les SoCs devraient devenir plus complexes lorsqu'on intègre des centaines des processeurs ou des IPs.

En effet, le problème de la communication dans les SoCs se pose. Certaines applications nécessitent un fort débit, d'autres une forte latence, donc le concepteur doit faire des optimisations architecturales pour répondre à ces compromis. La communication devient un goulot d'étranglement si elle ne suit pas la cadence maximale des unités de calcul.

6.2. Solution d'interconnexion actuelle

Traditionnellement, la communication dans les SoCs était assurée par des liaisons point à point ou des bus partagés (figure 2.13). Il s'agit tout simplement de lier un récepteur à son émetteur par une connexion physique. Le bus partagé est le moyen de communication le plus utilisé dans le monde industriel. Il existe plusieurs produits proposés par des fournisseurs d'IPs, on cite par exemple le bus AMBA d'ARM, STbus de STMicroelectronics, le Wishbone Interconnect, etc. Avec l'évolution incessante des applications, des problèmes de performances apparaissent dans les SoCs à base de bus tels que la bande passante, la latence, le parallélisme dans les communications etc. L'introduction des bus hiérarchiques offre la possibilité de surmonter quelques inconvénients reliés à la solution à base de bus. Il s'agit de relier plusieurs bus entre eux par l'intégration d'un pont et d'un arbitre. Cette solution reste complexe à cause du problème de la gestion d'adresse. En plus, il est difficile d'ajouter d'autres unités de traitement à un bus parce que, d'une part, la bande passante diminue en fonction du nombre des éléments et que d'autre part, le contrôle se complexifie.

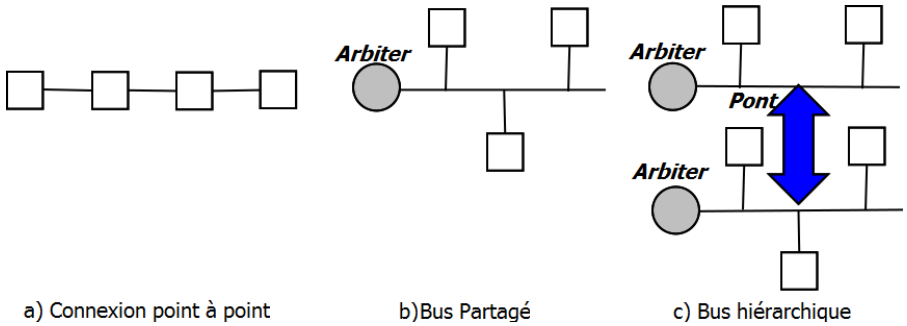

Figure 2. 13: Structures de communication traditionnelles.

Tous ces problèmes ont déjà été étudiés dans le domaine des réseaux locaux [44] et des réseaux d'interconnexion de machine parallèles [45]. En effet, l'idée consiste à distribuer la communication par un découpage en couches. Cela a permis de maitriser et de garder la compatibilité des structures de communication avec le système. Basée sur cette observation, l'intégration d'un réseau de communication dans un système est née au début des années 2000.

6.3. Les NoCs : les couches réseaux et l'évaluation de performances

L'idée des NoCs vient d'une observation basée sur le modèle OSI qui découpe la communication en sept couches : physique, réseau liaison de données, transport, session, application. Cependant, ce

découpage n'est pas adéquat pour les réseaux sur puce parce que seules quatre couches sont utilisées [46].

La couche physique dans les NoCs permet la définition du medium de transmission des paquets tout en tenant compte des caractéristiques de ce paquet. La couche réseau définit la topologie du NoC ainsi que le routage utilisé pour l'acheminement des messages dans le réseau. La couche liaison est chargée de la fiabilité de réseau par l'intermédiaire des techniques de détection et de correction d'erreur. Elle définit aussi le contrôle du flux d'information en se basant sur les techniques de poignée de main, de crédit d'émission. La couche transport permet de transformer les messages en paquets et inversement. L'adaptation des protocoles entre le réseau et les IPs s'effectue par le biais des adaptateurs réseau.

Plusieurs métriques ont été proposées pour évaluer les performances d'un NoC. On en cite la fréquence, le diamètre [47], la bande passante, la surface, la consommation, l'extensibilité, la latence, la flexibilité [48].

6.4. Concepts relatifs à l'architecture des NoCs

Plusieurs travaux dans la littérature ont proposé des architectures pour les NoCs. Cependant, les NoCs sont caractérisés par un modèle général qui décrit leurs éléments de base. Dans cette section, nous décrivons quelques éléments qui sont développés dans la partie expérimentale de cette thèse. Ce modèle est composé principalement de routeurs, également appelés nœuds, qui permettent l'acheminement des paquets selon des protocoles bien choisis en fonction de la topologie. Les données sont diffusées dans le réseau à travers les liens de communication qui peuvent être monodirectionnels ou bidirectionnels.

Les routeurs dans les NoCs [49] sont principalement construits à l'aide des ports d'entrée/sortie contenant des FIFOs *(First In First Out)* permettant de stocker les messages qui ne pourront pas être transmis tout de suite, d'un composant appelée « Crossbar » (Matrice de commutation) pour relier physiquement les ports d'entrée aux ports de sorties, un module d'arbitrage et d'un module pour le calcul de la fonction de routage. La figure 2.14 montre l'architecture générale d'un routeur.

La temporisation des paquets dans les NoCs se fait par des files d'attente. Le rôle de ces files, qui sont basées sur des FIFO, est de stocker temporairement les paquets qui sont en conflit. Les files d'attentes sont caractérisées par leur taille et leur position dans le routeur. La figure 2.15 montre les différentes possibilités de placer les files d'attente.

On distingue trois types d'emplacement des files d'attente : **Files d'attente en entrée** [50], **Files d'attente en sortie** [51]**, Files d'attente en sortie virtuelle** [52].

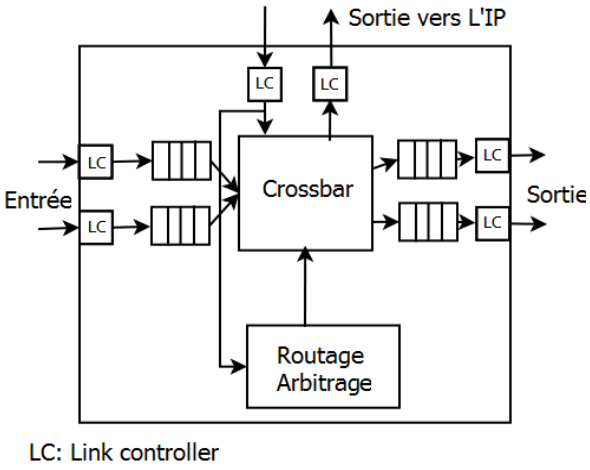

Figure 2. 14: Architecture générique d'un routeur

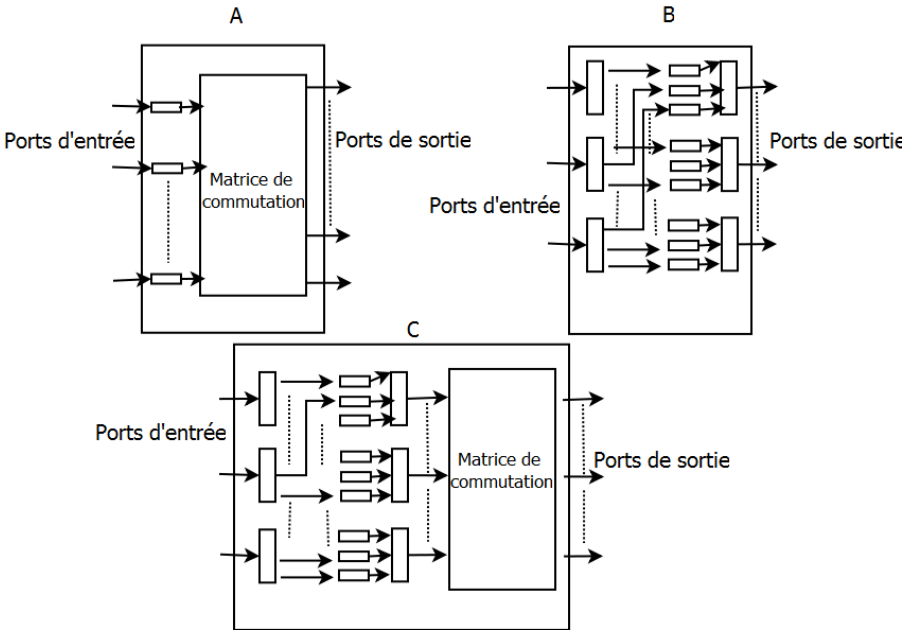

Figure 2. 15: Positionnement des files d'attentes dans un routeur : A en entrée, B en sortie, C en sortie virtuelle.

La matrice de commutation permet l'interconnexion des ports d'entrée aux ports de sortie du même routeur. Le module d'arbitrage assure l'ordonnancement d'acheminement des paquets suivant la priorité. Il y a plusieurs techniques d'arbitrage qui sont citées dans la littérature : Le Time-Division Multiple Access (TDMA), la priorité tournante et la priorité fixe [53].

Le module de routage dans les NoCs assure la transmission des paquets au sein du réseau en se basant sur une fonction de routage. Plusieurs algorithmes de routage ont été proposés dans la littérature, et ils peuvent être classés en trois familles : déterministe, adaptative et semi-adaptative [54].

Le Mécanisme de commutation dépend de la topologie choisie du NoC. L'objectif est de spécifier comment les données vont circuler dans le réseau. Deux types ou modes de commutation existent ; la

commutation par paquet et la commutation par circuit. La commutation de circuit n'est pas beaucoup utilisée dans la littérature, suite à sa complexité d'implémentation. Tandis que la commutation de paquet est souvent implémentée dans différentes architecture. Il y'en a 4 politiques de mémorisation des paquets:

- Sotore-and-forward (SAF).
- Cut-through (CT).
- Worhmhole.
- Switched virtual circuit (SVC).

La topologie des NoCs définit la façon géométrique dont les routeurs sont connectés par des liens physique [55]. Dans la conception des NoCs la première étape est de choisir une topologie convenant à l'application. La topologie conditionne la technique de routage et la méthode de contrôle de flux. Les topologies des NoCs sont inspirées de la théorie des graphes. En effet, on peut les classer suivant des critères topologiques. Les topologies sont au nombre de trois : directe, indirecte [56] et hybride tel que le réseau « Protéo » [57] (Figure2.16). Cette topologie est hiérarchique, nous voyons deux réseaux qui sont connectés par un pont.

La classification de la topologie peut se baser sur d'autres critères tels que la régularité. Dans cette thèse la classification des topologies est basée sur la définition du mot régulier dans le contexte du NoC. La régularité d'un réseau sur puce dépend essentiellement du degré de la topologie. En effet une topologie n'est dite régulière que lorsque tous les routeurs ont le même nombre des liens les reliant aux routeurs voisins. Dans le cas contraire, la topologie est dite irrégulière. Suite à cette définition nous identifions la topologie hiérarchique, comme un graphe construit par des topologies régulières ou irrégulières, et qui peut présenter une régularité locale [58]. Nous détaillons cette classification dans le chapitre suivant et lors de la modélisation des topologies des NoC utilisant le profil MARTE.

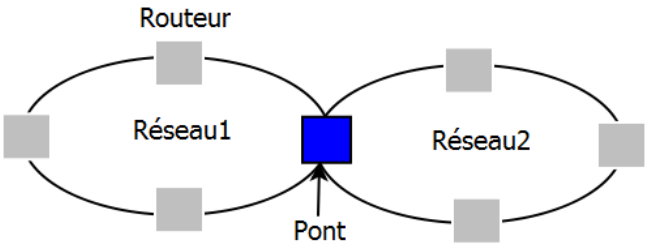

Figure 2. 16: Topologie hiérarchique

Dans la littérature, plusieurs topologies ont été proposées, qu'elles soient régulières, irrégulières, hiérarchiques, directes ou indirectes. Extraire la régularité d'une topologie permet de faciliter la conception de tout le réseau, parce que cela favorise l'identification des routeurs de base de tout le réseau et à partir de cette identification, on peut modéliser un routeur général et générer les autres.

Dans cette thèse, nous proposons la conception architecturale d'un NoC nommé FeRoNoC « Flexible, Extenssible Router implémentation for diagonal Mesh Topologies ». Nous allons détailler cette architecture dans un prochain chapitre.

7. Outil de modélisation des NoCs

La conception des NoCs passe par la modélisation dans un langage adapté à leur implémentation. Certains de ces langages sont relatifs à un niveau d'abstraction bien défini et ne sont manipulés que dans un domaine spécifique. Cela signifie que la conception des NoCs peut passer par l'utilisation de nombreux modèles. Dans un niveau d'abstraction élevé, la conception des NoCs peut se réduire à une modélisation formelle ou fonctionnelle. En effet le choix du langage dépend de ce qu'on souhaite faire, si par exemple en vise de la simulation, on utilisera le SystemC, si on vise la synthèse, on a besoin d'un langage de description HDL. Donc la question qu'on pose, c'est : comment on peut partir d'une modélisation haut niveau, et indépendamment de la technologie cible, pour arriver à une génération de code dans un domaine de travail spécifique ? Le langage UML qui est un langage unifié utilisé pour le développement de logiciel est devenu aujourd'hui un langage permettant la conception des SoCs, évidemment les NoCs. Suite à ces constats, les concepteurs ont mené des recherches pour développer des outils de conception assistée par ordinateur, dont l'objectif est de faciliter le processus de conception. La majorité des outils dédiés aux NoCs sont des outils spécifiques, qui peuvent être soit d'évaluation de performances, tels que le débit, la latence ou de simulation. Dans ce paragraphe nous mettons l'accent sur quelques outils dédiés aux NoCs, et nous positionnons nos travaux de thèse.

7.1. L'outil µSpider

Le µSpider est un environnement permettant d'automatiser des tâches dans la conception des NoCs, il est développé à l'INSA de Rennes. Cet outil est basé sur une approche orientée objet. Le format d'échange choisi est le XML. La méthodologie de modélisation dans µSpider est donnée par la figure 2.17 [59].

Cette figure met en œuvre la relation entre les composants ou les objets qui forment le système. Ce modèle est basé sur UML. Le système est découpé en deux parties ; application et architecture. L'architecture peut être de type NoC. Suivant ce flot, une application peut avoir plusieurs modes de fonctionnement. Le NoC est composé par un ensemble des routeurs, des NI (Network Interface), des ports et des liens. Le µSpider permet à l'utilisateur de saisir les spécifications architecturales et applicatives afin d'arriver à la génération de code. Les principales étapes décrivant le flot de l'outil µSpider sont les suivantes :

- Spécification et saisie des contraintes de l'application.
- Spécification de l'architecture NoC.
- Mapping des IPs avec les NIs convenables.
- Spécification des caractéristiques du NoC (routage, taille des FIFOs, dimensionnement de la table etc)
- Création de la bibliothèque des composants.
- Génération du code VHDL.

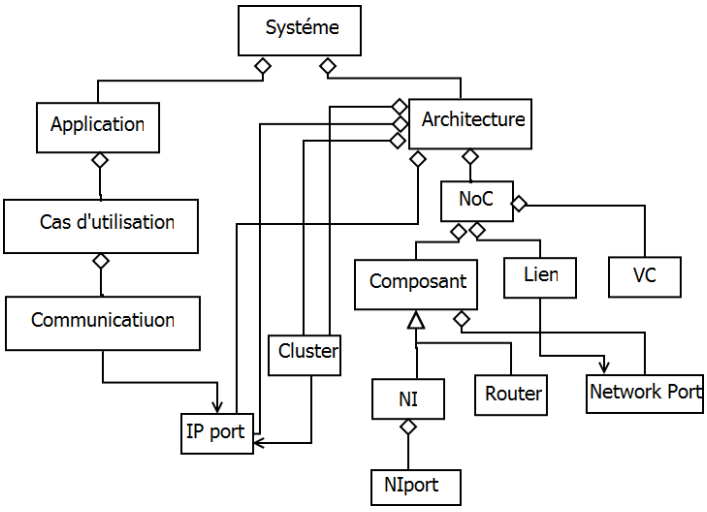

Figure 2. 17: Modélisation µSpider [59]

7.2. Outil de AEthereal

Philips a développé un flot de conception pour son réseau AEthereal [60]. Ce flot montré dans la figure 2.18 prend comme entrée les contraintes de la bande passante et la latence des communications afin de générer le code VHDL du NoC. Les concepteurs proposent d'unifier les phases, de placement des IPs et de sélection du chemin, par l'utilisation d'un algorithme nommé (Unified Mapping, Routin and Slot allocation).

Le flot de conception AEtheral a été validé par un SoC dédié à la télévision numérique

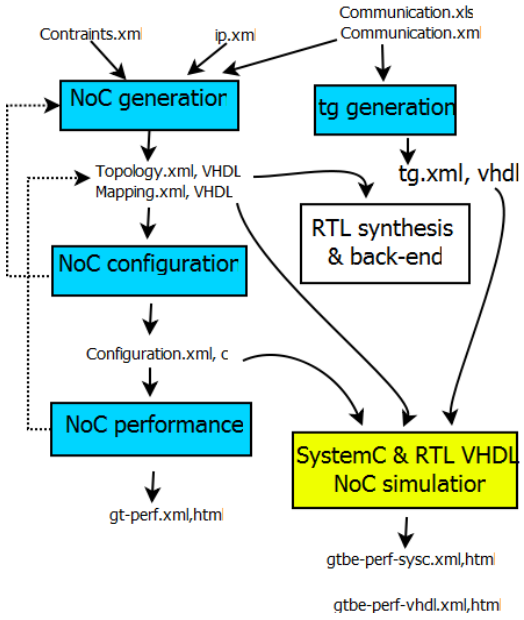

Figure 2. 18: Flot de conception de NoC Aethereal [60].

7.3. L'outil NS-2

L'utilisation de l'outil NS-2[61] résulte de l'idée d'une réflexion commune entre VINNOVA, NOKIA et Ericsson, dont l'objectif est la simulation et l'évaluation de performances du réseau NOSTRUM. Le simulateur a été transféré en SystemC, en utilisant les *canaux* de communication prédéfinis. Ce simulateur est divisé en deux grands domaines : Un domaine applicatif, qui génère et mappe le trafic et un domaine de communication qui émule un réseau sur puce par les 4 couches basses du modèle OSI. Il intègre un générateur de topologie en grille 2D (pour Nostrum), une stratégie de routage à priorité tournante (ou *round robin*). Il possède une interface standard afin de pouvoir modifier le modèle du réseau sans modifier la ressource, et inversement. Pour l'instant le générateur de trafic est simple. Un paquet (de taille fixe) est émis, puis la ressource attend un nombre de cycles aléatoire avant de réémettre. A noter que cela ne correspond pas à des *burst* de tailles variables.

7.4. Bilan des outils pour les NoCs

Comme nous l'avons déjà annoncé au début de cette partie, la plupart des outils de conception des NoCs sont dédiés à la simulation ou à l'évaluation de performances. En plus la plupart de ces outils sont taillés à la mesure des architectures bien définies. Cela veut dire que ces outils ne peuvent pas être un support d'exécution pour la communauté du NoC. La nécessité des outils de conception s'avère grandissante, suite à l'émergence des applications et aux contraintes de temps de mise sur le marché. Dans cette thèse, la modélisation d'une façon compacte des NoCs dans un langage haut niveau favorise une conception indépendante de la technologie et des caractéristiques spécifiques à une architecture. Après la modélisation, la génération de code est basée sur une bibliothèque d'IP intégrée dans GASPARD. Cette proposition se révèle importante avec l'approche du massivement parallèle et les architectures multi cœur.

8. Conclusion

Ce chapitre nous a présenté un large panorama des différents domaines concernant cette thèse. En effet partant du contexte de Co-conception des systèmes embarqués, nous décrivons une méthodologie basée sur le flot de GASPARD permettant la modélisation de l'application et de l'architecture. Nous avons présenté le domaine d'application qui est le traitement de signal intensif, et nous voyons après que cela va être le codeur H.264, puis nous avons détaillé les concepts reliés au standard MARTE que nous utilisons pour la modélisation de l'architecture, l'application et l'allocation. Nous avons vu aussi les caractéristiques du NoC et des éléments modélisés durant cette thèse.

C'est pourquoi, dans le chapitre suivant, nous allons détailler la modélisation des NoCs en utilisant le profil MARTE.

Chapitre 3

Modélisation de l'architecture du NoC en MARTE: Topologie, Routage

1. Introduction

Dans le premier chapitre, nous avons introduit les notions de base utilisées dans cette thèse, afin de donner au lecteur le vocabulaire et les éléments nécessaires à la manipulation de la thématique globale étudiée. Ce deuxième chapitre s'intéresse à la modélisation de l'architecture NoC dans une démarche IDM en utilisant le profil MARTE. Il présente les grandes lignes de la contribution à la modélisation de l'architecture. Nous mettons l'accent sur le contexte de notre travail, présenté dans le chapitre 1, afin de proposer une méthodologie pour la modélisation des NoCs. L'objectif de ce chapitre est de modéliser les NoCs dans un niveau d'abstraction élevé, par l'utilisation du package RSM du MARTE, tout en tenant compte des contraintes RTL. La méthodologie de modélisation des NoCs, proposée dans ce chapitre se base sur l'utilisation de l'IDM et se concentre sur la façon de traiter le problème plutôt que sur les détails d'implémentation.

2. Ingénierie dirigé par des modèles (IDM)

Tout système peut être décrit par un modèle qui permet la compréhension de son fonctionnement et facilite l'intervention au niveau description. Le modèle définit une vue simplifiée et/ou une abstraction du système, il s'agit donc de faire sortir les caractéristiques pertinentes afin de réduire les complexités relatives à la conception. Dans le domaine de la modélisation, plusieurs méthodes ont été définies telles que UML et le SSADM (Structured Systems Analysis and Design Methodology) [62]. Les méthodes d'IDM proposent des concepts et des notations permettant la description d'un tel système à concevoir. En conclusion, l'ingénierie dirigée par les modèles apparaît comme une solution face au problème rencontré lors de la modélisation avec les méthodes ordinaires. Dans la littérature, plusieurs termes ont la même signification que l'IDM. On trouve par exemple le MDD (Model Driven Development), MDE (Model Driven Engineering), etc. Parmi les notions de base exploitées dans le domaine de l'IDM, on trouve le modèle et le méta-modèle. Un modèle est une abstraction d'un système permettant le masquage des détails qui ne sont pas pertinents pour le système à concevoir. Ce concept est utilisé tout au long de la phase de modélisation. Un méta-modèle définit les concepts relatifs à un domaine d'abstraction dans lequel on exprime notre modèle. Donc, un modèle, qui est l'abstraction d'un système réel, est conforme à son méta-modèle. Un meta-modèle permet alors de préciser la syntaxe des modèles.

L'IDM permet donc de fournir un espace de modélisation dans lequel les modèles passent de l'état passif à l'état productif. Le modèle est dit productif lorsqu'il peut être exécuté d'une façon ou d'une autre, soit directement par la machine, soit par la production des codes exécutables.

2.1. Le package HRM (Hardware Resource Modeling) du MARTE

Le package HRM (Hardware Resource Modeling) du profil MARTE permet la modélisation et la description de plateformes matérielles ainsi que la modélisation architecturale des systèmes de différents points de vue. Ces points de vue peuvent être logiques ou physiques. La figure 3.1 représente une vue du domaine de HRM.

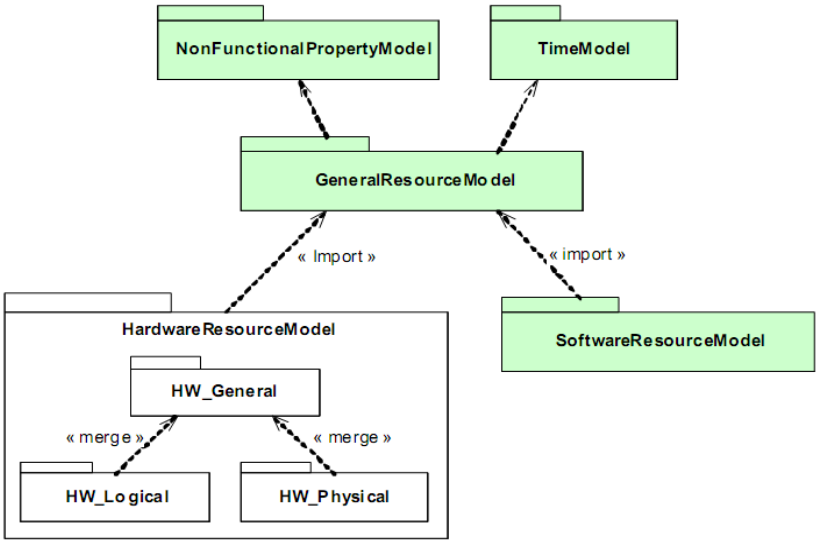

Figure 3. 1: Dépendances de HRM

La vue du domaine, présentée par la figure 3.1, montre la dépendance du HRM avec d'autres profils et librairies. HRM exploite la librairie NFP (Non Functional Property Model) qui englobe des types ayant des unités, tels que la durée, la taille des données, la puissance électrique, etc. Ces types sont utilisés pour la modélisation architecturale.

Le HRM est composé de deux sous package ou vues, une vue logique ou aussi fonctionnelle indiquée par le Hw_Logical, et une vue physique exprimée par le Hw_physical. Le package Hw_Logical permet d'expliciter le côté fonctionnel du matériel tandis que le Hw_physical explicite la partie physique. Ces deux packages sont regroupés dans un modèle général commun.

2.2. Modèle général du HRM

Le modèle général de HRM illustré dans la figure 3.2, décrit la structure d'une plateforme d'exécution en combinant les spécialisations différentes, relatives au modèle logique ou au modèle physique.

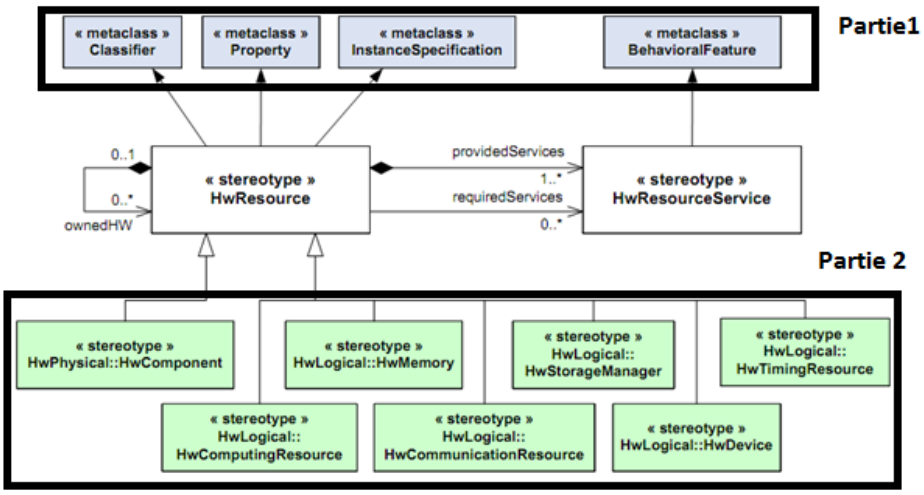

Figure 3. 2 : Vue globale de HRM.

On note la présence de plusieurs stéréotypes permettant la spécification du matériel. Le stéréotype *HwResource,* par exemple, spécifie une entité générique du matériel. Chaque *HwResource* peut contenir des *ownedHW* ressources. L'approche de composition offre la possibilité de modéliser les architectures ou les plateformes à différentes granularités. Le *HwResource* étend (Partie 1) les méta-classes générales d'UML *Classifier, Property* et Instance *Specification*. Par la suite il est possible de modéliser à l'aide de HRM en utilisant tous les diagrammes UML (diagramme de classe, diagramme de composant…). De la même façon, le *HwResource* étend la méta-classe UML *BehavioralFeature* qui permettra la description comportementale du système. Dans la seconde partie, les stéréotypes du HRM héritent de *HwResource*. Ils peuvent exploiter alors la même structure et les mêmes extensions.

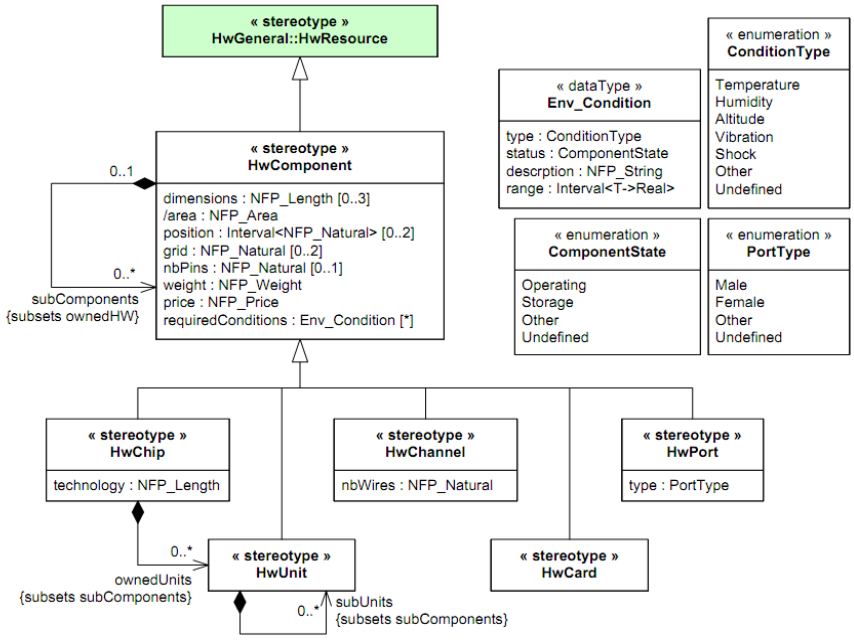

Figure 3.3 : vue globale de HwComponent

Ce qui nous intéresse dans cette partie, c'est le stéréotype HwComponent, qui est une spécification de la métaclasse *HwResource*. Il décrit toute entité matérielle comme étant une classe stéréotypée *HwComponent*. Un composant, comme on l'a vu dans le second chapitre, peut être élémentaire, composé ou répétitif. Notre objectif de base est de favoriser autant que possible la réutilisation des composants déjà modélisés. D'après le méta-modèle illustré dans la figure 3.3, un HwComponent est caractérisé par des propriétés non fonctionnelles telles que la dimension, la surface, le nombre de pins, etc.

Cette partie est introduite dans ce chapitre pour définir les concepts qu'on va utiliser dans la modélisation de notre système, soit la modélisation de l'architecture, soit la modélisation de l'application.

3. Un méta-modèle de Gaspard2 pour l'expression des systèmes répétitifs.

La plateforme de Co-conception Gaspard2 offre un méta-modèle pour décortiquer les différentes parties du système. Ce méta-modèle se base sur cinq paquetages [63] permettant la modélisation en suivant le modèle en Y. Nous citons les paquetages component, factorisation, application, HwArchitecture et l'association. Le premier et le deuxième ont comme objectif de regrouper les concepts nécessaires à la modélisation des composants matériels ou logiciel en favorisant leur réutilisation.

3.1. Le concept Component

Le paquetage component permet l'expression de la structure d'un composant indépendamment du domaine d'implémentation. Le but de ce mécanisme est d'offrir autant que possible la réutilisation des composants déjà modélisés. Ce concept de composant est très important dans un modèle Gaspard, vu qu'il représente la brique de base pour la modélisation de la totalité du système. Les composants, dans un modèle Gaspard2, communiquent à travers leurs *ports*. La modélisation structurelle ou hiérarchique d'un composant se fait par l'utilisation de la notion d'*Instance* et de *Connector* de paquetage GCM (Generic Component Modeling) de MARTE. Par ailleurs, Gaspard2 adopte une approche à base de composants basés sur le profil UML MARTE. D'une manière générale, comme on l'a dit dans le chapitre précédent, un composant peut être de type *élémentaire, composé ou répétitif.* Un composant de type *RepetitiveComponent* est basé sur la répétition de l'instance de ce composant.

Un composant structuré, d'après le paquetage GCM [64], peut définir plusieurs concepts, tels que l'*Assembly Connectors* et l'*Interaction Port*. L'*assembly connectors* a comme rôle de connecter des ports d'un composant structuré à des ports de sous composants.

Le concept *Interaction Port* définit l'interaction qui peut exister entre les différents composants à l'aide des connecteurs d'assemblage. Les ports suivant le paquetage GCM peuvent être classés en deux grandes catégories :

- *Flow Port* : il définit l'orientation de la communication c'est-à-dire la direction du port (in, out, inout). En Gaspard2 ce concept est très utilisé pour la modélisation basée composant.

- *Message Port :* il est utilisé pour orienter les messages (request/reply) lors de la communication

La figure 3.4 illustre un exemple d'un composant répétitif à base de composants élémentaires.

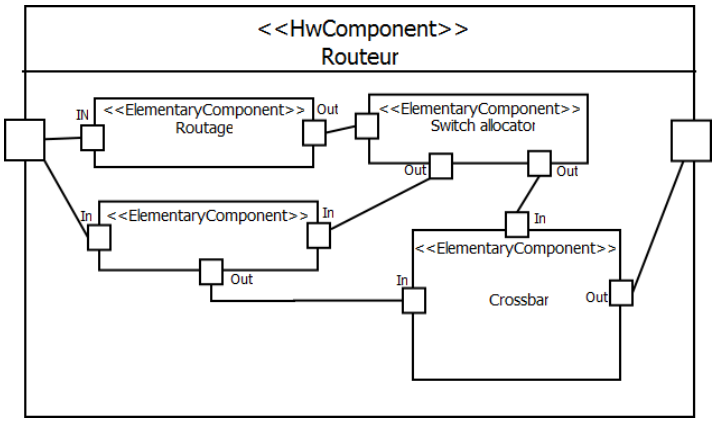

Figure 3. 3: Exemple de composant composé.

3.2. Le concept Factorisation

Le paquetage factorisation engendre des concepts permettant de modéliser d'une façon compacte des systèmes répétitifs réguliers. La sémantique dans ce paquetage est basée sur les concepts de RSM (Repetitive Structure Modeling) dédiés aux applications de traitement de signal intensif et aux architectures à aspects répétitifs. Les topologies répétitives sont à l'origine des répétitions de sous-composants interconnectés à travers des motifs réguliers. Les mécanismes fournis par le paquetage RSM du MARTE sont basés sur 2 aspects : la possibilité de décrire la forme (Shape) d'une structure répétée par une forme multidimensionnelle et aussi l'expression de l'ensemble des liens comme un tableau multidimensionnel. Le deuxième aspect est la modalité suivant laquelle représenter des informations topologiques sur des relations entre des entités. Un exemple d'application de Shape est de prendre une instance de composant ou un port et de spécifier la multiplicité à l'aide d'un vecteur strictement positif qui indique le nombre d'éléments dans chaque dimension du tableau. Une instance répétée 20 fois peut s'écrire sous forme d'un tableau bidimensionnel 2×10. Nous modélisons une seule instance avec une Shape de (4, 10).

Ce paquetage, qui est basé sur RSM, utilise les concepts Tiler, Reshape, Inter-repetition pour modéliser les liens qui existent entre les composants. La figure 3.4 montre ces liens. Nous présentons par la suite ces concepts de modélisation des liens topologiques.

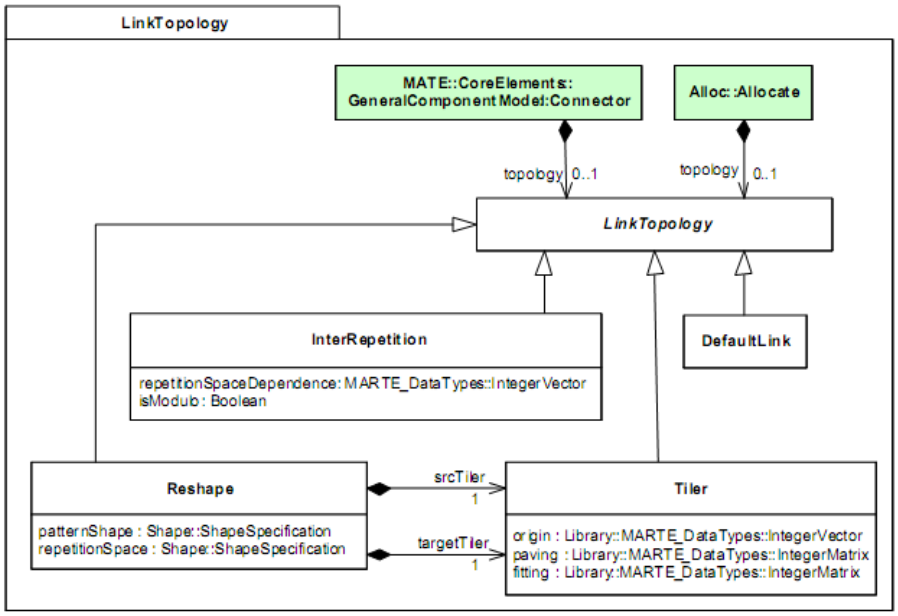

Figure 3. 4: Concepts relatif à l'expression des liens topologiques

3.2.1. Le Tiler

Le Tiler nous permet de construire les motifs à partir des informations du tableau d'entrée, ou de la façon de ranger les motifs dans un tableau. Donc, il contient des informations permettant de répondre à la question suivante : comment construire le motif ? Ces informations sont l'origine, le pavage et l'ajustage. Le tiler est un connecteur qui permet d'identifier des sous- tableaux (motifs) à l'intérieur d'un tableau défini par une forme (shape). La tuile est l'assemblage d'un ensemble de points du tableau utilisés pour créer le même motif. L'ensemble de la tuile est construit par des points régulièrement espacés et les tuiles sont aussi régulièrement espacées. La matrice d'ajustage nous donne une information sur l'espacement régulier des points d'une tuile. La matrice de pavage décrit l'espacement régulier des tuiles dans le tableau.

Notion de motif : soit le couple (T_1, A_1), où T_1 indique la tâche et A_1 le tableau. On a deux cas :

Premier cas :

- A_1 est un tableau d'entrée.
- T_1 prend en entrée un sous- ensemble fini des éléments d'A1 pour effectuer le traitement.

Deuxième cas :

- A_1 est un tableau de sortie.
- T_1 va produire pour lui un ensemble fini d'éléments qu'elle vient de calculer.

Les sous-ensembles de A_1 sont appelé motifs.

L'opération recommence, T_1 va produire ou choisir un nouveau motif qui doit être de même forme que le premier, comme l'indique la figure 3.5 :

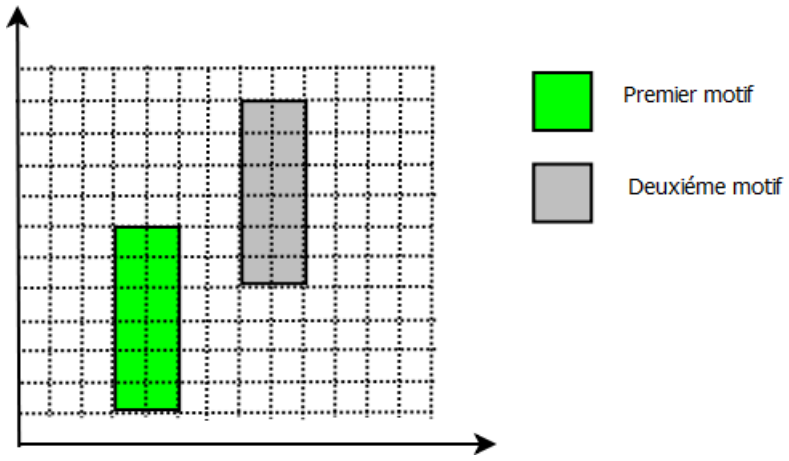

Figure 3. 5 : Exemples de motifs

A chaque couple (tâche, tableau), on associe un tiler.

Notion de pavage

La construction des motifs commence par trouver les coordonnées de leur origine. La matrice de pavage et un point de référence sont suffisants pour calculer les premiers points de chaque motif. Donc, la matrice de pavage permet d'identifier l'origine de chaque motif associé à chaque instance de tâche. Ce calcul est récursif. Il correspond à la somme des coordonnées de l'origine et d'une combinaison linéaire des vecteurs de pavage, le tout modulo la taille du tableau. La matrice de pavage a autant de lignes que le tableau a de dimensions, et autant de colonnes que l'espace de répétition a de dimensions. Donc leurs coordonnées sont construites à partir de cette relation :

$$\forall r, 0 \le r \le s_{répétition}, réf_r = O + P.R \bmod s_{tableau} \quad \textbf{Eq3.1}$$

Où $s_{répétition}$ est la forme de l'espace de répétition, P la matrice de pavage et $s_{tableau}$ la forme du tableau. La figure 3.7 représente un exemple de pavage d'un tableau, dans le cas d'un tiler défini par l'origine $(0,0)$ et une matrice de pavage $\begin{pmatrix} 20 \\ 03 \end{pmatrix}$. Les motifs sont construits à partir de leurs points d'origine par itérations successives. L'origine du premier motif est l'origine du motif de référence. Les autres origines des autres motifs sont construites par l'application de l'équation 3.1. Le tableau est donc parcouru de 2 en 2 horizontalement et de 3 en 3 verticalement.

Les origines des motifs suivant la direction horizontale sont données par les valeurs suivantes :

$$r_{(1,0)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 20 \\ 03 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 2 \\ 0 \end{pmatrix}$$

$$r_{(2,0)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 20 \\ 03 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ 0 \end{pmatrix} = \begin{pmatrix} 4 \\ 0 \end{pmatrix}$$

$$r_{(3,0)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 20 \\ 03 \end{pmatrix} \cdot \begin{pmatrix} 3 \\ 0 \end{pmatrix} = \begin{pmatrix} 6 \\ 0 \end{pmatrix}$$

$$r_{(4,0)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 20 \\ 03 \end{pmatrix} \cdot \begin{pmatrix} 4 \\ 0 \end{pmatrix} = \begin{pmatrix} 8 \\ 0 \end{pmatrix}$$

Les origines des motifs suivant la direction verticale sont données par les valeurs suivantes :

$$r_{(0,1)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 20 \\ 03 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 3 \end{pmatrix}$$

$$r_{(0,2)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 20 \\ 03 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 2 \end{pmatrix} = \begin{pmatrix} 0 \\ 6 \end{pmatrix}$$

$$r_{(0,3)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 20 \\ 03 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 3 \end{pmatrix} = \begin{pmatrix} 0 \\ 9 \end{pmatrix}$$

De la même façon on peut construire les autres motifs :

$$r_{(1,1)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 20 \\ 03 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$$

$$r_{(1,2)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 20 \\ 03 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 2 \\ 6 \end{pmatrix}$$

$$r_{(2,1)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 20 \\ 03 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 4 \\ 3 \end{pmatrix}$$

La figure 3.6 représente l'origine de chaque motif.

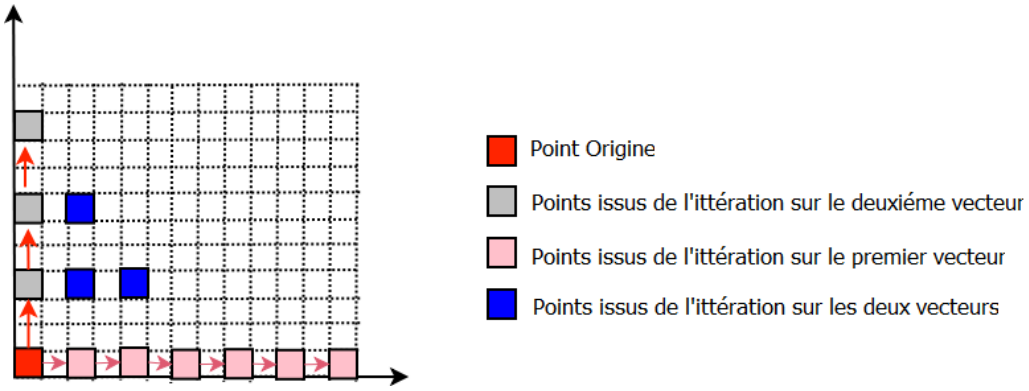

Figure 3. 6 : Exemple de pavage d'un tableau.

Notion d'ajustage

La deuxième partie de la construction, une fois qu'on a trouvé l'élément de référence du motif, consiste à extraire les autres éléments qui forment le motif relatif à l'élément de référence. La matrice d'ajustage permet donc de calculer les coordonnées des autres éléments par rapport à la référence. Cette matrice a autant de lignes que le tableau a de dimensions, et autant de colonnes que le motif a de dimensions. Les coordonnées des autres éléments du motif ($\mathbf{e_i}$) sont construites comme sommes des coordonnées de l'élément de référence et d'une combinaison linéaire des vecteurs d'ajustage, comme ci-dessous :

$$\forall i, 0 \le i < s_{motif}, e_i = réf + F.i \bmod s_{tableau} \qquad \textbf{Eq3.2}$$

Où s_{motif} est la taille des dimensions du motif, $s_{tableau}$ est la forme du tableau et F la matrice d'ajustage. La figure3.7 montre un exemple d'ajustage.

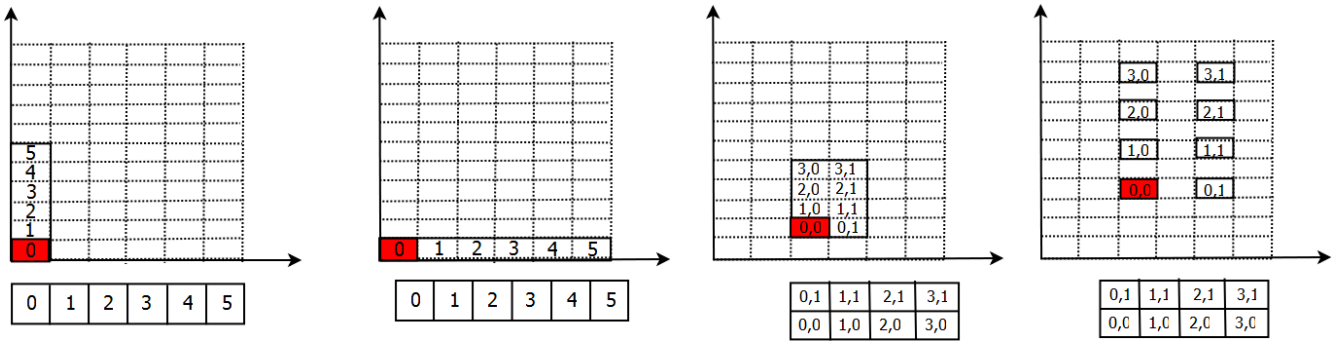

Figure 3. 7 : Exemple d'ajustage.

Les matrices d'ajustages de chaque cas sont données comme suit :

Premiers cas (gauche) : $S_{motif} = 6, F = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$

Deuxième cas : $S_{motif} = 6, F = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$

Troisième cas : $S_{motif} = \begin{pmatrix} 4 \\ 2 \end{pmatrix}, F = \begin{pmatrix} 01 \\ 10 \end{pmatrix}$

Quatrième cas : $S_{motif} = \begin{pmatrix} 4 \\ 2 \end{pmatrix}, F = \begin{pmatrix} 02 \\ 20 \end{pmatrix}$

Résumé :

En conclusion, pour définir la relation tâche- tableau, il nous faut l'origine, la matrice P de pavage et la matrice F d'ajustage. On peut combiner les deux équations citées en une seule formule. Soit **r** l'indice

de répétition, $\mathbf{0 \le r < S_{répétition}}$ et **i** l'indice de motif, $\mathbf{0 \le i < S_{motif}}$, les coordonnées du point dans le tableau sont données par la relation suivante :

$$O + (P.F).\begin{pmatrix} r \\ i \end{pmatrix} \bmod S_{tableau} \quad (3.3)$$

telle que $\mathbf{S_{tableau}}$ est la forme du tableau, $\mathbf{S_{motif}}$ est la forme du motif, $\mathbf{S_{répétition}}$ est la forme de l'espace de répétition, O est le vecteur qui définit les cordonnées de l'élément de référence. Voici des constatations qu'on peut en tirer :

- Le nombre de lignes du vecteur d'origine, de la matrice de pavage et de la matrice d'ajustage est égal au nombre de dimensions du tableau.
- Le nombre de colonnes de la matrice de pavage est égal au nombre de dimensions de l'espace de répétition.
- Le nombre de colonnes de la matrice d'ajustage est égal au nombre de dimensions du motif.

3.2.2. Le Reshape

Le Reshape [34] est un connecteur construit par deux Tilers mis en bout à bout. Cette construction est essentielle pour la modélisation en MARTE. Il permet l'expression des liens dans les topologies complexes. Ce concept exprime le réarrangement des éléments entre deux tableaux multidimensionnels source et destination. Dans notre cas il peut exprimer aussi des liens entre les éléments du même tableau.

Le Reshape permet l'expression de :

- La distribution de l'application sur l'architecture.
- Les liens dans la modélisation des architectures à aspect répétitif telles que la topologie des NoCs.
- Le parallélisme existant dans l'application.

3.2.3. Le concept de la dépendance d'inter-répétition.

La dépendance d'inter-répétition connecte un port de sortie d'une tâche T répétée avec un port d'une autre tâche, sachant que la forme des deux ports connectés doit être semblable. Ce concept offre la possibilité de modéliser un lien comme étant un connecteur de dépendance décrit par un vecteur de dépendance **d,** qui exprime la distance de dépendance entre les répétitions de la même tâche. Formellement, la dépendance d'inter-répétition est décrite par la formule suivante [65]:

$$\mathbf{r_{dép} = r - d}$$

où $\mathbf{r_{dép}}$ représente la répétition dépendante, **r** la répétition et **d** la distance.

La dépendance de toutes les répétitions est identique, cela signifie que lorsqu'une répétition **r** dépend d'une autre $\mathbf{r_{dép}}$, la répétition **r** consommera sur son port d'entrée des valeurs produites par le port de sortie de la répétition dépendante. En résumé, un connecteur de type *InterRepetition* est utilisé pour la

modélisation des liens entre les répétitions d'un même composant GASPARD2 dans une structure répétitive telle que les topologies des NoCs. Nous verrons dans la deuxième partie de ce chapitre des exemples d'utilisation de ce concept pour la modélisation des topologies du NoC, telles que le mesh et le torus.

4. Méthodologie proposée pour la modélisation des NoCs en MARTE

Dans cette partie de ce chapitre, nous présentons une de nos contributions. Il s'agit de la proposition d'une méthodologie pour la modélisation des concepts de base des NoCs en utilisant le profil MARTE et l'environnement GASPARD2. Dans notre cas, l'architecture cible est de type réseau d'interconnexion assurant la communication entre les différents éléments qui forment l'application qui est le codeur H.264. Dans la modélisation de l'architecture, les concepts de l'expression de la répétition sont un peu différents par rapport à la modélisation de l'application. Les liens qui relient des tâches répétées n'expriment pas la dépendance de données mais un lien physique. En effet, nous avons commencé par l'identification des concepts des NoCs, qui permettent de construire correctement un réseau de communication pour les SoCs. Puis, proposé une méthodologie permettant de fournir une démarche pour l'expression et la modélisation de chaque concept du NoC en utilisant le profil MARTE. Cette méthodologie a comme objectif de fournir un espace de développement génie logiciel dans lequel les modèles proposés, de la modélisation de l'architecture dans cette partie, passent de l'état contemplatif à l'état productif. L'abstraction des systèmes et la modélisation dans un niveau élevé comme c'est notre cas, correspond à l'état contemplatif, tandis que la génération du code exécutable correspond à l'état productif.

Notre méthodologie de modélisation commence par identifier les concepts des NoCs, nécessaires à l'implémentation, puis identifie l'espace d'exploration de chaque concept dans le profil MARTE. Elle finit par la modélisation, en utilisant la sémantique fournie par chaque paquetage, où on va exprimer le concept relatif au NoC. Dans cette section, nous illustrons pas à pas notre méthodologie de modélisation de la composition entière d'un réseau d'interconnexion. Nous citons les concepts nécessaires à la construction d'un NoC : le choix de la topologie, l'algorithme de routage, le protocole de communication et la technique de commutation. Le diagramme de la figure 3.8 montre la démarche proposée par cette méthodologie. Dans un premier temps, nous commençons par la caractérisation des concepts du NoC que nous voulons modéliser, à savoir : spécifier la topologie, fixer l'algorithme de routage etc. En second lieu, nous proposons les paquetages de MARTE capables de nous offrir la sémantique nécessaire à la modélisation. Entre autres, avec cette méthodologie nous voulons aider le concepteur d'un NoC à trouver directement où il faut modéliser tel concept d'un NoC. Cela signifie que, pour modéliser la topologie, il va falloir utiliser forcément le paquetage RSM du MARTE et non pas un autre. Nous détaillons dans les sous-sections qui viennent la modélisation des concepts cités dans notre méthodologie.

4.1. Modélisation de la topologie

Dans cette section, nous commençons par la modélisation de la topologie. Comme on l'a dit dans les premières parties de cette thèse, le paquetage RSM du profil MARTE permet la modélisation des

structures répétitives régulières. L'objectif est donc de vérifier si le package RSM permet la modélisation de toutes les topologies des NoCs, régulières ou autres. Pour ce faire, nous proposons une démarche pour aider à la modélisation. En premier lieu, et pour pouvoir appliquer notre méthodologie, nous définissons analytiquement la topologie. Dans un second lieu, nous faisons une classification des topologies des NoCs, afin de valider la méthodologie proposée par des topologies de la littérature.

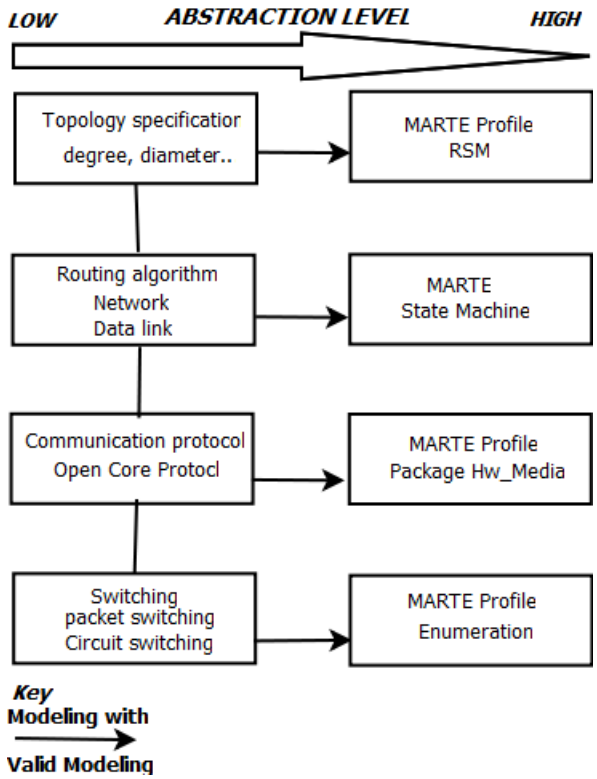

Figure 3. 8: Méthodologie proposée pour la modélisation des NoCs

4.2. Description formelle des topologies des NoCs

La topologie d'un réseau d'interconnexion spécifie l'organisation matérielle du réseau et définit la répartition structurelle de ses routeurs, etc.

Définition : *le graphe de la topologie du NoC est décrit [66] par P (U, F) tel que U est un ensemble non vide de routeurs et F un ensemble non vide de liens, chaque* $u_i \in U$ *représente un routeur de la topologie. Soit le couple* (u_i, u_j) *décrivant deux routeurs communiquants,* $f_{i,j} \in F$ *représente un lien direct entre le routeur* u_i *et* u_j.

La figure 3.9 montre quelques exemples de construction des topologies des NoCs.

4.2.1. Classification des topologies des NoCs

Dans la littérature, les topologies sont usuellement classées suivant plusieurs paramètres. En se basant sur la connectivité, les topologies des NoCs sont réparties en trois classes : directes, indirectes et hybrides. L'autre critère important est la régularité. Il permet de classer les topologies des NoCs en

deux : irrégulières ou régulières. Cependant, tant qu'on parle des topologies, on doit distinguer entre la définition de la régularité pour les NoCs et la régularité au sens mathématique du mot.

Définition de la régularité des NoCs: *une topologie est dite régulière lorsque tous les sommets (routeurs) ont le même nombre de liens qui les relie à leurs voisins, c'est-à-dire le même degré ou valence. Un graphe régulier dont les sommets sont de degré n est appelé un graphe n-régulier ou graphe régulier de degré n. Dans le cas contraire, la topologie est dite irrégulière.*

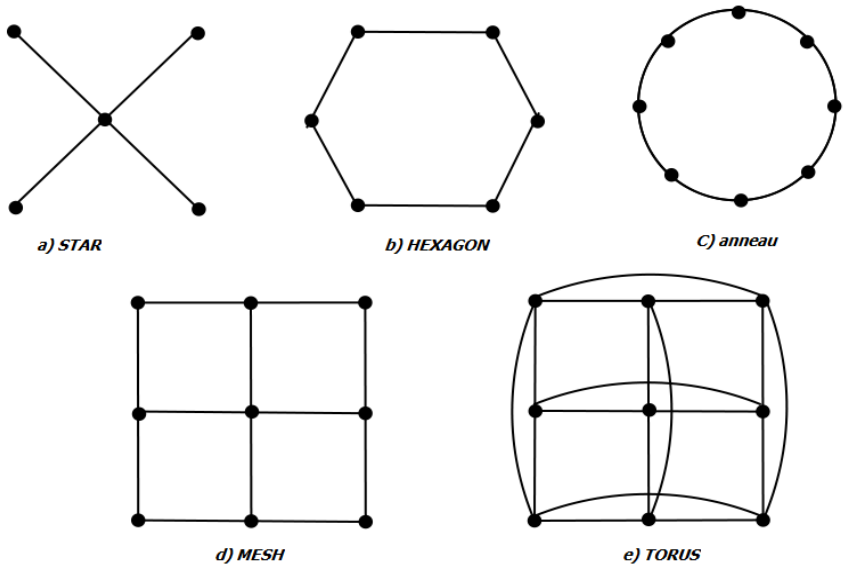

Figure 3. 9 : Exemple des topologies

Dans la littérature, il existe une variété de topologies du NoC, qu'elles soient régulières ou irrégulières, directes ou indirectes. Notons que la liste des topologies n'est pas exhaustive, mais on peut s'intéresser à celles qui sont souvent utilisées dans les NoCs. Dans cette thèse, on classe les topologies en 2 familles, une liste nommé standard [67] du fait que ces topologies sont les plus utilisées dans le monde académique et industriel et qu'elles peuvent être utilisées pour construire d'autres topologies, et une liste des topologies hiérarchiques. Les topologies standard peuvent être régulières ou irrégulières directes ou indirectes. On cite comme exemple la topologie Mesh, le Torus, l'anneau, etc. Cependant, on a défini les topologies hiérarchiques comme des constructions basées sur les topologies standards, la structure équivalente ainsi obtenue est un exemple de réalisation d'un réseau de type hiérarchique. L'objectif de cette classification est de prouver qu'on peut extraire une régularité qui aide à la modélisation d'une façon compacte, en utilisant le paquetage RSM du MARTE. Le fait de partir d'une topologie régulière et de construire une topologie hiérarchique ne garantit pas souvent la régularité de la nouvelle topologie, mais elle peut présenter une certaine régularité locale. Cette observation nous mène à définir les topologies globalement irrégulières localement régulières.

Définition : *une topologie est dite globalement irrégulière localement régulière lorsqu'on a des degrés différents des sommets qui forment la topologie et dans l'ensemble des degrés on trouve un nombre fini qui sont égaux.*

La figure3.10 illustre quelques topologies suivant notre classification.

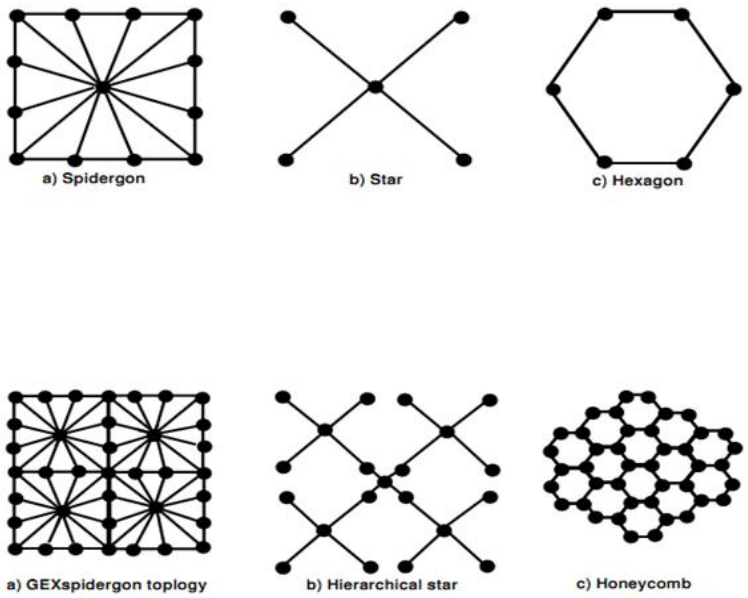

Figure 3. 10 :a),b),c) de la première ligne représentent des topologies standards, les autres sont hiérarchiques.

Habituellement, ce qu'on sait modéliser en RSM, ce sont les topologies régulières. Nous détaillons dans la section suivante la proposition d'une méthodologie pour la modélisation des topologies régulières, irrégulières ou globalement irrégulières localement régulières.

4.2.2. Méthodologie proposée pour la modélisation de la topologie

Notre méthodologie proposée est illustrée dans la figure3.11. L'objectif est d'aider les concepteurs et les utilisateurs du standard MARTE à la modélisation des structures répétitives, précisément les réseaux sur puce ou aussi les réseaux de processeur. Les difficultés rencontrées lors de la modélisation en utilisant le paquetage RSM sont relatives au savoir-faire, c'est-à-dire qu'à un certain moment nous ne savons pas comment commencer, par quoi commencer et quels sont les concepts, fournis par le RSM, que nous devons utiliser pour modéliser d'une façon compacte notre topologie. Comme ceci a été annoncé dans l'introduction de ce chapitre, notre méthodologie pour la modélisation des topologies est réalisée suivant des activités :

- Identification de la topologie : nous définissons la topologie en précisant la taille, le degré, le nombre des ports de chaque routeur.
- Classification des routeurs : préciser le type de routeur suivant le degré.

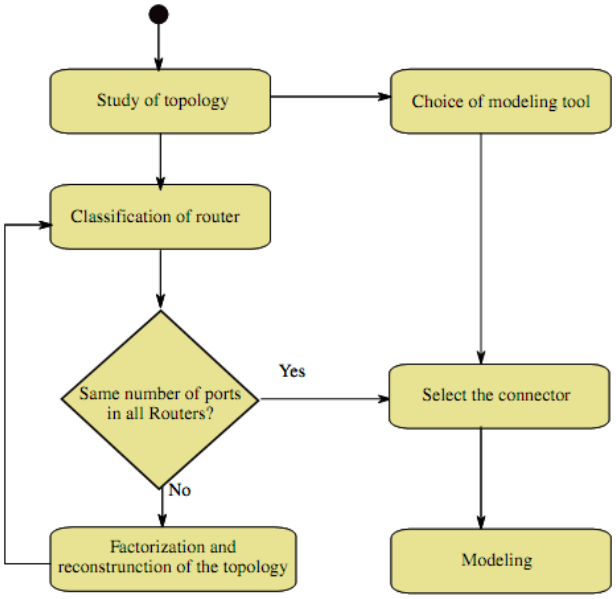

Figure 3. 11: Méthodologie proposée pour la modélisation de la topologie

- Effectuer un test sur le nombre des ports afin de spécifier le type du connecteur (Tiler, Reshape, inter-répétition).
- Si tous les routeurs ont le même nombre de routeurs, alors nous choisissons le connecteur et nous modélisons. Sinon, nous faisons une reconstruction de la topologie en regroupant les routeurs de la même forme et en les classant suivant leurs degrés afin de modéliser d'une façon compacte.

Cette méthodologie de modélisation fait appel aux concepts reliés au standard MARTE. Cela permet aux utilisateurs de ce standard et de l'environnement GASPARD2 de manipuler et de maitriser les différents concepts du paquetage RSM relatif à la modélisation des structures répétitives, afin de profiter d'une présentation visuelle des modèles des topologies. Ces modèles favorisent la réutilisation et la génération du code exécutable indépendamment de la technologie cible.

Dans cette optique, nous passons dans ce qui suit à la validation de cette méthodologie par des topologies de la littérature.

4.2.3. Etudes de cas

Dans la démarche de validation de la méthodologie proposée, nous modélisons la topologie en mesh, torus, nid d'abeille et le GEXspidergon.

La topologie Mesh : elle est construite par une matrice de N×M routeurs interconnectés via des liens physiques, caractérisée par un ensemble des routeurs qui n'ont pas le même degré. Donc, il s'agit d'une topologie standard irrégulière. Dans cette topologie, la brique de base est un routeur de 4 ports d'entrée sortie (In/Out) ; Nord, Sud, Ouest, Est. La structure du réseau Mesh est décrite comme étant une répétition de cette unité élémentaire sur deux dimensions, avec la valeur de répétition sur chaque

dimension donnée par la valeur du"*tagged value : shape*"du stéréotype "*Shaped*". Donc, nous utilisons le connecteur de dépendance d'inter-répétition. La classe routeur qui est stéréotypée "*HwComponent*" et "Shaped"est une propriété de la classe routeur. Nous remarquons aussi que cette topologie présente une dépendance non cyclique entre les liens. Cela se traduit dans l'expression de la dépendance d'inter-répétition par la sémantique ; *isModulo=false.* La figure 3.12 illustre la modélisation d'un exemple d'une topologie Mesh 3×3.

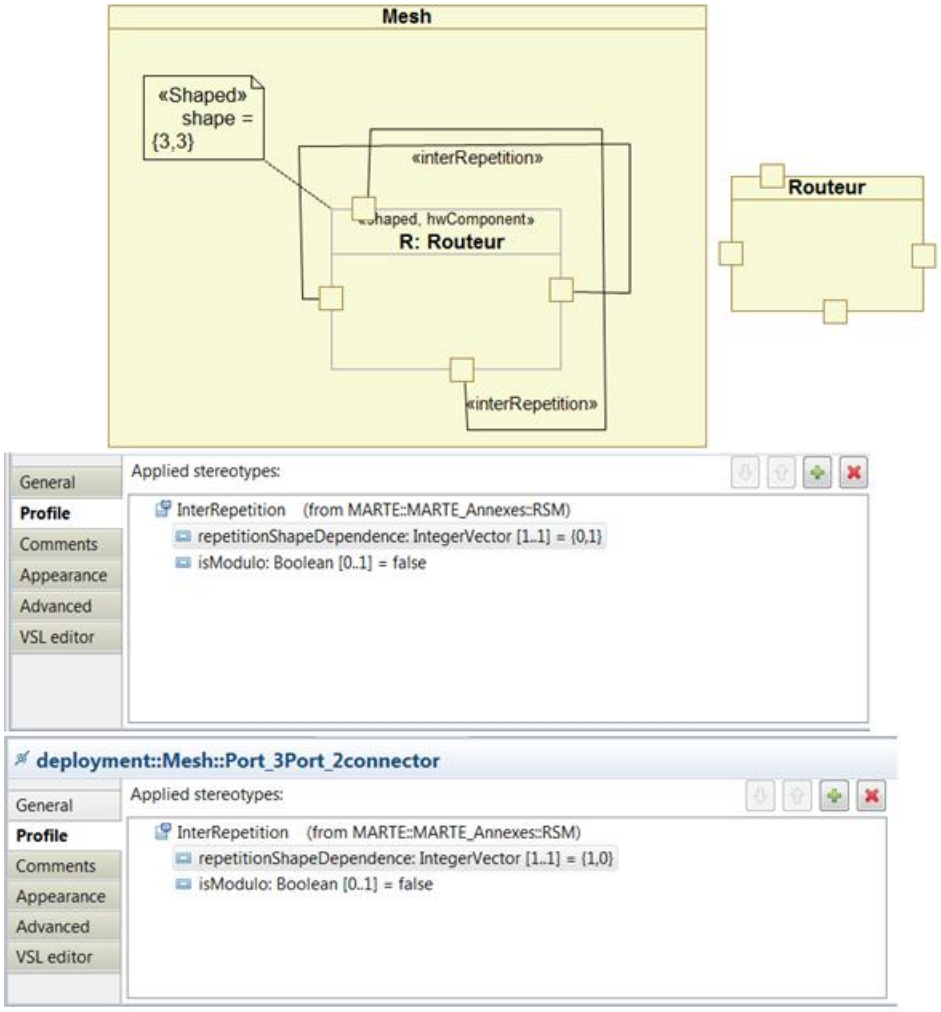

Figure 3. 12 : modélisation de la topologie Mesh.

Chaque routeur est connecté à ses voisins situés à l'est, à l'ouest, au nord et au sud. Le connecteur "*InterRepetition*" spécifie les liens entre les routeurs. La valeur booléenne *false* du "*tagged value :*" *isModulo*" montré dans la figure indique que les routeurs du bord ne sont pas connectés aux autres routeurs des autres bords. L'attribut "repetitionSpaceDependence"est un vecteur qui indique la position d'un routeur voisin dans la direction spécifique. Sa valeur est égale à {0,1} indiquant que, suivant la direction verticale, un routeur de position {i,j} est connecté à un de ses voisins de position{i, j+1}, tandis que le deuxième connecteur possède un vecteur de translation qui est égal à {1,0} indiquant que,

suivant la direction horizontale, un routeur de position {i,j} est connecté à un de ses voisins de position {i+1, j}.

La topologie Torus : Cette topologie est similaire à la topologie Mesh sauf que les routeurs de bord sont connectés aux autres routeurs du bord. Cela se traduit par la présence d'une dépendance cyclique entre les liens d'où l'expression de la dépendance d'inter-répétition par la valeur booléenne *True* du "*tagged value : isModulo*". La figure 3.13 montre la modélisation d'une topologie Torus 3×3.

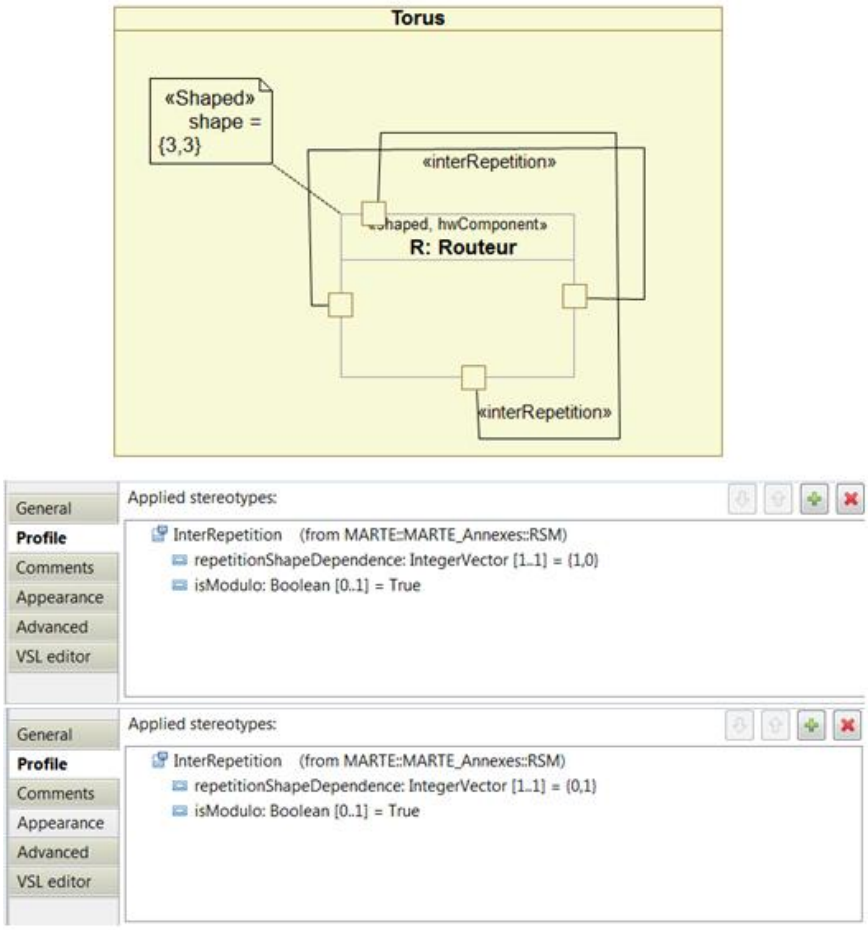

Figure 3. 13: modélisation de la topologie Torus.

Modélisation de GEXspidergon : Le NoC GEXSpidergon est développé au sein du laboratoire d'électronique et micro-électronique [68]. Il est construit par la répétition d'une maille élémentaire qui est le spidergon. Cette topologie est assez complexe en termes de graphes mais elle offre des qualités de services considérables pour la communication sur puce. En appliquant notre méthodologie pour la modélisation des topologies, nous arrivons à montrer, pour la première fois, la possibilité d'utiliser un connecteur "*Reshapé*" sur le même port.

Nous commençons la modélisation par l'identification de cette topologie en précisant le nombre de routeurs connectés et la régularité existante. Elle est caractérisée par 4 types de routeurs qui forment les composants élémentaires pour cette construction. Nous avons les routeurs des coins, les routeurs

horizontaux, les routeurs verticaux et les routeurs du centre. Chaque routeur possède un nombre de ports différents de celui de l'autre famille. On en tire que les connexions entre les routeurs ne sont plus uniformes. En résumé, cette topologie est hiérarchique et globalement irrégulière localement régulière, on a pu extraire une régularité, en termes de routeurs qui ont le même degré. Cela, nous permettes de modéliser cette topologie d'une façon compacte par l'utilisation du connecteur "*Reshape*". Ce connecteur qui existe entre les ports des éléments répétés, cela signifie que chaque port d'un élément répété est connecté à un port de l'autre élément répété. Donc le rôle de ce connecteur est d'indiquer comment distribuer les ports de chaque répétition. La figure 3.14 illustre la modélisation de cette topologie.

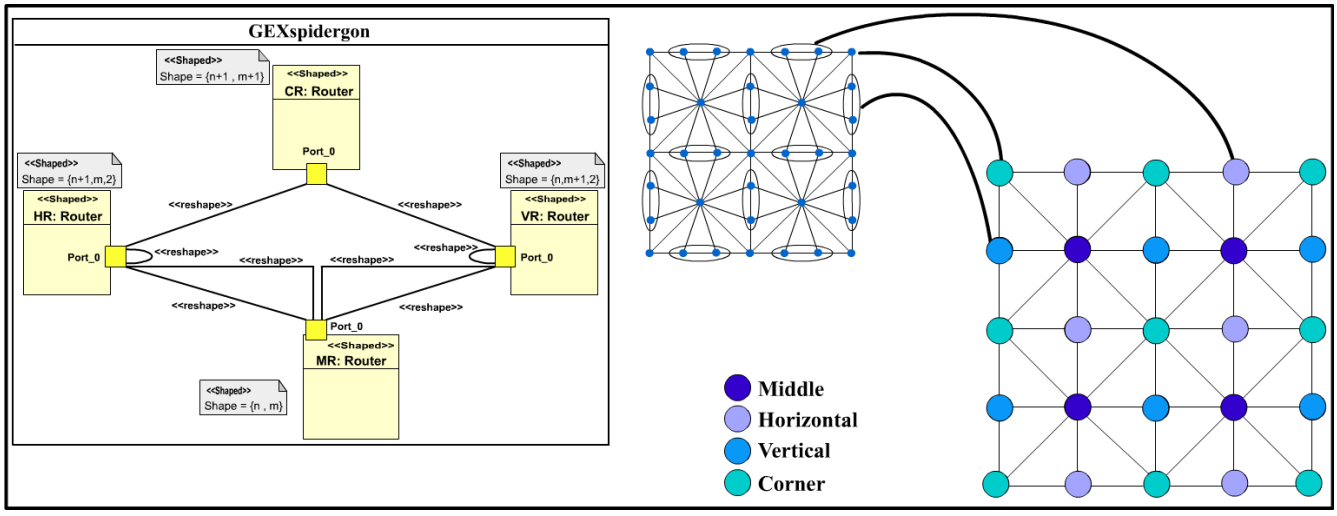

Figure 3. 14: modélisation de la topologie GEXspidergon par le profil MARTE.

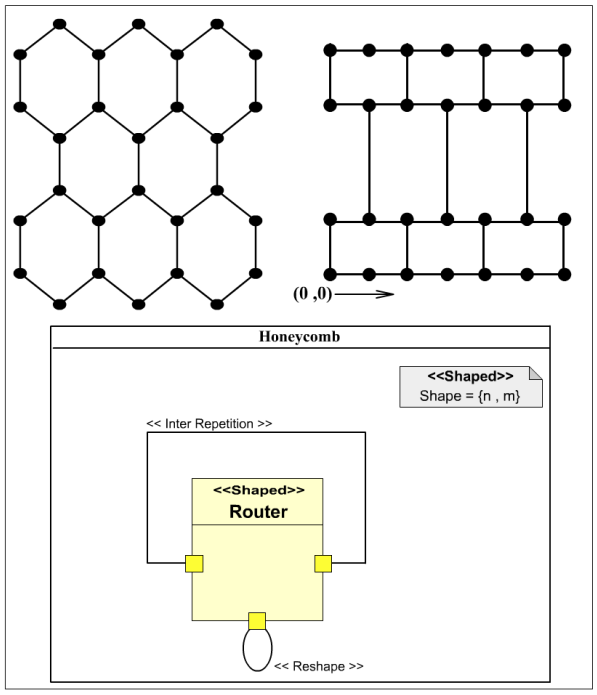

Figure 3. 15 : modélisation de la topologie Honeycomb par le profil MARTE.

Modélisation de Honeycomb

La topologie en nid d'abeilles est considérée comme un réseau d'interconnexion hiérarchique. Il est construit sur une forme hexagonale. Cette topologie est utilisée dans la littérature pour diverses applications comme les téléphones cellulaires. Le nid d'abeilles est obtenu en joignant le centre de chaque triangle dans la maille hexagonale avec le centre du triangle voisin. L'hexagone est un nid d'abeilles de taille 1. Un nid d'abeilles de taille 2 est obtenu par adjonction à six hexagones du bord de délimitation de la taille 1 nid d'abeilles. Par conséquent, un nid d'abeilles de taille m est obtenu à partir d'un nid d'abeille de taille 1. La topologie concernée est formé par une répétition d'un seul type de routeur. Dans notre cas, chaque instance est connecté à des voisins qui se trouvent au nord, sud, et ouest. La figure 3.15 illustre la modélisation en UML MARTE de cette topologie.

Reshape sur le même port

Le connecteur Reshape est utilisé pour spécifier des liens moins réguliers, ce que nous remarquons suite à la modélisation de la topologie GEXspidergon et d'autres telles que le nid d'abeille [69]. Ce connecteur permet aussi l'expression :

- Des liens de plusieurs éléments vers un, et vice versa.
- Des liens entre les éléments du même tableau, ce qui signifie l'utilisation du Reshape sur le même port.

Dans cette partie de la thèse, nous avons réussi à proposer une méthodologie pour la modélisation des topologies des NoCs, afin de vérifier et d'éclaircir la sémantique fournie par le paquetage RSM du profil MARTE pour la modélisation des structures répétitives. Dans la section suivante, nous passons à la modélisation d'un autre concept du NoC, qui est l'algorithme de routage.

4.3. Modélisation des algorithmes de routage

Le mécanisme de routage est l'un des concepts importants pour la modélisation des NoCs dans n'importe quel niveau d'abstraction. Le module de routage, implémenté dans le routeur, est responsable de la transmission efficace des informations. Dans cette section, nous nous intéressons à la modélisation de ce mécanisme par l'utilisation du profil MARTE. Les algorithmes de routage peuvent être implémentés de différentes manières. Les méthodes les plus utilisées consistent, soit à consulter une table de routage, soit à exécuter un algorithme de transmission, basé sur une machine d'états finis, qui peut être logiciel ou matériel. Dans la littérature plusieurs algorithmes de routages sont proposés. Ils peuvent être classés selon différentes catégories, et suivant plusieurs critères [70]. Parmi ces critères on cite :

- Le nombre de destinations, et dans ce cas on a un algorithme de routage où le paquet peut avoir une destination unique (unicast) ou de multiples destinations (multicast).
- L'endroit où les décisions sont prises. Principalement, la fonction de routage détermine le chemin qui peut être établi par un contrôleur centralisé et dans ce cas on parle d'un routage

centralisé. L'autre cas consiste à déterminer d'une façon distribuée le chemin pendant que le message transite à travers le réseau, c'est le routage distribué.

- Le mode de progression : On peut aussi classer les algorithmes de routages en algorithmes de routage progressif ou à rebroussement de chemin.
- L'adaptabilité, et dans ce cas on distingue entre un algorithme de routage déterministe, adaptatif et semi adaptatif.

Ces classifications dépendent essentiellement des critères d'implémentation, et dans un niveau d'abstraction élevé l'objectif est de modéliser indépendamment de tous les détails d'implémentation qui ne sont pas pertinents pour la conception des systèmes. Du coup, nous proposons une classification basée sur une description formelle permettant la spécification de l'algorithme de routage, afin de valider par la modélisation quelques algorithmes de routage en utilisant les concepts des machines d'états fournis par le profil MARTE.

4.3.1. Description formelle des algorithmes de routages et classification

Formellement, un algorithme de routage pour la transmission des paquets dans un NoC est donné par le formalisme suivant :

$$R[p,i]: P \times I \rightarrow O^n$$

Où P représente l'ensemble des paquets, I et O représentent respectivement l'ensemble des ports d'entrée/sortie d'un routeur. On considère que les informations de routage sont données dans l'entête de paquet, la fonction R permet de trouver un chemin entre un port de sortie et un port d'entrée de deux routeurs communiquants, pour transmettre un paquet [70]. En se basant sur cette équation, on peut classer les algorithmes de routages, indépendamment des détails d'implémentation ou d'exécution, en deux familles :

- Si n=1 et $\forall(p,i) \in (P,I), \exists! o \in O / R(p,i) = o$ alors l'algorithme de routage est déterministe.
- Si $1 < n \leq card(O)$ alors l'algorithme de routage est adaptatif et pour un paquet de données plusieurs ports de sortie $O_{\{0,\ldots,\, n-1\}}$ sont possibles.

En résumé, le but de cette classification est de modéliser le mécanisme de routage dans les NoCs loin de tout ce qui est critères relatifs à l'implémentation. Ce mécanisme peut être modélisé, dans un niveau d'abstraction, en utilisant les machines à états finis.

4.3.2. Les machines d'états en UML/ MARTE

La machine à états finis est utilisée pour la modélisation du comportement d'un système via un diagramme d'états. Ce diagramme représente l'ensemble des états possibles pour un système et les transitions entre ces états. Il est défini autour de trois concepts de base : les états, les transitions et les actions. Les états définissent les différentes étapes de son comportement, les transitions spécifient les changements d'états et l'évolution dans le comportement qui peut être exécuté par la machine à états. Le système doit être donc en mesure d'exécuter des actions, soit au moment du déclenchement d'une transition ou à l'intérieur d'un état. On distingue donc deux types de machines : la machine de Mealy et la machine de Moore. Dans le profil MARTE qui est une extension de l'UML, le paquetage

¨*CoreElements*˝ décrit les concepts nécessaires à la spécification et à la modélisation du comportement d'un système par l'utilisation des automates ou des machines à états.

4.3.3. Etude de cas

Dans cette section, nous prenons deux algorithmes de routage, adaptatif et déterministe, proposés dans la littérature pour valider la modélisation en utilisant les machines à états finis.

Modélisation d'un algorithme de routage déterministe

Dans les algorithmes de routages déterministes, le chemin est établi en fonction de l'adresse de destination en utilisant le même chemin entre chaque paire de nœuds. Ce routage possède différents noms tels qu'algorithme « XY » pour un réseau à deux dimensions et algorithme « e-cube » pour la topologie hypercube. Nous nous intéressons à la modélisation d'un algorithme de routage en XY. La figure 3.16 illustre un exemple de cet algorithme pour une topologie maillée 3×3.

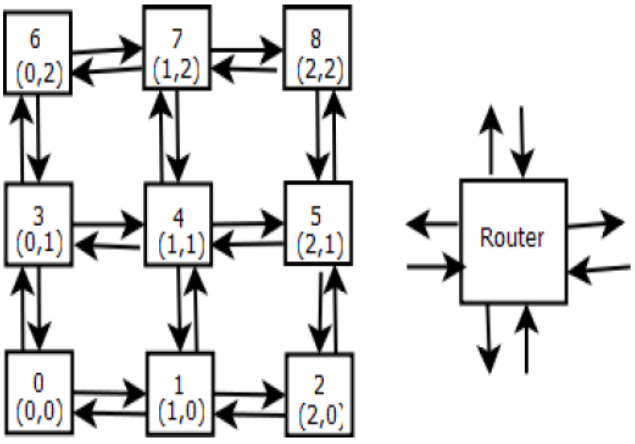

Figure 3. 16 : Exemple de routage déterministe

Le principe de cette technique de routage est décrit par l'algorithme 1. Chaque routeur dans le réseau possède une adresse unique indiquant son emplacement sur les deux axes X et Y. Le paquet circulant sur le réseau contient dans son entête les valeurs des composantes X et Y. Cet algorithme commence par comparer l'adresse du routeur courant avec l'adresse du routeur destination. Si $C_x > D_x$, le paquet va être routé vers l'Ouest. Dans le cas contraire, il prend la direction de l'Est. Lorsque $C_x = D_x$, le paquet tourne en direction du nœud ayant la même adresse Y. Dans le cas où la dernière condition est vraie, la fonction de routage passe à la comparaison des composantes suivant la direction en Y. Le paquet est en direction du Nord lorsque $C_y > D_y$ sinon, il tourne vers le Sud. Une fois que l'adresse de destination est atteinte, c'est-à-dire que l'adresse du nœud correspond à l'adresse donnée par l'entête du paquet, le paquet est routé vers le cœur récepteur. En conclusion, le paquet peut se diriger, soit vers le haut, soit vers le bas lorsqu'il circule sur l'axe des X, et vers la gauche ou la droite lorsqu'il circule sur l'axe des Y.

Algorithme1 :
Routeur destination: (Dx;Dy)
Routeur courant: (Cx;Cy)
If (Dx > Cx) // ***Move*** East
Return E
Else if (Dx < Cx) // ***Move*** West
Return w
Else if (Dx = Cx) // ***same*** direction
If (Dy < Cy) // **Move** South
Return S
Else if (Dy > Cy) // ***Move*** North
Return N
Else if (Dy = Cy) //***same*** router
Return C

Cet algorithme de routage déterministe peut être modélisé par une simple machine d'états en utilisant le profil MARTE. La figure 3.17 illustre une machine à états finis qui modélise le comportement de cet algorithme.

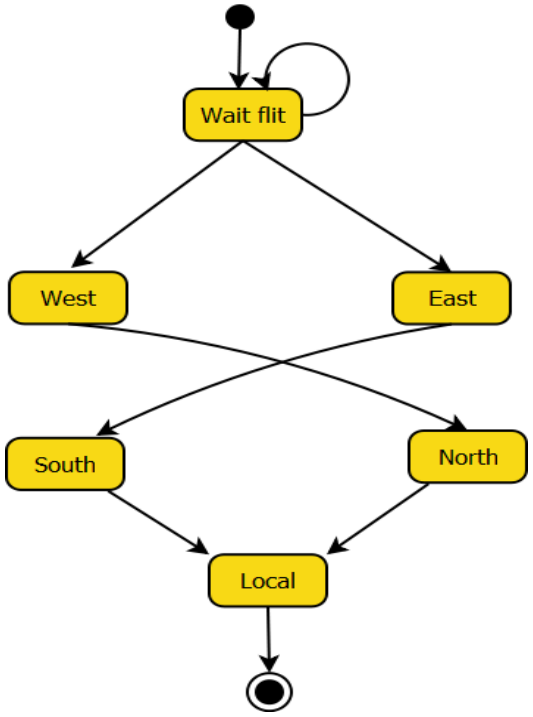

Figure 3. 17 : Modélisation d'un algorithme de routage déterministe

Modélisation d'un algorithme de routage adaptatif

Dans cette partie, nous détaillons le principe d'un algorithme adaptatif. Le principal avantage de cet algorithme est qu'il s'adapte à plusieurs topologies. Le principe est que les paquets peuvent circuler

dans le réseau facilement sans objections. Dans la stratégie de routage adaptatif, le chemin à emprunter par le paquet est décidé dans les routeurs intermédiaires avant chaque saut. Cela se traduit par l'utilisation d'un arbitrage dynamique, ce qui influe directement sur l'architecture du routeur. Un algorithme de routage adaptatif, pour la topologie Mesh, de la littérature [71] est présenté dans la figure 3.18

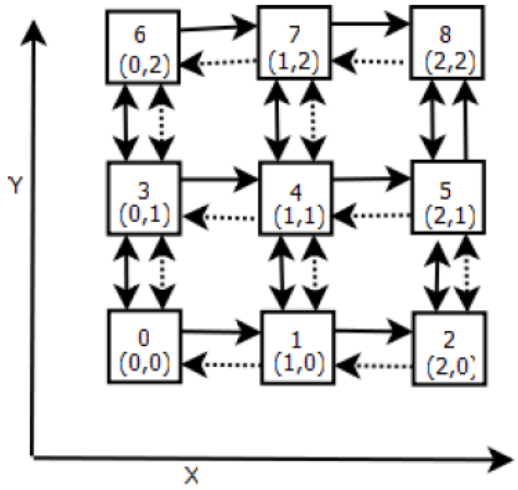

Figure 3. 18: Exemple de routage adaptatif

Dans un algorithme de routage adaptatif, les chemins sont dynamiquement changés suivant des conditions sur l'état du réseau telles que les problèmes de congestion. Pour remédier à ces problèmes, une fonction coût doit être implémentée dans l'unité de routage. Cette fonction permet de sélectionner quel chemin doit emprunter un paquet dans le cas où les chemins destinés sont occupés. Un algorithme de routage adaptatif augmente la complexité matérielle du routeur. En effet une modélisation à haut niveau s'avère importante. L'algorithme de routage adaptatif présenté dans la figure 3.18 est nommé « double-Ychannel routing ». Cet algorithme est appliqué à des topologies maillées à deux dimensions. Les routeurs sont connectés par un lien bidirectionnel dans la direction de X et par deux liens bidirectionnels dans la dimension de Y. Donc, le réseau est divisé en deux sous-réseaux X1 et X2. Le déplacement sur le premier sous- réseau est vers le haut tandis qu'il est vers le bas dans le second sous-réseau. Si on considère deux nœuds, source et destination de coordonnées respectives S_x et D_x, lorsque la destination du paquet est vers la droite, il va utiliser le sous-réseau X1 sinon, c'est-à-dire si la direction du paquet est vers la gauche, il va utiliser le sous réseau X2. Dans la condition où $D_x=S_x$ une fonction coût décide quel sous-réseau doit être emprunté. L'algorithme 2 décrit le fonctionnement de cet algorithme.

Algorithme 2 :

Routeur destination: (Dx;Dy)

Routeur source: (Sx;Sy)

If (Dx > Sx) // ***Move*** in X1

Return X1

Else if (Dx < Sx) // ***Move*** In X2

Return X2

Else if (Dx = Sx) // ***CF***

Return CF

Une simple machine à états finis est suffisante pour modéliser le comportement de cette stratégie de routage permettant une manipulation compacte et rapide. La figure 3.19 illustre la machine à états modélisant cet algorithme.

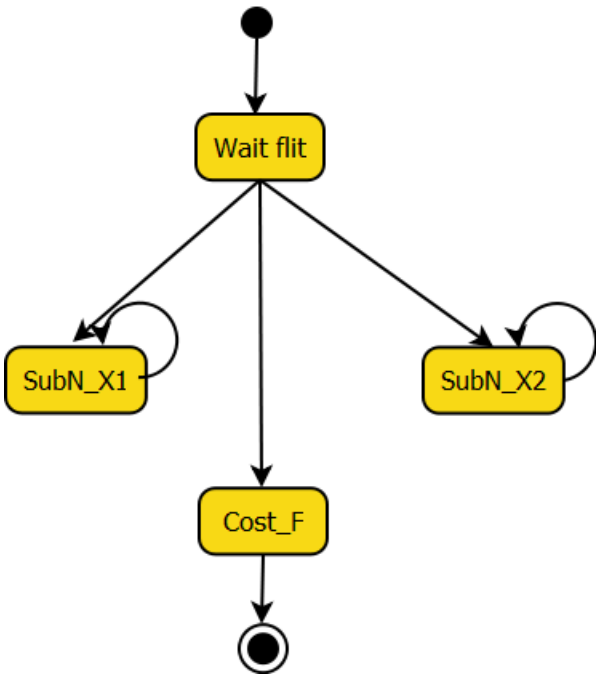

Figure 3. 19 : Exemple de machine à états pour la modélisation d'un algorithme de routage adaptatif

4.4. Modélisation du protocole de communication

Dans le réseau sur puce, on trouve le cœur ou l'IP, qui représente l'unité de traitement de données, et le routeur qui achemine les informations dans le réseau. Entre ces deux éléments, on trouve l'interface (Network Interface, NI), réseau qui sépare matériellement le cœur de la fonction communication. Son rôle est de rendre compréhensible les données envoyées par le cœur vers le routeur. Les standards de

communication deviennent de plus en plus utilisables dans les NoCs. C'est grâce à eux que les IPs sont conçus avec des NIs. Plusieurs standards de communication sont définis, parmi eux en trouve le VCI « Virtual Component Interface » [72] et l'OCP « Open Core Protocol » [73]. Le principal avantage de ces standards est de rendre les IPs compatibles avec n'importe quels systèmes de communications, cela favorise la réutilisation des IPs indépendamment de l'infrastructure de communication.

La modélisation à haut niveau d'abstraction du protocole de communication doit laisser la liberté au concepteur ou au chercheur de choisir n'importe quel protocole qu'il considère comme adéquat pour son système. En se basant sur l'objectif de ce concept, qui consiste à rendre standard les signaux de communication entre l'IP et le routeur, on peut modéliser le protocole de communication par des ports, interconnectant le routeur à un IP de type quelconque qu'il soit OCP, VCI, ou autre.

Le concept *"FlowPort"* étudié auparavant est un concept du paquetage GCM qui permet de supporter le paradigme communication entre les composants qui forment le système. La spécification de la nature de flux est garantie par ce concept. Par l'utilisation de ce concept, on peut modéliser les interfaces de communication comme étant des ports, ce qui favorise la réutilisation de la modélisation et aussi l'indépendance par rapport au type de standard. Le choix du standard et la caractérisation des signaux peuvent se faire au moment de la phase de déploiement, lorsqu'on affecte à chaque composante son code.

4.5. Modélisation des techniques de commutation dans les NoCs

Les techniques de commutations dans les NoCs ont fait l'objet de plusieurs publications. La plupart des solutions convergent vers l'utilisation des deux techniques : la commutation de circuits et la commutation de paquets.

4.5.1. Commutation de circuits

Le principe, dans ce mode de commutation [74], est d'établir entre la destination et la source de données un chemin durant toute la communication. En effet, des éléments du réseau sont alloués pour assurer le transfert et la communication entre les deux nœuds. Cette allocation est spécifique. Cela signifie qu'aucune autre communication ne peut être établie sur les éléments qui sont déjà alloués. Cet ancien mode est inspiré de la technique de communication sur les lignes téléphoniques. Le chemin est tout d'abord établi entre le récepteur et l'émetteur puis le paquet est envoyé en une seule fois à la destination en traversant le chemin établi. Ce mécanisme est important lorsqu'on veut transmettre une énorme quantité de données.

4.5.2. Commutation de message

Dans ce mécanisme [74], les messages à transmettre sont découpés en paquets qui sont envoyés indépendamment dans le réseau. Le message à transmettre sur le réseau est mémorisé et retransmis chaque fois vers le nœud suivant jusqu'à atteindre sa destination. Cela favorise l'accès rapide au réseau du fait que les paquets peuvent traverser des chemins différents. Au moment de l'arrivage à la destination, les paquets sont regroupés grâce aux informations sur leurs destinations. Cette technique nécessite des mécanismes de contrôle de flux car les paquets peuvent se perdre dans le réseau.

Concernant la modélisation au niveau système de ce concept du NoC, le profil MARTE ne permet pas sa modélisation, du fait qu'il s'agit d'un formalisme générale qui décrit la manière avec laquelle les routeurs commutent les paquets entre eux. Ainsi il ne s'agit pas d'un objet concret ni d'un protocole qu'on peut lui associé une machine d'état ou un code, etc. En effet, nous montrons dans la section suivante que ce formalisme peut être modélisé comme étant une classe énumérée dont la commutation de paquet et celle de circuit représentent ses deux champs.

5. Bilan sur la modélisation des concepts des NoCs

Les travaux présentés dans cette partie s'intéressent à la modélisation des concepts de base des réseaux sur puce. Nous venons de présenter notre méthodologie de modélisation des concepts de NoCs. Elle permet une modélisation compréhensible et loin de toutes les détailles d'implémentations matérielles. Cette méthodologie est innovante du point de vue de l'adaptation d'une démarche IDM. Par ailleurs, la modélisation des réseaux sur puce, en suivant notre méthodologie, est indépendante du langage cible. Elle assure la modélisation dans un niveau d'abstraction élevé, ce qui permet au concepteur de modéliser les NoCs sans être expert dans le domaine.
La méthodologie proposée approuve une identification des différentes caractéristiques du réseau sur puce en précisant pour chaque concept le paquetage du profil MARTE assurant sa modélisation. Nous avons montré tout au long de cette partie que le profil MARTE est une solution intéressante pour la modélisation dans un niveau d'abstraction élevé. Nous avons commencé par la modélisation de la topologie en utilisant le paquetage RSM et nous venons d'éclaircir quelques concepts et aussi de vérifier que ces concepts permettent de modéliser d'une façon compacte la topologie des NoCs. Ensuite, nous sommes passés à la modélisation des algorithmes de routages par le biais des machines à états finis. L'interface de communication, à son tour, est aussi modélisée indépendamment de tous les détails de mise en œuvre. Enfin, la technique de commutation n'a pas de description comportementale sur laquelle on puisse se baser pour la modéliser en utilisant le profil MARTE.

Les réseaux sur puce représentent, aujourd'hui, l'infrastructure de communication la plus intéressante pour les systèmes sur puce. C'est pour cette raison qu'ils font l'objet de nombreux travaux de recherche. Ils contribuent de plus en plus à l'essor de la communication sur puce, vu les performances qu'ils offrent au système, telles que la flexibilité, la qualité de service, le débit, la bande passante, etc.

Ces différentes constatations nous amènes à proposer une extension dans le profil MARTE, plus précisément dans le paquetage *Hw_Media.* Notre proposition repose sur le fait que dans ce standard, il n'y a pas une spécification permettant la modélisation des réseaux sur puce. Tandis qu'on trouve des stéréotypes spécifiques pour la modélisation des bus, des arbitres, etc. L'autre aspect intéressant est qu'il y a d'autres caractéristiques du réseau sur puce qu'on ne peut pas présenter au moment de la modélisation, telles que la notion de synchronisme, mode de commutation et le type de mémorisation. Nous montrons dans la suite notre proposition pour l'extension du profil MARTE.

Nous allons commencer par présenter le paquetage *Hw_communication* en mettant l'accent sur les différents stéréotypes qui forment ce paquetage, afin de proposer notre extension.

6. Le paquetage Hw_Communication

L'objectif du paquetage *HW_Communication* est de regrouper tous les éléments qui assurent la communication dans les systèmes sur puce. Il offre une vue de la communication autonome dans les architectures matérielles. La figure3.20 illustre les éléments formant ce paquetage.

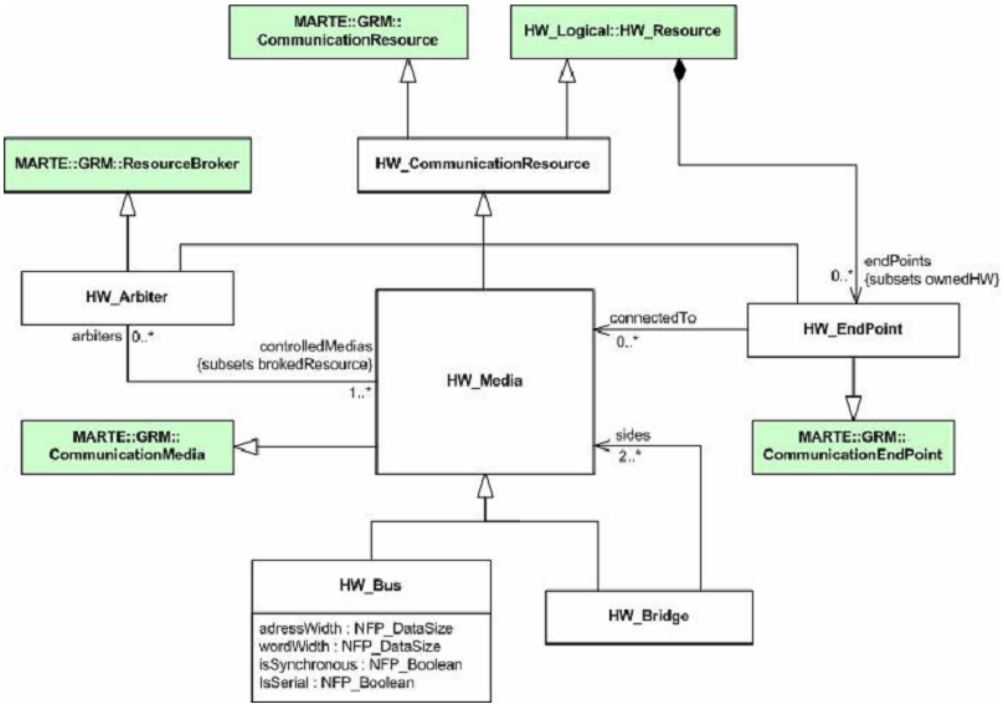

Figure 3. 20: paquetage Hw_Communication du profil MARTE.

Le Hw_Media est un concept central du paquetage *Hw_Communication*, il désigne toute source de communication capable de transférer des données tout en spécifiant des contraintes telles que le débit. Il pourrait être contrôlé par de nombreux HW_Arbitr et il peut aussi être relié à un autre medium de communication à l'aide du concept Hw_bridge. Les concepts regroupés dans ce paquetage sont définis comme suit :

HW_EndPoint : c'est un concept générique qui symbolise les points d'interface d'un *Hw_Resource*, donc il s'agit d'une partie qui sert d'interface pour communiquer avec d'autres ressources à travers le concept Hw_Media. *Hw_EndPoint* peut être un port ou un slot ou un pin.

HW_Bridge : Le HW_Bridge permet de gérer, typiquement, les communications entre deux ou plusieurs HW_Medias. Il peut réaliser une adaptation entre des protocoles de communication différents.

HW_Arbiter :HW_Arbiter est une ressource intermédiaire dans la communication. Il contrôle au moins un élément communiquant et effectue l'arbitrage pour assurer le bon déroulement de la communication entre des composants communiquant sur le bus. Il pourrait donc intégrer une stratégie d'arbitrage permettant de gérer l'ordonnancement pour accéder au bus.

HW_bus : c'est un composant matériel de *Hw_Media*, il est caractérisé par la taille de données, son adresse et le synchronisme.

A partir de cette description du paquetage *Hw_Communication,* nous remarquons qu'il tient compte des éléments pour la communication à travers les bus. Par ailleurs, une première observation nous a fait penser à ajouter un stéréotype qui décrit la brique de base pour construire et modéliser un réseau sur puce sans être un expert du domaine.

7. Proposition d'une extension dans le profil MARTE.

Cette partie représente une de nos contributions principales dans cette thèse. Suite à la modélisation des concepts des réseaux sur puce en utilisant le profil MARTE, nous arrivons à proposer cette extension. Nous justifions cette proposition en nous basant sur quelques observations. L'une d'elles, déjà citée dans la section précédente, c'est que ce standard fournit un paquetage qui s'intéresse à la modalisation des structures de communications dans les systèmes sur puce, mais il tient compte seulement des concepts relatifs à l'usage des bus comme medium de communication. D'un autre coté les bus ont montré leurs limites, lorsque le nombre des IPs communiquant est important. La solution proposée est de remplacer les bus par les réseaux sur puce et de profiter de ces services. Ce motif est l'une de nos motivations pour la proposition de cette extension.
Les questions qui se posent à ce niveau et qui justifient cette proposition peuvent être, principalement, de deux ordres :

Q1 : Pourquoi nous avons proposé cette extension ?

Comme nous l'avons déjà dit, un premier motif, c'est que les concepts relatifs à la modélisation de la brique de base, qui est le routeur, assurant la spécification des réseaux sur puce, ne sont pas pris en compte dans le paquetage décrivant la communication.
Autre point, lors de la modélisation des concepts des NoCs, nous avons montré que le profil MARTE est incapable de spécifier les techniques de commutations, c'est pour cette raison que nous avons dit qu'il peut être modélisé comme étant une énumération, tandis que c'est un concept indispensable dans la conception des NoCs. D'autres aspects, nécessaires pour les NoCs, qui sont des caractéristiques non fonctionnelles, telles que la synchronisation et la mémorisation ne peuvent pas être spécifiés via le profil MARTE. L'autre raison est de faciliter la tâche du concepteur. Sans être expert dans les NoCs et sans entrer dans les détails que nous avons proposés dans notre méthodologie, il va arriver à modéliser un réseau sur puce en stéréotypant une classe par un stéréotype *Hw_Router*, que nous avons proposé, afin d'utiliser uniquement le paquetage RSM pour représenter d'une façon compacte la topologie.

Q2 : Ce qu'on a proposé est-il suffisant ?

Dans la littérature, toutes les architectures proposées pour les routeurs sont issues d'un modèle global et générique. Nous nous sommes basés sur ce modèle pour proposer notre stéréotype. Le rôle fondamental d'un routeur est de router les messages dans le réseau, ce routeur est caractérisé par la taille des paquets, le type de mémorisation. Comme c'est indiqué dans le chapitre 2, il peut être aussi synchrone ou asynchrone, et le routage peut être adaptatif ou déterministe. La figure 3.21 illustre la structure de base d'un routeur sur laquelle reposent tous les routeurs de l'état de l'art [75, 76].

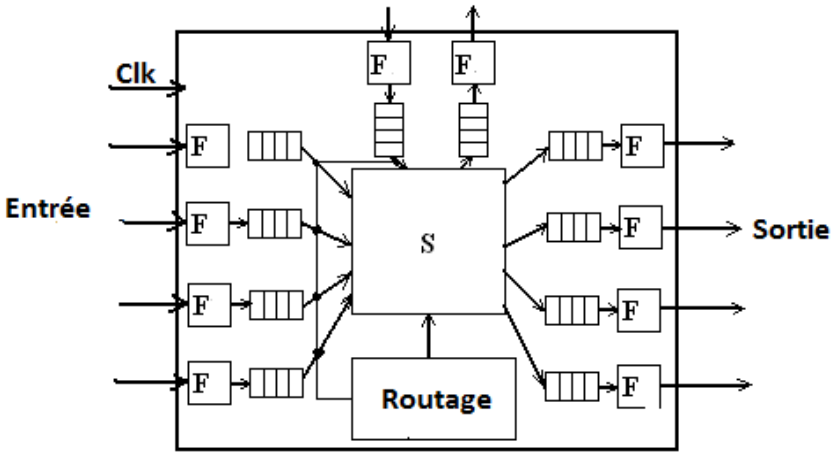

Figure 3. 21: Modèle d'un routeur générique

En se basant sur ce modèle global, nous avons pu proposer le stéréotype *Hw_Router* qui représente le composant élémentaire pour la construction d'un NoC. La figure 3.22 montre le stéréotype proposé.

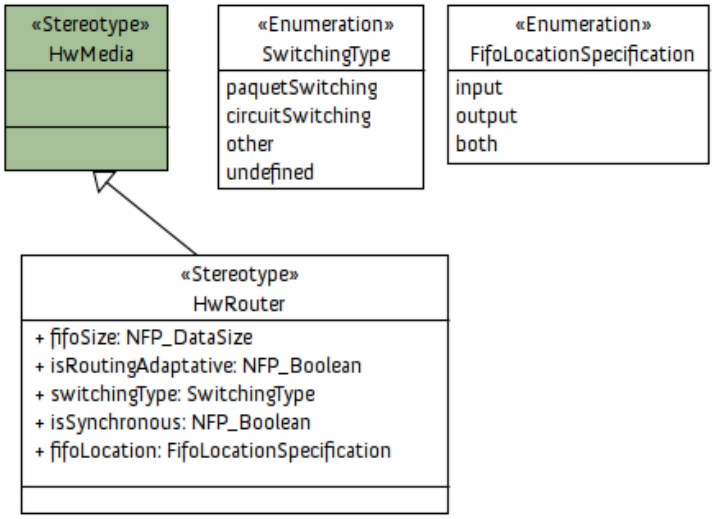

Figure 3. 22: Stéréotype proposé.

Le *Hw_Router* est un élément de paquetage *Hw_Media* du paquetage *Hw_communication* . Il est décrit par les champs suivants :

Les attributs : ce stéréotype possède les attributs suivants :

FiFoSize :NFP_DataSize : indique la taille de paquet, il est exprimé généralement en bits.
isRoutingAdaptatif :NFP_Boolean : spécification du type de routage.
switchingType :SwitchingType : définit le type de commutation donné par le stéréotype énumération
IsSynchronous : NFP_Boolean : spécifie si le routeur est synchrone ou asynchrone.
FiFoLocation :FifoLocationSpecification :indique l'emplacement de FIFO et le type de mémorisation.
Contrainte : Si le routeur est synchrone, il doit avoir une fréquence.

En résumé, l'objectif de cette proposition d'un stéréotype *Hw_Router* est de fournir au profil MARTE une balise assurant la modélisation fonctionnelle à partir d'une description graphique des réseaux sur puce.

8. Conclusion

Dans ce chapitre, nous avons présenté en détails la méthodologie proposée pour la modélisation des concepts des NoCs. Nous avons commencé par la modélisation de la topologie à l'aide du paquetage RSM. Nous arrivons dans cette partie à éclaircir quelques concepts afin de montrer pour la première fois dans la littérature la possibilité d'utiliser le connecteur Reshape sur le même port. Notre étude a prouvé la possibilité de modéliser les topologies régulières, irrégulières et globalement irrégulières localement régulières. Nous avons aussi abordé dans une démarche méthodologique la modélisation des autres concepts de base des réseaux sur puce. Notre étude a montré certaines limites dans le profil MARTE concernant la spécification de medium de communication pour les systèmes sur puce. En effet une proposition d'extension de ce standard, pour pouvoir supporter la modélisation des NoCs, a été définie.
Dans le chapitre suivant, nous nous intéressons à la description architecturale, en VHDL, d'un routeur, qui représente le composant élémentaire dans la modélisation en MARTE, afin de proposer une topologie nommée « Diagonal Mesh ». Ce routeur va être intégré dans la bibliothèque des IPs du GASPARD. La topologie étudiée permettra la communication entre les différents IPs du codeur H.264.

Chapitre 4

Vers un model fonctionnel VHDL d'un NoC DMesh

1. Introduction

La synthèse de communication est l'approche relative à la séparation entre les noyaux de calcul et le medium de communication dans les systèmes sur puce. Elle s'intéresse à la modélisation au niveau système ou au niveau RTL (Register Transfert Level) des structures de communication. Dans le monde académique et industriel, les chercheurs et les concepteurs des SoCs ont donné du temps et de l'argent pour obtenir des SoCs avec les meilleures performances par le biais d'un choix judicieux de l'architecture de communication. Dans la section 6 du deuxième chapitre, nous avons présenté un préambule sur les concepts relatifs à la conception architecturale des réseaux sur puce. Il est important de séparer les problèmes vu qu'il est utile, pour le concepteur de l'application, de connaitre les fonctions à intégrer pour envoyer les messages, mais il est inutile de savoir comment la communication se fait. En effet, les éléments de communication peuvent être conçus indépendamment des IPs de l'application. Les architectures de réseaux sur puce apparaissent comme une solution pertinente pour remédier aux problèmes liés aux besoins en communication d'un système sur puce complexe intégrant un nombre important d'IP. Dans la littérature, plusieurs travaux ont été menés, dont l'objectif est de proposer de nouvelles architectures offrant de meilleures performances. Comme nous l'avons vu dans le chapitre 2, les performances d'un système sur puce dépendent fortement des performances de son réseau de communication. Choisir la forme du réseau de communication compte parmi les difficultés majeures de conception des systèmes sur puce. Dans la littérature, plusieurs topologies ont été employées telles que le Mesh, l'octagone le spidergon, etc. Le choix de la topologie dépend principalement de l'exigence de l'application et des critères de décision. On cite la latence, le débit, la surface occupée, le coût en énergie consommée, la flexibilité, etc. Nous proposons dans ce chapitre une topologie inspirée de la littérature, nommée « Diagonal Mesh » avec les détails architecturaux de ces briques de base.

La première section de ce chapitre s'intéresse aux enjeux liés à la conception des réseaux sur puce. Nous expliquons ensuite les motivations et les objectifs pour concevoir un NoC. Nous détaillons dans la section suivante des travaux de littérature, pour consacrer la cinquième section à l'exposition de l'architecture proposée. Nous finissons par une validation et une évaluation de performances, consolidée par une étude comparative.

2. Les enjeux liés à la conception des réseaux sur puce

La difficulté de conception des réseaux sur puce est problématique dans la mesure où l'émergence et la variété des applications limitent la réutilisation du routeur qui forme le réseau. Donc pour faire face aux évolutions technologiques et pour répondre à l'exigence de l'utilisateur, la conception des systèmes sur

puce, basée sur les réseaux sur puce, doit être flexible permettant d'avoir les performances requises par l'application. Dans l'évaluation de l'architecture d'un réseau sur puce, plusieurs critères peuvent être pris en considération. Nous souhaitons une architecture qui offre une bande passante élevée, une faible latence, peu énergivore et le tout implémenté sur une faible surface en silicium.
Dans le flot de conception pratique, il s'agit de commencer par la mise en œuvre de l'architecture de la brique de base du NoC, c'est-à-dire le routeur. Il définit tous les concepts nécessaires pour assurer la communication entre les IPs d'une application. C'est dans le routeur que le routage et le mécanisme de communication sont définis. D'autres paramètres cruciaux tels que la taille des FIFOs et la taille des liens physiques peuvent notamment être dimensionnés au cours de cette phase de conception. Le système n'est complet que lorsque les éléments constitutifs du réseau sont mis en place. Les performances d'un réseau sur puce dépendent essentiellement du nombre de routeurs. La position des IPs va aussi avoir une influence sur la latence. En effet, pendant la phase de conception, il faut tenir compte de toutes ces contraintes afin de fournir une meilleure infrastructure de communication. Dans la littérature, plusieurs travaux dont l'objectif est de proposer des architectures garantissant les performances nécessaires, ont été proposés. La section suivante s'intéresse à la présentation de quelques travaux.

3. Etat de l'art des architectures de réseau sur puce

Pour spécifier les enjeux présentés auparavant, nous présentons dans cette section, concrètement, des architectures de NoC présentées dans la littérature.

SPIN

Le réseau Spin est une proposition académique conçue par une équipe de recherche du laboratoire LIP6 de l'université Pierre et Marie Curie de Paris [77]. Ce réseau est caractérisé par une topologie en arbre élargi, un mécanisme de commutation de paquets, un faible diamètre d'où une latence minimale. Par contre, c'est une topologie non régulière avec extensibilité limitée. La figure 4.1 illustre cette topologie.

La complexité de cette topologie vient de l'utilisation d'un algorithme de routage adaptatif, ce qui impose la présence d'un ordonnanceur pour pouvoir regrouper, à la réception, les paquets dans l'ordre convenable. La mémorisation dans le routeur se fait par des FIFOs placés à l'entrée. Ce réseau académique a été évalué par une comparaison avec un bus. L'étude a montré la transcendance du NoC Spin en termes de cycles d'horloge nécessaires à la transmission d'un paquet de taille donnée.

OCTAGON

C'est un réseau développé par STMicroelectronics [77]. La topologie prend la forme d'un anneau dont les routeurs diamétralement opposés sont reliés par des liens supplémentaires. Elle est destinée aux applications manipulant une quantité d'information énorme. Cette topologie garantit plusieurs avantages tels que : un diamètre faible, une fonction de routage simple à manipuler. Ce réseau est extensible par juxtaposition d'autres topologies OCTAGON via un routeur pont. La figure 4.2 illustre cette topologie.

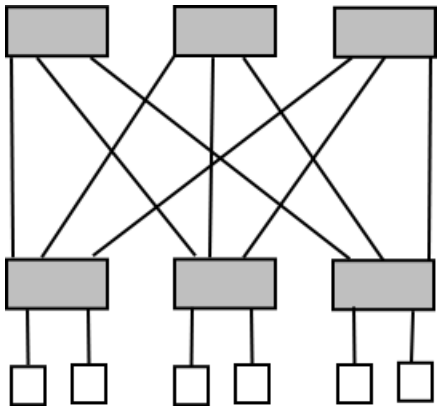

Figure 4. 1 : Topologie en arbre élargi.

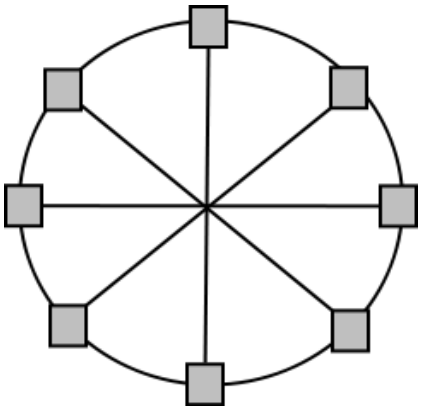

Figure 4. 2 : Topologie en Octagon

HERMES

C'est un réseau direct proposé par F.Moraes [78], caractérisé par une structure maillée à deux dimensions. Le mécanisme de commutation adapté est la commutation par paquets, une mémorisation en entrée, un arbitrage à priorité tournante, un routage déterministe permettent ensemble d'éviter les problèmes de conflit et de congestion sur le réseau. L'algorithme de communication adapté à ce réseau permet d'avoir un fonctionnement avec plusieurs cadences d'horloge, ce qui implique le paradigme Globalement Asynchrone Localement Synchrone (GALS). Il intègre la qualité de service grâce à l'adaptation de deux chemins virtuels et à la priorité de paquets.

ARTERIS

C'est un réseau industriel flexible développé par la société Arteris [79], figurant parmi les premières implémentations commerciales d'un réseau sur puce. La topologie de ce réseau est non définie, elle s'adapte suivant les besoins de l'application. C'est une architecture « scalable » qui utilise des interfaces réseau spécifiques appelées NIU (Network Interface Units) permettant la connexion à de nombreux IPs. L'architecture du routeur de ce réseau est conçue de façon à avoir la possibilité d'une variété d'horloges, ce qui favorise la présence d'une communication GALS. Une qualité de service à

priorité de paquets est intégrée dans ce réseau, elle permet le passage des paquets de plus haute priorité indépendamment de la taille des autres paquets.

MANGO

Une équipe de recherche à l'université de Danemark a présenté le fruit d'un travail sur les réseaux sur puce. Il s'agit d'un réseau sur puce nommé MANGO [80] (Message-passing Asynchronous Network-on-chip providing Guaranteed services over OCP interface). Il se base sur une topologie maillée à deux dimensions et s'appuie sur un routeur asynchrone.

4. Synthèse de l'état de l'art des réseaux sur puce

Comme cela a été dit dans la section précédente, les réseaux sur puce ont des origines variées, académiques et industrielles. Nous remarquons que chaque critère dans les NoCs peut faire l'objet d'une immense réflexion en fonction des besoins du SoC à concevoir. Les réseaux proposés peuvent être dédiés à des applications spécifiques, d'autres sont adaptatifs. Nous remarquons que la plupart des travaux ont tendance à adapter les réseaux maillés grâce à leur facilité d'implémentation.

La mémorisation en entrée est la plus utilisée dans les différentes propositions. Ceci s'explique aussi par sa simplicité lors de la phase de conception architecturale. De plus, le coût en matériel n'est pas élevé et la latence offerte est faible. Nous remarquons également que le transfert des données est toujours fait par la technique« wormhole », qui est simple à manipuler, à programmer et qui présente des performances acceptables. Le tableau 4.1 présente une récapitulation permettant de faire une comparaison entre les différents réseaux proposés dans la littérature.

Tableau 4.1 : Etat de l'art sur les réseaux sur puce proposés.

NoC	Topologie	Routage	Lien	Transfert de données
SPIN	Arbre élargi	Adaptatif	36 bits	Wormhole
Octagon	Anneau avec des cordes	Source	Variable	--
Hermes	Maillé	XY	10 bits	Wormhole
Arteris	Maillé	XY	--	Wormhole
Mango	Maillé	Source	34 bits	--

Dans cette première partie, nous avons présenté une brève description des travaux proposés et l'importance de développer de nouvelles architectures des réseaux sur puce. Les interconnexions à medium partagé ont montré leurs limites, puisqu'elles ne sont capables d'intégrer qu'un nombre réduit des IPs. Elles ne supportent pas la synchronisation GALS ni le parallélisme de communication. La solution à ces problèmes apparaît avec la conception et le développement des NoCs. Les NoCs sont extrêmement paramétrables, extensibles, flexibles, et ils offrent une qualité de service pour les communications de l'application. Plusieurs paramètres sont à prendre en compte lors de la conception

de la topologie et de la brique de base qui est le routeur. On peut en citer la topologie, le routage, la latence le surface, la consommation, etc.

Dans la section suivante, nous passons à la présentation de la topologie « DMesh » et à la conception des routeurs qui forment le réseau.

5. Architecture proposée

La topologie de réseau sur puce proposée dans cette thèse est inspirée de [68] [81]. Elle est nommée « Diagonal Mesh » (DMesh). Cette topologie peut être construite en ajoutant des liens en diagonale pour les routeurs d'un réseau Mesh ordinaire. Cette topologie est illustrée dans la figure 4.3.

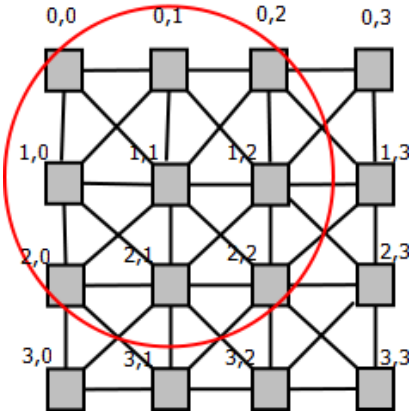

Figure 4. 3 : La topologie DMesh

Notre objectif est de proposer un réseau de communication, piloté par les contraintes de notre application, qui offre de bonnes performances en termes de diamètre, de flexibilité et de latence tout en simplifiant l'architecture du routeur. L'inconvénient majeur de cette topologie est le nombre de liens grandissant lorsqu'on augmente le nombre de routeurs. Notre idée consiste à extraire le motif cerclé en rouge, à éliminer les liens qui relient les nœuds (1,2) et (0,1), (1,0) et (0,1) etc. et à concevoir un routeur central permettant l'acheminement des données passant par un nombre réduit de liens et de routeurs.

5.1. Topologie de réseau

La topologie proposée, inspirée de la topologie DMesh, est un réseau d'interconnexion pour des applications orientées flot de données. Ce réseau, illustré dans la figure 4.4, est composé de deux types de routeurs : un routeur périphérique et un routeur central. Les liens existant dans cette topologie fournissent de meilleures performances telles que le diamètre qui est égal à 2, ce qui a un impact positif sur la latence.

Chaque routeur, sur le périmètre, est relié directement à son voisin, soit avec, soit contre le sens de l'aiguille d'une montre, ainsi qu'à un IP par un port local. Le routeur central est connecté directement à tous les routeurs du périmètre. Ce dernier peut représenter un pont entre les routeurs afin de favoriser un acheminement en traversant un nombre minimal de liens.

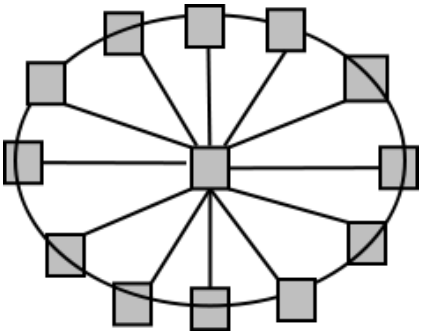

Figure 4. 4: Topologie proposée

Cette topologie peut être nommée « Ring Star ». Elle est constituée par douze routeurs sur le périmètre et un routeur central. Vu la flexibilité de l'architecture du routeur conçu, la topologie peut être extensible et le nombre des routeurs peut augmenter.

Ce réseau est construit à l'aide de deux types de routeurs. Dans le cas général, il y a N routeurs à 4 ports d'entrée/sortie numérotés de 0 à 3 (positionnés sur le périmètre) et un routeur central à N+1 ports d'entrée/sortie numérotés de 0 à N. Chaque routeur possède une adresse unique dans le réseau. Ainsi, les routeurs de l'anneau sont numérotés de 0 à N-1 tandis que le routeur central porte l'adresse N. Notre réseau est construit par 13 routeurs, 12 routeurs sur l'anneau et un routeur central. Dans cette topologie, chaque routeur est relié à ses deux voisins, dans le sens, ou contre le sens de l'aiguille d'une montre, et au routeur central.

Le routeur est l'élément de base de l'architecture du réseau. Son rôle principal est d'acheminer les données d'un IP lié au routeur, via un port bidirectionnel qui est le port « Local », à un autre en connectant des ports d'entrée à des ports de sortie. La stratégie de stockage utilisée dans notre cas est la mémorisation en entrée (déjà définie dans le chapitre 2) ; chaque port d'entrée possède un FIFO. Le contrôle de flux adopté dans ce réseau est le « crédit d'émission ». La technique de commutation utilisée est la commutation de paquets de type « wormhole » car il demande moins d'espace mémoire et il peut atteindre une latence plus faible. Dans le cas de la commutation de paquets, les paquets sont découpés en éléments de taille paramétrable, appelés « flits » (une unité de contrôle de flux dans le réseau). Un « flit » est un élément unitaire contenant des données et aussi des informations de contrôles qui peuvent être transmis sur le réseau. On distingue trois types de « flit » ; Le flit d'entête nommé BOP (Begin Of Paquet), le flit de données et le flit de fin de paquet dit EOP (End Of Paquet). L'architecture du routeur dépend de son emplacement dans le réseau.

Le routeur de l'anneau diffère du routeur central dans le nombre de ports ainsi que dans l'algorithme de routage mais ils ont le même fonctionnement concernant l'IC, la FIFO, l'OC et l'allocateur du port. Le routeur central possède 12 ports reliés chacun à un routeur de l'anneau et un port Local. Les flits passent d'un routeur à un autre dès qu'il y a de la place dans les FIFOs, et pas nécessairement de la place pour un paquet complet. Généralement, un paquet comprend un flit d'en-tête (header Flit), qui

peut être suivi par un ou plusieurs flits de données. Ce flit réserve le chemin de routage, en effet tous les autres qui suivent vont emprunter le même chemin que lui.

Les routeurs sont conçus et développés en logique synchrone. L'architecture du routeur se compose de six unités qui sont le bloc d'entrée, qui est composé lui-même par un contrôleur d'entrée, la FIFO et le contrôleur de sortie, la fonction de routage, l'allocateur du port de sortie et le commutateur. Le contrôleur d'entrée reçoit les flits d'un paquet afin de les stocker dans la FIFO, alors que le contrôleur de sortie permet d'envoyer les données au port de sortie adéquat. Le contrôleur d'entrée assure la génération des crédits qui indiquent à l'émetteur qu'il y a des places libres dans la FIFO. Chaque routeur peut envoyer les flits de données tant que le crédit du prochain routeur est au niveau haut.

Chaque port d'entrée du routeur possède une fonction de routage lorsque le paquet arrive au port d'entrée du routeur, le port de sortie est calculé et demandé par la fonction de routage en fonction de l'adresse de destination. L'algorithme de routage détermine le chemin emprunté par un paquet entre la source et la destination. Dans une architecture de communication, le routage joue un rôle important : le meilleur algorithme de routage donne les meilleures performances. Il est nécessaire de faire des compromis entre une utilisation optimale des liens de communication et un algorithme simple qui peut être facile à implémenter en RTL.

5.2. Architecture du routeur de l'anneau

Le routeur de l'anneau est composé d'un port bidirectionnel « Local » numéroté zéro, qui relie le routeur avec (IP), et de trois ports bidirectionnels vers les routeurs voisins numérotés de 1 à 3 qui sont :

- Le premier port est relié au routeur voisin dans le sens des aiguilles d'une montre (nommé CW).
- Le deuxième port est relié au routeur central (nommé C).
- Le troisième port est relié au routeur voisin dans le sens contraire des aiguilles d'une montre (nommé CCW).

Dans cette section, nous décrivons en détails les différents composants du routeur de l'anneau. La figure 4.5 illustre l'architecture interne du routeur.

Afin de garantir un acheminement de données fiable et une absence de panne, nous avons intégré un module de contrôle de flux au sein du routeur. Ce mécanisme utilise des signaux de contrôle indiquant à un routeur récepteur que la source possède suffisamment d'espace mémoire pour enregistrer les données. Ce contrôle est effectué pendant l'envoi des flits, il se base sur le principe de crédit d'émission. En effet, pour un récepteur libre, l'émetteur envoie une nouvelle donnée à chaque cycle d'horloge, et en indiquant sa disponibilité par un signal nommé **Crédit**. Si ce signal est actif, le routeur émetteur peut envoyer un nouveau flit. Sinon, le routeur émetteur doit attendre la transition de ce signal.

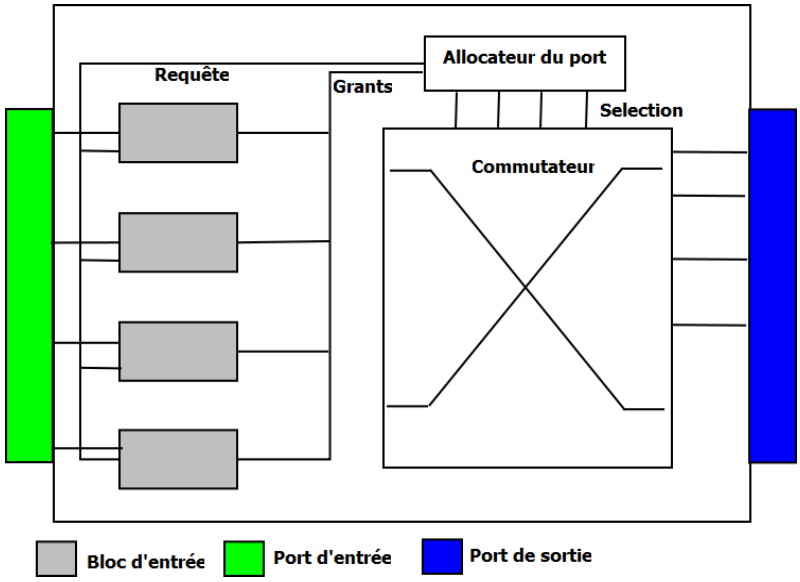

Figure 4. 5: Architecture interne du routeur de l'anneau

5.2.1. Bloc d'entrée

Ce bloc est constitué de deux sous-blocs et d'une mémoire FIFO. Son rôle principal est de gérer le flux d'entrée de données, via le contrôleur d'entrée, et d'assurer la bonne émission de paquets à travers le port adéquat. Nous détaillons dans cette partie le fonctionnement de ces deux modules. La figure 4.6 illustre un extrait de la figure de synthèse, après avoir décrit le routeur complet en VHDL, de ce bloc.

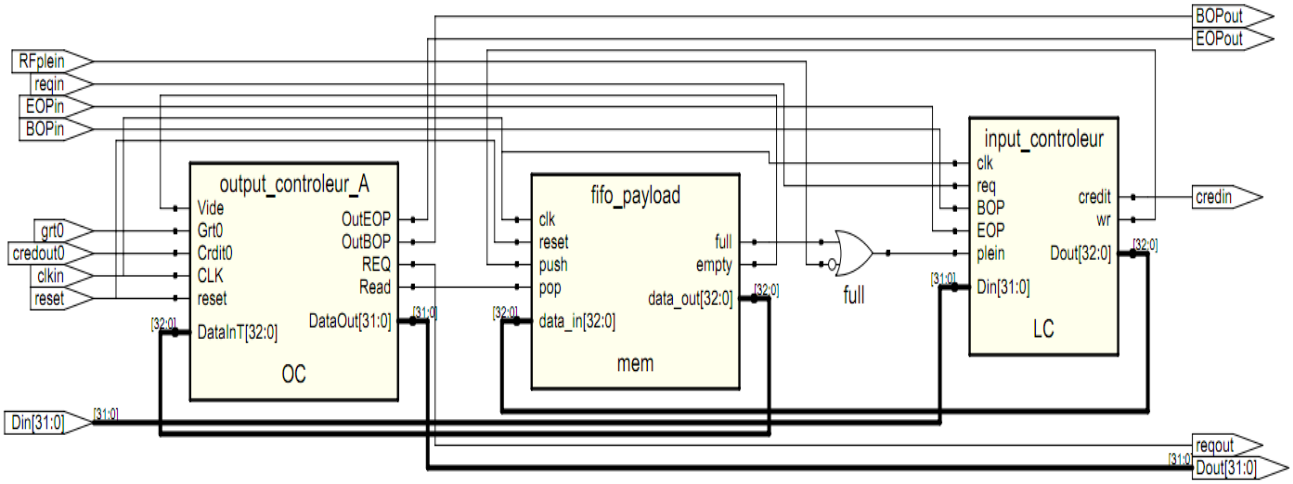

Figure 4. 6: Synthèse du bloc d'entrée

Les signaux d'entrées sont un bus de 32 bits transportant les données utiles, les signaux BOPin, EOPin, Reqin et le Grant venant de l'allocateur de port. Ce bloc est cadencé avec la même horloge que celle du routeur. Les signaux de sortie sont un bus de 32 bits, les signaux BOPout, EOPout, et Reqout. Il reçoit un signal « Credit » qui l'informe sur l'état du bloc d'entrée du routeur prochain, qui peut être un demandeur ou un pont, et génère également un signal « Credit » informant les autres routeurs sur son état.

5.2.2. Le contrôleur d'entrée (IC)

Il représente l'interface de communication avec le routeur émetteur, en reliant la sortie de l'émetteur avec le routeur courant qui est le récepteur. Il veille à la bonne réception des filts, en faisant un contrôle sur l'ordonnancement des flits afin de construire le paquet. Le contrôleur d'entrée (IC) est activé quand le signal BOP est mis à l'état haut, ce signal indique le début de l'émission d'un paquet. Son rôle se résume dans les trois tâches principales suivantes :

- Recevoir les paquets envoyés par le contrôleur de sortie,
- Ecrire les paquets reçus dans la FIFO,
- Assurer le contrôle de flux avec le routeur voisin.

En se référant à la figure 4.6, le bus de donnée d'entrée est codé sur 32 bits, tandis que l'information stockée temporairement dans la FIFO est codée sur 33 bits. En effet, le contrôleur d'entrée concatène le bit EOP avec le 32 bits de données et il les stocke dans la FIFO. Il est important de détecter le flit de fin de paquet, pour cela nous devrons conserver le signal EOP. Sans cette technique, on ne pourrait pas distinguer entre les données des différents paquets. Le contrôleur d'entrée utilise des signaux de contrôle (req, bop, eop) et génère un signal crédit. Si la FIFO n'est pas pleine, le contrôleur d'entrée génère un signal crédit égal à 1 sinon il le met à 0. Dans ce cas, le contrôleur de sortie du routeur précédent conserve les données à envoyer jusqu'à la réception d'un signal de la part de la FIFO, du routeur récepteur, indiquant qu'elle dispose de l'espace libre.

5.2.3. Le contrôleur de sortie

Les signaux de ce bloc sont montrés sur la figure 4.6 le contrôleur de sortie est comme un pont qui relie le port de sortie d'un routeur source au port d'entrée d'un routeur récepteur. Ses rôles sont de lire les données stockées dans la FIFO et de les envoyer après avoir examiné le signal crédit indiquant la disponibilité du routeur de destination. Quand le port de sortie demandé par la fonction de routage est alloué au contrôleur de sortie, ce dernier est actif et l'émission des filts commence. Une fois l'émission de données terminée, le contrôleur de sortie est en état de repos.

Il a comme ports d'entrée des signaux de contrôle qui sont le « Credit » et le « Grant ». Le signal « Credit » vient des routeurs qui veulent communiquer, tandis que le signal « Grant » est généré par l'allocateur du port du même routeur. Dès qu'un signal « Grant » et un signal « Credit » sont actifs, la transmission d'un paquet d'un routeur émetteur vers un routeur récepteur peut débuter.

5.2.4. La fonction de routage

Un algorithme de routage est nécessaire pour la détermination du port de sortie convenable. Dans notre cas, le routage est déterministe et les ports ne communiquent pas tous entre eux. En effet, le port local peut communiquer avec tous les ports, les deux ports CW et CCW communiquent seulement entre eux ou avec le port local. Le routeur central possède des liaisons directes avec tous les routeurs de l'anneau via le port C. Ce dernier ne communique qu'avec le port Local du routeur. Cette fonction est indispensable dans le routeur puisqu'il a un rôle essentiel dans la détermination du port de sortie. Elle extrait l'adresse de destination du paquet, effectue le calcul du chemin à suivre par le paquet et génère

donc un signal demandant le port choisi nommé req-p (p indique le numéro de port de sortie) comme le montre la figure 4.7. Ce signal assure la sélection d'un port de sortie convenable. Le signal BOP active le processus de routage, tandis que le signal EOP envoyé par l'OC remet la fonction de routage à l'état de repos. Comme le signal EOP provient du contrôleur de sortie, cela garantit l'envoi de tous les flits du paquet.

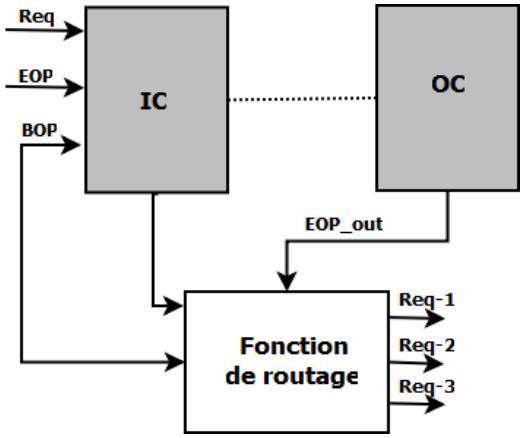

Figure 4. 7: Fonction de routage

Dans ce cas, l'élément d'arbitrage peut allouer le port de sortie à un autre port d'entrée. Donc, on est sûr que les données vont être envoyées sans risque de perte. Les fonctions de routage envoient des signaux servant comme requêtes pour le commutateur du port du routeur contenant les arbitres chargés d'accorder le port de sortie demandé au port d'entrée convenable. Les signaux de sortie ne sont pas les mêmes pour toutes les fonctions de routage parce qu'ils dépendent du port où elles sont placées. Par exemple, la fonction de routage du port local communique avec tous les ports, donc elle peut envoyer des requêtes à tous les autres ports.

Pour mieux expliquer le mécanisme du routage, nous prenons des extraits de notre code VHDL de ce module. L'exemple retenu correspond au cas où la fonction de routage du port local d'un routeur de l'anneau veut envoyer un paquet.

L'algorithme de routage proposé dans notre conception est déterministe avec un chemin minimal. Chaque routeur admet une adresse courante, et si l'adresse de la destination correspond à l'adresse du routeur central, la fonction de routage demande la réservation du port de sortie « across »correspondant au routeur central. Si la différence entre l'adresse source et l'adresse destination est égale à 1 ou 2, alors le port de sortie « ClockWise » est demandé. En se basant sur le même principe, et comme c'est indiqué sur le code, la fonction de routage peut demander un port de sortie «Counter Clockwise » ou encore le port de l'« across ».

```
    ad <= conv_integer(ADdestination);
    as <= conv_integer(ADRC);
-------------- go across -------------
    if ( ad = 12) then
        req1_CW <= '0';
        req1_A <= '1';
        req1_CCW <= '0';
------------ go clockwise ------------
    elsif(((ad - as ) mod 16)= 1 or ((ad - as ) mod 16)= 2 )then
        req1_CW <= '1';
        req1_A <= '0';
        req1_CCW <= '0';
-------- go counter clockwise --------
    elsif((ad - as)= -2 or (ad - as) = -1 )then
        req1_CW <= '0';
        req1_A <= '0';
        req1_CCW <= '1';
-------------- go across -------------
    else
            req1_CW <= '0';
       req1_A <= '1';
       req1_CCW <= '0';
    end if;
  else
     next_state <= idle;

       req1_CW <= '0';
       req1_A <= '0';
       req1_CCW <= '0';
  end if;
```

Dans le cas de la fonction de routage placée dans le port d'entrée « Clockwise » d'un routeur de l'anneau, elle peut demander uniquement le port de sortie « Counter Clockwise » ou le port de sortie local. Cela dépend de l'adresse de destination et de l'adresse du routeur courant. Le code VHDL ci dessous explique mieux le principe :

```
-------------- go Locale -------------
    if ( ad = as) then
        req1_L<= '1';
       req1_CCW <= '0';
-------- go counter clockwise --------
    elsif((ad - as )= -2 or (ad - as )= -1 )then

        req1_L<= '0';
        req1_CCW <= '1';
    end if;
  else
     next_state <= idle;
     req1_L<= '0';
     req1_CCW <= '0';
  end if;
```

5.2.5. L'allocateur de port

Pour gérer la communication et les accès simultanés, on a besoin d'un mécanisme d'arbitrage. Chaque port possède son propre arbitre. L'ensemble des arbitres se regroupent dans un composant commun nommé allocateur de port ou « Switch Allocator » en anglais.

Ce composant gère toutes les demandes d'accès au port de sortie envoyées par les fonctions de routage. Le bloc allocateur du port de sortie est un composant crucial qui contient 4 arbitres qui décident lequel des ports d'entrées peuvent gagner l'accès au port de sortie. La figure 4.8 illustre la structure de l'architecture interne d'un allocateur de port. Le principe est expliqué de la manière suivante :

- L'arbitre du port de sortie local peut recevoir des demandes à partir des fonctions de routage des ports 1, 2 et 3(indiqué par r1, r2, r3 ou reqCW_L, reqA_L, reqCCW_L sur la figure).
- L'arbitre du port ACW reçoit les requêtes à partir des fonctions de routage des ports (Local) ou de ports empruntant un chemin contre l'aiguille d'une montre (ACW).
- L'arbitre du port (AA, pour dire arbitre du port permettant d'établir un chemin avec le routeur central) reçoit une seule requête à partir de la fonction de routage du port (local).

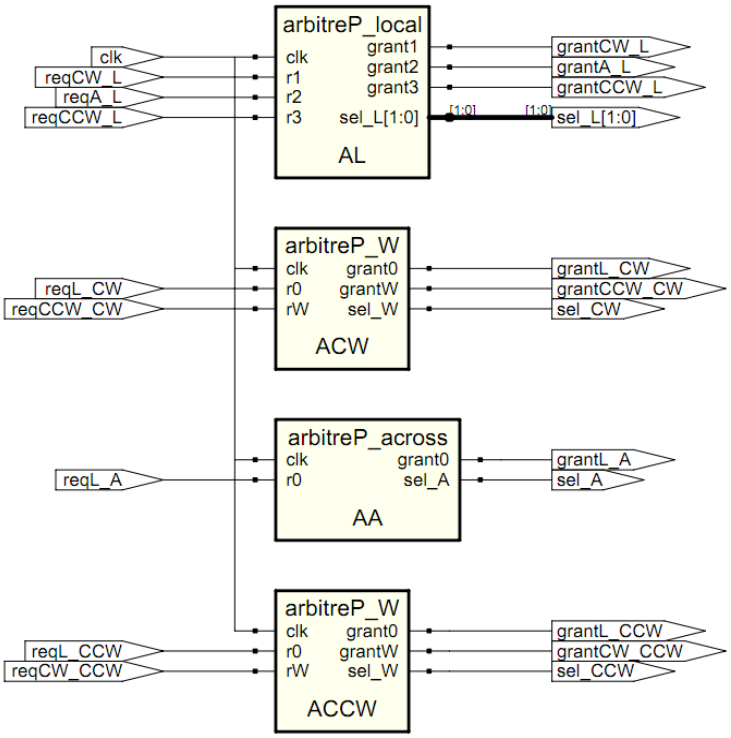

Figure 4. 8: Structure de l'architecture de l'allocateur de port

L'arbitre du port(ACCW) reçoit deux requêtes à partir des fonctions de routage des ports (local) et du port (CW).

Afin de bien montrer le mécanisme d'allocation des ports de sortie, nous expliquons dans le paragraphe qui suit la technique de l'arbitrage.

5.2.6. Arbitrage

Un arbitre, comme l'indique son nom, permet d'arbitrer ou de piloter différents signaux afin de contrôler le passage par un port de sortie. Dans notre construction, l'arbitre reçoit les différentes requêtes et les envoie en se basant sur l'algorithme Round Robin. Le bloc d'arbitrage, comme c'est illustré sur la figure4.9, possède un nombre bien défini d'entrées concurrentes envoyées par les modules de routage des ports d'entrées. Cependant, une seule requête gagne le passage au port de sortie convenable. Par la suite un des signaux « Grant » sera affecté à l'unité de gestion de port convenable en sélectionnant le commutateur. Ce commutateur, qui est un composant important dans l'architecture du routeur, indique le chemin à emprunter par le paquet. Le nombre des signaux pour chaque arbitre dépend de plusieurs paramètres tels que la position sur le réseau et la nature de l'algorithme de routage.

L'algorithme d'arbitrage adopté dans notre travail est l'algorithme de priorité tournante ou en anglais « Round Robin » [82]. Le principe est d'attribuer au routeur qui est en possession du port de sortie le niveau de priorité "k", la plus haute priorité suivante sera celle qui a le niveau "k-1" et ainsi de suite. Prenons l'exemple suivant : On considère deux demandeurs D0 et D1 qui peuvent utiliser le même port de sortie pour envoyer ou traverser le réseau. L'objectif est de concevoir un allocateur de port de sortie équitable. Chacun des ports d'entrée dispose d'un signal de requête r_i, permettant la demande d'un port de sortie, (r_0, r_1) sont indépendants et peuvent être actifs simultanément. Dans ce cas, l'allocateur alloue le port de sortie à l'un des demandeurs par l'utilisation d'un signal « Grant ».

Ă la fin de l'acheminement du paquet, l'allocateur de port désactive le signal r_i de celui alloué. Le port de sortie libre ne peut pas être immédiatement réutilisé, et l'allocateur attend au moins un cycle avant de l'allouer de nouveau. L'algorithme d'arbitrage « round-robin » assure un arbitrage équitable, le demandeur qui obtient le port de sortie devient le moins prioritaire pour la prochaine allocation. Le fonctionnement de l'arbitre est modélisé à l'aide d'une machine d'états.

5.2.7. Le commutateur

C'est un commutateur physique reliant l'entrée à la sortie, ayant N entrées et N sorties. Il est composé de multiplexeurs. Il est nommé aussi La barre croisée, largement utilisé grâce à sa nature non bloquante et à sa simplicité.

Il a pour rôle de commuter les liens provenant des OCs vers les ports de destinations choisis, en se basant sur les arbitres de l'allocateur de port. La figure 4.9 montre le schéma en bloc de notre commutateur, il est constitué de 4 multiplexeurs. Les bits de sélection proviennent de l'allocateur de port.

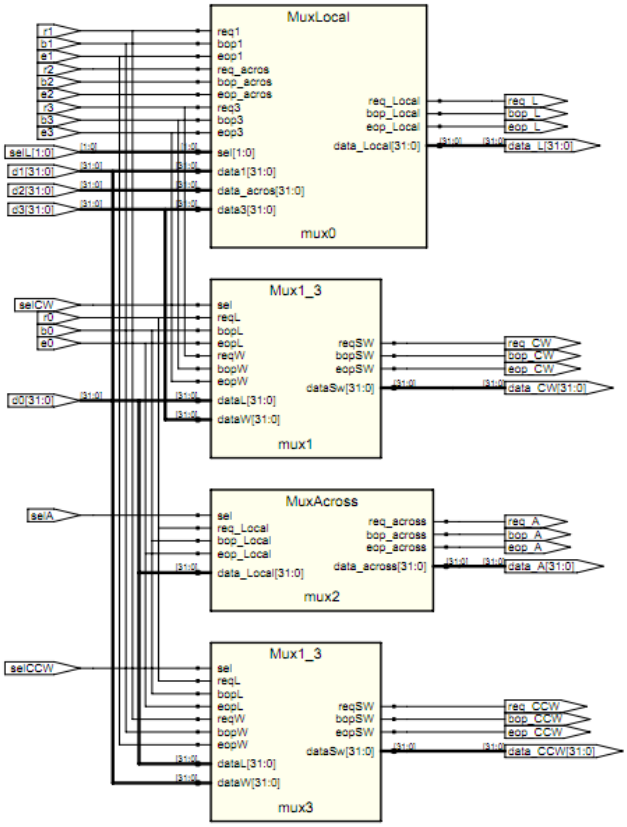

Figure 4. 9: Architecture de commutateur.

Dans notre architecture, on a ajouté un composant nommé commutateur de crédit. Il connecte le signal crédit de sortie du routeur récepteur aux ports d'entrée du routeur émetteur. Il est conçu à l'aide d'un démultiplexeur. Ă la demande d'un port de sortie, le démultiplexeur relie le signal d'entrée credit_IN de l'OC récepteur avec le signal de sortie credit_Out de l'IC du routeur adéquat.

5.2.8. Simulation du routeur

Le routeur a été développé en VHDL. Pour valider le fonctionnement du routeur, nous faisons la simulation suivant un scénario bien défini. Le chronogramme de la figure 4.10 décrit l'acheminement de quatre paquets successifs provenant du port local du routeur d'adresse égale à 2 vers quatre routeurs d'adresse différente. Dans notre cas, le routeur d'adresse 2 reçoit sur son port local le premier paquet de taille neuf flits. Ce paquet est destiné au routeur d'adresse 0, la différence entre les deux adresses est égale à -2, donc il est routé vers le port de sortie « Counter Clockwise ». Le deuxième paquet va être acheminé vers le routeur central à travers le port de sortie adéquat (across). Le troisième paquet est acheminé vers le routeur d'adresse égale à 4, la différence entre les deux adresses est égale à 2. Dans ce cas, le paquet est routé vers le port de sortie « Clock Wise ». Le même principe est appliqué pour le dernier paquet.

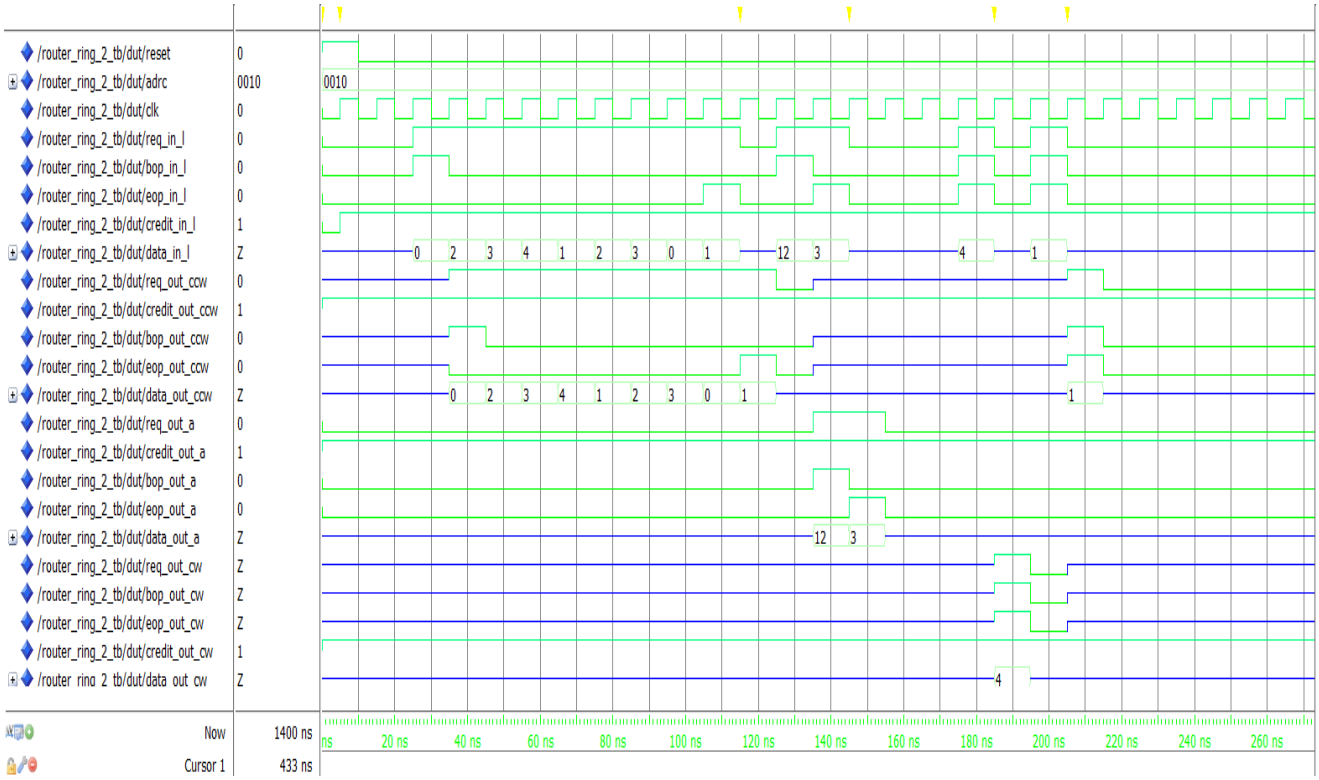

Figure 4. 10: Simulation du fonctionnement du routeur

5.3. Routeur central

Le routeur central est conçu pour minimiser le trafic et la charge imposée sur les routeurs de l'anneau. Son rôle est d'acheminer les paquets à leur destination, dans le cas où le nombre de sauts nécessaires pour qu'un flit atteigne sa destination est supérieur à deux sauts. L'architecture en bloc de ce routeur est la même que celle du routeur de l'anneau sauf qu'il y a des points de différence dans le fonctionnement du module de routage. L'allocateur de port et le nombre de ports est supérieur puisque chaque routeur de l'anneau est relié directement au routeur central. Le routeur central contient 12 ports le reliant aux routeurs de l'anneau et un port local qui le relie à son IP. Pour router un paquet reçu sur un port d'entrée i, l'unité de routage implantée dans ce port décode l'entête de ce paquet et extrait l'adresse du routeur destination.

5.3.1. Routage

La fonction de routage dans le routeur central de son port local peut demander tous les ports de sortie permettant d'envoyer un message aux routeurs de l'anneau. Cependant, les autres fonctions de routage des ports d'entrée de ce routeur peuvent envoyer des paquets à huit ports de sortie de ce même routeur. Le principe de la fonction de routage de ce routeur est expliqué de la manière suivante :

Soit p le numéro du port d'entrée du routeur central, il peut envoyer des paquets vers les ports de sortie allant de 0 à p-3, du p+3 à n-1ou au port local. N ici indique la valence du routeur central. Ce principe s'explique par le fait que notre fonction de routage est implémentée pour assurer un parcours minimal.

Le code VHDL ci dessous montre l'entité d'une fonction de routage du port d'entrée d'indice 4. Elle ne peut pas envoyer une requête vers les ports de sortie 2, 3, 4,5 et 6.

```
entity RF4_centrale is

    Port ( clk: in STD_LOGIC;
           ADdestination: in std_logic_vector(3 downto 0);
           bop: in STD_LOGIC;
           eop: in STD_LOGIC;
           requestL : out  STD_LOGIC;
           request0 : out  STD_LOGIC;
           request1 : out  STD_LOGIC;
           request7 : out  STD_LOGIC;
           request8 : out  STD_LOGIC;
           request9 : out  STD_LOGIC;
           request10 : out  STD_LOGIC;
           request11 : out  STD_LOGIC;
           pleine : out  STD_LOGIC ;
           ADRC : in std_logic_vector(3 downto 0));
end RF4_centrale;
```

5.3.2. L'allocateur de port

Dans le cas du routeur central, l'allocateur de port est constitué par treize arbitres, comme le montre la figure 4.11. Dans cette architecture, il y a douze arbitres qui peuvent recevoir huit requêtes et un arbitre du port local qui reçoit douze requêtes. Cela est dû aux fonctions de routages adoptées.

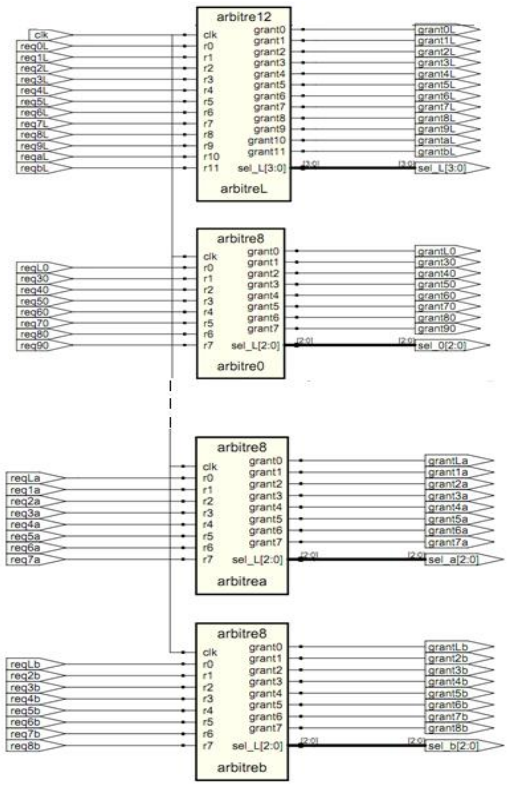

Figure 4. 11: Architecture de l'allocateur de port du routeur central

5.3.3. Simulation du routeur central

Le routeur central a été aussi développé en VHDL. Son fonctionnement peut être décrit par le scénario de simulation de la figure 4.12. Elle décrit l'acheminement de quatre paquets successifs provenant du port local du routeur central d'adresse égale à 12 vers quatre routeurs d'adresses différentes. Dans notre cas, le routeur central reçoit sur son port local le premier paquet de taille 10 flits. Ce paquet est destiné au routeur d'adresse 8, donc il est routé vers le port de sortie d'indice 8. Le deuxième paquet est destiné au routeur d'adresse 9, la figure de simulation nous montre le bon acheminement de ce paquet.

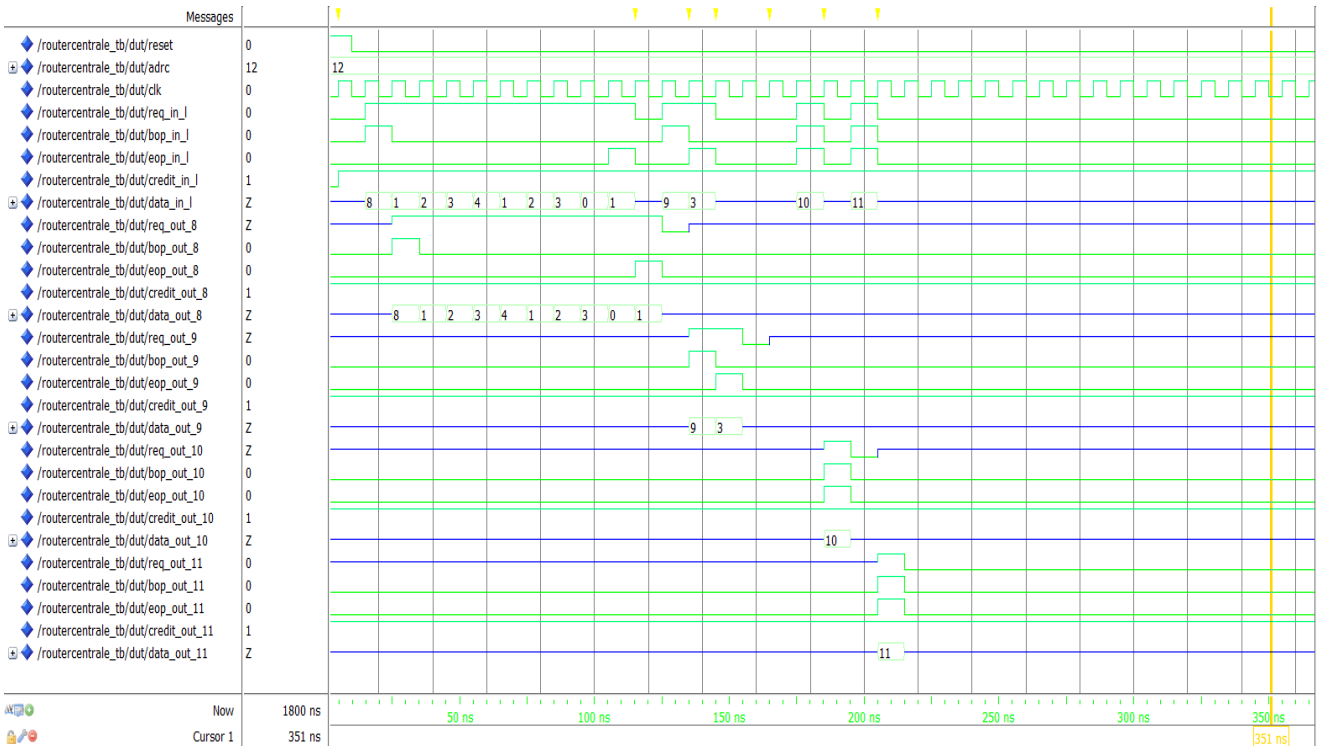

Figure 4. 12 : Simulation du routeur central

6. Evaluation de performances du routeur de l'anneau et étude comparative

Après la modélisation, une phase d'évaluation de performances s'avère importante. Dans les réseaux sur puce, l'évaluation de performances peut prendre en considération plusieurs critères. Nous pouvons les classer en deux parties : les critères de qualité de service et les métriques de conception. Nous nous intéressons dans notre évaluation à la deuxième classe. Nous évaluons le routeur en termes de fréquence, de surface et de latence. La latence globale du réseau fait intervenir dans sa formulation mathématique la valeur du diamètre. L'équation 4.1 [83] définit ce paramètre :

$$L = (\sum_{1}^{n}(Ri)) + P \times Clockcycle\,s \qquad \textbf{Eq4.1}$$

Où n est le nombre de liens à traverser par un paquet, Ri est le temps pour router un paquet et P la taille du paquet. La latence du routeur de l'anneau est égale à 2.

Un autre métrique d'évaluation est le « Peak Performance » il est défini par l'équation 4.2 :

$$P = (F\max \div T) \times Flitsize \qquad \textbf{Eq4.2}$$

Où F_{max} est la fréquence maximale du routeur, T est le temps nécessaire pour envoyer un flit et « Flitsize » représente la taille de flit.

Grâce à l'utilisation d'un flit de contrôle à base de crédit, le T=1, et dans notre conception un flit est de taille 32bits.

Le routeur conçu est synthétisé à l'aide de l'outil ISE de Xilinx, le tableau 4.2 montre l'évaluation des performances en termes de fréquence, de surface, de consommation et « Peak Performance ».

Tableau4.2 : Métrique de conception

FPGA/Performances	**Virtex5xc5vlx50-3ff676.**
Slice	4%
Flip Flop	2%
Lut	2%
Fréquence(Mhz)	264
puissance (mW)	33mw (200Mhz)
Peak.Per	8.44Gbit/s

Une étude comparative a été menée [83] dont l'objectif est de positionner ce travail avec les travaux proposés dans la littérature. Plusieurs topologies sont proposées dans la littérature, dans chacune il y a un ensemble différent de compromis en termes de paramètres, tels que le degré de réseau, l'extensibilité du réseau pour les topologies maillées. Néanmoins, en raison de la complexité croissante des applications, cette topologie ne peut pas fournir de bonnes performances. D'autre part, une topologie simple comme l'anneau consomme moins en surface, mais elle montre ses limites dès que le nombre de routeurs devient important. Le diamètre et la distance moyenne représentent un facteur important en termes de performance et de mise en œuvre de la topologie d'un réseau sur puce. La conception proposée du routeur de l'anneau est conçue pour offrir un bon compromis entre le coût en termes de métrique de conception et la formulation théorique des performances.

Tableau4.3 : Etude comparative de l'architecture du routeur

Perf/travail	**[84]**	**[85]**	**Ce travail**
Taille de flit	36bits	32bits	32bits
Latence du routeur	3	6	2
Slice	431	1464	989
Fréquence	166	166	218
Topologie	Mesh	Mesh	DMesh

En raison de la pertinente connectivité, notre topologie maillée peut intégrer un grand nombre de routeurs sans changer le diamètre. En effet, elle peut fournir une latence pour des applications multimédia telles que l'encodeur vidéo (H.264). L'inconvénient majeur de notre proposition apparaît dans le nombre des liens requis et la scalabilité.

Dans [84] l'auteur présente un réseau 2D mailles, le routeur proposé dans ce travail est implémenté sur une carte virtex2 et virtex4. Dans ce travail, un routeur générique basé sur la commutation de paquets et le contrôle de flux Wormhole, a été proposé. Comme c'est indiqué dans le tableau4.3, la largeur de bus des données est d'environ 36 bits et la fréquence maximale est inférieure à la fréquence de notre conception. Dans [85] les auteurs présentent deux architectures de deux routeurs un est de 3 ports et l'autre de 4 ports. Le but de cette proposition était d'aider les concepteurs à choisir entre le multiplexage temporel et la commutation par paquets. La taille de bus de données utilisée est d'environ 32 bits. Le tableau comparatif présenté dans cette section montre que notre conception est meilleure en termes de fréquence maximale et de surface utilisée.

7. Conclusion

Les architectures de réseaux sur puce apparaissent comme une solution pertinente pour remédier aux problèmes liés aux besoins en communication d'un système sur puce complexe intégrant un nombre important d'IP. Choisir la forme du réseau de communication compte parmi les difficultés majeures de conception des systèmes sur puce. Nous proposons une description architecturale de deux routeurs pour construire la topologie DMesh qui est proposée dans la littérature. Ce réseau sur puce peut représenter l'infrastructure de communication de notre SoC. La conception matérielle des routeurs de l'anneau et du routeur central a été détaillée, validée et synthétisée. Ces routeurs vont être intégrés, dans la bibliothèque du GASPARD, et exploité durant la phase de déploiement.

La contribution dans cette partie est apportée au niveau architecture des routeurs, plus précisément dans les modules qui construisent le routeur. L'avantage de notre routeur réside en une latence qui est égale à deux. Aussi bien le routage distribué est assuré dans cette conception. Le routeur est conçu en logique synchrone mais peut être un bon support pour la synchronisation GALS du fait que son protocole de communication est à base de crédit.

Jusqu'à maintenant nous avons entamé tous ce qui concerne la modélisation de l'architecture, dans les deux niveaux d'abstraction ; niveau système et niveau RTL. Le chapitre qui suit s'intéresse essentiellement à la modélisation de l'application qui est le codeur H.264.

Chapitre 5
Modélisation et conception architecturale du codeur H.264 à faible coût de consommation : vers l'association sur un NoC

1. Introduction

Ce chapitre permet au lecteur d'avoir une idée sur l'application H.264 précisément sur les IPs qui forment ce codeur. Nous allons commencer en premier lieu, par l'extraction du parallélisme locale qui peut exister dans les tâches du codeur H.264 et la modélisation de ces tâches en utilisant le profil MARTE, en second lieu d'améliorer les performances au niveau RTL.

La modélisation en MARTE du codeur H.264 est faite pour la première fois dans alittérature. Les modèles proposés peuvent être utilisés indépendamment du profil du standard H.264, ce qui favorise leur réutilisation.

Le problème de la conception architecturale abordé dans cette partie de la thèse est la consommation élevée due à la quantité énorme d'informations à traiter et la complexité algorithmique de certains IP tels que l'estimateur de mouvement. Ce composant qui représente 70% de la complexité globale du codeur va être bien étudié dans ce chapitre. Cette étude est consolidée par des propositions au niveau architecture dont le but est de réduire la consommation par l'intégration d'un module de contrôle de consommation basé sur la synchronisation GALS. Egalement, la réduction de la surface a été faite par la réutilisation de ce qui est déjà calculé et la partage de ressources. Les différents IPs sont décrits en VHDL et synthétisés sur FPGA. Les IPs développés vont être utilisés comme brique de base pour la génération de code.

2. Présentation de la norme H.264

L'émergence de l'AVC « Advanced Vidéo Coding », issu du regroupement de l'OSI (L'Organisme de Standardisation International) et de l'UTI (L'Union Internationale des Télécommunications) a donné naissance au standard de compression vidéo nommé H.264 [86]. Il a été développé au sein du MPEG. Son but original est d'améliorer la compression en permettant un codage efficace et robuste. Il est employé dans plusieurs domaines tels que la vidéoconférence, le stockage HD (Haute Définition) et le codage pour la transmission sur les réseaux. Il intègre plusieurs techniques différentes de ces prédécesseurs qui lui permettent d'offrir une compression meilleure que les autres standards de codage vidéo. Le codeur H.264 définit trois profils : le profil de base, le profil étendu et le profil principal. Chacun de ces profils supporte un ensemble spécifique de fonctions de codage et un domaine d'application précis. Ce standard de codage vidéo a fait l'objet de plusieurs travaux de recherche vu son importance et la qualité de service qu'il offre. En effet, une complexité au niveau architectural apparaît dans les différents modules qui le forment, notamment le module d'estimation de mouvement.

Ce qui représente pour les concepteurs et les chercheurs un défi à surmonter via l'optimisation algorithmique, architecturale, et également par la modélisation à haut niveau.

2.1. Principe de fonctionnement

Le H.264 présenté est une méthode, et aussi un format de compression vidéo, qui prend moins de capacité quand il est stocké ou transmis sur un réseau. La compression vidéo est une technologie essentielle pour des applications citées auparavant. La standardisation de compression vidéo rend possible l'inter-opération des encodeurs et des décodeurs au sein du même standard. Un codeur convertit la vidéo dans un format compressé et un décodeur convertit la vidéo compressée de nouveau dans un format non compressé. La figure 5.1 illustre le schéma synoptique du codage/décodage suivant la norme H.264.

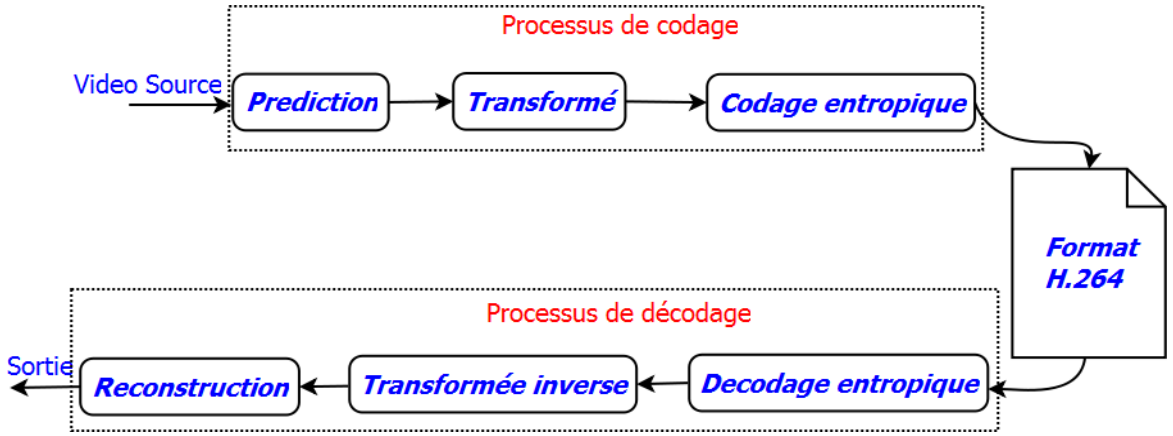

Figure 5. 1 : Le processus de codage et décodage suivant la norme H.264.

L'image vidéo source est d'abord divisée en blocs de taille égale. Dans le sens du codage, il passe par un module de prédiction dont l'objectif est de réduire la quantité d'informations à traiter. Ceci est fait soit par la prédiction temporelle, soit par la prédiction spatiale. Habituellement, la première image est codée en exploitant la redondance spatiale. La deuxième phase permet de passer d'une matrice de pixels vers la binarisation. Cette opération inclut la transformation en cosinus discrète (DCT en anglais) et la quantification. La phase finale dans le processus de compression vidéo est le codage entropique qui permet de générer le format ou le bitstream H.264.

Dans le sens du décodage, des opérations inverses doivent être effectuées. Le décodage entropique permet d'extraire l'information binaire qui va être envoyée à la quantification inverse et la DCT inverse afin de reconstruire la vidéo complète. La version décodée n'est pas en général identique à la séquence originale car le codeur H.264 est un format de compression avec pertes, c'est à dire qu'une certaine qualité d'image est perdue lors de la compression.

L'image dans n'importe quel codeur de compression est formée par trois informations de couleur. Un pixel d'une image contient trois sous-pixels : la chrominance rouge(C_b), la chrominance bleue (C_b) et la luminance (Y). En effet chaque MB de taille 16×16 est composé par 6 sous MBs, comme l'indique la figure 5.2, quatre pour la luminance, 1 pour la C_b et 1 pour la C_r. Généralement, le traitement est

effectué sur la luminance, et on entend par là le niveau de gris, vu que l'œil humain n'est pas sensible à l'information de couleur.

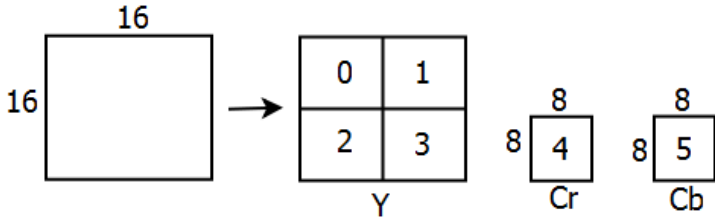

Figure 5. 2*: Partitionnement du couleur*

Les opérations intéressantes en termes de complexité algorithmique et architecturale existent dans la phase de codage, notamment la prédiction et la transformée. Nous allons détailler dans les sections qui viennent les différents modules du codeur H.264.

2.2. Prédiction

La figure 5.3 illustre la structure typique du sens d'encodage de la norme H.264 [87]. Les données sont traitées sous forme d'unités nommées *Macro-Bloc* (MB) correspondant à 16 × 16 pixels. Dans le processus de codage, un MB prédit est généré à partir du module de prédiction. Il est soustrait du MB courant afin de former un MB résiduel qui est ensuite transformé, quantifié et codé. En parallèle, les données quantifiées sont passées vers la transformée inverse, et ensuite ajoutées au MB prédit pour reconstruire une version codée de l'image. Elle est stockée dans la mémoire pour pouvoir l'utiliser comme image de référence dans les futures prédictions. Dans le processus d'encodage, on distingue entre deux types de prédictions ; la prédiction inter, qu'on appelle aussi temporelle et la prédiction intra c'est-à-dire spatiale.

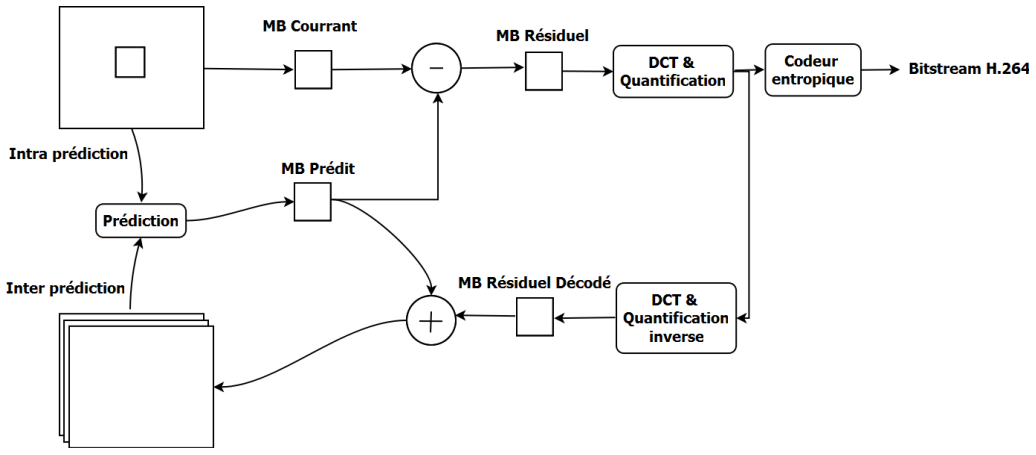

Figure 5. 3: le processus de codage suivant la norme H.264

Le codeur forme une prédiction du MB courant à partir de données préalablement codées, soit à partir de l'image courante en utilisant la prédiction intra, et dans ce cas on parle de l'image de type I, soit à partir d'autres images qui ont déjà été codées à l'aide de prédiction inter. Les méthodes de prédiction

supportées par le codeur H.264 sont plus souples que celles dans les précédentes normes, permettant des prédictions précises et donc une compression vidéo efficace.

2.2.1. L'intra prédiction

La norme H.264/AVC permet une large gamme d'options dans la prédiction intra en exploitant la redondance spatiale. Les MBs dans les images de type **I** sont codés sans se référer à des données en dehors de l'image actuelle. Ce type de codage se base sur le fait que les pixels dans la même image sont fortement corrélés, c'est-à-dire qu'il existe une corrélation relativement forte entre les échantillons dans le MBs et les échantillons qui sont immédiatement adjacents au MBs. La prédiction intra utilise donc des échantillons provenant de près, les MBs précédemment codés pour prédire les valeurs dans le MB courant.

Cette opération permet alors de coder la différence entre le MB réel et sa prédiction, ce qui favorise un codage avec un nombre réduit d'informations. Dans un MB codé intra, il y a trois choix de la taille du MB, à savoir 16 × 16, 8 × 8 ou 4 × 4. Par exemple, dans le mode 4 × 4 chaque MB peut être codé en utilisant un parmi neuf modes de prédictions [88]. Le tableau 5.1 explique les différents types de l'intra prédiction pour la luminance (niveau de gris) et la chrominance (les couleurs).

Tableau 5.1 : Types de l'intra prédiction

Mode de prédiction intra	Description
16 × 16 luma	Quatre modes de prédiction sont possibles
8 × 8 luma	Neuf modes de prédiction sont possibles
4 × 4 luma	Neuf modes de prédiction sont possibles
Chroma	Quatre modes de prédiction sont possibles

2.2.2. L'inter prédiction

Comme on l'a déjà dit dans la section 2.2, la technique de prédiction inter se base sur l'exploitation de la redondance temporelle. Cela signifie que dans le processus de codage, et vu que les images de la même séquence vidéo sont fortement corrélées, on peut ne pas coder toute l'image actuelle en exploitant ce qu'on a déjà codé. Cette technique est basée sur deux éléments critiques dans le codeur H.264 : l'estimation de mouvement et la compensation de mouvement. Dans ce type de prédiction, on parle des images de types **P.** Il s'agit d'un processus de prédiction d'un MB à partir des images qui ont été préalablement codées et transmises, on les appelle images de référence. L'inter prédiction contient 7 modes comme le montre la figure 5.4 : mode 4×4, 8×4, 4×8, 8×8, 16×8, 8×16 et 16×16. Ces modes décrivent la taille du MB actuel.

Le principe de l'exploitation de la redondance temporelle consiste tout d'abord à choisir l'image de référence parmi une liste d'images précédemment codées et de la stocker dans une mémoire. Le décalage d'adresse entre la position du MB actuel, dans une zone de recherche de l'image de référence, et le MB le plus ressemblant, est un vecteur de mouvement. Il s'agit de l'estimation de mouvement. L'estimateur de mouvement permet de générer un vecteur de mouvement qui décrit le déplacement du MB actuel, c'est-à-dire à coder, dans l'image qui a été déjà codée. Ce vecteur de mouvement est

ensuite envoyé à la compensation de mouvement pour générer le MB prédit à partir de l'image de référence.

En résumé, la prédiction est une phase importante dans le codage vidéo notamment dans la norme H.264. Elle consiste en quelque sorte à coder la différence entre l'image actuelle et une image de référence. La prédiction inter peut se faire à partir de deux images adjacentes. On parle donc de prédiction bidirectionnelle. Les images utilisées sont notées images de type B. L'importance de cette technique de prédiction réside dans le fait qu'une image prédite peut être efficacement codée, ce qui influe sur le taux de compression de la vidéo.

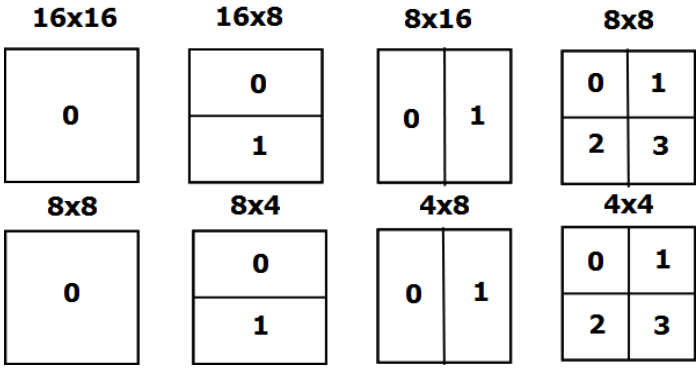

Figure 5. 4: Les modes de l'inter prédiction

2.2.3. L'estimation de mouvement

L'estimation de mouvement est une technique importante pour la compression vidéo vue la quantité énorme d'informations à traiter. Sur le codeur H.264, jusqu'à 70% de la complexité et de la charge de calcul est dédiée à cette opération [89]. En effet, le nombre d'opérations à effectuer dans l'estimation de mouvement, pour la compression du nouveau format vidéo tel que la compression haute définition, dépasse les capacités des nouveaux processeurs.

Le principe de l'estimation de mouvement est de capturer les mouvements des MB dans les images au cours du temps. Elle peut être faite de deux façons : « avant » ou « arrière » [90]. Dans le cas de l'estimation « avant », nous supposons que le mouvement est compris entre deux images n-1 et n. Il exprime un changement dans l'image n-1 par rapport à l'image n. Dans ce cas d'estimation, ça revient à répondre à une question clé : où se trouve un MB de l'image courante ? Pour décrire le déplacement, nous attribuons un vecteur de mouvement à chaque MB de l'image n-1, tandis que l'estimation « arrière » procède d'une façon inverse. Cela signifie que, pour chaque MB de l'image n, on associe un vecteur de mouvement désignant son emplacement dans l'image précédente. Il s'agit dans ce cas de répondre tout simplement à la question : d'où vient chaque MB ?

Dans le codeur H.264, l'estimation de mouvement a un impact significatif sur les qualités de compression en termes de gain et de taux. Les hypothèses clés dans cette technique sont les suivantes :

- La luminosité est uniforme et ses variations sont négligeables.
- Les problèmes d'occlusion sont négligés.

Suite à l'importance de l'estimation de mouvement dans le codage vidéo, plusieurs méthodes ou points de vue ont été proposés. Ces méthodes sont regroupées en trois classes :

- Les méthodes basées sur l'équation des flux optiques : elles s'appuient sur l'invariance de la luminosité de l'objet en mouvement. A l'aide de la DFD (Displaced Frame Difference) ou de la différence inter-image, ces méthodes peuvent décrire le mouvement d'un MB de l'image n dans l'image n-1 [91].
- Les méthodes fréquentielles : elles caractérisent le mouvement par un déphasage obtenu à l'aide des transformations. Ces transformations prennent en compte les propriétés fréquentielles du mouvement, à savoir qu'une translation dans l'image correspond à un déphasage dans le domaine fréquentiel [92].
- Les méthodes par correspondance de blocs (Block Matching Algorithm, BMA) : elles sont basées sur la corrélation entre deux images.

Parmi ces trois classes, les BMAs sont souvent utilisés par les normes de compression vidéo, vu leur efficacité de codage. Dans cette thèse, nous ne nous sommes intéressés qu'à la dernière méthode.

La correspondance de bloc

La méthode de correspondance de blocs est la plus utilisée en estimation de mouvement. Elle est basée sur la corrélation entre l'image courante et l'image de référence. Habituellement, elle est utilisée en estimation « arrière ». Le principe de cette méthode est de comparer un MB de l'image courante à une région de l'image de référence appelée zone de recherche. Les valeurs des pixels sont exploitées afin de trouver la correspondance, et la position du MB de l'image courante dans l'image de référence donne directement le mouvement. Chaque image est segmentée dans un premier temps en MB de taille N×N. Ensuite, pour chaque MB de l'image courante, le MB réduisant un critère d'erreur est recherché dans la zone de recherche de l'image de référence. La figure 5.5 illustre le principe de la correspondance de blocs.

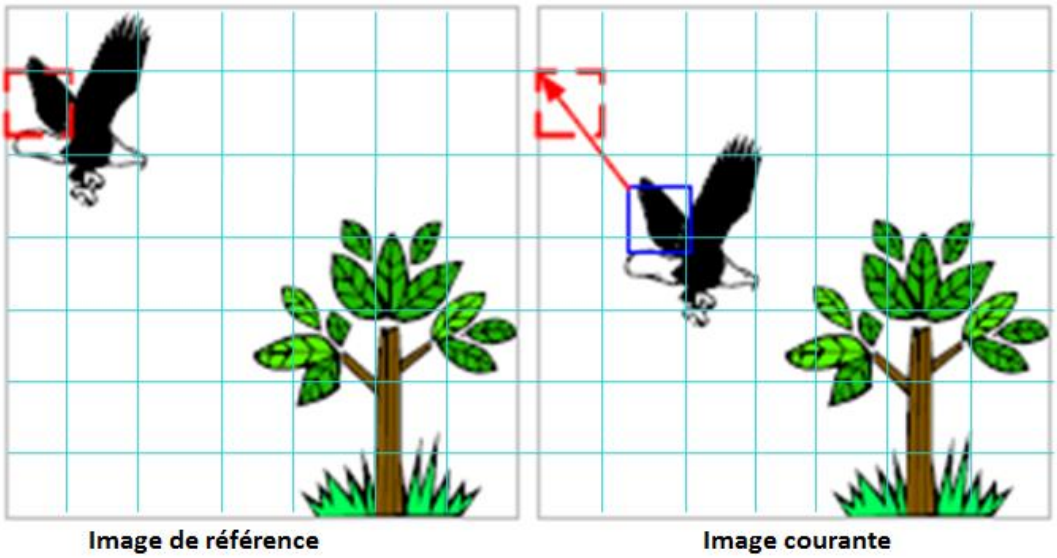

Figure 5. 5: Mise en correspondance de blocs

Le but de cette méthode d'estimation de mouvement est de trouver le MB le plus ressemblant au MB courant en se basant sur une formule mathématique décrivant la distorsion ou l'erreur entre deux MBs.

Cette formulation n'est pas imposée par le comité de MPEG mais c'est la plus utilisée dans la littérature. La somme des différences absolues ou SAD (Sum of Absolute Difference) est un critère permettant de mesurer l'erreur entre deux régions. Elle est donnée par l'équation 5.1 :

$$SAD(x,y) = \sum_{i=1}^{N}\sum_{j=1}^{M}\left|MBcur(i,j) - MBref(i+x, j+y)\right|$$
$$MV = (x,y) \qquad \textbf{Eq5.1}$$
$$SAD_{\min} = \min(SAD(i,j))$$

Généralement l'image est découpée en MB de taille 16×16 pixels.
MBcur est le MB de l'image courante.
MBref est le MB de l'image de référence.
MV : représente le vecteur de déplacement ou aussi le vecteur de mouvement du MB de l'image courante dans l'image de référence.
Les BMAs imposent une hypothèse supplémentaire, que les pixels du MB à estimer ont tous le même mouvement de translation. Dans la littérature, plusieurs algorithmes de recherche ont été inventés afin d'augmenter les performances, en termes de temps de calcul et aussi de qualité de compression, et de réduire la complexité architecturale de l'estimateur de mouvement. On cite parmi ces algorithmes : le TSS (Three Step Search), DS (Diamond Search) et le FS (Full Search).
Nous décrivons brièvement quelques algorithmes et nous allons détailler l'algorithme utilisé dans cette thèse.
Three Step Search : C'est un algorithme rapide [93] dont le fonctionnement commence en premier lieu par le choix d'un MB du centre entouré par huit MB à une distance de quatre pixels du pixel central, afin de les tester et de déduire le MB qui minimise le SAD. Ce nouveau MB trouvé sera pris comme la nouvelle origine. En second temps, la même opération est répétée tout en choisissant les huit points situés à une distance de deux pixels. Cet algorithme est itératif, nous procédons de la même manière dans la troisième étape, mais avec un déplacement d'un pixel. Le nombre de calculs effectués dans ces trois étapes est constant (25 candidats), ce qui rend possible le calcul du temps d'exécution si le temps d'accès aux mémoires est déterministe. Plusieurs variantes ont été proposées et chacune d'elles vise à optimiser et à améliorer cet algorithme. Dans [94], l'auteur a proposé une approche permettant de rendre dynamique le nombre de pas grâce à un seuil. La réduction du nombre de candidats [95], à son tour, a fait aussi l'objet d'un travail de recherche et, pour atteindre ce but, l'auteur a fixé un seuil pour lequel la recherche de correspondance s'arrête dès que le résultat trouvé est acceptable. De nombreux calculs inutiles sont donc évités. L'algorithme TSS s'adapte bien aux mouvements de faible amplitude tandis qu'il peut trouver des problèmes dans le cas de grands déplacements, c'est-à-dire de séquences vidéo rapides.

Diamond Search

Il est proposé dans [96] et nommé « algorithme d'estimation de mouvement par descente de gradient basé bloc » puis « recherche en diamant » par Shan Zhu [97]. Cet algorithme se base sur un motif en diamant. Il permet le suivi d'une direction de descente du critère de distorsion. Il existe deux formes du DS le Large Diamond Search Pattern (LDSP) et le Small Diamond Search Pattern (SDSP) [98]. Le

LDSP teste les huit positions qui entourent le centre afin de déduire celle qui minimise le SAD. Celle trouvée sera prise comme centre dans l'itération suivante. Cette opération se répète jusqu'à trouver un SAD qui coïncide avec le centre du diamant. Il y aura moins de SAD à calculer quand on passe d'une itération à la suivante. Dans le cas du SDSP, le même principe est adapté sauf que le pixel qui a le SAD minimum donne le vecteur de mouvement. Lorsque les mouvements dans une séquence vidéo sont rapides, cet algorithme perd son efficacité et le taux de calcul augmente.

De nombreux algorithmes, autres que ceux qu'on a cités, ont été inventés dans la littérature suite à l'importance de l'estimation de mouvement dans les codeurs vidéo, notamment le H.264. Chacun d'eux vise à offrir une qualité d'image nette et un taux de compression meilleur. Les algorithmes que nous venons de détailler sont simples à mettre en œuvre et également à implémenter, mais ils présentent des inconvénients qui peuvent toucher la qualité de l'image. L'algorithme de recherche complet ou le FS (Full Search) est lui-même un algorithme de recherche de correspondance de bloc. Il offre une qualité meilleure que les autres algorithmes, mais une complexité architecturale grandissante. Dans cette thèse, nous nous intéressons à cet algorithme et nous allons proposer une amélioration architecturale dans les sections qui viennent. La section suivante décrit en détail cet algorithme.

Recherche exhaustive

Cet algorithme de mise en correspondance par recherche complète a l'avantage de donner une qualité d'image meilleure que les autres algorithmes et ainsi de trouver la solution optimale dans une région donnée. En comparant de façon exhaustive tous les blocs de référence dans la fenêtre de recherche, le FS donne le vecteur de mouvement le plus précis qui réduit le SAD. L'algorithme 1 décrit le principe de fonctionnement de cette méthode de recherche avec un déplacement de +/- P dans l'image de référence, afin de construire la fenêtre de recherche, et un MB de taille N×N. L et H désignent respectivement la largeur et la hauteur de l'image.

Cet algorithme calcule le SAD pour tout déplacement possible d'amplitude p (Figure 5.6), soit globalement $(2p+1)^2$ candidats. En effet, il demande une capacité de calcul énorme, notamment pour les images à haute définition. Pour minimiser le nombre d'opérations à effectuer, plusieurs variantes ont été proposées. On cite :

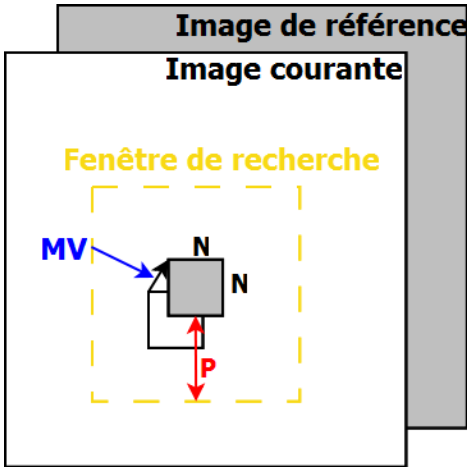

Figure 5. 6: Structure de la fenêtre de recherche

Algorithme 1 : Recherche exhaustive.

Pour L= 0 à L/N ***faire***
Pour H= 0 à H/N ***faire***
MV (L, H) = (0,0)
SAD (L, H) = ∞
Pour i= -p…p ***faire***
Pour j= -p…p ***faire***
Pour x =1 à N ***faire***
Pour y= 1 à N ***faire***

$$SAD(i, j) = \left| MBCur(x, y) - MB\,\mathrm{Re}\, f(i + h, j + y) \right|$$

Fin pour
Fin pour
Si $SAD(i, j) < SAD(L, H)$ *Alors*
MV (L, H) = (i,j)
SAD (L, H) = SAD (i,j)
Fin si
Fin pour
Fin pour
MV (L, H)
Fin pour
Fin pour

- Eviter les opérations inutiles.
- Réduire le nombre de candidats en se basant sur le fait que le SAD varie d'une façon monotone dans la zone de recherche.
- L'élimination de candidats [99], en effectuant une estimation rapide du SAD afin de ne pas effectuer la totalité des calculs.

2.2.4. La compensation de mouvement

La compensation de mouvement est l'opération permettant de construire l'image courante MB par MB à partir d'une image de référence et d'information du vecteur de mouvement. Elle permet cette reconstruction en déduisant le MB prédit dans l'image de référence, puis en le soustrayant du MB de l'image courante pour obtenir le MB résiduel qui a été déjà codé et envoyé avec le vecteur de mouvement et l'emplacement de la meilleur concordance, c'est-à-dire le MB le plus ressemblant. Cela réduit considérablement la quantité de données à transmettre. La compensation de mouvement est un composant en soi, mais qui peut être regroupé avec l'estimation de mouvement. Il prend, dans un premier temps, le vecteur de mouvement et l'image de référence comme entrée pour générer, en second temps, le MB prédit c'est-à-dire le MB correspondant au MB le plus ressemblant au MB actuel. Dans le sens du codage, le MB résiduel est codé, puis décodé et ajouté à la meilleure concordance pour former un MB reconstruit qui va être stocké et utilisé comme référence pour des futures prédictions de mouvement. Cette étape est importante pour assurer l'utilisation de la même image de référence par le

codeur et le décodeur. L'algorithme 2 décrit le fonctionnement du module de compensation. La compensation, comme l'estimation de mouvement peut se faire pour les images de type P ou B (images bidirectionnelles).
De nombreux travaux ont été proposés dans la littérature en vue de réduire soit le nombre de calculs effectués, soit l'accès mémoire [100, 101]. L'architecture proposée dans cette thèse va être développée en profondeur dans la partie de la conception architecturale.

Algorithme 2 : compensation de mouvement.

```
Pour i= 1 à  ligne  faire
  Pour j= 1 à Col faire
Imgref(i,j) :=img(i,j)
 Fin pour
Fin pour
refMBVer :=MVy
refMBVer :=MVy
Pour i= 0 à  MBsize-1  faire
  Pour j= 0 à MBsize-1  faire
ImagComp(i,j) :=imgref(refMBVer+1,refMBhor=j)
 Fin pour
Fin pour
MBP :=Imagcomp
Fin
```

2.3. La transformation et la quantification

La transformée est une étape de traitement d'image dans le codeur vidéo. Comme c'est déjà illustré dans la figure 5.1, le codeur H.264 peut être composé de deux grandes parties : une phase de prédiction assurée par l'estimation de mouvement et la compensation de mouvement, et une phase de traitement d'image décrite par la transformée en cosinus discret et la quantification. Généralement, la transformation est réalisée par MB. Elle est appliquée sur le MB résiduel obtenu après la prédiction.

2.3.1. La transformation en cosinus discret

La transformation permet de convertir l'information contenue dans l'image résiduelle dans un autre domaine, le domaine de transformation. Le choix de la transformation dépend de plusieurs critères. De nombreuses transformations ont été proposées pour la compression d'image et de vidéo. Les plus en vogue sont la transformée cosinus discrète (TCD) et la transformée en ondelettes discrète. Le but donc de la TCD par blocs 8×8 utilisée dans toutes les normes actuelles de codage vidéo est la dé-corrélation des blocs 8×8 de pixels originaux ou de pixels différentiels à mouvement compensé et la compression de l'énergie avec le moins de coefficients possibles.

La transformation est appliquée sur les blocs du résidu obtenu après prédiction, c'est-à-dire une fois soustraite la prédiction au bloc original. H.264/AVC utilise trois transformations suivant le type de données résiduelles à coder. Pour les MB prédits en mode 16×16, les composantes continues sont placées dans une matrice 4×4 et une transformation de Hadamard lui est appliquée. Pour tous les MB, les coefficients continus de chrominance sont placés dans une matrice 2×2 et une transformation de Hadamard lui est également appliquée. Enfin, pour le reste des blocs 4×4, une transformation DCT est appliquée.

Plusieurs algorithmes ont été proposés pour mettre en œuvre le DCT. On définit l'algorithme Loeffler [102] qui a donné une nouvelle classe de DCT-1D en utilisant seulement 11 multiplications et 29 additions. Pour mettre en œuvre un tel algorithme, une ou plusieurs multiplications doivent être intégrées. Cela nécessite une surface de silicium de haute occupation. Aussi la technique de la distribution arithmétique est-elle largement utilisée pour de tels algorithmes. On définit également une nouvelle technique proposée, basée sur la fusion de la distribution arithmétique avec l'algorithme de Loeffler. La formulation mathématique de la DCT est donnée par l'équation 5.2.

$$F(k) = C.a_k \sum_{n=0}^{7} (f(n).\cos \frac{(2.\Pi(2n+1).k}{4N} \qquad \textbf{Eq (5.2)}$$

Avec: $a_k = \begin{cases} \cos\frac{\pi}{4} & \text{pour} \quad k = 0 \\ 1 & \text{pour} \quad k = 1,\ldots,7 \end{cases}$

f(n): Valeur reçue à l'entrée de la transformée, F(k) : coefficient de la DCT-1D.

Pour un bloc de 8×8 pixels, on applique tout d'abord une DCT-1D sur les lignes, puis on applique de nouveau, sur la matrice de pixels résultante, une autre DCT-1D pour donner finalement la matrice transformée, comme le montre la figure 5.7 ci-dessous. Cette implémentation permet de réduire la complexité des calculs et le nombre d'opérateurs nécessaires aux calculs.

En se basant sur cette équation, nous remarquons la complexité de ce module, DCT, est due à la quantité énorme de données à traiter ainsi qu'aux opérations arithmétiques à effectuer. La distribution arithmétique [103] est une technique qui permet de réduire la complexité en termes de ressources. En stockant d'abord un nombre fini de résultats intermédiaires, une somme de produits peut être obtenue par des opérations d'additions répétées et des décalages sans l'utilisation d'une multiplication.

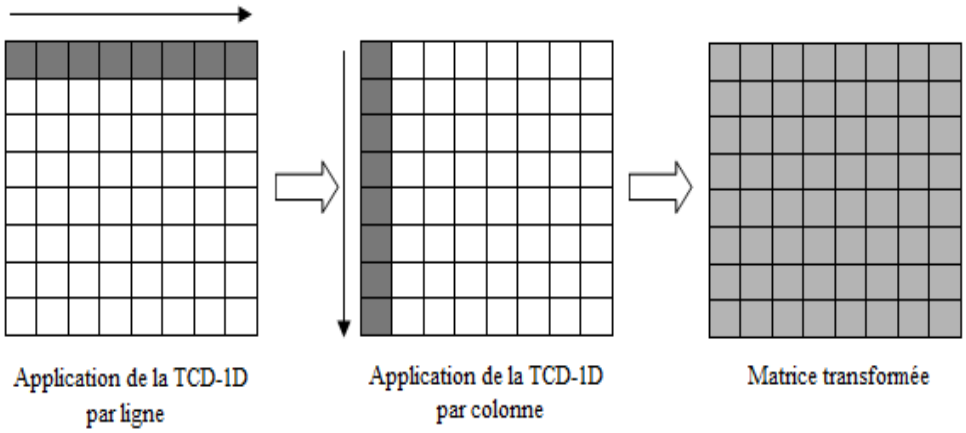

Figure 5. 7: Principe de la DCT.

2.3.2. La quantification

La quantification se résume à faire correspondre un signal contenant un ensemble de valeurs X à un signal quantifié contenant un nombre réduit de valeurs Y. Le but de cette opération est de représenter le signal quantifié avec moins de bits que l'original puisque le champ des valeurs est plus petit. Donc, la quantification a pour but de discrétiser les amplitudes des coefficients issus de la transformée. Elle permet de réduire la taille des coefficients sans apporter de détériorations visuelles à l'image. Vu que l'œil humain est plus sensible aux basses fréquences, le résultat de la DCT est quantifié pour supprimer les hautes fréquences [104]. La quantification fait correspondre à chaque coefficient de transformation une valeur numérique. Celle-ci peut prendre un nombre fini et limité de valeurs.

Le codage par quantification est donc l'une des sources de perte d'information dans le système de compression. Son rôle est en effet de réduire le nombre de bits nécessaires à la représentation de l'information. Elle est réalisée avec la prise en compte de l'aspect psycho-visuel (l'œil humain), ce qui permet de déterminer la distorsion tolérable à apporter au signal à coder.

Dans la norme H.264, 52 quantificateurs sont prévus [105, 106]. Le parcours des données quantifiées consiste à mettre les coefficients les plus énergétiques au début du train et à placer les données nulles à la fin pour minimiser la taille des informations à coder.

2.4. Le codage entropique

Il s'agit des données résiduelles venant de la quantification, aussi bien des paramètres, des vecteurs de mouvement etc. Dans la norme H.264 le codage entropique peut être réalisé de trois manières différentes. Une première méthode utilise une table universelle de mots de code (UVLC pour *Unified Variable Lenght Coding*). Cette table est utilisée pour coder la plupart des éléments de synchronisation comme les en-têtes. Les deux autres méthodes sont utilisées pour coder presque tous les autres éléments syntaxiques (coefficients, vecteurs mouvements). Il s'agit d'une part d'un codage à longueur variable VLC (*Variable Length Coding*) adaptatif au contexte (CAVLC pour *Context Adaptative*

Variable Length Coding) et d'autre part d'un codage arithmétique adaptatif contextuel (CABAC pour *Context Adaptative Binary Arithmetic Coding*) [105].

Le codage entropique est le dernier élément de la présentation de la norme H.264. Il donne comme résultat le bistream H.264. Cela permet de réduire la quantité d'informations à transmettre en donnant moins de place aux éléments récurrents. La norme H.264 peut être conçue à l'aide de plusieurs types de codeurs entropiques. Tout dépend du profil choisi. Par exemple, le CAVLC est adapté au profil de base tandis que le CABAC est mieux adapté au profil principal.

Dans cette partie, nous avons présenté brièvement la norme H.264. Cette présentation permet au lecteur de cette thèse d'avoir une idée sur l'application qu'on a utilisée. Dans la partie suivante, nous passons à la modélisation à haut niveau et à l'extraction du parallélisme local du codeur H.264.

3. Modélisation du H.264 en MARTE sur l'environnement Gaspard2

L'objectif maintenant est de proposer des modèles fonctionnels du codeur H.264 sans tenir compte d'une part de son profil et d'autre part des détails d'implémentation physique. Cette méthode permet d'avoir un ensemble de modèles permettant de synthétiser ou de simuler ce codeur dans un langage cible. Nous optons pour une implémentation matérielle afin d'implémenter les tâches du codeur H.264. La figure 5.8 illustre le schéma fonctionnel, dans le sens du codage seulement, du codeur H.264 pour la prédiction inter c'est-à-dire les images de type P.

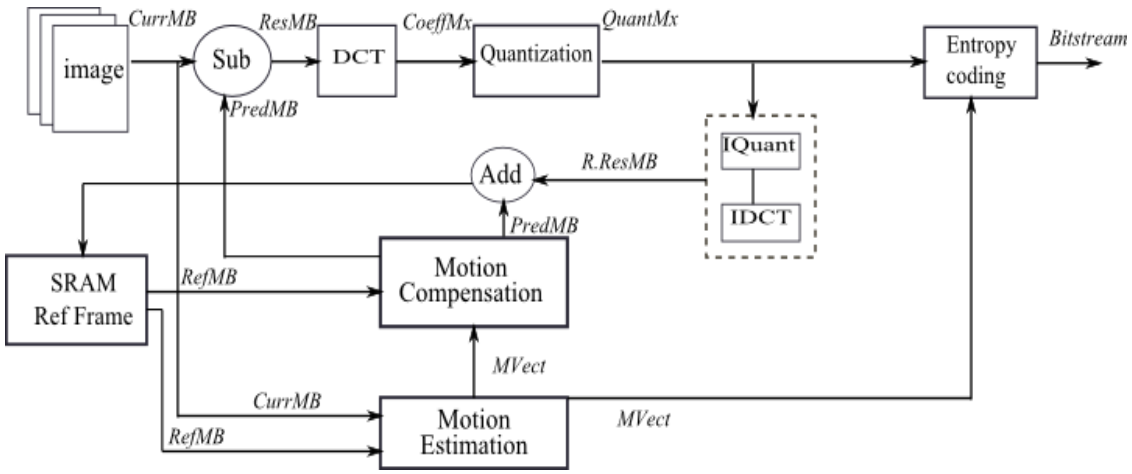

Figure 5. 8 : codage P frame suivant la norme H.264.

Dans le processus de codage des images de type P et suivant la norme H.264, les données à manipuler sont des MB de pixels. Nous pouvons à ce stade définir un parallélisme de données local, cela signifie que dans chaque tâche élémentaire, qui forme le codeur, nous cherchons le parallélisme possible. Plusieurs travaux ont proposé de paralléliser ce codeur en vue d'implémentation sur un réseau de processeur afin d'améliorer le temps d'exécution. Dans [107] l'auteur a proposé de diviser le codeur en deux grandes parties : une partie de traitement d'image nommée *MainDivix* contenant la DCT, la quantification, la quantification inverse, la DCT inverse et la compensation de mouvement. La deuxième partie s'intéresse à la compression par codage entropique. Cette approche favorise le traitement en parallèle par la distribution de ces deux tâches sur un réseau de processeurs.

En utilisant le profil MARTE et l'environnement de modélisation Gaspard2, nous avons modélisé le codeur en commençant par la modélisation des tâches élémentaires. Le parallélisme de tâche ou de donnée est exploité afin d'exprimer les répétitions. Cette modélisation est indépendante de tout détail d'implémentation et de nature de séquence vidéo. Nous procédons par hiérarchie. Dans le premier niveau, les tâches élémentaires de l'application sont présentées comme étant des IPs matériels. Nous ajoutons à ces tâches deux composantes. La première permet la lecture des images, elle réalise le fonctionnement d'une caméra, et la deuxième pour découper les images en MB de tailles égales. La figure 5.9 illustre ce niveau.

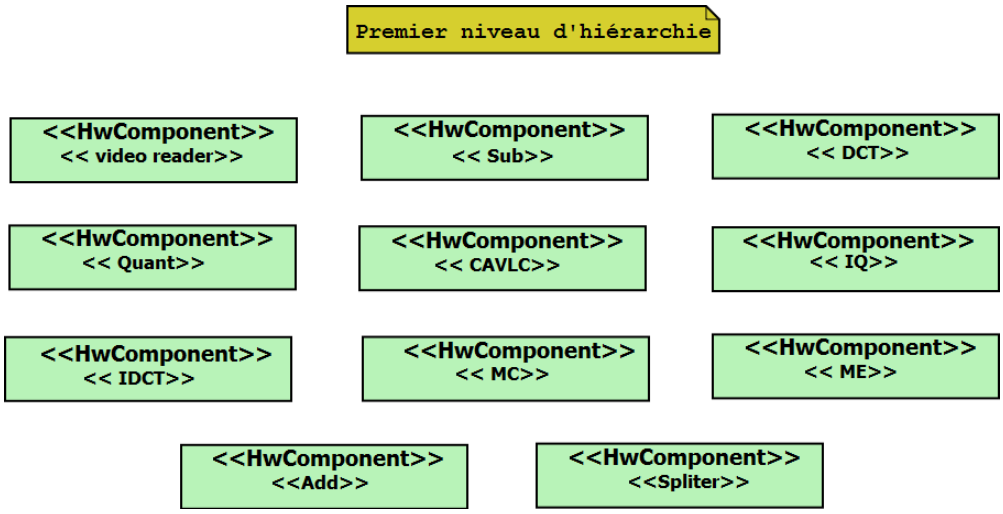

Figure 5. 9: Tâches élémentaires de l'application.

Quant à la DCT et la quantification, il s'agit d'exploiter l'indépendance des données à traiter et la répétition des tâches effectuées par un DCT unidimensionnel et un quantificateur, et multiplier ces unités effectuant des opérations sur des vecteurs de 8 pixels, pour traiter 8 vecteurs simultanément. A l'entrée de la DCT, un bloc de 8×8 pixels sera traité en premier lieu par la DCT qui prend 8 cycles pour la production de 64 coefficients simultanément. Ensuite ces coefficients seront traités par le bloc de la quantification qui prend de même 8 cycles pour les quantifier, huit coefficients par cycle. Le même principe est appliqué à la DCT inverse et la quantification inverse. Par conséquence les quatre tâches sont modélisées, dont chacune est composée par une sous tâche-élémentaire représentant l'unité de traitement. Pour la DCT par exemple, la sous-tâche permettant le calcul de coefficient est répétée quatre fois, du fait que les MBs à traiter sont de tailles 16×16, donc nous appliquons le stéréotype *Shape* pour exprimer cette répétition. Des Tilers sont nécessaires pour découper le MB 16×16 en entrée puis de reconstruire le MB en sortie. En se basant sur ce principe, nous modélisons les autres tâches telles que la quantification, le codage entropique, et l'estimation de mouvement. Le codage entropique est appliqué sur des MBs de taille 4×4 sachant que les données venant de la quantification sont des MBs de tailles 16×16. Cela se traduit par une répétition de l'unité de traitement du codage entropique.

Dans le cas de l'estimation de mouvement, plus précisément pour l'algorithme FS, la lecture des MBs de taille 16×16 de l'image courante et l'image de référence se fait pixel par pixel suivant la première dimension. Chaque pixel du MB courant est comparé avec d'autres pixels du MB de référence. La

tâche élémentaire dans l'estimation de mouvement est le SAD, qui est répétée 16 fois. Les Tilers permettent la répartition des pixels vers les processors élémentaires (PE). Après avoir générer les SADs un module de comparaisons, dont le port d'entrée est stéréotypé *Shaped* : qui est égale 16 afin de pouvoir lire les 16 SADs généré par les 16 PEs, se charge à la déduction de la plus petite valeur du SAD afin de trouver la meilleur correspondance et le vecteur de mouvement qui lui est associé.

Les autres tâches élémentaires telles que la compensation de mouvement, le soustracteur et l'additionneur ne peuvent pas subir le parallélisme. Pour mieux expliquer le calcul, nous prenons le cas de la DCT, le Tiler a une origine de $\begin{pmatrix}0\\0\end{pmatrix}$, une matrice d'ajustage égale a $\begin{pmatrix}10\\01\end{pmatrix}$ et un pavage de $\begin{pmatrix}80\\08\end{pmatrix}$. L'ajustage correspond dans ce cas à un déplacement par un élément sur la première et la deuxième dimension. Le pavage permet de lire le motif suivant, qui est un MB de 8×8 pixels, en se déplaçant dans le tableau par 8 pixels, cela est fait sur les deux dimensions. La modélisation, d'une façon compacte, de ces tâches élémentaires donne naissance à une seconde hiérarchie. La figure 5.10 illustre ce niveau d'hiérarchie.

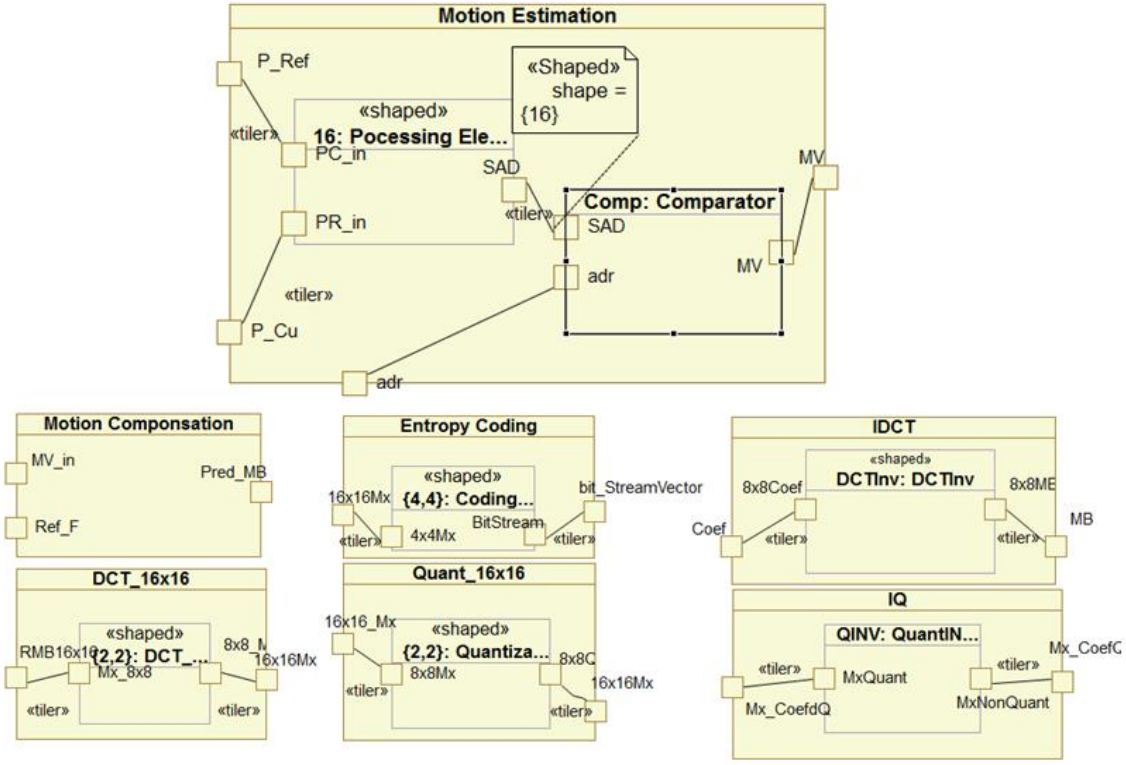

Figure 5. 10: le second niveau d'hiérarchie, la modélisation des tâches élémentaire du codeur H.264

La tâche de codage P frame suivant la norme H.264 est modélisée dans un autre niveau d'hiérarchie. Elle permet le codage en suivant tout l'algorithme qui est décrit par des tâches élémentaires. Les autres composants responsables au contrôle, au stockage à l'acquisition vidéo sont aussi modélisés afin d'obtenir le modèle complet d'une application vidéo à base du codeur H.264. La figure 5.11 illustre le modèle complet de cette application.

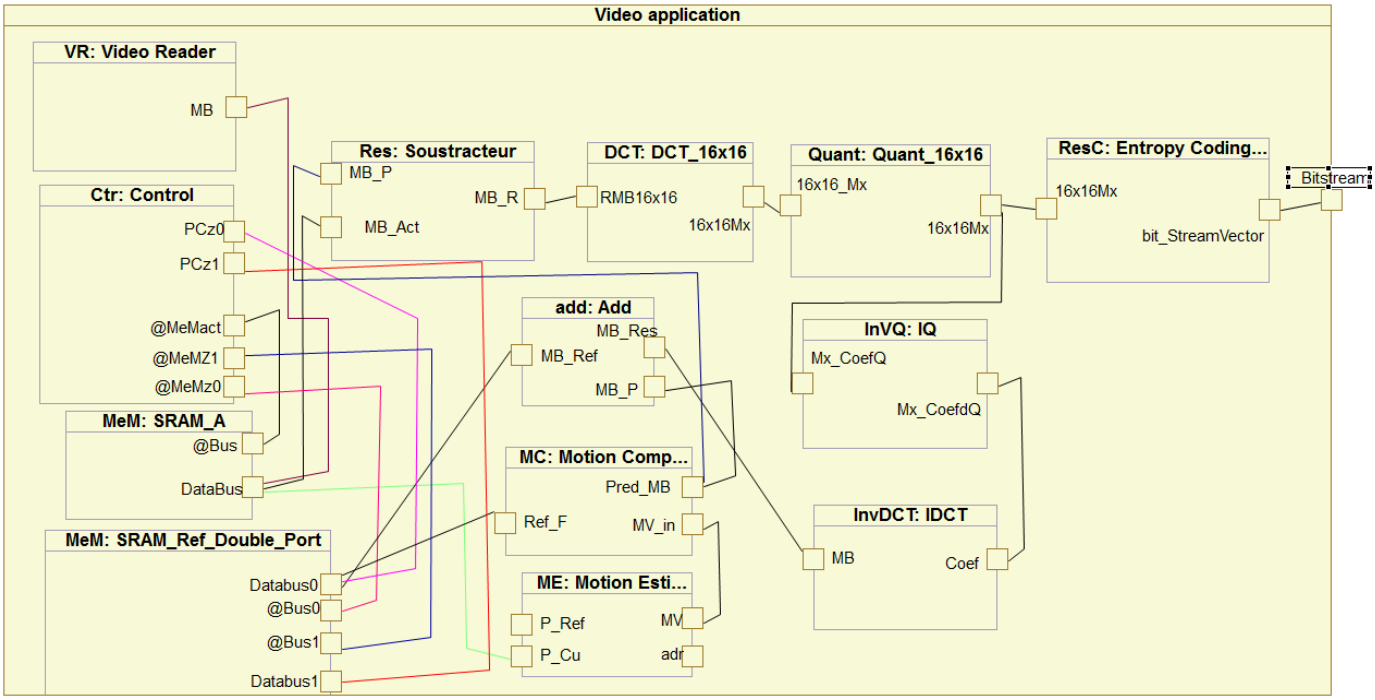

Figure 5. 11: le troisième niveau d'hiérarchie : modélisation de l'application vidéo basée sur le codeur H.264

La tâche *video application* est modélisée en utilisant le profile MARTE et l'environnement Gaspard2. Elle permet le codage vidéo, dans le cas de l'inter prédiction, conformément à la norme H.264. D'autres composants sont ajoutés tels que les mémoires, le contrôle, et l'acquisition du vidéo afin de décrire un système complet pour le traitement vidéo.

Dans cette partie de la thèse, nous avons proposé des modèles décrivant les tâches élémentaires du codeur H.264. Le parallélisme local, possible, a été exprimé en utilisant des répétitions par le biais de la sémantique du package RSM, ainsi que d'autres packages pour la modélisation des mémoires. Suite à ce travail de modélisation, une étape de modélisation en VHDL semble nécessaire pour décrire les tâches élémentaires afin de générer tout le système (Application/ Architecture).

4. Conception matérielle des tâches élémentaires du codeur H.264.

L'objectif de cette partie est de modéliser en VHDL les tâches élémentaires du codeur H.264 pour l'inter prédiction afin de définir une bibliothèque d'IPs qui va être utilisée pour la génération de code VHDL du SoC complet. Néanmoins, notre contribution dans cette partie ne prend pas en compte que le développement des IPs. Nous proposons également des améliorations en termes de performances des architectures conçues. La consommation d'énergie, la fréquence, la surface font l'objet de notre contribution. Une étude comparative avec de nombreuses variantes proposées dans la littérature a été menée. Nous détaillons, respectivement, dans les sous-sections suivantes l'architecture de l'estimation de mouvement, de la DCT, de la quantification, et la compensation de mouvement.

4.1. Estimateur de mouvement à taille de bloc fixe et variable

Afin d'aboutir à des performances convenables pour une application de traitement vidéo, de réduire la surface, d'augmenter la fréquence et de contrôler la consommation, des processeurs élémentaires dédiées ont été développés. Les architectures VLSI implémentées sur des ASICs ou FPGAs, permettent de tenir compte des différents défis relatifs à l'estimation de mouvement. Dans les architectures VLSI, plusieurs problèmes peuvent être résolus grâce à la réutilisation des ressources pour réduire par exemple la surface et augmenter le degré de parallélisme. Dans le cas de l'estimation de mouvement, le parallélisme peut être vu en plusieurs niveaux. L'algorithme 5.1 de l'estimation de mouvement en utilisant l'algorithme FS montre la possibilité d'extraire le parallélisme par la réorganisation des boucles imbriquées. Des travaux de recherches ont montré que le processus de l'estimation de mouvement prend entre 50% à 70% du temps total de codage H.264 [108]. A l'heure actuelle, l'inconvénient de H.264 est que sa complexité de calcul est très élevée par rapport à d'autres normes de codage vidéo. En effet, la plupart des chercheurs dans ce domaine s'orientent vers la réduction de complexité en proposant des architectures régulières, flexibles et qui favorisent la réutilisation des ressources.

Dans la norme H.264 l'estimation de mouvement peut être faite par plusieurs méthodes. Cela peut être une estimation de mouvement pour des blocs de taille fixe, c'est-à-dire de taille 16×16 ou de taille variable comme le montre la figure 5.4. Nous commençons en premier lieu par détailler l'architecture d'un estimateur de mouvement pour des blocs de taille fixe, et en second lieu pour des blocs de taille variable.

L'entité d'un estimateur de mouvement est déjà définie depuis la fin des années 80 [109]. Cependant, ce qui change dans les nouveaux travaux c'est l'architecture du processeur élémentaire (Processing Element, PE) qui est un accélérateur matériel dédié pour le calcul du SAD.

4.1.1. Les architectures systoliques

L'approche basée sur les architectures systoliques est largement utilisée dans la littérature. Elle se base sur un ensemble des PE interconnectés sous formes des vecteurs. Ce type d'implémentation favorise la réutilisation successive des données, en réduisant les goulots d'étranglement dû à l'accès aux pixels stockés dans les mémoires. En effet les pixels sont diffusés d'un PE à l'autre à travers des lignes basées sur des bascules. Le parallélisme, dans ce cas, est réalisé sur des intervalles de temps. Tous les PEs de l'estimateur de mouvement traitent l'opération du SAD. Les architectures systoliques sont classifiées en deux familles ; les architectures à une seule dimension dans laquelle un PE est connecté à son voisin, tandis que, dans le cas d'une architecture à deux dimensions, chaque PE est connecté à ses voisins horizontalement ou verticalement. Généralement ce type d'architecture est dédié à l'algorithme FSBM (Full Search Block Matching). La figure 5.12 illustre les deux familles des architectures systoliques.

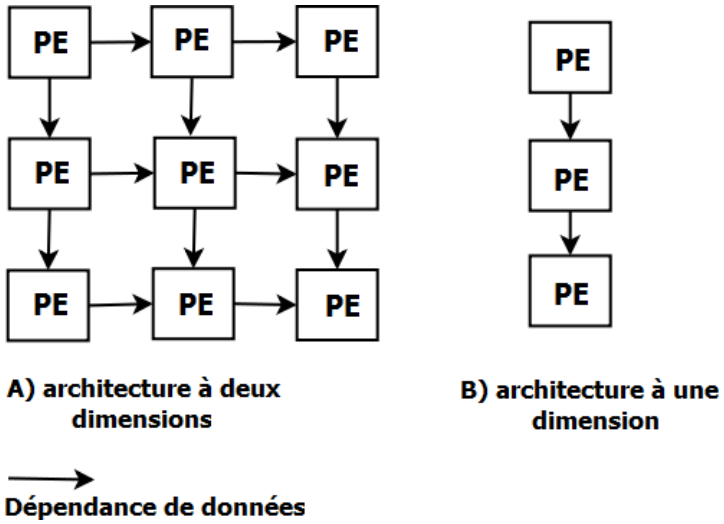

Figure 5. 12: Exemple d'architecture à une et deux dimensions

4.1.2. Parallélisme dans l'algorithme FS

Le parallélisme de l'algorithme FS réside dans les boucles imbriquées montrées dans l'algorithme 5.1. Dans cet algorithme, les PEs calculent le SAD pour un MB donné de l'image courante en exploitant une fenêtre de recherche de l'image de référence. Le tableau 5.2 décrit la propagation des pixels, du MB courant et de la fenêtre de recherche à travers les PEs.

Tableau 5.2 : Le flow des pixels [Yap et al]

Clk	PE_0 (t-0)	PE_1 (t-1)		PE_{14} (t-14)	PE_{15} (t-15)
0	C(0,0)-R(0,0)	C(0,0)-R(1,0)		C(0,0)-R(14,0)	C(0,0)-R(15,0)
1	C(1,0)-R(1,0)	C(1,0)-R(2,0)		C(1,0)-R(15,0)	C(1,0)-R(16,0)
…	…	…	…	…	…
14	C(14,0)-R(14,0)	C(14,0)-R(15,0)		C(14,0)-R(28,0)	C(14,0)-R(29,0)
15	C(15,0)-R(15,0)	C(15,0)-R(16,0)		C(15,0)-R(29,0)	C(15,0)-R(30,0)
			…		
240	C(0,15)-R(0,15)	C(0,15)-R(1,15)		C(0,15)-R(14,15)	C(0,15)-R(15,15)
241	C(1,15)-R(1,15)	C(1,15)-R(2,15)		C(1,15)-R(15,15)	C(1,15)-R(16,15)
…	…	…	…	…	…
254	C(14,15)-R(14,15)	C(14,15)-R(15,15)		C(14,15)-R(28,15)	C(14,15)-R(29,15)
255	C(15,15)-R(15,15)	C(15,15)-R(16,15)		C(15,15)-R(29,15)	C(15,15)-R(30,15)

Le nombre de PE utilisé dans l'algorithme FS est généralement égal à la taille de MB de l'image courante. Dans ce cas, et suivant le tableau de flow de pixels, nous avons 16 PEs. Dans ce tableau, les pixels du MB courant sont désignés par C et le R désigne les pixels de la fenêtre de recherche.
En se basant sur ce tableau, beaucoup de travaux ont été proposés. Yang et al. [110] ont inventé la première architecture permettant d'exécuter l'algorithme FSBM. La figure 5.13 illustre l'architecture proposée. Chaque PE se charge de calculer le SAD entre un MB de l'image courante et MB de l'image de référence.

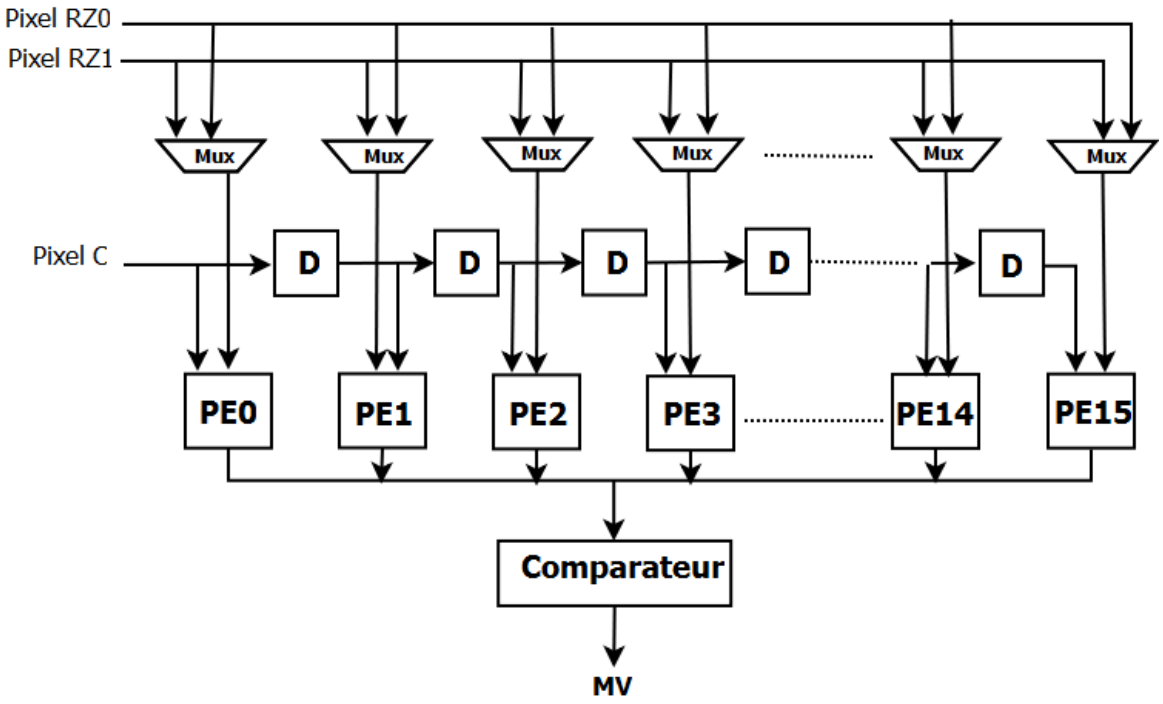

Figure 5. 13: Architecture de l'estimateur de mouvement proposé par Yang et al. [110]

Les Pixels du MB courant sont propagés à travers les registres à décalage tandis que les pixels de l'image de référence qui est découpée en deux zones, z0 et z1, blocs diffusés à tous les PEs. Cette conception favorise un traitement en parallèle avec une utilisation de 100% des ressources. Par ailleurs, cette conception réduit le trafic et les accès mémoire.

Dans la même optique, Yeo et al ont proposé [111] une architecture à deux dimensions comme le montre la figure 5.14. Les pixels du MB courant sont diffusés à travers les registres à décalage tandis que les pixels de la fenêtre de recherche sont propagés dans les deux dimensions. L'ensemble de N×N PEs est responsable au calcul de N×M régions dans la fenêtre de recherche, avec N=4 est la taille du MB courant. Par conséquent il y a au total $(2.SR_v/N) \times (2.SR_V/N)$ PEs pour trouver la correspondance dans la fenêtre de recherche. Par la diffusion des pixels dans les deux directions, cette conception favorise la réutilisation des données.

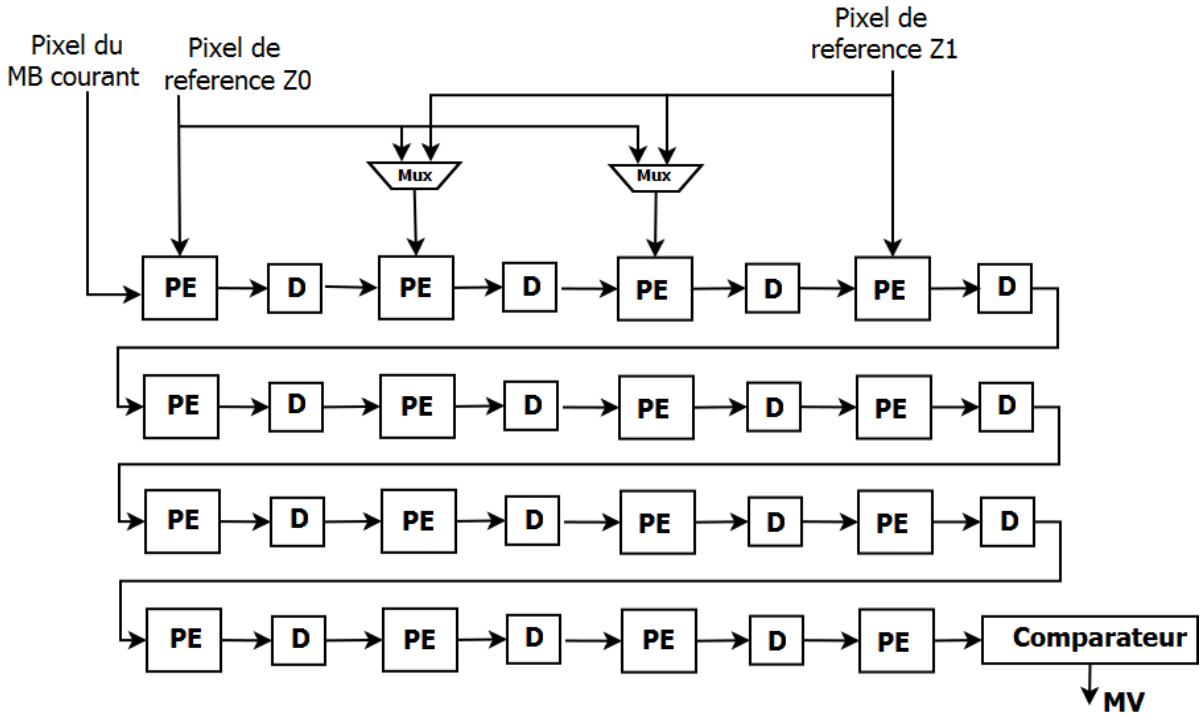

Figure 5. 14: Architecture proposée par [111]

4.1.3. Conception d'un ME à faible coût de consommation pour des blocs à taille fixe

Dans cette partie de la thèse, et suite à notre objectif global de la génération de code VHDL par l'environnement GASPARD2, nous proposons une modélisation VHDL d'un estimateur de mouvement à faible coût de consommation. Nous rappelons que l'estimation de mouvement est une tâche élémentaire de l'application H.264 donc la description dans un langage de programmation s'avère nécessaire afin de générer le code VHDL de tout le système. La partie contrôle et l'architecture du PE et du comparateur sont toutes détaillées dans cette section. La validation expérimentale est réalisée par une implémentation sur un ASIC. Nous montrons que la réduction de la consommation est assurée par l'intégration d'un module permettant de mettre en veille les composants qui ne sont pas en cycle de traitement par rapport aux autres élément de l'estimateur.

Les hypothèses de notre proposition sont les suivantes :

- Effectuer l'estimation de mouvement pour des macros blocs de 16×16 pixels.
- Utiliser une fenêtre de recherche de taille [-7,7].

Avec ces conditions, la zone de recherche doit inclure le bloc 16×16 pixels de l'image de référence plus un déplacement de 7 pixels sur les deux directions. Ceci implique que la taille de la fenêtre de recherche est de 30×30 pixels, qui seront utilisés pour l'estimation de mouvement de tous les blocs de l'image courante. L'image courante et la fenêtre de recherche qui est divisée en deux sous- zones sont stockées dans des SRAM externes à l'estimateur de mouvement. Avec ces conditions, la taille de la

fenêtre de recherche est de 30×30 pixels. L'estimation de mouvement conçu répond aux exigences de l'algorithme FS. Pour un déplacement de 7 pixels, il y a 255 SAD à calculer pour chaque MB.

Le « Top Level » de l'architecture proposée est inspiré de celui qui est présenté dans [110], plus précisément, de la conception du PE. Le schéma en bloc de l'estimateur comprend 16 PEs, deux chaînes de retard composées par 16 flip-flop, 16 multiplexeurs pour la sélection de la fenêtre de recherche, 3 mémoires locales (une pour le bloc de référence et 2 pour la fenêtre de recherche), une unité de contrôle, un bloc d'adressage et un comparateur. Le schéma de l'ensemble du système est illustré par la figure 5.15. Nous détaillons brièvement les éléments qui forment l'architecture de l'estimateur.

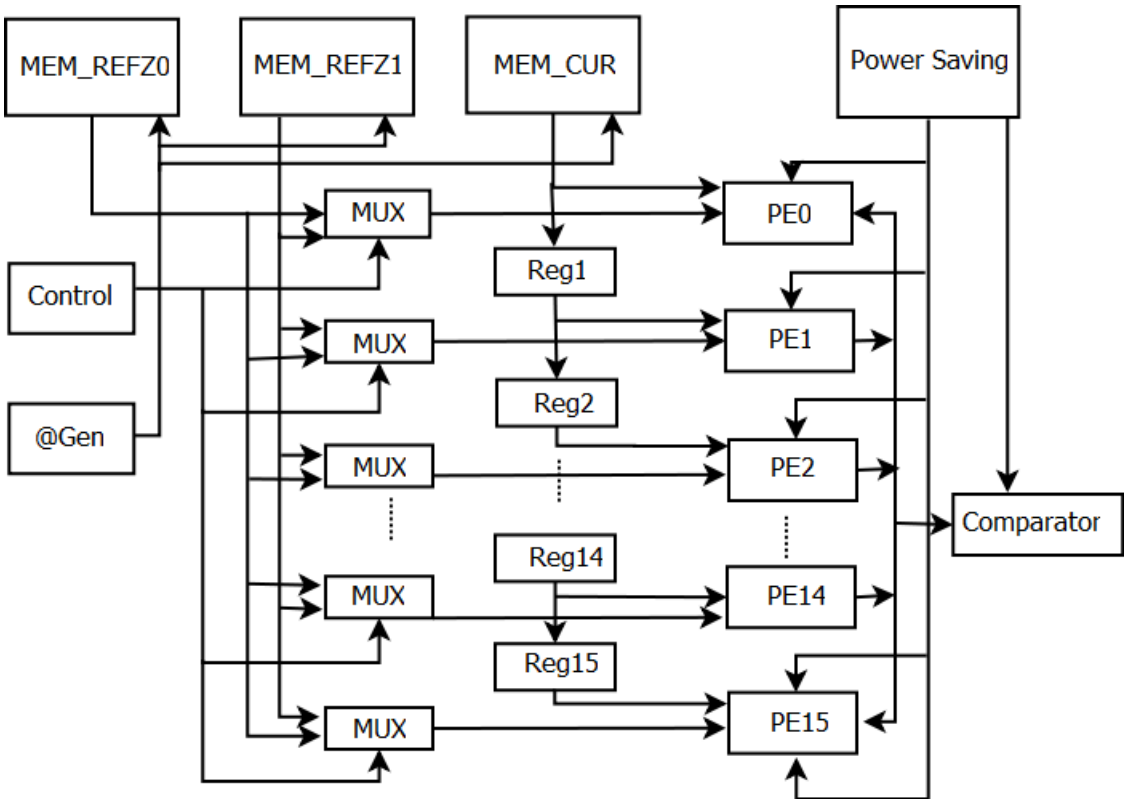

Figure 5. 15: Architecture de l'estimateur à faible coût de consommation

Fonctionnement du PE

En se basant sur l'équation du SAD, le PE doit effectuer la somme des valeurs absolues des différences entre un MB de l'image courante et un MB appartenant à une fenêtre de recherche de taille M×N pixels d'une image de référence. Les pixels correspondant à la luminance de l'image sont codés sur 8 bits. Le fonctionnement du PE est décrit par l'organigramme de la figure 5.16.

Au début du processus, un choix du MB qui servira comme origine est effectué, habituellement c'est le MB du centre. Ensuite la lecture du MB de référence à partir de la mémoire spécifique doit être effectuée. Le chargement des deux MBs se fait en parallèle vu l'utilisation de la communication par

mémoire distribuée. Après avoir lu les données, le calcul du SAD aura lieu jusqu'à la fin du parcours de la fenêtre de recherche.

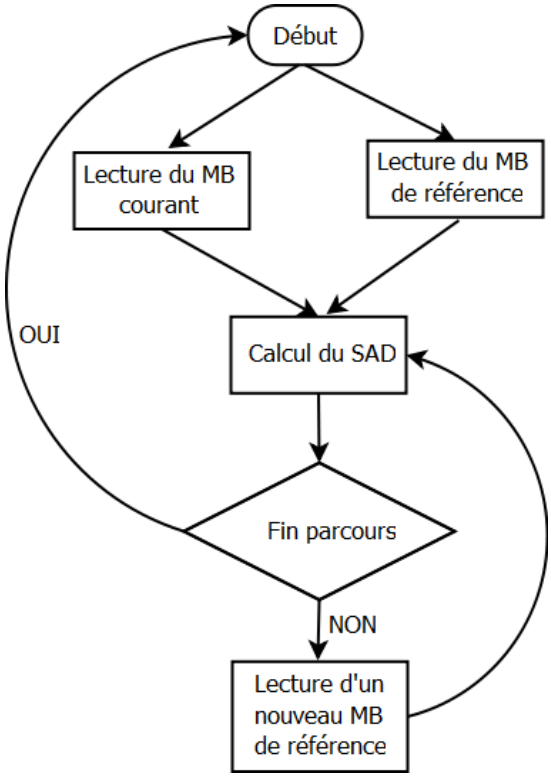

Figure 5. 16: Principe de fonction du PE

Module d'adressage

Concernant le générateur d'adresse, qui permet la lecture correcte des pixels suivant le flux indiqué sur le tableau 5. 2, il est divisé en deux sous-composants. Le premier sous-composant assure l'accès à la mémoire contenant les pixels du MB courant, tandis que le second est destiné à la lecture de la mémoire du MB de référence. Son architecture interne est basée sur des compteurs.

Le module de contrôle

Il contrôle le bon fonctionnement de l'estimateur en assurant l'inspection attentive de la lecture des pixels dans l'ordre défini par le tableau 5.2. Il permet également la sélection des pixels, de la fenêtre de recherche, qui doivent passer à travers les PEs.

Le comparateur

Il est chargé de la détermination du SAD ayant la valeur la plus basse et de sa mémorisation ainsi que des informations du vecteur de mouvement. Il est composé par un encodeur 16 :4, un multiplexeur 16 :1, des registres et un simple comparateur de bits. La figure 5.17 illustre son architecture.

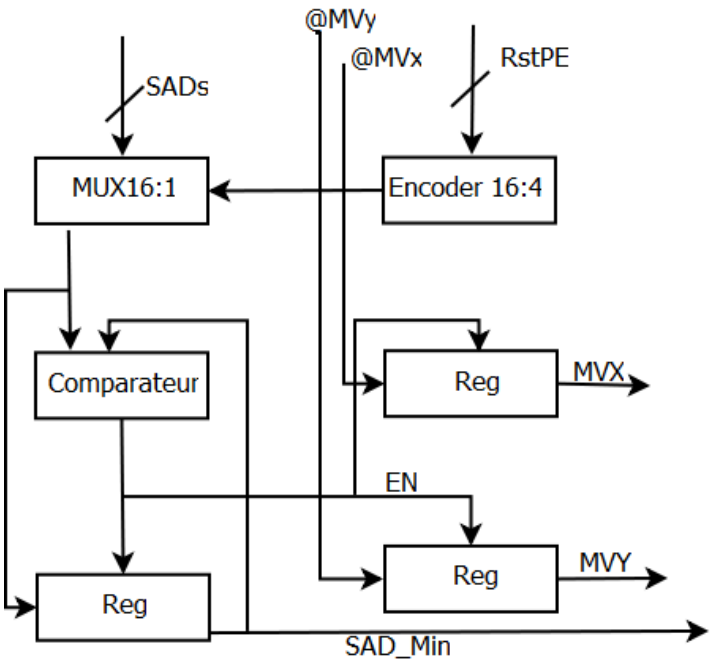

Figure 5. 17: Architecture du comparateur

Les signaux RstPE indiquent qu'un SAD est prêt pour être comparé avec d'autres valeurs qui sont déjà calculées. L'encodeur transmet le code au multiplexeur 16:1. La sortie du multiplexeur 16:1 est une valeur de SAD à 16 bits qui correspond au PE qui a reçu l'information de mise à zéro (RstPE). Le SAD du PE convenable est sélectionné et transféré vers le comparateur à 16 bits. Si celui-ci est inférieur au SAD déjà stocké dans le registre Reg, alors ce dernier sera activé par le signal EN et le nouveau SAD sera sauvegardé dans ce registre. De plus, pour démarrer le processus, il faut comparer la première valeur du SAD avec la valeur FFFF pour que ce premier SAD soit retenu et comparé avec le SAD qui vient de suite. De même, si la comparaison est valide, les registres de vecteurs de mouvement (MVX et MVY) rechargent le nouveau vecteur de mouvement.

Contrôle de consommation

Cette section explique comment, en se basant sur une observation, on a pu mettre en veille les PEs qui ne sont pas en cycle de traitement. Nous rappelons que la puissance dynamique est due à l'activité continue des portes logiques. D'après la table de flot de pixels (Figure 5.2), on constate que les PE ne fonctionnent pas simultanément tout le temps. Pendant les 16 premiers cycles d'horloge, on a uniquement le PE0 qui fonctionne pendant tout l'intervalle de temps. Par contre, les autres ne commencent à fonctionner qu'après un nombre bien défini de cycles. Le PE2 commence son traitement à t+2, le PE3 à t+3, etc. Le PE15 ne fonctionne qu'à la fin de ce premier cycle. Donc, notre idée est de concevoir un module qui permet de mettre en veille les PEs qui ne sont pas en cours de traitement. L'horloge de chaque PE ne sera activée qu'à l'instant où ce dernier commence à faire son calcul, que ce soit pendant les premiers cycles d'horloge ou les derniers cycles. L'organigramme de la figure 5.18 présente les différentes étapes à suivre pour obtenir un module capable de stopper les PE pendant un temps bien défini sans influer sur le bon fonctionnement de l'estimateur.

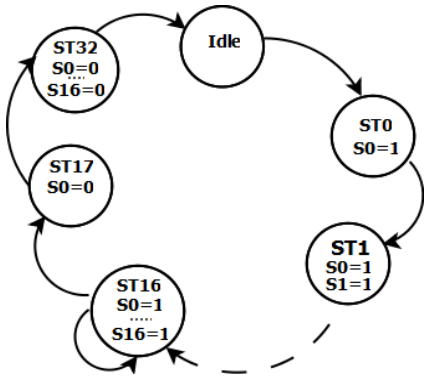

Figure 5. 18: Organigramme pour le contrôle de la consommation

Implémentation de l'estimateur de mouvement sur un ASIC

Les deux versions de l'estimateur, synchrone et GALS, ont été conçues par Synopsys et Cadence en utilisant la technologie CMOS 130 nm. La fréquence de fonctionnement maximale obtenue pour les deux versions est de l'ordre de 264 Mhz. Les résultats de placement/routage des deux circuits sont donnés par la figure 5.19. Les résultats de conception en termes de nombres de portes logiques, surface en silicium et puissance totale consommée par les deux estimateurs sont donnés par la table 5.3. L'étude des performances effectuée pour les deux versions (table 5.4) montre que l'estimateur de mouvement GALS nécessite une surface légèrement supérieure à celle de l'estimateur synchrone (augmentation de l'ordre de 3,76%). Cette étude montre aussi que la consommation de l'estimateur de mouvement intégrant le module du contrôle de consommation offre un gain de 37,18% par rapport à la puissance consommée par l'architecture de base de l'estimateur.

Table 5.3: Résultats de conception de deux versions de l'estimateur de mouvement

Estimateur	Nombre de portes logiques	Surface totale (mm^2)	Puissance totale (mW)
Synchrone	13408	0,186	44,21
GALS	14472	0,193	33,66

Table 5.4**:** Comparaison des deux versions de l'estimateur en termes de surface et consommation

$(100-\frac{SurfaceGALS}{SurfaceSynchrone}\times 100)\%$	-3,76%
$(100-\frac{P\_DynamiqueGALS}{P\_DynamiqueSynchrone}\times 100)\%$	+23,86%

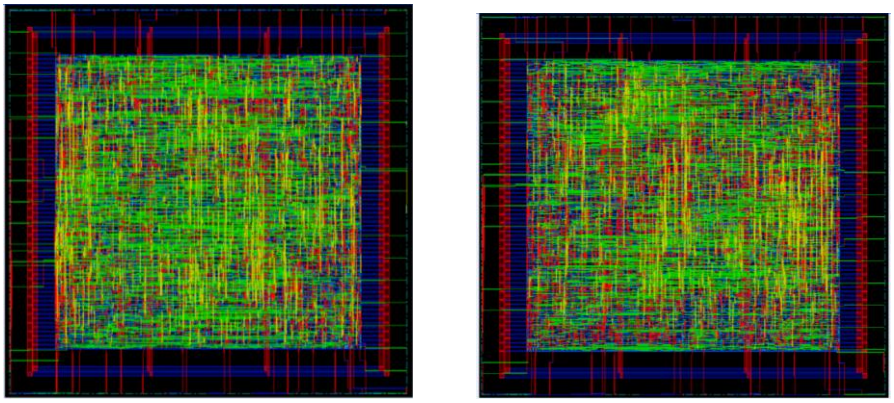

Figure 5. 19 : Résultats de placement/Routage de deux versions de l'estimateur: (Gauche) version synchrone et (Droite) version GALS.

Etude comparative

La contribution dans l'architecture de l'estimation de mouvement est évaluée en termes de performances d'implémentation matérielle [112]. Elle est comparée avec d'autres travaux de la littérature suivant des critères tels que la fréquence, la technologie et l'occupation en surface. Les éléments comparatifs sont regroupés dans le tableau 5.5.

Table5.5 : Etude comparative

	[113]	[114]	[115]	[116]	[112]
Nombre de PE	16×16	64	16	16×16	16
Taille de la fenêtre de recherche	16×16	16×16	32×32	32×32	30×30
Technologie	0.6um	0.35um	0.6um	0.18um	0.13um
Fréquence	72MHz	28MHz	60MHz	100Mhz	264Mhz
Surface	263K	33.2K	67K	154K	14.4K
Consommation	--	189mW	423mW	--	33.66mW

Notre architecture optimisée en termes de consommation énergétique fournit des performances acceptables en le comparant avec les architectures des travaux cités ci-dessus. Nous remarquons aussi que le module de contrôle ajouté n'influence pas la surface utilisée par cette architecture.

4.1.4. Conception d'un ME à faible coût de consommation pour des blocs à taille variable

L'estimation de mouvement à taille de blocs variable (VBSME, Variable Block Size Motion Estimation) se base sur la segmentation du MB de taille 16×16 en 16 MB de taille 4×4. A partir de cette segmentation, la norme H.264 définit les 7 modes (comme l'indique la figure 5.4), de l'inter prédiction, possibles. Nous observons sur la figure que dans les 7 modes il y a 41 sous- partitions, ce

qui signifie 41 vecteurs de mouvement à estimer, soit seize de 4×4, deux 16×8 et 8×16, quatre 8×8 et 4×8 et un 16×16. Cette variante d'estimation de mouvement permet de tenir compte des mouvements de tous les objets dans un MB. Cela assure la bonne prédiction mais en contrepartie une intensité de calcul énorme. Plusieurs techniques d'estimation de mouvement à taille de bloc variable ont été adoptées. La plus simple est la méthode exhaustive. Elle se résume dans la conception de l'estimation pour chaque mode. Cela conduit à la redondance en termes de ressources et n'exploite pas ce qui est déjà calculé. Néanmoins, cette solution n'est pas très utilisée dans la littérature.

La plupart des travaux de la littérature exploitent la redondance de calcul qui peut exister dans l'estimation de mouvement à blocs de taille variable [117]. Cela consiste à réutiliser le SAD de la brique de base, c'est-à-dire le SAD de 4×4, pour calculer les SADs des autres modes et également à exploiter complètement des PEs afin de générer les SADs de différents modes. Cette technique de réutilisation a un impact positif sur la performance des architectures proposées, en réduisant le nombre de calculs.

Les travaux proposés dans la littérature se basent sur cette idée et ils s'orientent vers la proposition des architectures du PE pour le rendre capable de réutiliser ce qui est déjà calculé. Ils ont tous gardé le même « Top Level » utilisé dans [109]. Le principe de la réutilisation de SAD se résume dans la figure 5.20.

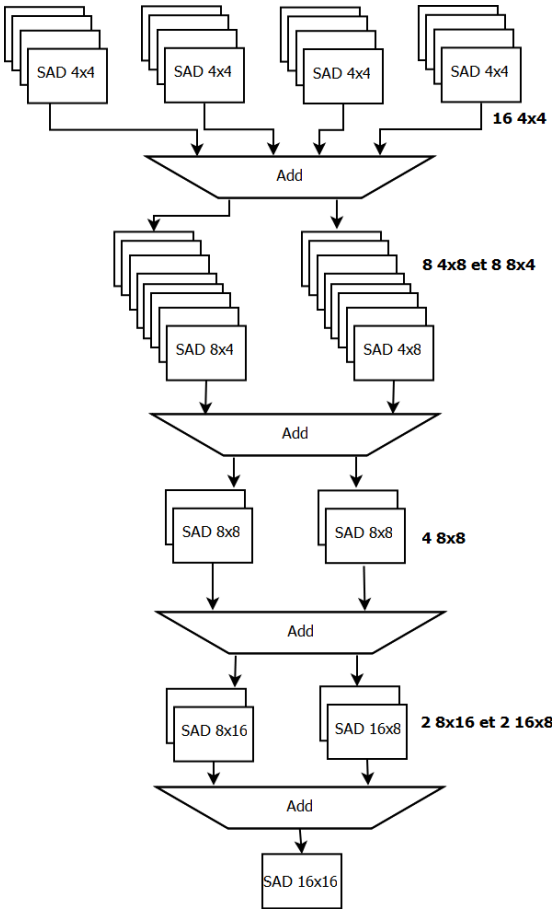

Figure 5. 20: Principe de réutilisation de SAD

Notre contribution consiste en premier lieu, à proposer un module permettant d'estimer le vecteur de mouvement d'un MB de taille précise à partir des vecteurs de mouvement déjà calculés, et en second lieu de favoriser le partage de ressources en termes de CE (Comparator Element). Avec l'estimation à taille des blocs variable, nous avons 112 SAD à comparer afin d'estimer 41 vecteurs de mouvement. Donc la question que nous avons posée, c'est de savoir combien de comparateurs il nous faut. Et comment faut-il arranger les groupes des SADs trouvés de façon à ne pas confondre les modes ?

Architecture de PE

L'architecture interne de PE, utilisée dans notre conception, est inspirée du [118]. Il favorise la réutilisation de SAD déjà calculés dans de précédents cycles d'horloge. Elle est composée par des multiplexeurs, des registres et des additionneurs. Le principe est simple. Il s'agit de calculer les SADs du bloc de plus petite taille et de déduire les SADs des blocs de grande taille. L'architecture de ce PE (Figure 5.21) est divisée en 3 parties : une partie (sélectionnée en bleu) pour calculer la valeur absolue des différences (AD, Absolute Difference), la deuxième partie est composée par des registres pour stoker les ADs déjà calculés. Une boucle de retour, décrite par la ligne de liaison entre un multiplexeur et l'additionneur (FA), assure la réutilisation de l'AD pour calculer la double somme. Après avoir calculé les SADs et les avoir stockés dans des registres, un processus de multiplexage aura lieu afin de les envoyer vers la troisième partie. Cette partie permet la réutilisation des SADs des MBs, de taille plus petite, déjà calculés, pour déduire des SADs des MBs de plus grande taille. Cette architecture est présentée par Yap et al.

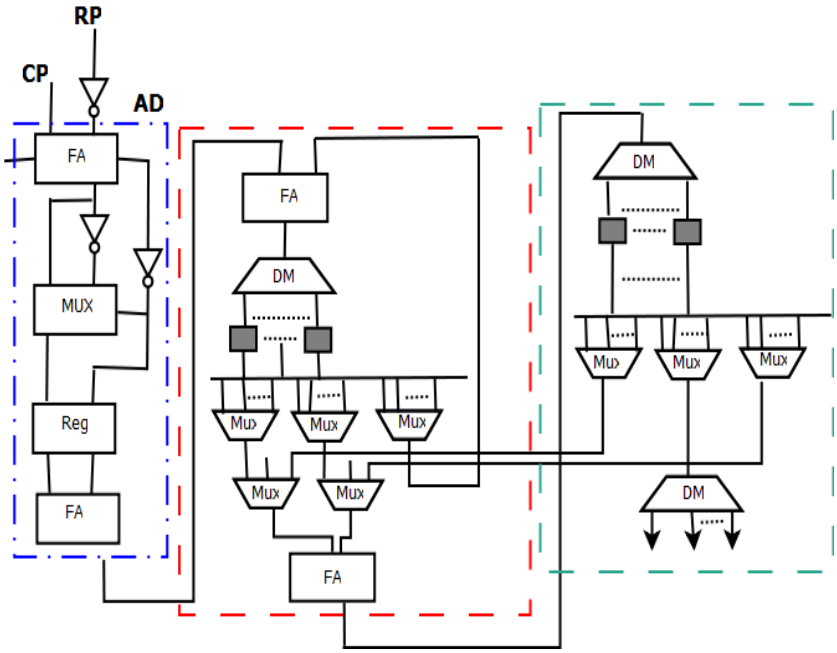

Figure 5. 21: Architecture interne de PE pour l'algorithme VBSME proposé par Yap et al

Les SADs, des MBs 4×4 stockés sont ajoutés pour calculer les SADs des MBs 4 × 8 et 8× 4. Les SADs des MBs 4 × 8 et 8× 4 sont combinés pour en déduire les SADs des MBs 8 × 8, et ainsi de suite. Cette méthode permet de réduire le nombre de calculs effectués pour le VBSME. Les SADs ainsi

trouvés sont envoyés à travers des buses vers un tableau de comparateurs afin de générer les 41 vecteurs de mouvement. Cela nous mène à la description de l'architecture du comparateur élémentaire (CE).

Architecture du CE

Pour concevoir l'architecture du CE nous nous basons sur les observations suivantes :

- Les SADs, du même mode, générés par les 16 PEs doivent partager le même ensemble de CE.
- Les SADs d'un bloc, calculés par les 16 PEs, sont obtenus avec un retard d'un cycle d'horloge. Par exemple le PE0 calcule la SAD du bloc 4×4_0 en 51 cycles d'horloge, alors que pour le même bloc, le PE1 la calcule en 52 cycles d'horloge etc.
- Le calcul des SADs du bloc 4×4 au sein du même PE se fait avec un retard de 4 cycles d'horloge. La même observation est appliquée pour les autres blocs, sauf que le cycle de retard change d'un mode à un autre.

De ces observations, nous pouvons adopter l'approche de partage des ressources. En effet, quatre comparateurs sont réservés pour effectuer la comparaison entre les SADs, du même bloc 4×4, générés par les 16 PES. Pour mieux comprendre le principe, nous prenons l'exemple du bloc 4×4_0.

La valeur du SAD du premier bloc est calculée après 51 cycles d'horloge par le PE0, en 52 cycles d'horloge par PE1, en 53 cycles d'horloge par PE2 et 54 cycle d'horloge par PE3. Par conséquent, certains PEs peuvent partager le même comparateur qui est le CE0. Le même principe est appliqué sur les autres PEs comme c'est indiqué sur le tableau5.6. En se basant sur la même philosophie, nous arrivons à utiliser 16 CEs partagés entre les PEs, en respectant les 7 modes. Les 16 CEs sont répartis respectivement comme suit : 4, 4, 2, 2, 2, 1et1 CEs pour les modes 4×4, 4×8, 8×4, 8×8, 8×16, 16×8 et 16×16. La figure 5.22 illustre l'architecture interne du comparateur.

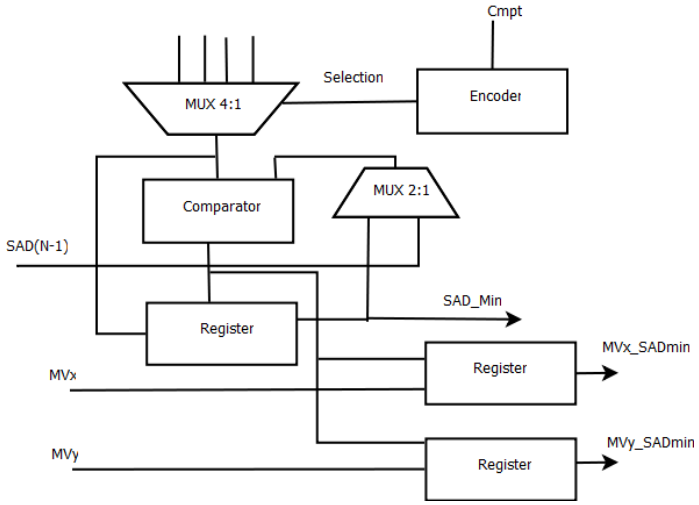

Figure 5. 22: Architecture interne d'un comparateur

L'encodeur prend en entrée le signal 'cmpt', qui est associé à un compteur d'horloge, afin de contrôler la diffusion des valeurs de SAD. Le rôle principal de ce codeur est de générer des adresses qui

correspondent au temps exact où les SADs doivent être comparés. Le signal SAD_{N-1} permet de transmettre la valeur du SAD minimal déjà trouvé par le comparateur précédent. Le tableau 5.6 présente le principe du partage de comparateur.

Tableau5.6 : Principe du partage de comparateur

Clk	CE0	CE1(t-55)	CE2(t-59)	CE3(t-63)	CE8(t-63)	CE9(t-64)	……..	CE15(t-261)
51	4×4_0PE0	4×4_0PE4	4×4_0PE8	4×4_0PE12	8×4_0PE0	8×4_0PE8	…..	16x16_0PE0
52	4×4_0PE1	4×4_0PE5	4×4_0PE9	4×4_0PE13	8×4_0PE1	8×4_0PE9	…..	16x16_0PE1
53	4×4_0PE2	4×4_0PE6	4×4_0PE10	4×4_0PE14	8×4_0PE2	8×4_0PE10	…..	16x16_0PE2
54	4×4_0PE3	4×4_0PE7	4×4_0PE11	4×4_0PE15	8×4_0PE3	8×4_0PE11	…..	16x16_0PE3
55	4×4_1PE0	4×4_1PE4	4×4_1PE8	4×4_1PE12	8×4_0PE4	8×4_0PE12	…..	16x16_0PE4
56	4×4_1PE1	4×4_1PE5	4×4_1PE9	4×4_1PE13	8×4_0PE5	8×4_0PE13	…..	16x16_0PE5
57	4×4_1PE2	4×4_1PE6	4×4_1PE10	4×4_1PE14	8×4_0PE6	8×4_0PE14	…..	16x16_0PE6
58	4×4_1PE3	4×4_1PE7	4×4_1PE11	4×4_1PE15	8×4_0PE7	8×4_0PE15	…..	16x16_0PE7
59	4×4_2PE0	4×4_2PE4	4×4_2PE8	4×4_2PE12	8×4_1PE0	8×4_1PE8	…..	16x16_1PE8
60	4×4_2PE1	4×4_2PE5	4×4_2PE9	4×4_2PE13	8×4_1PE1	8×4_1PE9	…..	16x16_1PE9
61	4×4_2PE2	4×4_2PE6	4×4_2PE10	4×4_2PE14	8×4_1PE2	8×4_1PE10	…..	16x16_1PE10
62	4×4_2PE3	4×4_2PE7	4×4_2PE11	4×4_2PE15	8×4_1PE3	8×4_1PE11	…..	16x16_1PE11
63	4×4_3PE0	4×4_3PE4	4×4_3PE8	4×4_3PE12	8×4_1PE4	8×4_1PE12	…..	16x16_1PE12
64	4×4_3PE1	4×4_3PE5	4×4_3PE9	4×4_3PE13	8×4_1PE5	8×4_1PE13	…..	16x16_1PE13
65	4×4_3PE2	4×4_3PE6	4×4_3PE10	4×4_3PE14	8×4_1PE6	8×4_1PE14	…..	16x16_1PE14
66	4×4_3PE3	4×4_3PE7	4×4_3PE11	4×4_3PE15	8×4_1PE7	8×4_1PE15	…..	16x16_1PE15

Par exemple, pendant le 51ème cycle d'horloge, le SAD du 4 × 4_0 est généré par PE0, et il est ensuite comparé, par CE0, avec les SADs générés par PE1, PE2, et PE3 pendant les 52, 53 et 54 cycles d'horloge. Le SAD minimal est donc généré par CE0 et ensuite envoyé au CE1 via le signal $SAD_{(N-1)}$. Pendant le 55ème cycle d'horloge, le SAD du 4 × 4_0 est calculé par PE4 et le PE0 génère le SAD du 4 × 4_1. Le CE1 se charge par le SAD minimal produit par CE0. Il compare cette valeur avec celle générée par PE4, PE5, PE6 et PE7, le nouveau SAD minimal est diffusé au CE2, et ainsi de suite. La même idée est appliquée aux autres modes.

Prédiction de vecteur de mouvement

En parallèle avec la prédiction du SAD, un accélérateur matériel est conçu dans l'objectif d'estimer le vecteur de mouvement (MV, Motion Vector) d'un bloc à partir du vecteur de mouvement d'un bloc de plus petite taille. Selon [119], la corrélation spatio-temporelle peut être adoptée pour estimer le MV du mode actuel en utilisant le vecteur de mouvement déjà calculé. Par conséquent, cette méthode peut être prise afin de réduire le calcul. Le MV correspondant à la Min_SAD des blocs 4 × 4 est réutilisé pour calculer le MV de Min_SAD des blocs 8 × 4. Pendant le 66ème cycle d'horloge, le SAD minimal du bloc 4 × 4_0 et son MV sont générés et stockés. Parallèlement, le comparateur des blocs de taille 8 × 4 commence le processus de comparaison et le SAD minimal est généré durant le 71ème cycle d'horloge. En effet, le MV correspondant est calculé comme la moyenne de deux MVs des blocs 4 × 4_0 et 4 × 4_1. Le même principe est appliqué sur les autres blocs. L'architecture de cet accélérateur peut être décrite par l'algorithme 5.3.

Algorithme 5.3
If $SAD_{(4\times4 \text{ or } 8\times8)} < min\_SAD_{(4\times4 \text{ or } 8\times8)}$ **then**
$Min\_SAD = SAD_{(4\times4 \text{ or } 8\times8)}$
$PMV_{(4\times8 \text{ or } 8\times16)} = 1/2\ (MV_{BLKi0} + MV_{BLKi1}) / i\epsilon\{0,1\}$
$PMV_{(8\times4 \text{ or } 8\times16)} = 1/2\ (MV_{BLK0i} + MV_{BLKi1}) / i\epsilon\{0,1\}$
$PMV_{(8\times8 \text{ or } 16\times16)} = 1/4 \sum(MV_{BLK\_Hi} + MV_{BLK\_Vi})$
End if;
Else
Return (Min_SAD, $MV_{(4\times8) \text{ or } (8\times16)}$, $MV_{(8\times4) \text{ or } (16\times8)}$, $MV_{(8\times8) \text{ or } (16\times16)}$)

Où MV_{BLK_Hi} et MV_{BLK_Vi} représentent respectivement, les blocs 8×4 ou 16×8 horizontalement et 4×8 ou 8×16 verticalement.

Contrôle distribué pour réduire la consommation

La consommation d'énergie dans le ME peut atteindre 70% de la consommation d'énergie globale du H.264. Par conséquent, il est nécessaire d'intégrer un système de contrôle afin de réduire la consommation. La technique « Gating clock » est parmi les techniques qui sont utilisées pour réduire la consommation d'énergie. Le modèle de contrôle de consommation proposé dans cette partie est fondé sur les machines à états finis (FSM, Finit State Machine). L'objectif de ce modèle est de réduire l'activité continue de cadence d'horloge par la mise en place de portes logiques permettant de mettre en veille le composant. Nous remarquons, d'après les deux tableaux 5.2 et 5.6, que les PE et les CE ne sont pas en fonctionnement durant tous les cycles d'horloge. Par conséquent, nous intégrons des FSMs dans l'estimateur, en profitant du flux de fonctionnement de chaque tableau. Notre proposition est basée sur deux observations:

- Le flux de données de pixels est en pipeline. Par conséquent, stopper les PES qui ne sont pas en cours de traitement s'avère nécessaire afin de réduire l'activité continue de cadence d'horloge.
- Les comparateurs sont inactifs pendant 51 cycles d'horloge pour chaque MB de l'image courante qui devrait être traité. Par ailleurs, le fonctionnement des CEs est en pipeline, ce qui rend possible de stopper l'horloge dans un ordre bien déterminé.

La figure 5.23 illustre le modèle proposé pour la réduction de la consommation. Les deux machines à états finis présentés dans cette figure décrivent les transitions entre les états permettant de contrôler l'activité de cadence d'horloge. Les transitions sont conditionnées par un signal de commande qui est associé à un signal d'horloge. Le FSM qui contrôle les CEs contient trois états, «Idle», « Nactive », et « Active ». Le FSM intercepte le signal de contrôle qui indique si la comparaison doit être "active" ou "Nactive". Ensuite, selon la valeur du signal de commande, il y aura une transition vers un de ces deux états. Si l'état est "Nactive" le comparateur est mis en pause par le signal "s" indiquant de stopper l'élément convenable.

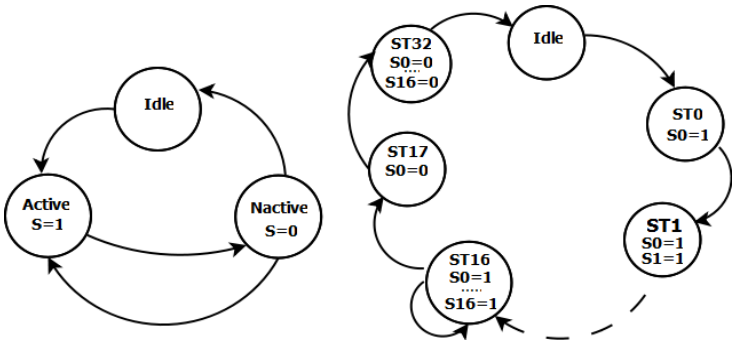

Figure 5. 23: Les machines d'états contrôlant la consommation énergétique

Suite à la deuxième observation, nous remarquons que pendant les 16 premiers cycles d'horloge seul le PE0 fait du calcul, de t = 0 jusqu'à t = 15. Les autres PEs commencent le traitement après un certain nombre de cycles d'horloge. Par exemple, le PE2 commence le traitement à l'instant t = 2, le PE3 à t =3, et ainsi de suite. De plus, nous remarquons que PE0 termine le traitement pendant le $255^{ème}$ cycle d'horloge. Un cycle d'horloge après, le PE1 termine son calcul et ainsi de suite.

L'architecture associée à ce modèle de contrôle de consommation est développée dans un langage de description matérielle. Il est simple à concevoir et pas cher en termes de puissance et surface.

Validation expérimentale et étude comparative

L'architecture proposée pour l'estimateur de mouvement à taille de blocs variable est décrite en VHDL, simulée et synthétisée respectivement par ModelSim et ISE 12.4. Cette conception a été prototypée en utilisant la technologie FPGA, précisément le composant Virtex-6. Le tableau 5.7 présente les résultats de synthèse en termes de surface, de consommation et de fréquence maximale. La puissance est mesurée en utilisant le « Digital Power Supply » de Texas Instruments. La fréquence maximale de fonctionnement est d'environ 123 MHz et la valeur de la consommation d'énergie est de 260 mW. La puissance de l'architecture de base, c'est à dire sans module de contrôle, est d'environ 293mW, donc le gain obtenu est de 11%.

Pour évaluer la performance, nous introduisons la notion de débit «S» définie dans [120].

$$S = M/T \qquad \text{Eq 5.3}$$

Soit M le nombre de sous-blocs, selon la norme H.264, M = 41 pour le VBSME. Soit T la latence de l'architecture matérielle FSBMA. Elle est définie comme le nombre de cycles d'horloge nécessaires pour identifier les 41 MVs. Elle est définie par ($T= 4p^2$). Le tableau 5.8 montre une étude comparative entre différents travaux de la littérature en termes de bande passante, de nombre des PEs utilisés et de taille de blocs supportée par l'architecture proposée. Nous observons que notre architecture est la meilleure en termes de latence et de bande passante.

Tableau5.7 : résultats de synthèse

Slice Logic Utilization	Design metrics
Number of Slice Register	2%
Number of Slice LUTs	7%
Number of IOB	14%
Power consumption	260mW
Frequency	123 Mhz

Tableau5.9 : Etude comparative

Architecture	[118]	[121]	[122]	[123]	Notre travail
Num. of PEs	16	96	256	256	16
Latency(T)	4096	240	1129	256	196
Throughput	41/4096	41/240	41/1129	41/256	41/196
Block size supported	16×16 to 4×4	16×16 To 4×4	16×16 to 4×4	16×16 to 4×4	16×16 to 4×4

4.2. Compensation de mouvement

En se basant sur l'information du vecteur de mouvement, le module de compensation de mouvement (Motion Compensation, MC) récupère à partir de l'image de référence un MB pointé par la partie entière du MV. Cela réduit considérablement le nombre de calculs à effectuer, car cela réduit la quantité d'informations à compresser, par l'extraction des informations redondantes dans l'image courante par rapport à l'image de référence qui est déjà codée. L'architecture proposée dans cette partie se base essentiellement sur l'algorithme 5.2, la figure 5.24 illustre le schéma en bloc du MC.

Plusieurs architectures ont été proposées ces dernières années. Elles peuvent être classées suivant deux critères majeurs: la réduction du trafic de la mémoire et l'interpolation des pixels.

Dans [124] on définit une architecture permettant la réduction des accès redondants à la mémoire par l'intégration des registres internes. Ces registres vont être partagés par des MBs voisins de taille 4×4.

La séparation entre l'interpolation de la luminance et la chrominance est utilisée dans [125]. Cette séparation mène à la réutilisation des ressources telles que les registres, ce qui a un impact positif sur les performances de l'architecture proposée.

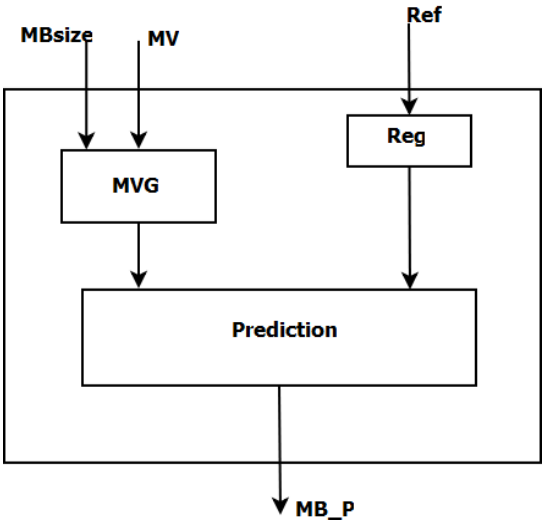

Figure 5.24 : Architecture interne de MC

Dans cette architecture, le composant MVG permet la lecture de MV et de son emplacement dans l'image de référence afin d'être transmis à la prédiction. L'image de référence stockée dans une mémoire externe est lue pixel par pixel. Le registre « reg » a comme rôle de regrouper les pixels sous une forme matricielle et de l'envoyer au module de prédiction. A partir des informations reçues, le module de prédiction place le vecteur de mouvement dans la région adéquate de l'image de référence et génère à la fin le MB prédit.

Les résultats issus de la synthèse RTL sur FPGA Virtex5 du module de la compensation de mouvement sont les suivants :

- Slice: (129/149760) 1%
- IOB: (556/680) 81%
- Fréquence maximale: 400MHZ

4.3. Conception RTL de la DCT et de la quantification

Dans cette section, nous détaillons les architectures conçues pour la DCT et la quantification afin de les valider par une implémentation expérimentale.

La DCT

Le module de la DCT est implémenté pour assurer le calcul de 64 coefficients simultanément en exploitant le parallélisme de tâches. Chaque ensemble de 8 coefficients est calculé par un bloc. Le schéma en bloc du circuit assurant la DCT est composé de 8 modules DCT-1D, de 8 multiplexeurs à 16 entrées et 8 sorties pour faire la sélection entre les données provenant de l'extérieur du module et les données provenant des sorties de chaque DCT-1D.
Un composant DCT-1D sera appliqué sur un vecteur de 8 pixels, dont chaque pixel est codé sur 8 bits. Le schéma bloc d'un DCT-1D est composé de trois éléments de calculs implémentés pour exécuter les trois étapes de l'algorithme de l'approximation entière [126]. La figure 5.25 représente le schéma bloc du système.

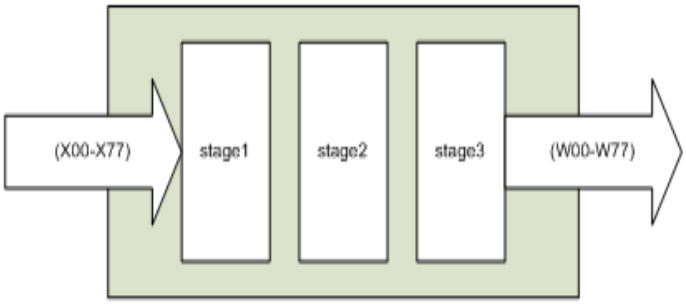

Figure 5. 25: Schéma bloc de la DCT-1D

Les résultats issus de la synthèse RTL par ISE10.1 sur FPGA Virtex5 XC5VLX50T du module de la DCTsont les suivants :

- Flip flop: (5124/28800) 17%
- LUTs: (5094/28800) 17%
- slices: (5094/28800) 17%
- Fréquence maximale: 540MHZ

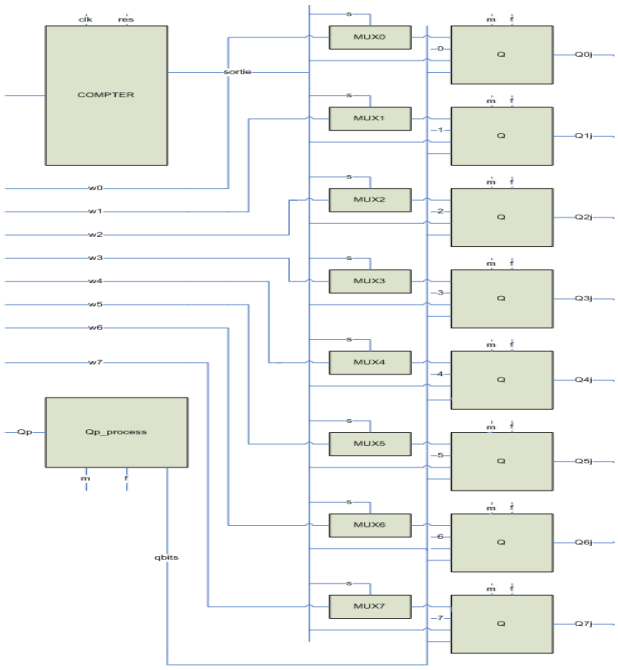

Figure 5.26 : Architecture du module de quantification

La quantification

Le module de quantification permet de quantifier, simultanément, 8 coefficients, ce qui fait que toute une ligne est traitée durant un cycle d'horloge. Chaque coefficient est calculé par un bloc de quantification. Donc le module de quantification contient 8 blocs de quantification. Le schéma bloc du module de quantification (Figure 5.26) comprend un bloc de « QP-processing », 8 blocs de multiplexeurs, un compteur qui permet la sélection des données qui vont être traitées.

Les résultats issus de la synthèse RTL par ISE 10.1 sur FPGA Virtex5 XC5VLX50T du module de quantification sont les suivants :

- Flip flop: (5/28800) 1%
- LUTs: (2741/28800) 9%
- slices: (2741/28800) 9%
- Fréquence maximale: 400 Mhz

5. Bilan sur la conception matérielle

Nous avons décrit dans cette partie de la thèse notre effort d'optimisation architecturale des tâches élémentaires du codeur H.264. Cela représente la bibliothèque d'IP exprimées dans un langage HDL, qui vont servir par la suite comme briques de base de la modélisation à haut niveau afin de mapper ses tâches sur un NoC. Nous avons choisi de faire des interventions dans ces modules du fait qu'ils sont les plus critiques, dans cette norme de codage vidéo, en termes de complexité architecturale et de consommation énergétique. Nous avons pu extraire du parallélisme local qui existe dans chaque module en se basant sur les modèles MARTE, déjà créés auparavant, et l'expression des répétitions.

Les tâches élémentaires que nous avons conçues sont l'estimation et la compensation de mouvement, la DCT, la quantification. Les autres tâches, le codage entropique, la DCT et la quantification inverse, sont des codes open source.

6. Conclusions

Dans ce chapitre, nous avons exposé l'application adoptée dans cette thèse. Nous avons commencé par une présentation du codeur H.264. Dans cette présentation, l'algorithme de codage vidéo suivant cette norme est défini en détails, en partant d'une vue globale de l'algorithme jusqu'à la convergence vers le fonctionnement de chaque tâche élémentaire. Nous sommes ensuite passés à la modélisation à haut niveau d'abstraction de ce codeur. Dans la modélisation, le parallélisme de données est exprimé par l'optimisation de boucle. Les modèles définis en MARTE de la chaine de codage P-Frame suivant la norme H.264 peuvent être réutilisés chaque fois qu'il y aura une évolution dans la technique de compression suivant cette norme de codage. Donc, l'utilisateur peut intervenir au niveau de la conception d'IP pour définir son choix qui est piloté par les besoins du domaine d'application.

Dans la seconde partie de ce chapitre, nous avons proposé des améliorations architecturales au niveau des tâches élémentaires du codeur H.264. L'amélioration se base précisément sur la réduction de la consommation énergétique, l'expression du parallélisme, et le partage des ressources. Les IPs développées vont servir ensuite dans la définition de la bibliothèque de notre système.

Dans le chapitre suivant nous passons à l'association de l'application et l'architecture suivant le flot de Co-conception du GASPARD.

Chapitre 6

Co-modélisation en GASPARD2 par association de l'application et l'architecture

1. Introduction

Ce chapitre permet au lecteur la prise en main de la conception des systèmes sur puce à l'aide de l'outil Gaspard. Après les phases préliminaires de la Co-modélisation suivant le diagramme en Y du Gaspard, à savoir la modélisation application/architecture en MARTE et la description en VHDL des différentes tâches élémentaires, une phase de modélisation de l'association et d'intégration s'avère nécessaire pour suivre tout le flot. Dans un premier temps, nous avons détaillé la modélisation de l'architecture du NoC en MARTE, puis la description architecturale du routeur. En second lieu, nous avons modélisé l'application H.264 au niveau système afin de proposer la description RTL des tâches élémentaires. Dans ce chapitre, nous abordons la phase de modélisation de l'association et le déploiement des IPs afin d'intégrer le système dans l'environnement GASPARD2. Dans l'association, une étude a été menée dans l'objectif de définir une fonction coût permettant la bonne distribution des IPs sur les routeurs. Nous procédons ensuite à une phase de déploiement assurant la spécification des composants modélisés au niveau système.

2. IDM de la modélisation à la génération

La conception d'un SoC souffre de la pénurie des outils offrant la possibilité de suivre la technologie des semi-conducteurs et de la fabrication. Cela induit à un gap de productivité (figure 6.1) assez important, ce qui laisse les méthodes de conception errer dans l'instabilité. En effet les environnements couramment utilisés n'arrivent pas à suivre l'évolution incessante de la technologie et la complexité grandissante des applications.

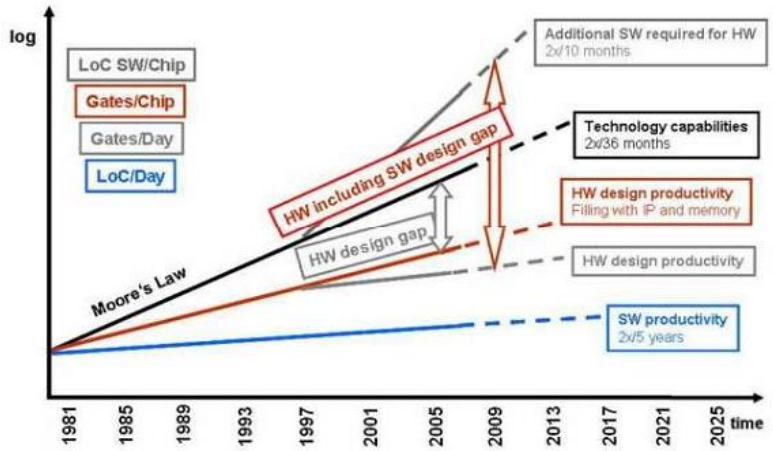

Figure 6. 1: L'écart de productivité entre le matériel et le logiciel [131]

D'après la figure6.1 qui montre une étude publiée par l'ITRS [131], le taux d'intégration des SoCs double tous les 36 mois, la productivité du logiciel est multipliée par deux tous les cinq ans et la demande du logiciel double tous les dix mois. Cela approfondit l'écart entre l'évolution de la technologie de fabrication des systèmes et la mise sur le marché des applications. Pour remédier à ces problèmes, une approche de modélisation à un niveau d'abstraction élevé basé sur la notion de modèle se révèle indispensable. Cette approche favorise la réutilisation des modèles et permet de s'affranchir de tout détail d'implémentation. Donc la modélisation à un haut niveau d'abstraction vient pallier les problèmes rencontrés lors de l'implémentation. Elle permet d'étudier le système tout en s'éloignant des détails de mise en œuvre. Cependant, l'objectif est d'étudier le système à un haut niveau d'abstraction afin de faciliter sa réalisation physique, donc toute démarche d'augmentation de niveau d'abstraction doit être nécessairement accompagnée par une démarche de raffinement. L'IDM apparaît comme un espace de travail permettant l'augmentation du niveau d'abstraction et aussi le raffinement des modèles jusqu'à la génération de code. Dans une démarche IDM l'environnement Gaspard, comme c'est indiqué dans le cinquième paragraphe du deuxième chapitre, fournit un flot de Co-modélisation dont l'objectif est d'aller de la modélisation à haut niveau jusqu'à la génération de code. La modélisation de l'application, de l'architecture et de l'association, définissent les grandes lignes de la Co-modélisation suivant Gaspard.

3. Modélisation de l'association

Dans ce paragraphe, nous détaillons la notion d'association suivant Gaspard et le « mapping » souvent utilisé par les concepteurs, afin d'expliquer la différence entre les deux.

3.1. Le « mapping »

Cette étape consiste à affecter chaque IP de l'application, qui est le codeur H.264 dans notre cas, à une tuile de l'architecture. Cela peut se faire de plusieurs façons et suivant plusieurs critères imposés par l'application. Cette étape devient difficile lorsque le nombre des IPs est énorme. Il s'agit d'une difficulté liée au choix de l'emplacement d'un IP sur une tuile, pour k IPs et l tuiles il existe k^l choix d'emplacements possibles. Pour résoudre ce problème, plusieurs variantes ont été proposées dans la littérature. Elles sont de deux sortes : les méthodes exactes et les méthodes approchées [127].

Les méthodes exactes : elles assurent la résolution du problème dans un espace de solutions optimales

Les méthodes approchées : permettent de trouver une solution optimale ou proche de la solution optimale en un temps réduit.

Plusieurs algorithmes ont été proposés dans la littérature pour placer des IPs sur une architecture. Dans la référence [128], l'auteur a proposé une approche nommée PMAP (Physical Mapping) basée sur le « mapping » des IPs d'une application sur un réseau sur puce afin de réduire les coûts relatifs aux communications. En premier lieu, il commence par placer les IPs les plus communiquant sur des routeurs voisins. En second lieu, les autres IPs sont placés en fonction du placement déjà effectué pendant la première étape.

L'algorithme de placement en SPIRAL [129], décrit un placement des IPs de l'application sur les tuiles de l'architecture selon une forme en spirale. Elle commence par définir des fonctions coût pour assurer un placement optimal en termes de coût de communication.

Dans notre travail, nous adaptons l'algorithme de placement en spirale, pour affecter les IPs de l'application H.264 sur les routeurs de notre réseau sur puce. Pour ce faire, une phase de modélisation sous forme de graphe, de l'application et de l'architecture, est nécessaire afin de définir une fonction coût permettant un placement optimal. La figure 6.2 illustre le graphe orienté qui modélise les IPs de l'application H.264.

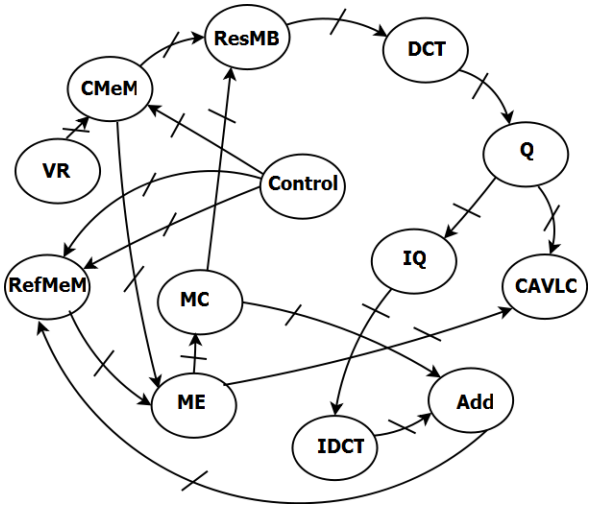

Figure 6. 2: Graphe d'application

Ce graphe d'application G (V, E) est un graphe orienté où chaque $v_i \in V$ correspond à un IP de l'application. Les sommets du graphe sont reliés entre eux par des liens indiquant la communication entre des IPs voisins. Sur chaque lien, un poids est affecté représentant la taille de paquet envoyé. De même l'architecture, qui représente l'infrastructure de communication dans notre cas, est modélisée à l'aide d'un graphe orienté.

Le graphe d'architecture du NoC, G (R, L) est un graphe orienté où chaque sommet r_i désigne un routeur de l'architecture du NoC, et chaque lien l_{ij} définit un lien de communication, physique entre r_i et r_j. La figure 6.3 illustre ce graphe.

Après avoir modélisé en utilisant les graphes orientés, le problème du mapping peut être décrit par la formule suivante :

La fonction asso ( ), qui consiste à affecter chaque IP de l'application à une et une seule tuile de l'architecture, est définie comme suit :

$$\text{Asso}() : \begin{cases} V \rightarrow R \\ V_i \rightarrow r_j \end{cases}$$

$$\forall\, vi \in V, \exists\, rj \in T : Asso(vi) = rj$$

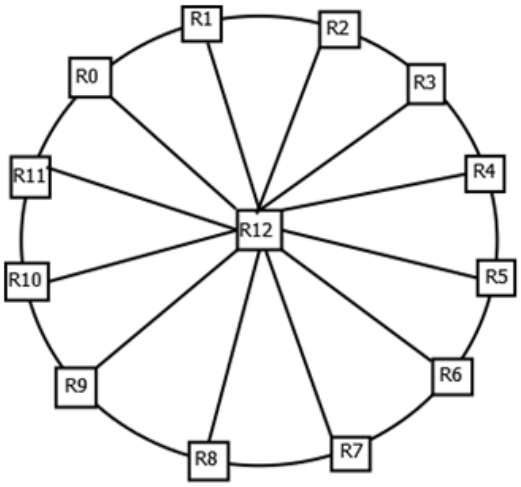

Figure 6. 3: Graphe d'architecture

Dans le graphe d'application, chaque communication est traitée comme un flux à chemin unique, représenté par d^k. La valeur Val (d^k) définit le coût total de la communication tel que :

$$d^k = Commij, \quad k = 1,2, \dots., \forall e_{ij} \in E$$

Où $Comm_{ij}$= la communication entre IP_i et IP_j.

Lorsque le placement est effectué, le but est de minimiser le coût de communications de l'application sur le réseau sur puce. En revanche, les IPs qui communiquent le plus entre elles doivent être placées sur des routeurs proches. La fonction de coût de communication est donnée par l'équation suivante:

$$CommCost = \sum_{k=1}^{E} vl(d^k) \times nbsauts(S(d^k), D(d^k))$$

Où nbsauts(a,b) désigne le nombre minimal de sauts entre les routeur a et b sur le réseau sur puce. Dans notre cas nbsauts=2.

3.2. Association suivant Gaspard

Le paquetage association permet l'expression du placement de l'application sur une architecture qui va être un support d'exécution. Il définit des concepts nécessaires au placement des IPs sur les tuiles de l'architecture. L'allocation et la distribution sont deux concepts de paquetage d'association qui sont chargés de l'affectation des tâches sur des processeurs.

Allocation : il s'agit d'allouer une tâche sur un élément d'exécution, qui est généralement un processeur. Cette opération est réalisée, suivant le profil Gaspard, en utilisant un stéréotype « TaskAllocation ». Elle est modélisée par un lien qui doit exister entre deux éléments ayant la même multiplicité.

Distribution : elle permet la distribution des éléments répétés sur la répétition d'autres éléments. Ceci veut dire qu'elle prend un élément de la tâche répétée et le place sur la répétition des processeurs. Le

stéréotype distribution est construit autour de deux Tilers ; Tiler source et Tiler destination. Le premier montre comment est construite la table des répétitions des tâches à l'aide des indices de répétitions, tandis que le Tiler destination exprime comment distribuer les tâches sur la répétition des éléments de la plateforme d'exécution. La figure 6.4 illustre les concepts de la distribution en se basant sur le profil MARTE.

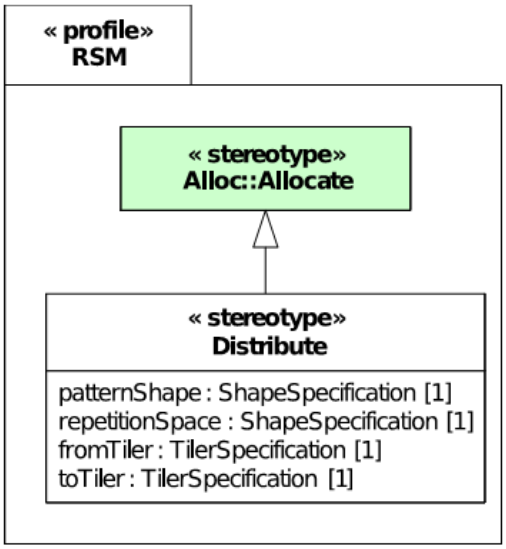

Figure 6. 4: La distribution [38]

Ce concept est généralement utilisé pour exprimer des répétitions sur des processeurs ou des matrices de données sur des mémoires. Le placement d'une répétition sur une unité de calcul est unique, c'est- à dire que chaque répétition est placée sur un unique processeur.

Dans notre cas, et contrairement à ce qu'on a l'habitude de faire sur Gaspard, il s'agit de mapper des IPs en VHDL sur des répétitions d'un routeur également décrit en VHDL. Le parallélisme de la tâche est exprimé à l'intérieur de l'IP. Les routeurs ne représentent pas des éléments d'exécution des IPs mais un élément assurant l'acheminement de l'information.

4. Association de l'application H.264 sur un NoC

Après avoir achevé la modélisation de l'architecture et de l'application, nous avons procédé à une phase d'association en utilisant les concepts reliés à cette opération. La figure 6.5 présente l'association ou le mapping que nous avons adopté dans cette thèse. La distribution des IPs sur notre réseau sur puce est effectuée en respectant la fonction coût déjà définie dans la section du 3 de ce même chapitre. Tout d'abord, la classe « Video application » est présentée et elle contient les tâches élémentaires ainsi que les éléments de contrôle avec les mémoires nécessaires. La seconde classe présente la répétition des routeurs périphériques et l'instance du routeur central de la topologie adoptée.

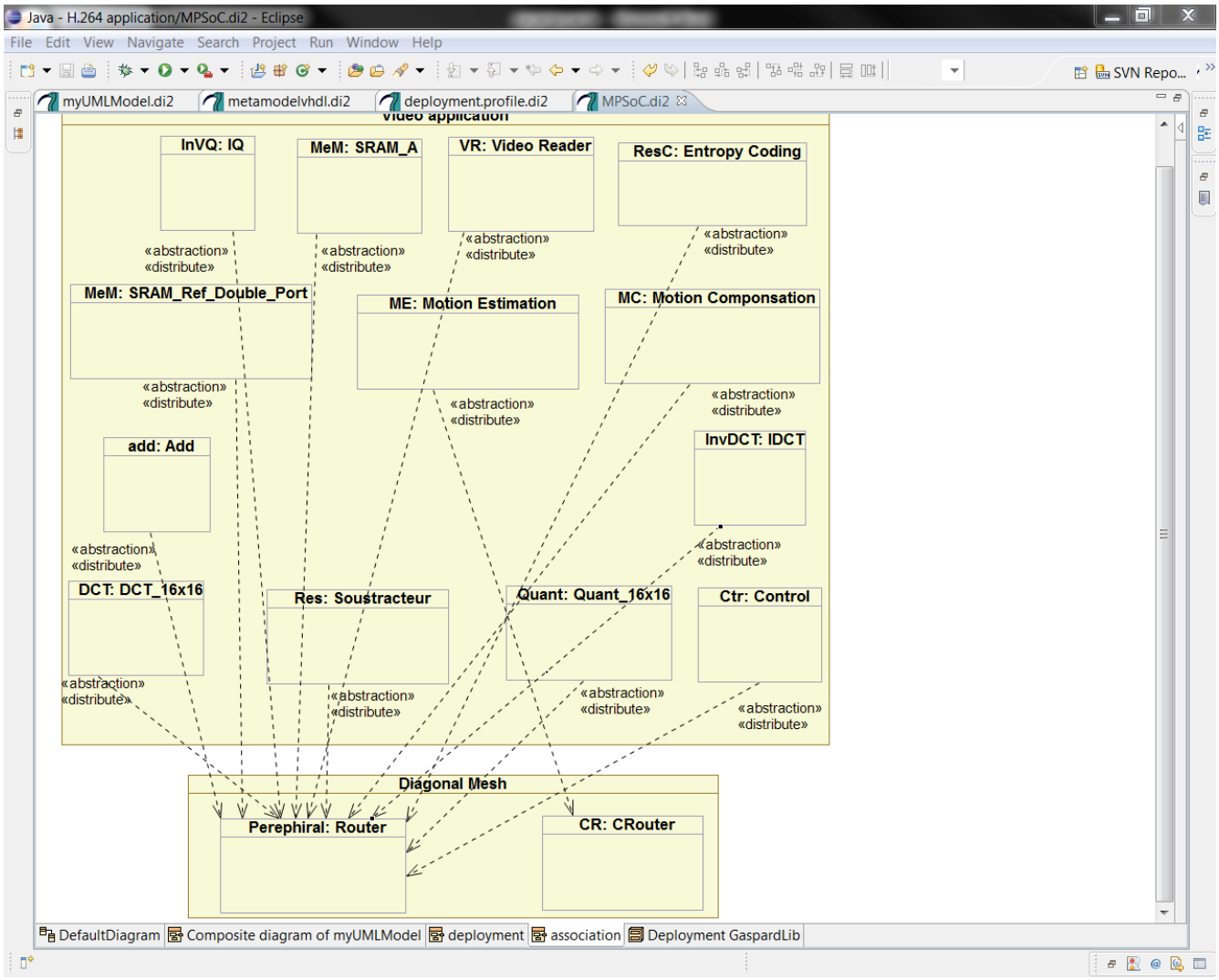

Figure 6. 5: association de l'application H.264 sur un NoC

Comme c'est déjà défini dans la section précédente, le stéréotype « Distribute » contient deux Tiler ; source et destination. Dans notre modélisation de l'association, nous avons utilisé ce stéréotype pour modéliser l'emplacement des IPs sur notre réseau sur puce. La figure 6.6 illustre un exemple de calculs nécessaires pour placer un IP sur un routeur.

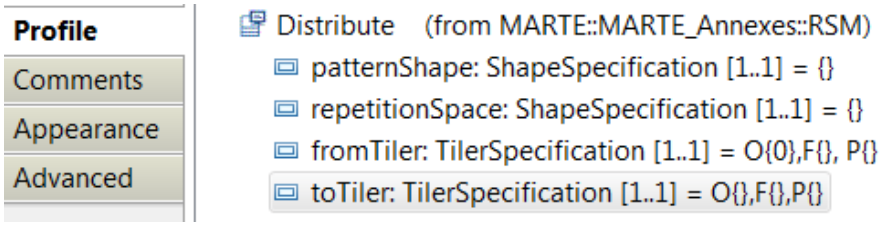

Figure 6. 6: Un extrait du calcul de « Distribute » de l'estimateur sur le routeur central

Dans notre cas, il ne s'agit pas d'exprimer les répétitions d'une tâche sur une répétition de processeur, il s'agit tout simplement d'allouer des IPs sur une répétition de routeur. Le routeur central est spécifié par la valeur du vecteur de l'origine qui est égale à zéro. Les matrices de pavage et d'ajustage sont vides du fait qu'il ne s'agit pas de placer des répétitions sur d'autres répétitions.

5. Méta-modèle de déploiement

La modélisation de l'association permet la modélisation de haut niveau de notre SoC, et les étapes abordées jusqu'à maintenant, à savoir la modélisation de l'architecture, celle de l'application et celle de l'association, sont indépendantes de toute plateforme d'exécution. La dépendance envers une technologie ou une plateforme cible est exprimée dans la phase de déploiement. Dans le déploiement, il s'agit de définir et de lier les tâches élémentaires de l'application et de l'architecture avec un IP intégré dans la bibliothèque.

La modélisation de haut niveau des tâches élémentaires de l'application ou de l'architecture (IP) a été définie dans Gaspard par la mise en place d'une bibliothèque appelée GaspardLib. Cette bibliothèque contient différents éléments permettant la génération automatique de code. Dans notre cas, elle contient le code VHDL du routeur, ainsi que le code des tâches de l'application H.264. Le langage VHDL définit pour nous une cible pour la compilation assurant le passage de notre SoC de l'état contemplatif à un état productif. Donc la phase de déploiement permet d'associer les composants élémentaires de haut niveau à des codes VHDL intégrés dans notre bibliothèque GaspardLib. Les liens reliant les composants modélisés en utilisant le profil MARTE vont être exprimés par les concepts fournis dans le méta model de déploiement. Les principaux avantages de ce méta modèle sont :

- Favoriser la réutilisation des modèles déjà effectués.
- Permettre au concepteur de faire des mises à jour indépendamment de la modélisation de haut niveau.
- Pouvoir intégrer des informations liées à l'estimation de performance et à la consommation énergétique [63].

Le méta modèle de déploiement se base sur des concepts. Nous détaillons par la suite ce que nous avons utilisé pour achever cette phase.

5.1. Le concept d'Implémentation

Il définit une méta-classe assurant la liaison entre le code généré par le système et le code des IPs de la bibliothèque. Il permet de rassembler les différentes implémentations (dans un langage cible) du même composant dans un seul ensemble. Il représente un IP pour une technologie cible, qui peut être orienté vers la simulation par l'utilisation de SystemC [63] ou vers la synthèse en utilisant le langage VHDL. Par le biais d'un lien nommé Implementation, un composant peut être implémenté dans plusieurs espaces de travail. Cela a comme avantage de fournir des modèles indépendants de la plateforme d'exécution.

5.2. Le concept CodeFile

Après avoir appliqué le concept Implementation, nous avons besoin de compiler les fichiers sources de chaque code utilisé. Le concept CodeFile assure la précision des informations concernant ces fichiers sources. Ce concept, illustré sur la figure 6.7, est introduit afin de représenter un fichier en spécifiant des attributs bien déterminés. Parmi ces attributs, il y a sourceFilepath, qui définit le chemin d'accès au fichier source.

Notons qu'il n'y a pas une relation directe entre les deux concepts, Implementation et CodeFile parce qu'une Implementation peut avoir plusieurs CodeFile. De même, un CodeFile peut être partagé par plusieurs implementations.

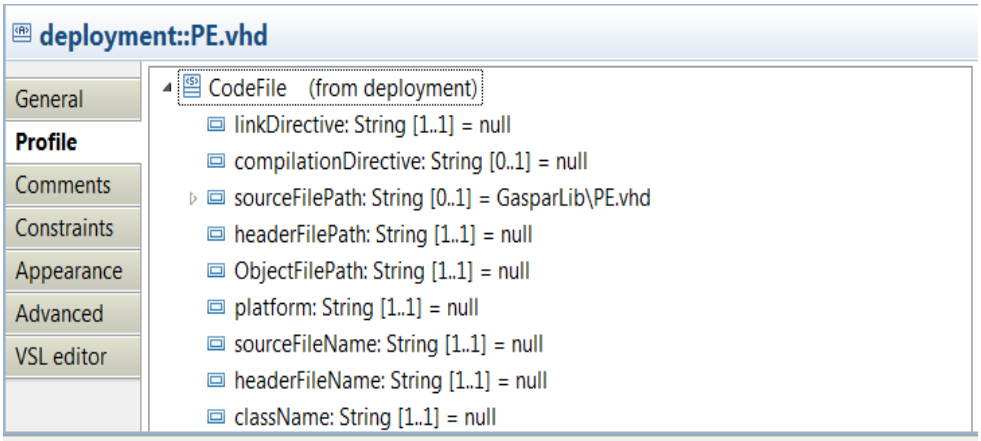

Figure 6. 7: Le concept CodeFile

6. Déploiement des IPs

Afin d'assurer le passage vers un état productif de notre modèle du SoC qui dédié au traitement vidéo, nous avons défini notre langage d'exécution qui est le VHDL. Nous prenons comme objectif la génération du code automatique pour la synthèse RTL du système. Donc, comme nous l'avons expliqué dans la section précédente, la phase de déploiement est nécessaire. Les IPs intégrés dans la bibliothèque vont servir comme briques de base pour la génération du code du système. A chaque modèle d'une tâche élémentaire du codeur H.264, ou de notre réseau sur puce, correspond un code VHDL disponible dans la bibliothèque. Les liens permettant la représentation de la modélisation de haut niveau de cette phase sont exprimés à l'aide des concepts fournis par ce méta modèle. Commençons par l'application : La figure 6.8 illustre une partie du déploiement des tâches élémentaires du codeur H.264 afin de permettre la génération du code du système.

Nous prenons un exemple pour expliquer concrètement la démarche à suivre dans cette phase de déploiement. *Motion Compensation* est déployé sur l'implémentation abstraite *Virtual IP MotionCompensation* qui contient une implémentation d'IP de compensation de mouvement. Le CodeFile de cette implémentation contient une seule implémentation nommée MC.vhd. De la même façon, nous avons procédé au déploiement de notre réseau sur puce. La figure 6.9 illustre le déploiement du NoC.

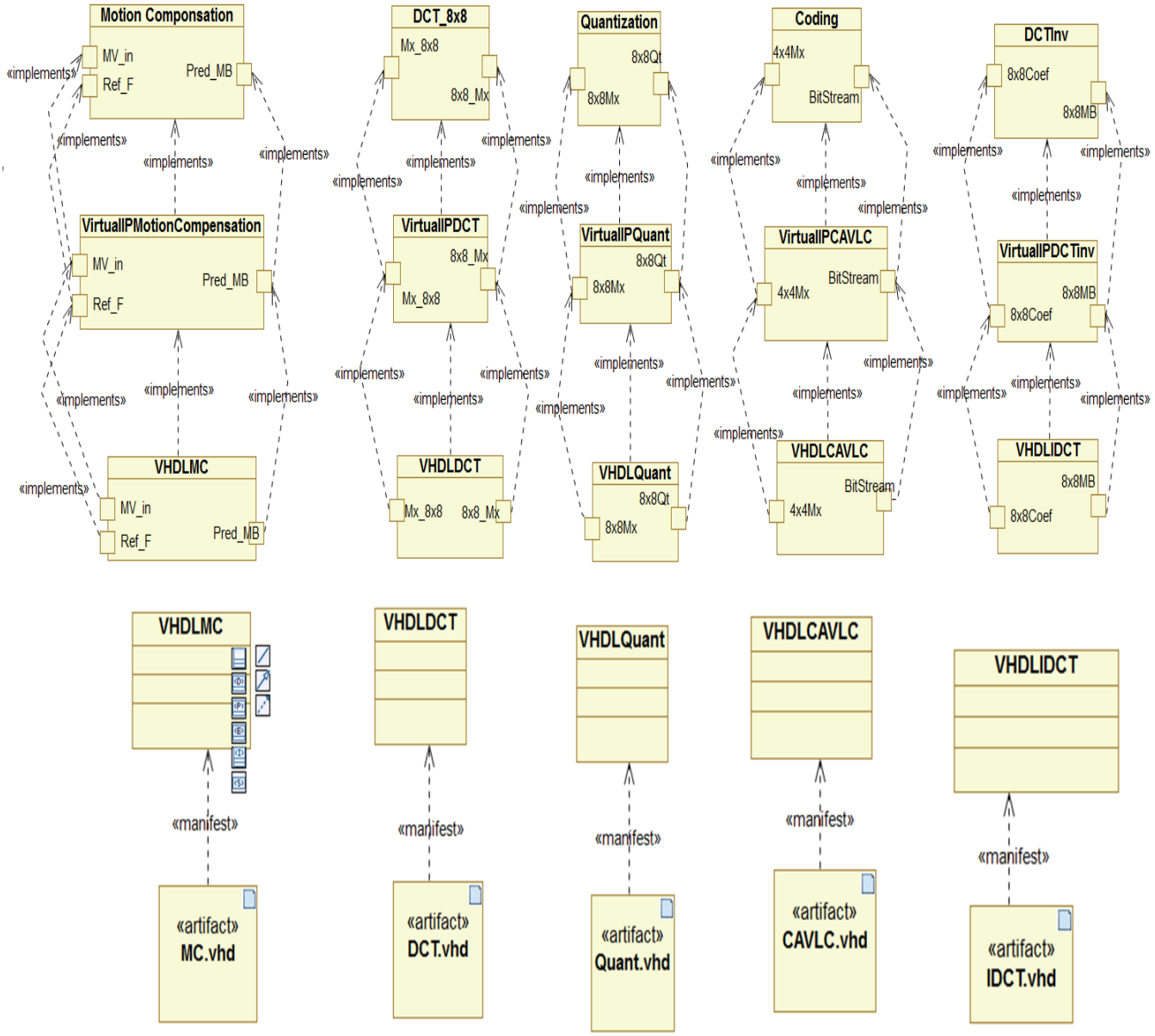

Figure 6. 8: Déploiement des tâches élémentaires du codeur H.264

Dans cette phase de déploiement, nous arrivons à décrire comment créer un modèle d'un SoC en utilisant le profil Gaspard. La modélisation a été réalisée sur l'environnement Gaspard. Les modèles fournis sont compatibles avec les règles et la chaîne de transformations de Gaspard, ils peuvent assurer la génération automatique du code. La chaîne actuelle du Gaspard n'automatise pas la génération du code VHDL. Nous avons alors défini notre chaîne à partir du modèle en Y du Gaspard, afin de montrer ce qu'il faut générer pour quelques éléments et non plus le système complet.

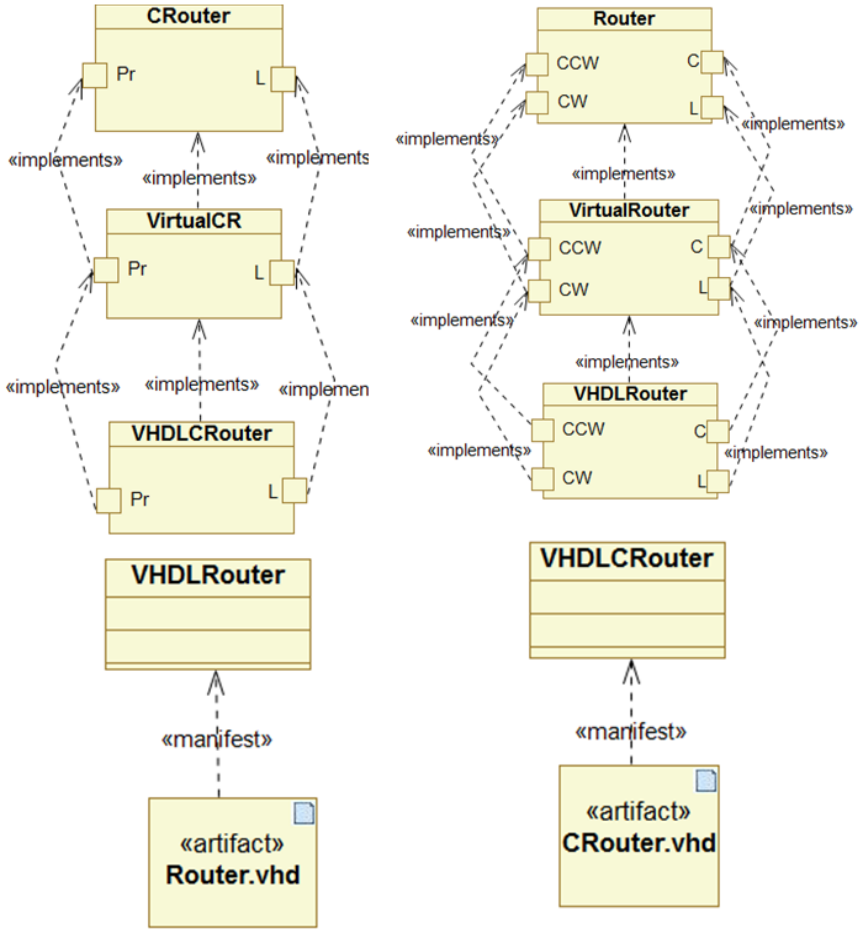

Figure 6. 9: Déploiement du NoC

7. Intégration d'une chaine pour la génération du code VHDL

Cette chaîne est déjà définie dans [130]. Elle s'intègre dans la démarche en Y de Gaspard telle que le montre la figure 6.10. Jusqu'à présent, nous sommes à l'étape de déploiement. La modélisation de l'architecture, de l'application et de l'association sont déjà effectuées ainsi que la description VHDL des tâches élémentaires. Dans la chaîne que nous voulons suivre et intégrer dans Gaspard, notre point de départ est la phase de déploiement, et nous voulons aller Jusqu'à la génération du code VHDL synthétisable. En revanche, nous avons utilisé les transformations de Gaspard pour faire pivoter des modèles UML jusqu'à la génération du code. Ă ce stade, nous considérons que le modèle est proche de la génération automatique du code.

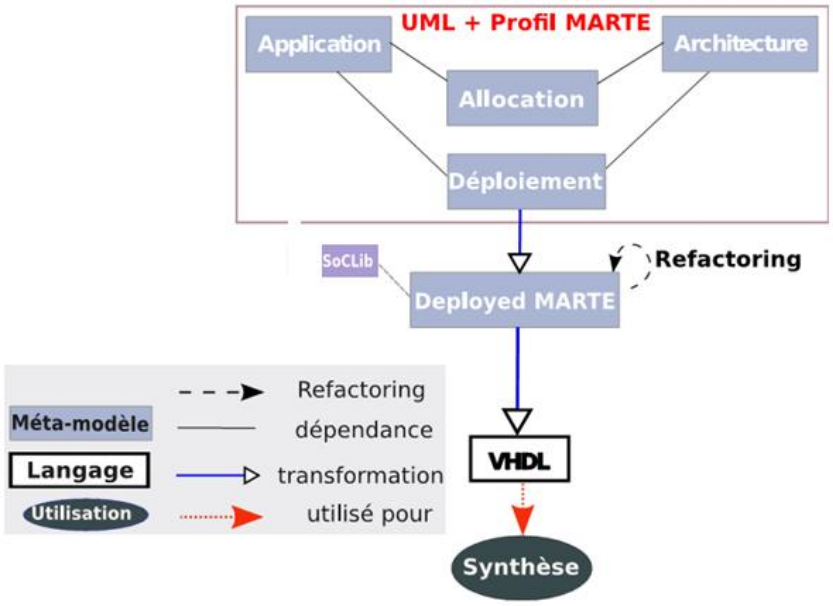

Figure 6. 10: Chaîne pour la génération du code

7.1. Organisation du GaspardLib

Dans la bibliothèque, deux types d'IP peuvent exister : les IPs qui représentent la brique de base de notre système et des IPs qui vont être générées automatiquement. Ces IPs correspondent aux IPs d'assemblage, c'est-à-dire le « Top Level » de l'architecture. La bibliothèque GaspardLib est incluse dans la bibliothèque GaspardLib. Afin de faciliter la manipulation de la démarche de la génération du code, les composants VHDL sont regroupés d'une façon hiérarchique. Pour bien organiser cette bibliothèque, nous avons séparé les IPs de l'architecture et celles de l'application. La figure 6.11 illustre la hiérarchie qui permet la construction de la bibliothèque.

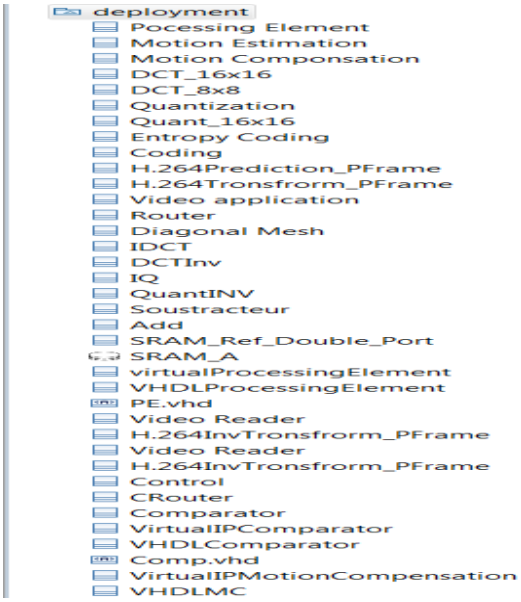

Figure 6. 11: Organisation des IPs

7.2. Etude de cas

Dans ce paragraphe, nous présentons une étude de cas pour la génération du code appliqué sur quelques parties de notre système. Comme nous avons déjà dit, la chaine actuelle du Gaspard n'automatise pas la génération du code VHDL, alors nous allons montrer un exemple du code qu'on doit générer pour l'application et un autre pour l'architecture.

7.2.1. La DCT

Nous avons parlé dans la section 3 du chapitre précédent de la modélisation des tâches élémentaires du codeur H.264. Parmi ces tâches, il y a la DCT et nous avons montré qu'on peut extraire du parallélisme de tâches en répétant une brique de base qui assure le calcul de la fonction élémentaire de la DCT. Nous rappelons au lecteur que la DCT parallèle, proposée, est mise en œuvre pour assurer le calcul de 64 coefficients simultanément. Le modèle MARTE de ce composant est basé sur la tâche élémentaire qui est répétée huit fois. La phase de déploiement de cette tâche élémentaire est déjà montrée dans la figure 6.8. Dans cette génération du code, nous avons exploité le calcul matriciel de l'origine, de l'ajustage et du pavage afin de l'introduire dans une boucle « Generate » du langage VHDL. Le code que nous devrons générer après avoir écrit les règles de transformation doit avoir la même forme que celui illustré dans la figure 6.12.

```
1  use IEEE.STD_LOGIC_1164.ALL;
2  use IEEE.STD_LOGIC_ARITH.ALL;
3  use IEEE.STD_LOGIC_UNSIGNED.ALL;
4  entity Tiler_DCT is
5  port(datain: in t1;dataout: out t1 );
6  end Tiler_DCT;
7  architecture Structural of Tiler_DCT is
8  signal in1,out1: t1;
9  component DCT port(inDCT:in std_logic_vector(0 to 7);-- La tâche élementaire de la DCT
10 outDCT:out std_logic_vector(0 to 7));
11 end component;
12 begin
13 gen1: for r in 0 to 7 generate -- L'instanciation du composant
14 gen2: for i in 0 to 7 generate
15 in1(i)(r) <= datain(i)(r);
16 dataout(i)(r)<=out1(i)(r);
17 end generate;
18 Config: DCT port map(indct=>in1(r)(0 to 7),outdct=>out1(r)(0 to 7));
19 end generate;
20 end structural;
```

Figure 6. 12: Code VHDL généré

En passant par les différentes phases de conception suivant le flot de Gaspard et en exploitant les règles de transformation, nous arrivons à générer le code VHDL d'un composant à partir de la répétition de sa tâche élémentaire. L'avantage d'une telle représentation est que le code est optimisé en termes de nombre de lignes. En le comparant avec un code écrit à la main, nous constatons que le nombre de lignes de « port mapping » est égal au nombre de répétitions de la tâche élémentaire. En exploitant les informations de la modélisation de haut niveau en MARTE, nous pouvons avoir un code comportant un nombre réduit de lignes. La synthèse sur l'outil ISE de Xilinx vérifie la construction correcte d'un IP de calcul de DCT en favorisant le parallélisme. La figure 6.13 illustre le schéma de synthèse de cet IP.

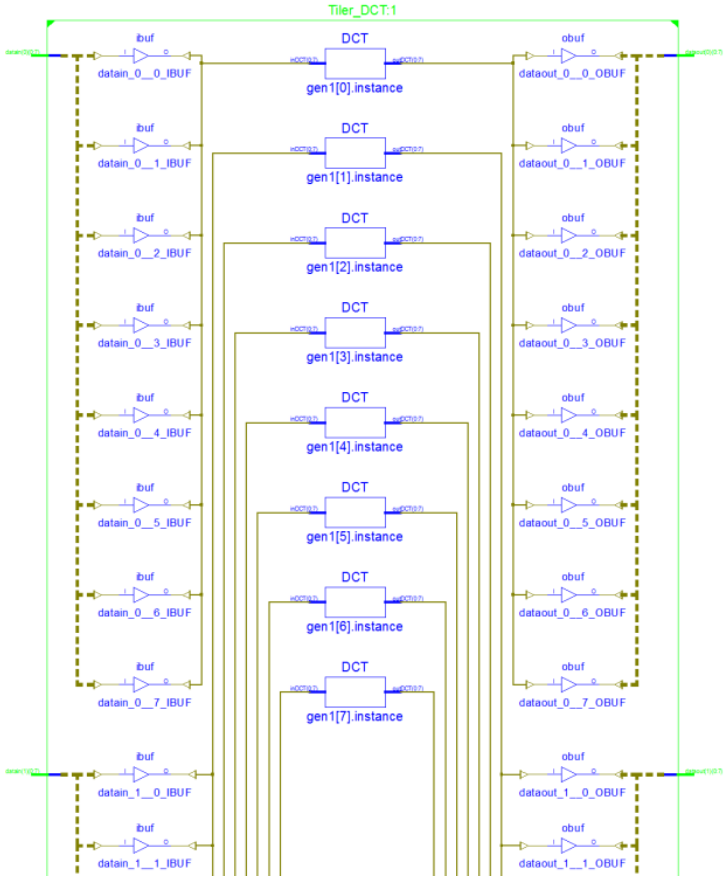

Figure 6. 13: Un extrait de schéma de synthèse du code généré

7.2.2. La topologie Mesh, Torus

Dans le chapitre 3, nous avons montré la modélisation de quelques topologies des NoCs. Parmi ces topologies, il y a le Mesh, le Torus et le diagonal Mesh. Les modélisations en MARTE de la topologie en Mesh et Torus sont illustrées respectivement sur les figures 3.14 et 3.15. Dans cette modélisation, nous avons utilisé la notion de dépendance d'inter répétition. Nous voyons maintenant comment exploiter les informations existant dans la modélisation pour arriver à un code VHDL.

La topologie Torus présente une dépendance cyclique entre les liens, d'où l'utilisation d'un connecteur de dépendance d'inter-répétition (comme le montre la figure 3.15). Cette topologie est construite autour d'une brique de base qui est un routeur de quatre ports. Donc, pour générer le code VHDL d'un Torus de taille M×N, il suffit d'avoir le code de ce composant de base. Les liens reliant les routeurs peuvent être exprimés à l'aide de tableaux multidimensionnels. Afin d'exprimer les connexions entre les routeurs de bord, l'application de l'opération modulo par la taille de l'espace de répétition est nécessaire pour assurer toutes les connexions. Le code de la figure 6.14 correspond à une modélisation VHDL d'un Torus de taille 3×3.

```
architecture Behavioral_Torus of Torus is

Type tab_3to3 is array(0 to 2) of std_logic_vector(0 to 2);-- Array that contains links.

signal tab_h, tab_v:tab3to3; -- vertical and horizental links.

component router is -- component declaration.

port(Clk,rst: in std_logic; North, Sud, West, East:inout std_logic_vector(31 downto 0));

end component;

Begin

-- Instantiation in both direction.

GenerateX: for i in 0 to 2 generate
GenerateY: for j in 0 to 2 generate

dependancecycle: router port map(clk, rst, tab_h(i)(j), tab_h(i)((j+1) mod 3),
tab_v((i+1) mod 3)(j),tab_v(i)(j));

end generate;
end generate;
end behavioaral_Torus;
```

Figure 6. 14: Code généré de la topologie Torus

Nous procédons de la même manière pour la topologie en Mesh. La différence dans la modélisation de ces topologies apparaît dans l'utilisation du concept « Modulo » pour le Mesh, le « Modulo= False », c'est-à-dire que cette topologie n'est pas exprimée dans un espace torique. Les routeurs de bord ne sont donc pas connectés. Cela introduit une diversité dans les routeurs qui construisent cette topologie. Nous trouvons en effet trois types de routeurs pour toute topologie de taille M×N telle que (M, N)> (2, 2). Ils sont de deux ports, de trois ports et de 4 ports. Ces constatations issues de la modélisation de haut niveau peuvent être exprimées en VHDL par l'utilisation de trois boucles, dans chaque boucle la répétition d'un routeur est exprimée. La figure 6.15 illustre un extrait des figures de synthèse.

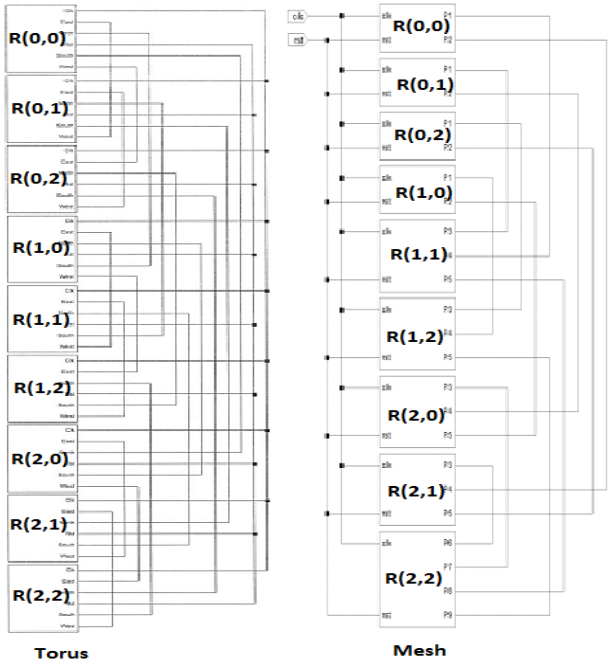

Figure 6. 15: Figure de synthèse exprimant les connexions entre les routeurs

8. Conclusion

Dans ce chapitre, nous avons décrit les phases finales de notre flot de Co-modélisation. Nous avons commencé par aborder les problèmes d'allocation des IPs sur un réseau sur puce. Ces problèmes sont liés au bon choix de l'emplacement des IPs sur les routeurs. En effet, nous avons utilisé un algorithme de placement, développé dans la littérature, en spirale. Pour bien adapter cet algorithme, une fonction coût a été définie, dans l'objectif de définir un placement qui permette de tenir compte des caractéristiques de notre réseau sur puce. Les IPs les plus communicants sont placés l'un à côté de l'autre. Le processus de placement des IPs sur le réseau est statique, donc nous avons commencé par placer l'estimateur de mouvement sur le routeur central, puis en fonction de ce placement les autres IPs sont placés. Suivant cette démarche, la modélisation de haut niveau d'abstraction de l'association est effectuée. Il s'agit de mettre des liens en appliquant le stéréotype « distribute » afin de définir le placement géographique de l'IP sur le réseau. Après cette phase d'association, nous avons entamé la phase de déploiement qui consiste à relier les tâches élémentaires modélisées et leurs composants exprimés en langage VHDL. La chaîne actuelle de Gaspard n'automatise pas la génération automatique du code VHDL. En revanche, nous avons utilisé la chaîne développée dans la section 6 de ce même chapitre afin de la valider par un exemple du code VHDL qui doit être généré par Gaspard après avoir mis en place une chaîne complète pour la génération automatique du code VHDL des systèmes complexes.

Chapitre 7 : Bilan et perspectives

Bilan

Les travaux présentés dans cette thèse constituent une contribution au développement d'un environnement de co-conception pour les SoCs via l'exemple de Co-design de l'application H.264 sur un NoC, qui peut être de type globalement asynchrone localement synchrone (GALS). Il résulte de ces travaux la proposition d'une méthodologie pour la modélisation des NoCs et un flot de conception permettant l'automatisation de la génération de code VHDL. Afin d'atteindre ces objectifs, il a été nécessaire d'intervenir dans différents domaines, partant de la modélisation de haut niveau jusqu'au niveau RTL. Entre ces deux niveaux, plusieurs travaux ont été menés dans cette thèse, tels que la modélisation de haut niveau des NoCs, du codeur H.264, la conception matérielle des IPs de l'architecture et de l'application, l'intégration de ces IPs dans un flot de Co-conception etc. Cette intégration a mené à une chaîne de génération de code VHDL à partir de la modélisation en MARTE.

Nos travaux de thèse ont commencé par une étude bibliographique détaillée sur la modélisation de haut niveau en utilisant le profil MARTE. Via cette étude, nous avons remarqué que la modélisation des NoCs n'a jamais été faite, donc nous n'avons pas trouvé de méthodologie claire qui définisse les étapes nécessaires à la modélisation des réseaux sur puce. Pour surmonter ce problème, nous avons commencé par l'identification des concepts de base des NoCs, afin de trouver le paquetage adéquat à leur modélisation. Les étapes effectuées dans cette phase ont été reformulées autour d'une méthodologie permettant à un concepteur, même s'il n'est pas un expert en MARTE ou NoC, de faire la modélisation. La première étape de notre méthodologie est la modélisation de la topologie. Pour ce faire, nous avons développé une démarche qui explique comment exploiter les concepts de paquetage RSM afin de modéliser une topologie de façon compacte. Cette démarche a été validée par la modélisation de plusieurs topologies de la littérature. En suivant la méthodologie, les autres concepts des NoCs sont aussi modélisés en MARTE. Il a résulté de ces étapes la proposition d'une extension dans le paquetage Hw_Media du standard MARTE. Cette extension consiste à ajouter un stéréotype nommé Hw_Router qui peut être utilisé comme brique de base pour la modélisation au niveau système des NoCs.

Apres ces étapes, nous avons passé à la modélisation, toujours en haut niveau d'abstraction, de l'application H.264. Dans cette étape, nous avons tiré profit des concepts de paquetage RSM pour exprimer le parallélisme potentiel qui peut exister dans le codeur. Le parallélisme est identifié au niveau des modules qui constituent la norme H.264, pour cela nous avons parlé de parallélisme local.

Afin de pouvoir intégrer les briques de base de notre architecture et l'application, nous sommes passés à la conception matérielle du routeur, du NoC, et des IPs de la chaîne de codage conforme à la norme H.264.

Durant cette phase, nous avons proposé une description VLSI d'un routeur modulaire qui représente la brique de base de la répétition qui construit notre réseau. L'architecture du routeur a fourni des performances qu'on peut dire pertinentes suivant l'étude comparative effectuée dans la thèse. Cette architecture intègre une fonction de routage intelligente permettant de réduire le chemin parcouru par un paquet. La description de l'architecture du routeur a été faite en VHDL. Cette étape prépare la construction de la bibliothèque de notre système.

Le codeur H.264 est composé de plusieurs modules. Nous nous sommes intéressés dans cette thèse au développement d'une chaîne de codage vidéo conforme à cette norme, et qui intègre le sens de l'encodage suivant la prédiction temporelle. En effet, nous avons développé le module de la prédiction temporelle, qui est composé de l'estimateur et de la compensation de mouvement, et le module de traitement d'image, qui est constitué par la DCT et la quantification. Dans cette thèse, et en se basant sur la modélisation en MARTE, qui nous a permis d'extraire le parallélisme, nous avons pu concevoir une architecture parallèle de la DCT et de la quantification. Un autre module très critique dans le codeur H.264 est l'estimateur de mouvement. Ce module fait l'objet de plusieurs travaux de recherche malgré que son architecture globale, c'est-à-dire l'entité, existe depuis les années 80. Notre contribution dans ce module peut se résumer de la manière suivante : dans un premier temps, nous avons développé l'architecture régulière, au niveau RTL, d'un estimateur de mouvement de taille de blocs fixe et à faible coût de consommation par l'utilisation de la technique « Gated Clock ». Dans un second temps, nous sommes passés à la description de l'architecture modulaire d'un estimateur de mouvement de taille de blocs variable. Cette architecture repose sur le principe de la réutilisation de ce qui est déjà calculé, ce qui peut nous faire gagner au niveau temps de calcul, et sur le principe de partage des ressources. Cette proposition assure l'utilisation complète des processeurs élémentaires(PE) et des comparateurs élémentaires(CE) de l'estimateur. Nous avons intégré aussi dans cette architecture le principe du contrôle distribué de la consommation énergétique. Cette architecture a été développée en VHDL prototypé sur une carte FPGA virtex6 équipée par un « Kit » de mesure de la consommation énergétique.

Suivant notre flot de conception, qui est celui de Gaspard, une phase d'association a été abordée. Dans cette phase, nous avons étudié le problème de la distribution géographique des IPs sur le NoC. Elle a été faite en deux étapes, en premier lieu nous avons étudié ce problème en reformulant notre allocation géographique à l'aide d'une fonction coût réduisant la communication dans le système. En s'appuyant sur cette distribution, cette phase d'association est achevée par l'utilisation des concepts du profil MARTE.

Après toutes ces étapes, le déploiement des IPs est modélisé et nous arrivons à la fin de notre flot de co-conception. Dans cette dernière étape, nous avons décrit une démarche IDM pour la génération de code automatique, mais malheureusement la chaîne de compilation VHDL de notre environnement Gaspard n'est pas complète et nous n'avons pas pu générer automatiquement le code. Mais nous avons fourni dans cette thèse tous les éléments nécessaires pour arriver à la génération automatique de code. Des exemples (DCT, Mesh, Torus) de ce qu'on doit générer automatiquement sont expliqués dans cette thèse.

Perspectives

Ce travail de thèse peut être poursuivi et peut servir de ligne directrice dans d'autres perspectives. Concernant la modélisation des NoCs, et comme complément de ce qu'on a commencé, une intégration du stéréotype Hw_Router s'avère nécessaire. Cette intégration peut conduire à une phase de test et de génération de code à partir d'une modélisation des NoC, en se basant sur les attributs de ce stéréotype. De plus, elle permet d'étudier la possibilité de modéliser des architectures de NoC GALS.

Actuellement, la chaîne VHDL de Gaspard n'est pas complète mais elle est destinée à la génération de code exécutable, c'est-à-dire synthétisable sur un FPGA ou un ASIC. Ce travail de thèse fournit tout ce qui est nécessaire pour arriver à l'implémentation d'un SoC dédié au traitement vidéo sur un ASIC.

Orthogonalement, la phase d'exploration peut constituer une perspective de ce travail. Par exemple, pour modifier certaines caractéristiques d'un composant qu'on a utilisé lors de la modélisation de haut niveau, nous devrons modifier le modèle de départ. Donc une automatisation de l'exploration s'avère intéressante, afin d'accélérer le processus de conception. Pour valoriser l'automatisation de cette phase, il faut définir l'ensemble des paramètres qui peuvent être modifiés dans un SoC.

La consommation énergétique est un autre paramètre important, qui peut être pris en compte lors de l'extension de ce travail. L'intégration des modèles de contrôle de consommation dans le flot de Gaspard permettra au concepteur de fournir des systèmes de hautes performances et peu énergivores.

Références Bibliographiques

[1] G. D.Micheli, L.Benini, "Networks on Chip" Livre publié par Elseiver, 2006.

[2]S.K Tewksbury, M.Uppuluri, L.A.Hornak "interconnction/micro-letworks for integrated microelectronics ", global telecommunication conference, 1992, pp 180-186, Orlando, USA.

[3] I.Richardson, "The H.264 advanced video compression standard", second edition, par Wiley, 2010.

[4] DaRT. Graphical Array Specification for Parallel and Distributed Computing. 2010.

[5] UML MARTE. The UML Profile for MARTE : Modeling and Analysis of Real-Time and Embedded Systems. http://www.omgmarte.org/.

[6] G.D.Micheli, R.K.Gupta, "Hardware/Software Co-Design", IEEE Proceedings of the IEEE, Special Issue on Hardware/Software co-design, mars 1997.

[7] D. D. Gajski and R. H. Kuhn, "New VLSI tools",IEEE Computer, pp, 11–14, 1983.

[8] O. Adeluyi, E. ok Kim, J. Lee, and J.-G. Lee. The use of fair Y-sim for optimizing mapping set selection in hardware/software co-design. In ISOCC '08: SoC Design Conference, volume 2, pages 174–178, Busan, Corée du Sud, 2008. 19

[9] E. Riccobene, P. Scandurra, A. Rosti, and S. Bocchio, "A model-driven design environment for embedded systems," In DAC '06: Proceedings of the 43rd annual Design Automation Conference, pages 915–918, San Francisco, CA, USA, 2006. ACM. 19

[10]I. R. Quadri, S. Meftali, and J.-L. Dekeyser, "High level modeling of dynamic reconfigurable FPGA" Int. J. Reconfig. Comput., 2009:1–15, 2009. 19.

[11] S.I. Han, S.I. Chae, L. Brisolara, L. Carro, K. Popovici, X. Guerin, A. A. Jerraya, K. Huang, L. Li, and X. Yan. "Simulink based heterogeneous multiprocessor SoC design flow for mixed hardware/software refinement and simulation," Integr. VLSI J., 42(2):227–245, 2009.19

[12] D.W. Franke, M.K.Purvis, "Hardware/Software codesign : a prespective", proc 13th International Conference on Software Engineering, IEEE CS Press, Los Alamitos, California, pp 344,352, 1991.

[13] W.H.Wolf., "Hardware/Software codesign of embedded systems", proc of IEEE, vol.82, PP 967-989, July 1994.

[14] E.Stoy, " A petri net based unified representation for HX/SX codesign", Master Thesis, Dept.of Computer and information Science, Linkoping University, Sweden, 1995.

[15] M.D. Edward, J.Forrest, A.E.Whkean, "Acceleration of software algorithms using Hardware/software co-design techniques", journal of systems Architecture, Vol42, pp 697-707, 1997.

[16] G.D.Micheli , Nato AS, and Sami M.G, "hardware/software Co-design", Proceding of the IEEE, Vol.85 pp349-365, 1997.

[17] Theo. A.C.M. Claasen. "System on a Chip: Changing IC design today and in the future". IEEE MICRO, pp 20-26, Juin 2003.

[18] MMAlpha, www.irisa.fr/cosi/ALPHA

[19]Mathematica, www.wolfram.com

[20] H.Verge, C.Mauras, P.Quinton. "The alpha language and its use for the design of systolic arrays," The journal of VLSI signal Processing, pp 173-182, 1991.

[21] projet INRIA-CNRS COSI. ALPHA home page

[22] Tanguy Risset, "Contribution à la compilation de nids de boucles sur silicium. Habilitation à diriger des recherches," université de Rennes1, 2000.

[23] R.M.karp, R.E.Miller, S.Winograd, "The organization of computations for uniform recurrence equation," J. ACM, pp 563-590, 1967.

[24] C.Mauras, "Alpha: un langage équationnel pour la conception et la programmation d'architecture paralléles synchrones," Thèse de l'université de Rennes1, 1989.

[25] C.Dezan, "Géneration automatique de circuit avec ALPHA du CENTAUR," Phd thésis, IRISA, 1993

[26]P.le Moenner, L.Perraudeau, P.Quinton, S.Rajopadhye and Tanguy Risset, "Generating regular arithmetic circuit with ALPHARD," Technical report, IRISA, mars 1996.

[27] E.A. Lee and D.G.Messerschmitt, "Static scheduling of synchronous data flow programs for digital signal processing," IEEE trans. on Computers, janviers 1987.

[28] M.C.Williamson, "Synthesis of parallel Hardware Implementation from Synchronous Dataflow Graph Specification," PhD thesis. University of California, Berkley, 1998.

[29] T.Filiba, M.K.Leung, and V.Nagpal, "VHDL code generation in the ptolemyII environement" Technical report, Electrical Engineering and Computer Sciences, university de Berkly, décembre 2006.

[30] Y.sorel and C.Lavarenne. Syndex Documentation index. INRIA 2000, http:// www-rocq. Inria.fr/ syndex/doc

[31] L.Kaouane, M.Akil, Y.sorel, and T.Grandpierre, "From algorithm graph application to automatic synthesis of FPGA circuit: a seamless flow of graph transformations,"In 13th international conference on Fild-Programable Logic and Application. FPL'03, Lisbon, Purtugal, septembre 2003.

[32] L.Kaouane, M. Akil, T.Granpierre, and Y.Sorel, "Amethodology to implement real-time application onto reconfigurable circuits," Journal of supercomputing, 2004.

[33] J.P.Calvez, O.Pasquier, D.Isidoro, D.Jeuland, "Co-Design with the MCSE methodology," Proceedings of the 20th EUROMICRO Conference, pp 19 – 26, 1994.

[33] A.Demeure, A.Lafarge, E.Boutillon, D.Rozzonelli, J.C Dufoourd,J.Louis Marro, " Array-OL : proposition d'un formalisme tableau pour le traitement de signal multi-dimensionnel," GRETSI, Groupe d'Etudes du Traitement du Signal et des Images, 1995.

[34] C. Glitia, "Optimisation des applications de traitements systématique intensives sur system-on-chip," Université des sciences et technologies de Lille, 2009.

[35] J-M. Favre, J. Estublier, and M. Blay-Fornarino, "L'ingénierie dirigée par les modèles,au-delà du MDA," Hermès Science, Lavoisier, pp 42-44, 2006.

[36] Malcon Eva, SSADM Version 4: A User's Guide (McGraw-Hill International Series in Software Engineering), avril 1994.

[37] Object Management Group, Inc, editor. UML2 infrastructure (Final Adopted Specification). Http:/www.omg.org/cgi-bin/doc.pct/2003-90-15.

[38] OMG. Modeling and analysis of real-time and embedded systems (MARTE).http://www.omgmarte.org/ pp 555-557, 2009.

[39] Rabie Ben Atitallah, "Modèles et simulation des systèmes sur puce multiprocesseurs Estimation des performances et de la consommation d'énergie," Thèse de l'université de Lille, 2008.

[40] Sébastien le beux, "Un flot de conception pour applications de traitement du signal systématique implémentées sur PGA à base d'ingénierie dirigée par les Modéles,"Thèse de l'université de Lille, 2007.

[41] H. Yu. A MARTE based reactive model for data-parallel intensive processing: Transformation toward the synchronous model. Thèse de l'université de Lille, 2008. 62, 68

[42] J. Taillard. Une approche orientée modèle pour la parallélisation d'un code de calcul éléments finis. Thèse université de Lille, France, 2009.

[43] GaspardLib. GaspardLib: An open platform for modelling and simulation of multi-processors system on chip, 2009. https://www.GaspardLib.fr/trac/dev/wiki.

[44] A. S. Tanenbaum, Computer networks. Prentice-Hall, 1996.

[45] D. Culler, A. Gupta, and J. P. Singh, "A Hardware/Software Approach," Parallel Computer Architecture Morgan Kaufmann Publisher, 1999.

[46] M. Millberg, E. Nilsson, R. Thid, S. Kumar, and A. Jantsch, "The Nostrum backbone-a Communication protocol stack for Networks on Chip," in Proc. 17th International Conference on VLSI Design, pp. 693-696, 2004.

[47] W. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks,"in Proc. Design Automation Conference, pp. 684-689. 2001

[48] J. Quartana, "Conception de reseaux de communication sur puce asynchrones : application aux architectures GALS, " thèse à l'institut national Polytechnique de Grenoble. 2006.

[49] G. D. Micheli and L. Benini, Networks on Chips : Technology and Tools (Systems on Silicon). Morgan Kaufmann Publishers, 2006.

[50] M. Karol, M. Hluchyj, S. Morgan, "Input Versus Output Queueing on a Space-Division Packet Switch," IEEE Transactions on Communications, vol. 35, no. 12, pp 1347-1356, 1987.

[51] L. Ni and P. McKinley, "A survey of wormhole routing techniques in direct networks,"
IEEE Computer, vol. 26, pp. 62-76, 1993.

[52] W. Dally, "Virtual-channel flow control, " in Proc. 17th Annual International Symposium on Computer Architecture, pp. 60-68, 1990.

[53] R.Lemaire, "Conception et modélisation d'un système de contrôle d'applications de télécommunication avec une architecture de réseau sur puce (NoC)," Thèse de l'INP Grenoble, 2006.

[54] J. Duato, S. Yalamanchili, and N. Lionel, "Interconnection Networks : An Engineering Approach," Morgan Kaufmann Publishers, 2002.

[55] A. Leroy, "Optimizing the on-chip communication architecture of low power systems-on-chip in deep sub-micron technology," Thèse, Universite de Bruxelles,2006.

[56] L. Fan, M.C. Wu, H.C. Lee, P. grodzinski, "Optical interconnection network for massively parallel processors using beam-steering vertical cavity surface-emitting-lasers," Workshopon Massively Parallel Processing Using Optical Interconnections,pp 28-35,1995.

[57] M. Alho, J. Nurmi, "Implementation of Interface Router IP for Proteo Network-On- Chip,"The6th IEEE InternationalWorkshopon Designand Diagnostics of Electronics Circuits and Systems, 2003

[58] M.Elhaji, P.Boulet, A.Zitouni, S.Meftaly, J.L.Dekeyser, R.Tourki, "An MDE approach for Modeling network-on-chip topologie," DTIS, 2009.

[59] S.Evain, "µSpider environnement de conception de réseau sur puce," Thèse, Institut national des sciences appliquées de Rennes. 2006.

[60] K. Goossens, J. Dielissen, O. P. Gangwal, S. Gonzalez Pestana, A. Radulescu et E. Rijpkema, "A Design Flow for Application-Specific Networks on Chip with Guaranteed Performance to Accelerate SOC Design and Verification," In Proc. Design, Automation and Test in Europe Conference and Exhibition (DATE), 2005

[61] University of Southern California, "The network simulator ns-2," On-line, disponible sur http://www.isi.edu/nsnam/ns/.

[62] The use of SSADM (Structured Systems Analysis and Design Methodology) as a standard methodology on information systems Projects. Computer Science-Theory, 2001.

[63] Adolf Abddallah, "Conception de SoC à base d'horloge abstraites : vers l'exploration d'architectures en MARTE," thèse université de Lille, 2011.

[64] OMG. OMG Unified Modeling Language (OMG UML), Superstructure, V2.1.2, 2007.
http://www.omg.org/spec/UML/2.1.2/Superstructure/PDF/. 119

[65] C.Glitia, P.Boulet,E.Lenormand, M.Barreteau, "Array-Ol with delays, a domain specific specification language for multidimensional intensive signal processing," in Multidimensional Systems and signal processing, 2009.

[66] S. Murali, Designing Reliable and Efficient Networks on chips.

[67] D.Atienzaa, F.Angiolini, S.Murali, A.Pullini, Luca Benini, G.Micheli,"Network-on-Chip design and synthesis outlook, INTEGRATION, " the VLSI journal, pp 340–359,2008.

[68] M.Zid, A.Zitouni, A.Bagane, R.tourki, "Nouvelles architectures génériques de NoC, " Technique et Science Informatiques, pp 101-133, 2009.

[69] M. Elhaji and P. Boulet and A. Zitouni and R. Tourki and J.L. Dekeyser and S. Meftaly, " Modeling Networks-on-Chip at System Level with the MARTE UML profile," Proceedings of the Model Based Engineering for Embedded Systems Design in DATE, 2011.

[70] T.Marscaux, Mapping and Management of Communication Services on MP-SoC Platforms, CIP-DATA LIBRARY TECHNISCHE UNIVERSITEIT EINDHOVEN, 2007.

[71] X.Lin, Ph.K. McKinley, L.M. Ni, "The Message Flow Model for Routing in Wormhole-Routed Networks," International Conference on Parallel Processing, pp 294-297, 1993.

[72] VSI Alliance, "On-Chip Bus Virtual Component Interface Standard (OCB VCI)".

[73] OCP-IP Association, "Open Core Protocol (OCP) specification," http://www.ocpip.org/

[74]S.Riso, " Evaluation des paramètres architecturaux des réseaux sur puce," thèse de l'université de Montpellier,

[75] N. W. Eum, T. Kim and C. Kyung, " CeRA: A Router for Symmetrical FPGAs Based on Exact Routing Density Evaluation," IEEE TRANSACTIONS ON COMPUTERS, 2004.

[76] C. Bobda and A. Ahmadinia, "Dynamic Interconnection of Reconfigurable Modules on Reconfigurable Devices", Copublished by the IEEE CS and the IEEE CASS September–October 2005, pp 443-451.

[77] P. Guerrier, A. Greiner,"A generic architecture for on-chip packet-switched interconnections," Proc. Design, Automation and Test in Europe Conference and Exhibition, p. 250–256, 2000.

[78] F.Moraes, "an infrastructure for low area overhead packet switching networks," Vlsi integration journal, pp 69-93, 2004.

[79] Arteris, "Networks on Chip for Managing On-Chip Communications,"SOC-central,2006.

[80] T. Bjerregaard, "The MANGO clockless network-on-chip : Concepts and implementation," Thèse de doctorat, University of Denmark, 2005.

[81] Ouyang Y M, Zhu B, Liang H G, "Networks on chip based on diagonal interlinked mesh topology structure," Computer Engineering, pp 100-102, 2009.

[82] E.S. Shin, Vincent J. L.Mooney and G.F. Riley, "Round-robin Arbiter Design and Generation," ISSS, 2002.

[83]M.Elhaji, B.Attia, A.Zitouni, R.Tourki, S.Meftali, J.L.Dekeyser, "FeRoNoC: Flexible and extensible Router implementation for diagonal mesh topology," DASIP 2011.

[84]A.Ehliar, D.Liu, "An fpga based open source network-on-chip architecture," in Field Programmable Logic and Applications. IEEE, pp. 800–803, 2007.

[85] N.Kapre, N.Mehta, M.deLorimier, R Rubin, H Barnor, M.J Wilson, M Wrighton, and A DeHon, "Packet switched vs. time multiplexed fpga overlay networks," in IEEE Symposium on Field-programmable Custom Computing Machines. IEEE, pp. 800–803, 2006.

[86] JVT. Advanced video coding for generic audiovisual services. Rapport technique, ISO/IEC 14496-10 and ITU-T Rec. H.264, 2003.

[87] I.E.Richardson,"The H.264 Advanced Video compression Standard," Wiley publisher, 2003.

[88]Vcodex, "H.264/MPEG-4 Part 10 : Intra Prediction," H.264 / MPEG-4 Part 10 White Paper, Octobre 2002.

[89] Fabrice Urban,"Implantation optimisée d'estimateurs de mouvement pour la compression vidéo sur plates-formes hétérogènes multicomposants," Thèse de l'institut d'électronique et de télécommunications de Rennes, 2007.

[90] G. Robert, “Représentation et codage de séquences vidéo par hybridation de fractales et d’éléments finis,” Thèse de l’université Joseph Fourier, 1997.

[91] F. Kelly and A. Kokaram, “Graphics hardware for gradient based motion estimation,” Electronic Imaging - Embedded Processors for Multimedia and Communications, 2004.

[92] E. D. Castro and C. Morandi, “Registration of translated and rotated images using finite Fourier transforms,” In IEEE Transactions on Pattern Analysis and Machine Intelligence, pp 700-703, Sept 1987.

[93] R.X. Li, B. Zeng, M.L. Lion, “A new 3-step search algorithm for block motion estimation,” IEEE transactions on Circuits and Systemsfor Video Technology, pp 438-442, 1994.

[94] Subhash C. Kwatra, Chow-Ming Lin, andWayne A.Whyte, “An adaptive algorithm for motion compensated color image coding,” IEEE Transactions on Communications, pp 747-754, 1987.

[95] J. Lu, M.L. Liou, “A simple and efficient search algorithm for block-matching motion estimation,” IEEE Transactions on Circuits and Systems for Video Technology, pp 429-433, 1999.

[96] L.Kuo,E.Feig, “A block-based gradient descent search algorithm for block motion estimation in video coding,”In IEEE Transactions on Circuits and Systems for Video Technology, pp 292-296, 1996.

[97] Shan Zhu and Kai-Kuang Ma, “A new diamond search algorithm for fast block matching motion estimation,” In International Conference on Information, Communications and Signal Processing, pp 292-296, 1997.

[98] W.Choi, B. Jeon, J. Jeong, “Fast motion estimat ion with modified diamond search for variable motion block sizes, ” International Conference on Image Processing, ICIP, pp371-374, 2003,.

[99] Y-S Chen, Y-P Hung, C-S Fuh, “Fast Block Matching Algorithm Based on the Winner-Update Strategy, ” In IEEE Transactions on Image Processing, pp 1212-1222, 2001.

[100] T .akizawa, M .Hirasawa, “An efficient memory arbitration algorithm for a single chip MPEG2 AV decoder,”international conference on consumer electronics, pp 182-18, 2001.

[101] T .Takizawa, J.Tajime, H.Harasaki, “High performance and cost effective memory architecture for an HDTV decoder LSI,” In Proceedings of IEEE international conference on acoustics, speech, and signal processing, pp 1981–1984, 1999.

[102] K.Bukhari, G.Kuzmanov, S.Vassiliadis, “DCT and IDCT Implementations on Different FPGA Technologies,” Computer Engineering Lab, Delft University of Technology, pp 232-235.

[103] J.B.Lee, H.Kalva, “The VC-1 and H.264 Video Compression Standards for Broadband Video Services,” 2008.

[104] T.TOTOZAFINY, "Compression d'images couleur pour application a la a la télésurveillance routière par transmission vidéo a très bas débit,"

[105] Y.L.Steve, C.Y.Kao, H.C.Kuo, J.W.Chen, "VLSI Design for video coding H.264/AVC Encoding from Standard Specification to Chip," 2010.

[106] H. Malvar, A. Hallapuro, M. Karczewicz, and L. Kerofsky, "Low-Complexity Transform and Quantization in H.264/AVC," IEEE Trans. Circuit Syst. Video Technol, pp 598- 603, 2003.

[107] M.P.Bonaciu, "Plateforme flexible pour l'exploitation d'algorithmes et d'architectures en vue de la réalisation d'application video haute définition sur des architectures multiprocesseurs mono puce," Thèse de L'INPG, 2006.

[108] Ng Ka Ho, "Highly efficient Motion Estimation Algorithms in video coding," these à l'université de HONG KONG.

[109] T. Komarek and P. Pirsch, "Array architectures for block matching algorithms",
IEEE Transactions on Circuits and Systems, pp. 1301-1308, 1989.

[110] K.M.Yang, M.T.Sun, L.Wu, "A family of VLSI designs for the motion compensation block-matching algorithm," IEEE Trans Circuits Syst Video Technologies, pp 1317–1325, 1989.

[111] H.Yeo, Y.H. Hu, "A novel modular systolic array architecture for full-search block matching motion estimation," IEEE Trans Circuits Syst Video Technologies, pp 407-416, 1995.

[112] M.Elhaji, A.Zitouni, R.tourki, "A Low Power *ASIC* Design of a FSBM Motion Estimator for H.264AVC," PDCS journal, 2009.

[113] L. de Vos and M. Schobinger, "VLSI architecture;for a flexible block matching processor," IEEE Trans. Circuits Syst. Video Technol., vol. 5, pp. 417–428, Oct. 1995.

[114] Y. Huang, S. Chien, B. Hsieh, and Liang-Gee Chen" Global Elimination Algorithm and Architecture Design for Fast Block Matching Motion Estimation" IEEE Trans. Circuits and Syst.Video. Technol., Vol. 14, No. 6, June 2004.

[115] M. Kim, I. Hwang and S. Chae," A Fast VLSI Architecture for Full-Search Variable Block Size Motion Estimation in MPEG-4 AVC/H.264", In Proc. of the ASPDAC, volume 1,pages 631–634, Jan. 2005.

[116] Yap, SY, Mccanny, J.V, "A VLSI Architecture for Advanced Video Coding Motion Estimation"; In Proc. of Application-Specific Systems, Architectures, and Processors, 2003. IEEE International Conference , pp 293–301, 2003.

[117] Joaquin Olivares, "Reconfigurable VBSME architecture using RBSAD", Journal of universal Computer Science, vol 18,2, pp 264-285, 2012.

[118] S.Y.Yap, J.V.McCanny, "A VLSI architecture for variable block size video motion estimation, IEEE Transactions on Circuits and Systems," pp.384-389, 2004.

[119] Z.Yang,J.Bu, C.Chen, X.Li, "Fast predictive variable-block-size motion estimation for H.264/AVC," International Conference on Multimedia and Expo, pp. 1185-1195, 2005.

[120] C.Ou, C.F.Le and W.J.Hwang, "An Efficient VLSI Architecture for H.264 Variable Block Size Motion Estimation," IEEE Transactions on Consumer Electronics, pp.1291-1299, 2005.

[121] S.C.Hsia, P.Y.Hong, "Very large scale integration (VLSI) implementation of low complexity variable block size motion estimation for H.264/AVC coding," IET Circuits, Devices & Systems, pp. 414-424, 2010.

[122] C.Wei, H.Hui, T.Jiarong, L.Jinmei, M.Hao, "A high-performance reconfigurable VLSI architecture for VBSME, in H.264," IEEE Transactions on Consumer Electronics, pp. 1338-1345, 2008.

[123] C.Ou, C.F.Le and W.J.Hwang, "An Efficient VLSI Architecture for H.264 Variable Block Size Motion Estimation," IEEE Transactions on Consumer Electronics, pp.1291-1299, 2005.

[124] Li Y, Qu Y, He Y, "Memory cache based motion compensation architecture for HDTV H.264/AVC Decoder," IEEE international symposium on circuit and systems, pp 2906-2909, 2007.

[125] S.Z.Wang, T.A.Lin, T.M.Liu, C.Y.Lee,"A new motion compensation design for H.264/AVC decoder," Proceedings of IEEE international symposium on circuits and systems, pp 4558–4561, 2005,

[126] I.Amer, W. Badawy, and Graham Jullien, "A high-performance hardware implementation of the h.264 simplified 8x8 transformation and quantization," Advanced Technology Information Processing Systems, ICASSP, pp 1137-1140, 2005.

[127]D.Xavier, "Optimisation combinatoire et problèmes de capacité d'infrastructure ferroviaire," Thèse de doctorat, Université de Valenciennes et du Hainaut Cambrésis, Valenciennes, France, Décembre 2003.

[128]K.Nectarios, R.Michael, T.Panayiotis, P.George, "An Efficient Algorithm for the Physical Mapping of Clustered Task Graphs onto Multiprocessor Architectures," Proceeding of 8th Euromicro Workshop on Parallel and Distributed Processing, IEEE, pp 406-413, 2000.

[129] M.Armin, S.Samira, K.Ahmad, A. Ali, "SPIRAL: A heuristic mapping algorithm for network on chip," IEICE Electronics Expres, 478–484, 2007.

[130] M.Baklouti, "Méthode de conception rapide d'architecture massivement parallèle sur puce : de la modélisation à l'expérimentation sur FPGA," Thèse de l'université de Lille, 2010.