Systolic Ring : une nouvelle approche pour les architectures reconfigurables dynamiquement

Systolic Ring: A new approach for dynamical reconfigurable architectures

par Gilles SASSATELLI[1], Pascal BENOIT[1], Lionel TORRES[1], Gaston CAMBON[1], Jérôme GALY[1], Michel ROBERT[1], Camille DIOU[2]

1 LIRMM, UMR C5506, Université Montpellier II 161 Rue ADA 34392 Montpellier Cedex 5

2 LICM-CLOES, Université de Metz, 7 rue Marconi, 57070 Metz

Tél : 33 0 4 67 41 85 85 Fax : 33 0 4 67 41 85 00 e.mail : sassate torres cambon galy robert pbenoit@lirmm.fr

résumé et mots clés

Motivé par les exigences grandissantes en puissance de traitement auxquelles les architectures actuelles ne seront bientôt plus à même de faire face, cet article présente une nouvelle approche pour la réalisation de circuits à caractère traitement du signal. Après avoir énoncé le problème, nous soulignerons les limitations respectives des structures classiques telles que processeurs et FPGA, et présenterons une architecture hybride de ces deux familles affichant un niveau de performances sans précédent. Nous détaillerons les principes de reconfiguration dynamique sur lesquels notre architecture est basée, puis présenterons des résultats comparatifs sur un algorithme caractéristique des applications multimédia (DCT). Enfin nous exposerons les résultats obtenus par prototypage de la structure ainsi que les travaux en cours concernant les problématiques de compilation ciblant notre architecture.

Architectures reconfigurables, Processeurs, Traitement du signal, compilation matérielle

abstract and key words

Motivated by the growing requirements in performances which the current architectures will not soon be able any more to face, this article presents a new approach for the design of digital signal processing IC. Having expressed the problem, we underline the respective limitations of the classic structures such as processors and FPGA, and we present hybrid architecture of these two families presenting a level of unprecedented performances. We detail the principles of dynamical reconfiguration on which our architecture is based, then present comparative results on a well known multimedia applications algorithm (DCT algorithm). Finally we describe the results obtained by fast prototyping and the current works concerning the problems of compilation targeting of our architecture.

Reconfigurable architecture, Processors, Digital Signal Processing, Architceture Compilation

1. introduction

L'évolution rapide et constante des technologies de la microélectronique jointe à l'explosion des marchés des technologies de l'information pose aujourd'hui des contraintes de conception sans précédents. Par exemple, l'avènement des réseaux de troisième génération, tel l'UMTS (Universal Mobile Telecommunication System) autorisant des bandes passantes conséquentes (jusqu'à 2 Mbit/s) ouvre la porte à des services nouveaux comme la diffusion de musique, de vidéo (streaming) ou encore la visioconférence. Les solutions architecturales actuelles (présentées par la suite) retenues par les concepteurs de terminaux mobiles ne seront plus à même de répondre aux contraintes de plus en plus fortes : aux impératifs actuels de coût et de consommation, viendront s'ajouter ceux de puissance de traitement et de flexibilité. Le multimédia est le dénominateur commun des applications qui sont et seront vouées à s'exécuter sur ces dispositifs. Image haute définition, son qualité CD, vidéo, visioconférence, téléphonie IP (VoIP : Voice over IP) sont autant de médias que ces architectures devront incontestablement être aptes à prendre en charge [JPEG].

Les architectures (micro-)électroniques actuelles dédiées au traitement du signal ne pourront pas aborder ces nouvelles applications, excessivement exigeantes en puissance de calcul, sans être profondément repensées. En effet, si les technologies silicium sont en évolution constante, ce n'est pas toujours le cas des paradigmes employés. Le modèle d'exécution du microprocesseur actuel est celui de Von Neumann [ARVI] [HARDENBERGH] et date des années soixante. Celui-ci est basé sur l'exécution séquentielle d'algorithmes et, bien que les fréquences de fonctionnement dépassent aujourd'hui les 2 GHz, la réduction du temps de cycle ne suffira pas à compenser les insuffisances de ce mode de fonctionnement. La figure 1 illustre les densités d'intégrations obtenues sur trois familles de circuits microélectroniques. Alors que les mémoires et les FPGA (*Field Programmable Gate Array*) subissent une croissance exponentielle du nombre de transistors dont ils sont constitués, les processeurs, eux, ne connaissent pas de véritable explosion de complexité : le modèle de Von Neumann qui est séquentiel, n'autorise qu'une forme de parallélisme relativement limitée et n'exploite donc pas les capacités potentielles de multiplication des unités de traitement autorisées par les nouvelles technologies. La figure 1 illustre cet état de fait sur trois familles de circuits que sont les mémoires, les FPGA et les microprocesseurs. Alors que les deux premiers profitent pleinement de l'évolution des capacités d'intégrations (loi de Moore, évolution exponentielle), le nombre de transistors par microprocesseur suit lui une loi plus lente.

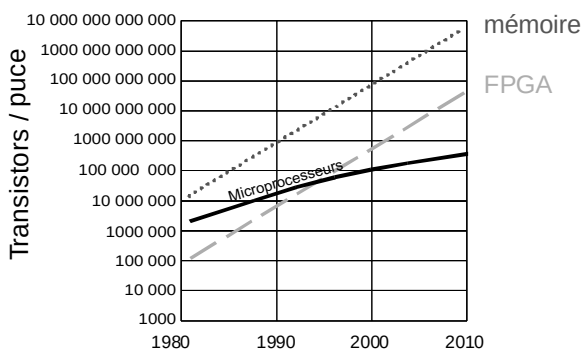

Figure 1. – Évolution des densités d'intégration.

La solution souvent retenue, quand les impératifs de performances sont particulièrement marqués, consiste en l'utilisation d'un cœur dédié (matériel ou logique câblée) conjointement à celle d'un processeur. Celui-ci est alors un accélérateur dédié au traitement des algorithmes chronophages des applications envisagées. Cette solution, bien que performante, n'est cependant pas compétitive sur un marché où les standards et les applications défilent à une cadence soutenue ; le cruel manque de flexibilité rend cette approche inadéquate.

Une approche alternative réside dans l'utilisation de circuits reconfigurables. Ceux-ci proposent un compromis intéressant entre flexibilité du processeur et performances de la logique câblée (ASICs). Les FPGA [BROWN][XILINX], circuits reconfigurables à granularité fine, sont les plus utilisés d'entre eux. Ces technologies sont arrivées à maturité ; des modèles de plusieurs millions de portes systèmes sont récemment apparus et sont, de ce fait, devenus des outils de choix notamment pour le prototypage de systèmes microélectroniques numériques complexes. Leur utilisation en tant qu'accélérateurs dans un contexte de système sur puce a souvent été proposée dans la littérature [HARTENSTEIN] et validée avec cependant des performances ou/et des coûts de mise en œuvre souvent prohibitifs.

Le paragraphe 2 sera consacré aux architectures dédiées au traitement du signal. Il détaillera de ce fait les différentes architectures de processeurs et FPGA mais aussi les architectures des composants reconfigurables alternatifs. Une étude des facteurs ayant un impact sur les performances et les capacités de l'ensemble de ces composants sera menée, et servira de base de réflexion pour l'élaboration de notre architecture.

Le paragraphe 3 sera consacré au Systolic Ring. Nous exposerons en détail son architecture, et donnerons quelques exemples de traitement illustrant ses atouts. Nous présenterons également les outils logiciels développés pour sa programmation.

Le paragraphe 4 présentera des résultats comparatifs d'implantation de la DCT [JPEG]. Le prototype réalisé utilisant une plate-forme de prototypage Altera SOPC (System On a Programmable Chip) [ALTERA] sera également exposé.

Le paragraphe 5 évoquera les principes que nous utiliserons afin de construire une chaîne logicielle de portage algorithmique (compilation) dédiée à cette architecture.

2. architectures dédiées au traitement du signal

La conception d'architectures vouées à répondre à un besoin de traitement passe impérativement par la notion de modèle d'exécution, ou paradigme [HARTENSTEIN]. Considérons un concepteur confronté à la réalisation d'un système complexe ; celui-ci devra impérativement choisir les composants de son système parmi les trois familles suivantes :

• Circuits dédiés (ASIC) [ARVI] ; dans ce cas le modèle d'exécution est structurel, concurrent et autorise l'exploitation du parallélisme potentiel de l'application. De ce fait, les performances sont aisément ajustables aux spécifications initiales, au détriment cependant de la flexibilité.

• Processeurs [SHIVA] ; dans ce cas le modèle d'exécution est procédural : c'est le paradigme de « Von Neumann » avec l'utilisation sous-jacente d'un contrôleur, et d'un chemin de données. Le paradigme de machine basée sur de la mémoire vive autorise une souplesse supérieure à celle des circuits dédiés, au détriment cette fois-ci des performances (exécution séquentielle).

• Composants reconfigurables [HARTENSTEIN] ; la présence de multiples unités fonctionnelles indépendantes autorise une exécution parallèle, le modèle d'exécution est donc structurel. Le paradigme de machine étant également basé sur de la mémoire vive, ceux-ci proposent un compromis attrayant entre performances de la logique dédiée et flexibilité des processeurs.

Dans la suite de cette partie, nous nous attacherons à qualifier les processeurs et les architectures reconfigurables actuellement utilisées, et à dégager des perspectives d'architectures nouvelles dédiées au traitement du signal. L'objet de cet article étant, entre autres, la recherche de structures proposant un compromis attrayant entre performances et flexibilité, les circuits dédiés à une application seront donc volontairement laissés de côté.

2.1. processeurs et DSP

Les processeurs sont basés sur le paradigme de Von Neumann [HARDENBERGH]. Ils se présentent sous la forme d'une architecture comportant deux éléments distincts : le contrôleur et le chemin de données. Le premier, appelé aussi séquenceur, se charge à chaque cycle de lire une instruction depuis la mémoire, puis d'appliquer au chemin de données la séquence de micro-instructions correspondante avant de passer à l'instruction suivante. L'ensemble des opérations est donc exécuté dans le chemin de données, et ce de manière séquentielle.

2.1.1. CISC et RISC

Les premiers processeurs apparus étaient appelés CISC (*Complex Instruction Set Computer*) [SHIVA]. De par la présence courante de centaines d'instructions, de multiples modes d'adressages, de multiples formats d'instructions, le contrôleur correspondant occupait une surface silicium imposante, usuellement entre 70 % et 90 % de la surface globale. Par ailleurs, le nombre moyen de micro-instructions nécessaires à l'exécution d'une instruction assembleur était important, d'où un temps de cycle machine accru et donc des performances relativement faibles. Sont ensuite apparus les processeurs RISC (*Reduced Instruction Set Computer*), comportant donc un jeu d'instruction largement réduit. La présence d'un, voire deux, modes d'adressage et d'opérations simples, a abouti à une surface silicium du contrôleur largement inférieure à celle d'un processeur standard, et des performances en hausse : la *granularité* des instructions, plus fine, autorise l'exécution de n'importe quelle instruction en un ou deux cycles d'horloge. Un corollaire direct est la taille du code généré : pour l'exécution d'une tâche donnée, un nombre plus important d'instructions RISC est nécessaire comparativement à son équivalent CISC.

2.1.2. processeurs superscalaires

Les appétits grandissants en performances des nouvelles applications ont motivé l'apparition d'architectures autorisant l'exploitation du parallélisme au niveau instruction (*ILP : Instruction Level Parallelism*) [SHIVA]. Ainsi, par opposition aux processeurs scalaires (exécutant une unique instruction par cycle machine), les processeurs superscalaires sont capables d'en exécuter plusieurs, par déduction dynamique des dépendances de données, tâche prise en charge par le contrôleur (et non par le compilateur). Plusieurs pipelines de traitement sont donc présents, ainsi que plusieurs unités de traitement. Les performances obtenues sur ces processeurs sont généralement supérieures à leurs homologues scalaires au prix toutefois d'une surface silicium bien supérieure pour gérer les nécessaires déductions de dépendances, autorisant l'exécution simultanée de plusieurs instructions non nécessairement adjacentes (*OOO : Out Of Order execution*).

2.1.3. processeurs VLIW

L'architecture VLIW est de plus en plus utilisée dans les DSP actuels [DEMIGNY] : celle-ci est également apte à effectuer du parallélisme au niveau instruction. La différence essentielle réside dans le fait que la construction du graphe de flot de données est effectuée à la compilation : le parallélisme au niveau instruction n'est donc pas pris en charge au niveau matériel mais au niveau logiciel. Une instruction VLIW est constituée de l'agrégation de plusieurs instructions RISC (figure 2), chacune s'exécutant dans une unité dédiée.

Les performances proposées par les architectures VLIW sont potentiellement élevées, mais les compilateurs ciblant ces architectures arrivent rarement à atteindre des performances opti-

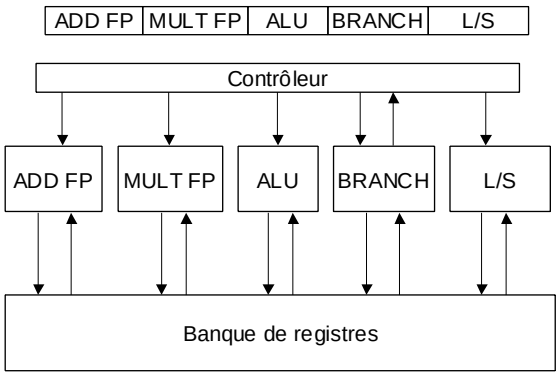

Figure 2. – Architecture de base de processeur VLIW.

males compte tenu de la complexité de l'architecture. Des rapports de performances allant jusqu'à 20 [LEE] sont constatés entre une description en langage C compilée et un code assembleur optimisé [TEXAS]. À noter qu'il est souvent possible de programmer la structure en assembleur linéaire (comme si l'architecture cible était scalaire), le compilateur déployant ensuite le parallélisme au niveau VLIW, avec des performances souvent élevées dans ce cas de figure.

2.1.4. architectures SIMD

Toujours dans le but d'accroître les performances dans les applications orientées flot de données, telles que les applications multimédia, des processeurs ou des unités intégrées à des processeurs SIMD (*Single Instruction Multiple Data*) sont apparus. Le principe est simple : chaque unité de traitement reçoit ainsi une même séquence de micro-instructions, issue d'une même unité de contrôle. Chaque unité de traitement échange des données avec une banque mémoire propriétaire. Dans ce type de structures, une unique instruction est appliquée à différentes données. Les extensions MMX et SSE d'Intel [INTEL] en sont un exemple : ces instructions supplémentaires sont vouées à accélérer le traitement des algorithmes multimédia, comme le traitement d'image 2D ou 3D ou encore le traitement du son. Néanmoins, aucun compilateur ne les prend en charge, et seul l'utilisation explicite de bibliothèques propriétaires autorise l'utilisation des unités SIMD.

2.2. architectures reconfigurables

Les évolutions exposées ci-dessus (superscalaires et VLIW) n'autorisent l'exploitation que d'une forme de parallélisme limité au niveau instruction. L'accélération est donc de ce fait relativement restreinte, et ne peut se comparer à celle fournie par des circuits dédiés autorisant l'intégration de chemins de données complexes et pipelinés.

Les architectures reconfigurables, quant à elles, permettent la synthèse de tels chemins de données et donc d'ajuster finement les performances du système intégré aux besoins. Leur flexibilité autorise par ailleurs de modifier la fonctionnalité comme dans le cas de processeurs.

Les architectures reconfigurables se distinguent entre elles par trois caractéristiques essentielles qui sont la granularité, l'architecture de la couche opérative et le mode de configuration.

Une architecture reconfigurable peut être schématisée par deux couches distinctes comme illustré figure 3 :

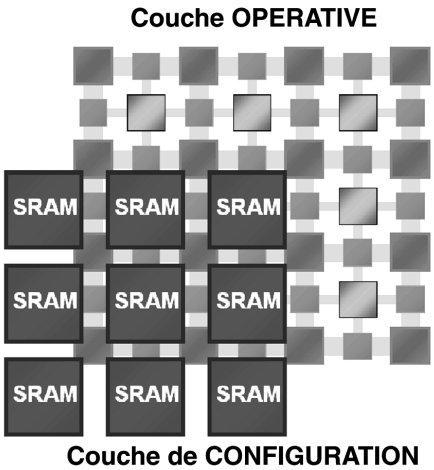

Figure 3. – Structure bi-couche des composants reconfigurables.

• La couche de configuration est un élément mémoire contenant la configuration de l'architecture à un moment donné. C'est elle qui détermine la fonctionnalité implémentée dans le réseau. Dans le cas d'architectures reconfigurables dynamiquement, le contenu de cette couche est modifié en cours même de traitement.

• La couche opérative est constituée d'un arrangement, le plus souvent régulier, d'unités fonctionnelles opérant sur les données incidentes au circuit. Le type des unités fonctionnelles (granularité) ainsi que la topologie de l'arrangement (architecture de la couche opérative) varie d'une réalisation à l'autre.

Granularité

La granularité est une caractéristique essentielle des réseaux reconfigurables (RA : *Reconfigurable Array*). Elle qualifie la donnée atomique que ces architectures sont capables de manipuler. Les FPGA, qui incarnent les architectures reconfigurables les plus connues, sont à grain fin, au niveau *bit*, ce qui leur procure cette si grande flexibilité : il est possible d'émuler le fonctionnement de n'importe qu'elle structure numérique.

Architecture de la couche opérative

Ici encore, plusieurs sous-familles de topologies existent. Le choix d'une structure donnée a un impact majeur sur la flexibilité et la scalabilité de l'architecture. On peut distinguer trois types majeurs d'arrangements :

• « *Mesh-based RAs* ». [HARTENSTEIN] Ce sont des arrangements réguliers bidimensionnels avec des connexions horizontales et verticales autorisant un routage efficace et dense. Ces architectures encouragent fortement les communications entre voisins (NN pour « *Nearest Neighbour* ») des CFB (*Configurable Functionnal Block*). Les communications sur de plus longues distances sont prises en charge par des lignes dédiées, voire des bus.

• « *Linear Arrays RAs* ». [KUNG] Ce sont des arrangements uni- ou bi-dimensionnels de CFB en série ; les communications sont essentiellement uni-directionnelles. Ces structures sont essentiellement vouées à l'implantation d'algorithmes à caractère flot de données (chemins de données pipelinés). Les divergences dans ces flots sont exclusivement gérées par les réseaux bi-dimensionnels.

• « *Crossbar-based RAs* ». [HARTENSTEIN] Ce sont des structures basées sur des matrices d'interconnexions ou *crossbar switches* globaux ou locaux gérant les interconnexions à l'échelle du circuit. Ce genre de composant est déjà utilisé dans les FPGA où il prend en charge l'ensemble du routage. Ceux-ci sont souvent consommateurs de surface de silicium, et le délai de propagation d'une réalisation à l'autre n'est généralement pas maîtrisable ce qui pénalise la fréquence de fonctionnement du circuit d'un algorithme à l'autre.

Mode de configuration

Trois modes de reconfiguration sont à distinguer :

• Configuration statique : les circuits reconfigurables de cette catégorie conservent leur configuration inchangée durant toute la phase de fonctionnement. Une majorité de FPGA fonctionne suivant ce mode. Ce type de reconfiguration se qualifie par *une architecture par application.*

• Configuration pseudo-dynamique : un nombre fini de configurations locales ou globales est stocké dans une mémoire auxiliaire et le traitement des données s'interrompt durant un temps relativement court (inférieur à 1/10 s) pour reconfigurer la structure avant de reprendre le traitement. Cette phase s'accompagne généralement d'un stockage des résultats intermédiaires dans des mémoires tampons. Ce type de reconfiguration se qualifie par *plusieurs architectures par application*.

• Configuration dynamique : tout ou partie de l'architecture peut être modifiée en cours même de traitement sans nécessiter l'interruption du traitement de quelque durée que ce soit. Dans ce cas de figure l'*architecture est une fonction continue du temps*.

2.2.1. les architectures reconfigurables à grain fin

Les architectures reconfigurables à grain fin, comme les FPGA manipulent des données à l'échelle du bit. La réalisation des fonctions logiques est souvent obtenue par des générateurs de fonctions, ou LUT (*Look-Up Table*). Le principe est simple, un ensemble de points mémoires est démultiplexé à travers un arbre de multiplexeurs, les entrées d'adresses de ceux-ci constituant les entrées du circuit. La figure 4 expose la structure d'un CLB (*Configurable Logic Block*) de la famille Xilinx XC4000.

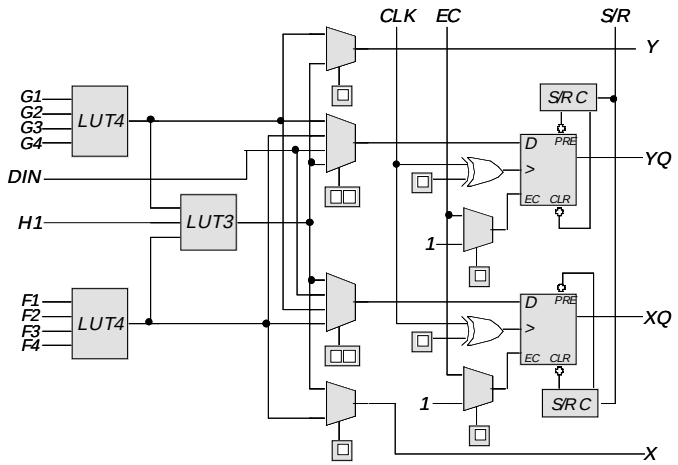

Figure 4. – Structure d'un CLB de FPGA Xilinx.

Le routage entre les différents CLB du FPGA est assuré par des matrices d'interconnexions nommées PSM (*Programmable Switch Matrix*) chez Xilinx. L'élément de base de la PSM est une cellule à 6 transistors, autorisant une large gamme de routages différents (Figure 5).

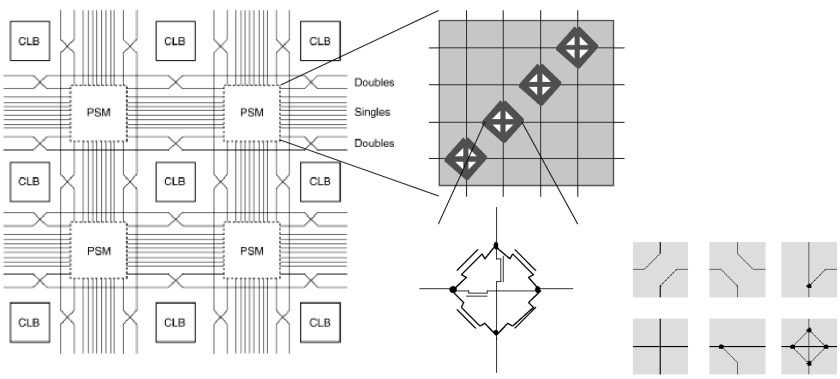

Figure 5. – Structure d'une matrice d'interconnexion de FPGA Xilinx.

Si les FPGA, de par leur constitution, s'avèrent très souples, ils souffrent de problèmes majeurs qui sont majoritairement dus à la technologie reconfigurable elle-même. Le critère de Dehon [DEHON] permet de comparer des architectures implantées sur différentes technologies silicium. Comme le facteur d'échelle d'une technologie à l'autre reste constant, il est possible d'obtenir des résultats cohérents, exprimés en λ^2. λ représente la moitié de la taille du motif minimal admissible sur une technologie donnée, soit la demi-largeur du canal d'un transistor MOS aux dimensions minimales. Le tableau 1 expose en λ^2, selon le critère de Dehon [DEHON], les surfaces requises par 3 composants majeurs de FPGA actuels : la mémoire de configuration, la logique active (arbre de multiplexeurs du générateur de fonctions) et les diverses ressources d'interconnexions. Si on considère que la surface totale est la somme des surfaces de chacun de ces éléments, il apparaît que plus de 97 % de la surface totale est occupée par la mémoire de configuration et les ressources d'interconnexions, ce qui qualifie le surcoût dû à l'utilisation d'une technologie reconfigurable.

Tableau 1. – Surface des différents composants d'un FPGA.

Fonction	Aire (λ^2)
LUT Mux + bascule	20 K
Mémoire de configuration	80 K
Interconnexion	700 K

La figure 6 [HARTENSTEIN] reprend ces données et distingue FPGA physique et FPGA logique. La première courbe, illustrant le FPGA physique exprime le nombre réel de transistors dont est constitué un FPGA. La seconde courbe, quant à elle, illustre le FPGA logique, c'est-à-dire le nombre de transistors équivalents réellement disponibles d'un point de vue concepteur ciblant une technologie FPGA.

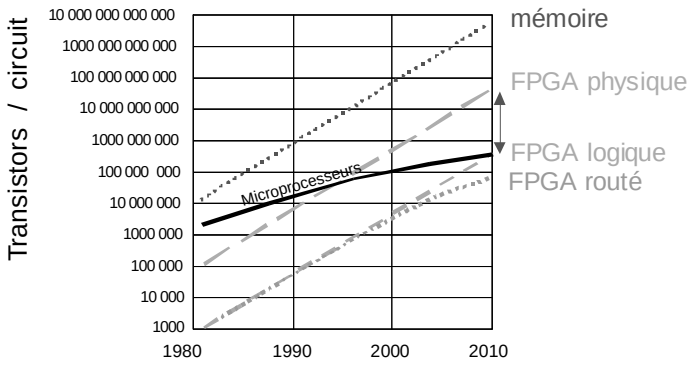

Figure 6. – Densités des FPGA physique, logique, et routé.

De manière comparable, la présence d'une architecture de routage basée sur des matrices d'interconnexions diminue également la capacité maximale disponible pour un FPGA donné, comme illustré figure 6. Il s'avère en effet difficile de router chacune des connexions de l'architecture intégrées dans le FPGA quand le taux de remplissage est important. Ainsi, dans les modèles de FPGA actuels de plusieurs millions de portes équivalentes, il s'avère souvent impossible de dépasser 90 % d'utilisation des ressources logiques.

Une architecture de routage à base de matrices d'interconnexions comporte également un autre inconvénient majeur, qui est celui du délai de propagation : un signal traversant de multiples matrices d'interconnexions impliquera un délai combinatoire important. C'est ce phénomène qui limite bien souvent les fréquences de fonctionnement des FPGA, et qui rend les phases de placement-routage si longues, tout particulièrement lorsqu'on approche la capacité maximale du composant : de multiples itérations de placement-routage sont nécessaires pour trouver une solution ne dégradant pas les performances de manière trop importante. Le tableau 2 présente les caractéristiques de routage d'un FPGA de la famille 10KV de chez Altera. Dans un cas standard de routage à travers un canal vertical (*row*) et un canal horizontal (*column*), 87 % des retards proviennent des ressources de routage.

Tableau 2. – Délai dû au routage dans un FPGA de famille Altera 10KV.

FPGA	Chemin de routage	Délai total	Délai LUT	Interconnexions
Altera 10K130V-2	LUT-LUT (local)	2.5 ns	2.1 ns	16 %
	LUT-ligne-LUT	6.6 ns	2.1 ns	68 %
	LUT-colonne-LUT	11.1 ns	2.1 ns	81 %
	LUT-ligne-colonne-LUT	15.6 ns	2.1 ns	87 %

C'est ce délai, majoritairement dû à l'architecture en matrices d'interconnexions, qui est à l'origine des faibles fréquences de fonctionnement obtenues comparativement à des circuits dédiés ou des processeurs. L'avantage de l'exécution spatiale est en partie perdu, ce qui explique que des solutions à base de FPGA ne sont pas systématiquement plus performantes que des solutions uniquement logicielles. Il est à noter que sont apparus récemment des FPGA intégrant directement des blocs dédiés au traitement du signal, comme des multiplieurs câblés ou encore des cœurs de processeurs ou DSP comme le Virtex II Pro de Xilinx [XILINX]. Ces solutions allient donc flexibilité et performances et bien que les fréquences de fonctionnement envisageables dans la logique à grain fin ne puissent être très importantes, ces solutions sont néanmoins très prometteuses.

2.2.2. architectures reconfigurables à grain épais

Dans le but de limiter le surcoût dû à l'utilisation d'une technologie reconfigurable certaines approches ont pris le parti de ne plus se baser sur des CFB à grain fin, mais à grain épais. Ces unités ne manipulent plus des données à l'échelle du *bit*, mais du *mot* (multiple de l'octet). Elles intègrent donc le plus souvent des opérateurs arithmétiques câblés tels que multiplieurs et additionneurs, mais aussi des registres dédiés au stockage de résultats intermédiaires.

Dans les dix dernières années, de nombreux travaux ont été menés dans le domaine des architectures reconfigurables. Nous ne détaillerons pas pour des raisons de concision l'ensemble des approches existantes. Les principales approches proposées au niveau universitaire et industriel depuis 10 ans sont répertoriées dans [HARTENSTEIN]. Ces architectures présentent des niveaux de performances systématiquement supérieurs aux FPGA dans le domaine du traitement du signal, avec bien évidemment un compromis fait sur la flexibilité.

À titre d'exemple représentatif, KressArray de l'université de Kaiserlautern [KRESS] est un réseau systolique de type « Mesh-based ». Des bus parcourant l'ensemble des unités fonctionnelles sont néanmoins disponibles pour d'éventuelles communications entre unités non voisines et pour la propagation du contrôle. Ce réseau fut à l'origine de la réalisation du MoM-3 Xputer [NAGELDINGER] : l'ajout d'une unité de contrôle supplémentaire rendit cette architecture apte à supporter tous les opérateurs du langage C. Cette structure est également dynamiquement configurable au niveau local. Cependant, pour des problèmes de complexité de gestion, aucune des applications portées jusqu'à maintenant n'en fait usage.

Synthèse de l'état de l'art

Les architectures reconfigurables présentées dans ce paragraphe et le « *reconfigurable computing* » en général se profilent réellement comme des solutions alternatives aux approches classiques, procurant des puissances de traitement potentiellement supérieures aux microprocesseurs conventionnels. Le choix des multiples structures disponibles doit cependant se faire de manière réfléchie pour ne pas tomber dans les travers des architectures reconfigurables à grain fin pour lesquels le surcoût d'utilisation par rapport à une technologie silicium reste prohibitif (facteur 50 à 100), avec des performances souvent dégradées.

Il est à noter que ces architectures sont inexploitables si des outils de programmation efficaces ne leur sont associés. Le succès de l'industrie du logiciel ces 20 dernières années est probablement dû en grande partie à la disponibilité de compilateurs, autorisant la génération automatique de la configuration de l'architecture (le programme) à partir d'une description séquentielle de haut niveau (langage C par exemple). Ainsi, l'utilisation des ordinateurs actuels ne nécessite pas de connaissance intime de l'architecture de la machine, ce qui n'est pas le cas pour les composants reconfigurables où aucun compilateur efficace n'est disponible. La capacité d'exécution spatiale de ces architectures en est à l'origine, et la réalisation d'un outil apte à porter efficacement une description séquentielle vers une architecture parallèle n'est pas triviale. Les déductions de dépendances et la construction du graphe de flot de données doit être prise en charge automatiquement, et la relative inefficacité de compilateurs ciblant des architectures à niveau de parallélisme limité (processeurs *VLIW*) laisse présager de nombreuses difficultés. Les caractéristiques, propres à chaque architecture reconfigurable considérée, influent largement dans la réalisation de tels outils, notamment en ce qui concerne le mode de reconfiguration (statique, dynamique).

De même, l'avènement de nouvelles technologies silicium doit pouvoir augmenter les performances envisageables d'une AR (Architecture Reconfigurable) d'un ordre de grandeur équivalent à l'accroissement des capacités d'intégrations ; toute architecture pour laquelle un problème donné croîtra (routage, etc.) de manière non linéaire sera à proscrire.

De ces réflexions, nous avons tiré les impératifs suivants :

– une AR doit être de granularité en rapport avec les applications visées : nous optons donc pour une granularité épaisse, au niveau arithmétique, pour notre domaine d'application du traitement numérique du signal, ramenant ainsi le coût relatif dans des proportions acceptables.

– nous n'optons pas pour un réseau super systolique ou plus généralement « Mesh-based », ou encore basée sur des matrices d'interconnexions ; bien que leur architecture soit excessivement souple, les complexités de gestion associées sortent du cadre du traitement numérique du signal. Notre choix se portera donc sur une architecture linéaire, c'est-à-dire avec des capacités de routage essentiellement unidirectionnelles. Les insuffisances rencontrées dans les architectures linéaires de la littérature étant essentiellement liées aux nécessités de communication des algorithmes divergents ou récursifs, nous proposerons une architecture particulière, non plus basée sur des bus, permettant d'apporter une solution à ce problème critique.

– la reconfiguration dynamique sera retenue pour la flexibilité de gestion du matériel qu'elle autorise ; cependant, à cause des problématiques de portage constatées dans la littérature, nous la développerons dans le soucis constant de la faisabilité des outils de compilation.

3. architecture du « Systolic Ring »

3.1. principe de l'approche

Les limitations des FPGA soulignées précédemment ont orienté les choix technologiques vers un nouveau type de cellule : notre

approche consiste en premier lieu à implémenter un réseau reconfigurable à granularité épaisse qui soit également apte à effectuer des manipulations bit à bit sur des mots. La donnée atomique manipulée par les applications orientées traitement du signal et flot de données étant essentiellement le *mot* par opposition au *bit*, c'est un bloc orienté traitement arithmétique que nous retenons.

3.2. architecture de la couche opérative

3.2.1. architecture de la brique de base : Le Dnode

3.2.1.1. chemin de données du Dnode

Le cœur de la brique reconfigurable de base, ou *Dnode*, est donc une ALU (*Arithmetic and Logic Unit*) proposant toutes les fonctions logiques et arithmétiques classiques comme addition, soustraction mais aussi multiplication. Nous lui adjoignons également une banque de registres qui sera principalement utilisée pour le stockage des résultats de calculs intermédiaires. La taille des mots manipulés choisie est de 16 bits, et s'adapte de ce fait à un grand nombre d'applications du traitement du signal. Dans un contexte applicatif ce choix doit néanmoins s'effectuer précautionneusement, car il fixe une limitation intrinsèque de l'architecture. La figure 7 détaille l'architecture globale du *Dnode*. Celle-ci a été développée dans un soucis d'efficacité ; l'ensemble des opérations disponibles s'exécutant en un unique cycle d'horloge. Les opérateurs de l'ALU sont complètement combinatoires et la structure de l'ensemble des éléments séquentiels (banque de registres et registre de sortie) est de type maître-esclave autorisant ainsi simultanément une lecture et une écriture de leur contenu. Dans l'objectif d'accélérer l'exécution des opérations de type multiplier-accumuler, souvent utilisées dans les applications TSI, une opération spéciale MAC (Multiplication-ACcumulation) a été ajoutée au jeu d'instructions ; celle-ci autorise le chaînage purement combinatoire du multiplieur et de l'additionneur. Cette instruction permet donc l'exécution d'une somme de produits $\sum(a_i.b_i)$ de n termes en n cycles d'horloge.

3.2.1.2. contrôleur local du Dnode

L'utilisation d'une logique à granularité épaisse, si elle optimise l'exécution d'algorithmes à dominante arithmétique, comporte tout de même des limitations : toute forme de réalisation au niveau bit est exclue. Les architectures au sens général sont modélisables sous un formalisme contrôleur/chemin de données. Le chemin de données est dans notre cas directement présent dans les cellules de base de notre architecture mais le contrôleur, systématiquement réalisé sous forme de machines d'états, nécessite impérativement la réalisation de logique au niveau bit, hors de portée des architectures à granularité épaisse. Toutes les opérations de contrôle doivent alors être prises en charge au niveau configuration (dynamiquement), ce qui conduit rapidement à une surcharge de contrôle saturant les structures de gestion de la configuration. C'est ce type de limitations intrinsèques dues au principe même des architectures reconfigurables à granularité épaisse qui a motivé l'extension du principe de reconfiguration dynamique par la définition de multiples modes de fonctionnement. Le **mode local** est dédié à la gestion des opérations mettant en jeu des modifications permanentes de la configuration de *Dnodes* : chaque *Dnode* peut se comporter comme une unité autonome par l'adjonction d'un petit contrôleur dédié au séquencement de la configuration, il s'apparente donc à un petit processeur.

Chaque *Dnode* se voit donc doté d'un séquenceur de 8 instructions et d'une petite MEF (machine d'états finis) pour contrôler l'ensemble. Ainsi transformé, le *Dnode* est apte à modifier sa

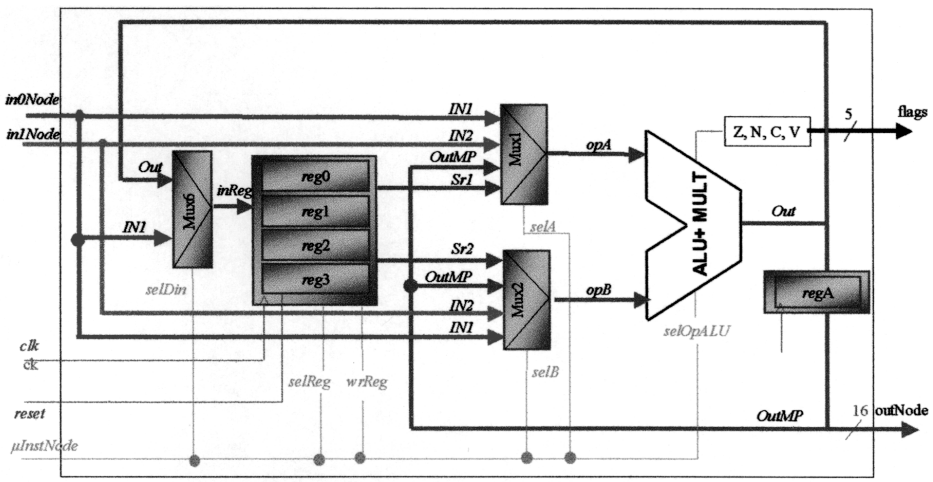

Figure 7. – Architecture du *Dnode*.

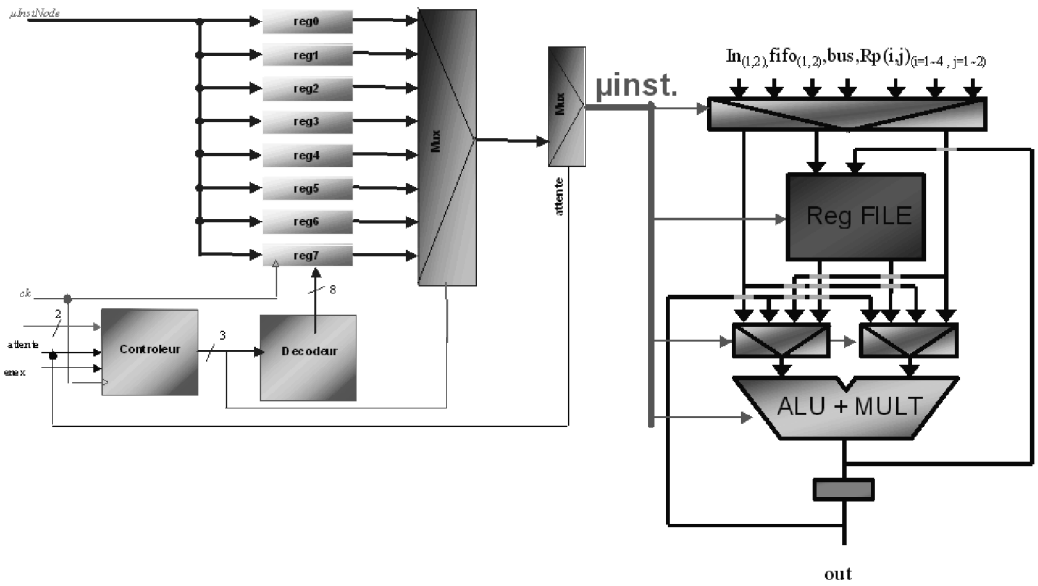

Figure 8. – Le *Dnode* avec son contrôleur local.

configuration à chaque cycle d'horloge (il se comporte alors comme un « nano-processeur », processeur avec un jeu restreint d'instructions) et donc à exécuter de manière autonome des opérations hautement répétitives du type filtres RIF, RII, opérations trigonométriques (développements limités) mais aussi lignes à retard, etc... Le séquenceur peut commander l'exécution en boucle ou non de micro-programmes constitués de 8 instructions au maximum, sachant que la partie opérative du *Dnode* autorise l'exécution de l'ensemble des opérations disponibles en un seul cycle d'horloge. L'impact en surface est minimal et se résume pour chaque *Dnode* à une dizaine de registres et à un peu de logique combinatoire. Le principe de fonctionnement du mode local est schématisé sur la figure 8. Trois versions du mode local sont disponibles :

– ***Fixed*** : le *Dnode* applique la même instruction aux données incidentes, c'est-à-dire celle qui est stockée dans le premier registre (reg0).

– ***One-way*** : le *Dnode* exécute son microprogramme, puis s'arrête une fois arrivé à l'adresse de fin, mémorisée dans un registre dédié du *Dnode*.

– ***Loop*** : le *Dnode* exécute indéfiniment son microprogramme en boucle ; une intervention du contrôleur de configuration exposée plus loin est nécessaire pour stopper l'exécution.

3.2.2. structure du réseau systolique

3.2.2.1. flot direct

Concernant la structure de la couche opérative, nous choisissons une structure linéaire, conformément au cahier des charges présenté à la fin du deuxième paragraphe. Les *Dnodes* seront ainsi organisés en couches et la propagation des données sera unidirectionnelle. Chaque *Dnode* étant doté de son propre contrôleur local, il s'avère possible dans cette structure d'effectuer du parallélisme d'instructions, à savoir d'appliquer simultanément différentes instructions à différentes données. Pour parer aux limitations classiques des architectures linéaires exposées précédemment, la couche de traitement – ou couche opérative – a une architecture particulière. La structure réalisée est de type linéaire bi-dimensionnel (plusieurs *Dnodes* par couche) et rebouclée, autorisant donc un unique sens de propagation des données. Celui-ci prend donc la forme d'un anneau dans le but

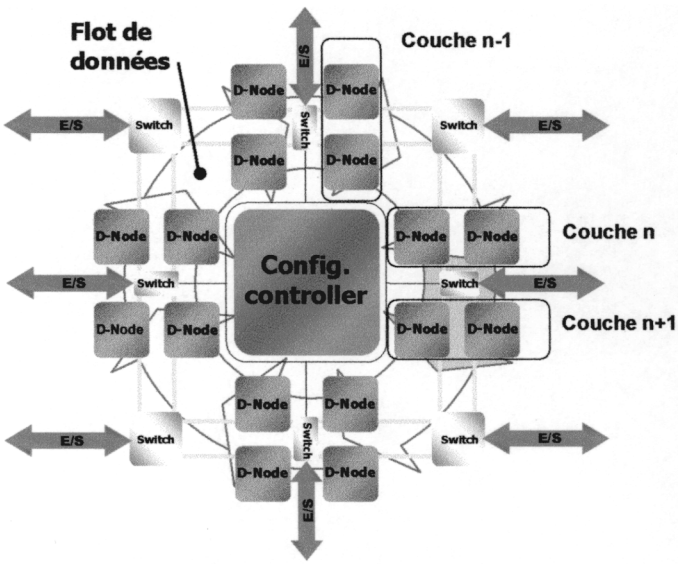

Figure 9. – L'architecture en anneau.

de faciliter le transfert des données ; le flot de données est « tournant ». Alors que le degré de parallélisme est directement lié au nombre de *Dnodes* par couche, le degré de pipeline, lui, n'est pas limité par l'utilisation d'une structure connexe. Les *Dnodes* sont organisés en couches, chacune étant reliée à la précédente et la suivante par le biais d'un composant spécialisé : le ***Switch*** (figure 9). Il se charge des injections de données dans la structure mais aussi du routage de la couche amont vers la couche aval. Il est dynamiquement configurable et gère ainsi les transferts de données avec le processeur central.

Le fait d'utiliser une structure rebouclée comporte de nombreux avantages, et élimine notamment tout problème de profondeur du chemin de données à implanter ; les données peuvent ainsi faire autant de tours que le nécessite le traitement avant d'être renvoyées vers le processeur hôte.

3.2.2.2. flot récursif

L'architecture présentée en figure 10 est uni-directionnelle, tout routage de données en sens inverse du flot est interdit [HARTENSTEIN] ; c'est le problème dont souffrent toutes les architectures linéaires. Cette incapacité fait que ces structures se prêtent généralement mal à la réalisation de flots de données récursifs dont les algorithmes de traitement du signal font un usage intensif (filtres RII…). Les approches classiques solutionnent en partie ce problème par l'utilisation de ressources partagées [HARTENSTEIN] comme des bus (figure 10).

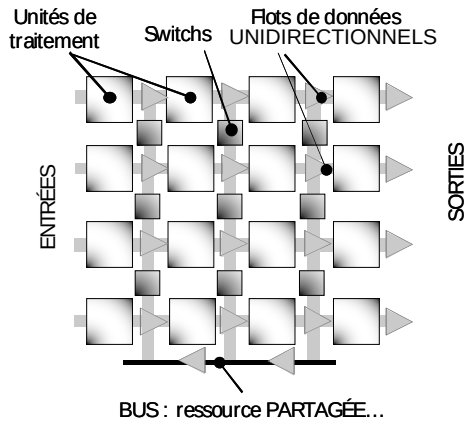

Figure 10. – Les architectures linéaires.

Ces solutions sont bien évidemment limitatives, dégradant de manière quasi-systématique les performances temporelles (latences importantes). L'avènement de technologies d'intégration toujours plus performantes autorisant la création de réseaux de capacité toujours plus grande rendra, à n'en pas douter, ce problème de routage de plus en plus critique.

La solution que nous avons retenue pour répondre à ces problèmes consiste en l'emploi d'un second chemin de données allant à contresens du premier. Nous l'appelons le réseau de rétro-propagation : chaque *switch* possède son propre pipeline

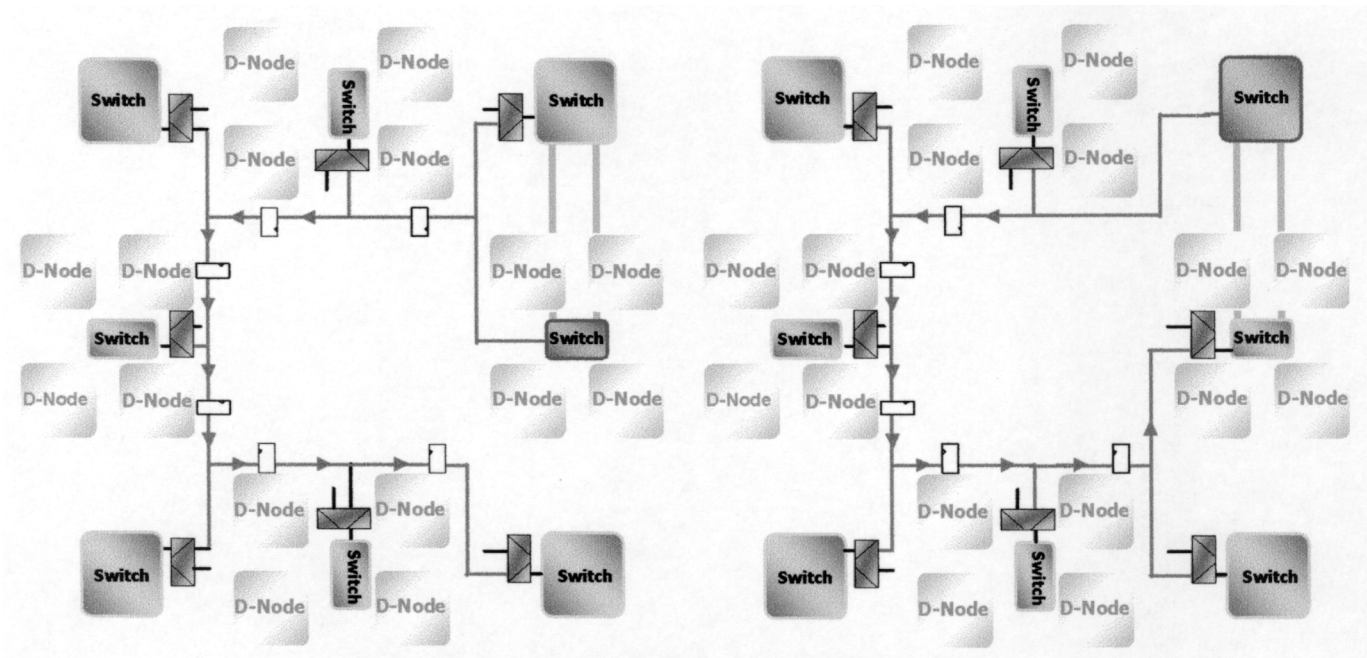

Figure 11. – Le réseau de rétro-propagation.

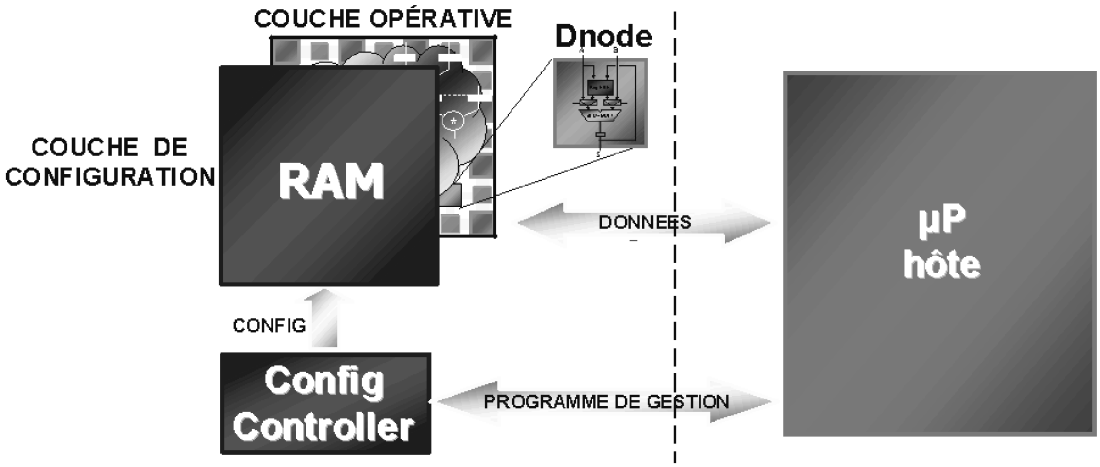

Figure 12. – L'architecture globale.

dans lequel il écrit de manière systématique les données traitées en provenance de la couche de *Dnodes* amont. Tous les autres *switchs* de l'architecture ont un point de lecture sur ce pipeline et peuvent ainsi acquérir les données y transitant. La figure 11 schématise les pipelines de deux des 8 *switchs* d'une version à 16 *Dnodes*.

3.3. architecture de la couche de configuration

3.3.1. principe de fonctionnement

Le principe de fonctionnement est exposé schématiquement en figure 12. Le contrôleur de configuration est un processeur de type RISC (*Reduced Instruction Set Computer*) avec un jeu d'instructions hybride pour être capable de contrôler la structure reconfigurable. Le Systolic Ring évolue sous le contrôle d'un processeur hôte qui lui communique pour chaque tâche envisagée un programme dédié de gestion dynamique de la configuration. Une fois ce programme chargé, l'hôte commande l'exécution de ce programme et peut débuter l'envoi des données à traiter vers la structure reconfigurable.

3.3.2. modes de fonctionnement

Trois modes distincts d'exécution sont disponibles :

– ***Mode local*** : comme exposé précédemment, chaque *Dnode* est doté d'un contrôleur local, et peut gérer sa configuration de manière dynamique et autonome, au même titre qu'un processeur standard. Dans ce mode de fonctionnement de la structure, le contrôleur de configuration intervient dans la phase d'initialisation de la structure en programmant chaque contrôleur local de chaque *Dnode*. Une fois que tous les contrôleurs locaux sont chargés, le contrôleur de configuration commande le passage en mode local de chacun des *Dnodes*.

– ***Mode global*** : le contrôleur de configuration étant apte à modifier la configuration en cours même de traitement, il peut assumer seul la reconfiguration des *Dnodes*. Dans ce cas, le contrôleur local des *Dnodes* est en mode figé (*fixed*), et le premier registre de chacun de ceux-ci fait office d'élément mémoire stockant la configuration. De manière synthétique, à chaque cycle d'horloge, la configuration d'une couche de *Dnodes* est modifiée sur front montant, le traitement étant synchronisé sur front descendant.

– ***Mode hybride*** : le contrôleur de configuration commandant l'évolution de la totalité de la structure, il peut imposer à un sous-ensemble des *Dnodes* un fonctionnement en mode local (*one-way ou loop*), et aux autres un fonctionnement en mode global.

3.4. exemples d'utilisation

3.4.1. mode local

À titre d'exemple illustrant l'intérêt du mode local, considérons le calcul polynomial suivant :

$$P(x) = a.x + b.x^2 + c.x^3$$

Ce type de calcul est intensément utilisé dans le calcul d'opérations mathématiques complexes, comme les fonctions trigonométriques (développements limités). Une implantation possible dans un *Dnode* en mode local est exposée figure 13. Le calcul complet du polynôme est effectué en seulement 5 cycles d'hor-

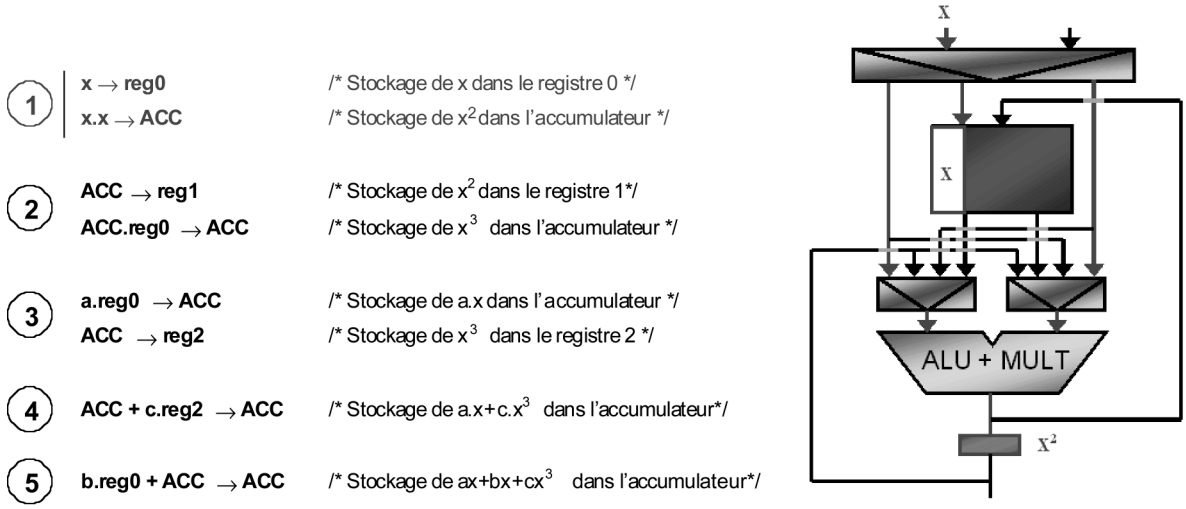

Figure 13. – Premier cycle du calcul polynomial.

loge, et ce, de manière totalement autonome. Le code micro-instruction appliqué au *Dnode* change à chaque cycle, ce qui souligne l'intérêt du mode local : aucune intervention du contrôleur de configurations n'est nécessaire. La présence d'une instruction MAC (multiplier-accumuler, cycles 4 et 5) permet de plus d'accélérer le traitement et ainsi d'économiser deux cycles d'horloge par rapport à une implantation classique n'utilisant que des additions et multiplications. La figure 13 détaille le programme ainsi que l'exécution du premier cycle sur un *Dnode*.

3.4.2. réseau de rétro-propagation

Le réseau de rétro-propagation implanté dans l'architecture contribue de manière significative à la flexibilité de la structure. Nous allons illustrer son intérêt par l'implantation d'un filtre à réponse impulsionnelle finie (RIF). Les échantillons de sortie de ce type de filtres sont de la forme :

$$y_n = \sum_{i=0}^{N-1} ai.x_{n-i-1}$$

Nous retiendrons pour l'exposé un filtre d'ordre 3, ainsi qu'une version du Systolic Ring à 3 *Dnodes* par couche. Les résultats exposés seront cependant aisément transposables pour des filtres de dimensions quelconques, et ce, sur des versions quelconques du Systolic Ring. Il existe de nombreuses façons d'implanter de manière performante (mode local) un tel filtre dans le Systolic Ring ; nous en exposons une afin de bien montrer l'intérêt du réseau de rétro-propagation.

L'architecture optimisée du *Dnode* permet d'effectuer des accumulations de résultats avec des sommes partielles non présentes dans le registre d'accumulation ; ce qui autorise l'implantation d'une version *pipelinée* du filtre présenté. Celui-ci est détaillé en figure 14, 15 et 16.

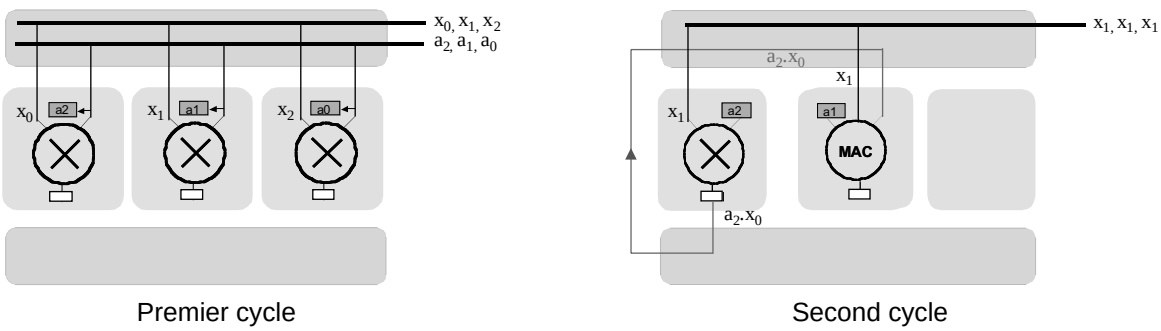

Figure 14. – Implantation pipelinée (1).

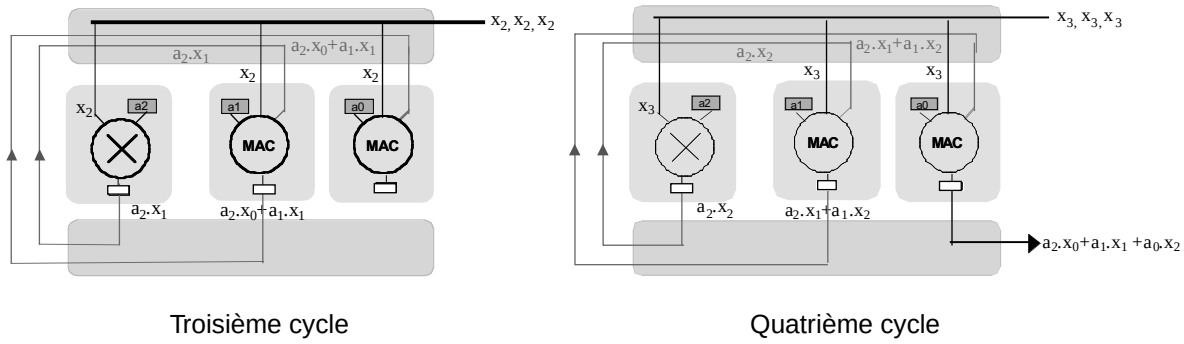

Figure 15. – Implantation pipelinée (2).

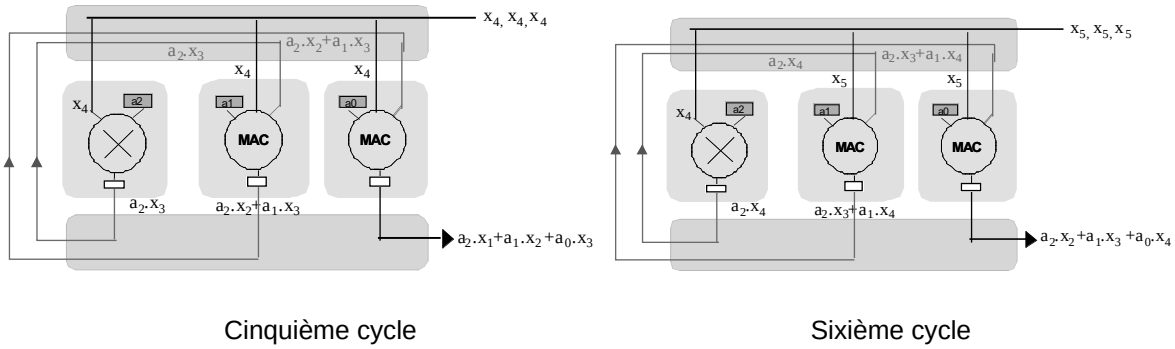

Figure 16. – Implantation pipelinée (3).

À partir du quatrième cycle (figure 15), la structure est capable de fournir un échantillon par cycle, ce qui constitue une implantation optimale compte tenu des dépendances de données. Les cycles cinq et six illustrent ce fait. L'implantation de filtres de dimensions supérieures au nombre de *Dnodes* par couche ne présente pas non plus de problème : dans l'exemple exposé ci-dessus la rétro-propagation s'effectue sur un seul niveau mais il est également possible de l'envisager sur plusieurs niveaux. La figure 17 expose l'implantation d'un filtre de dimension 6 dans la structure. Dans ce cas de figure, le niveau de performances obtenu est également maximal, un échantillon étant calculé par cycle.

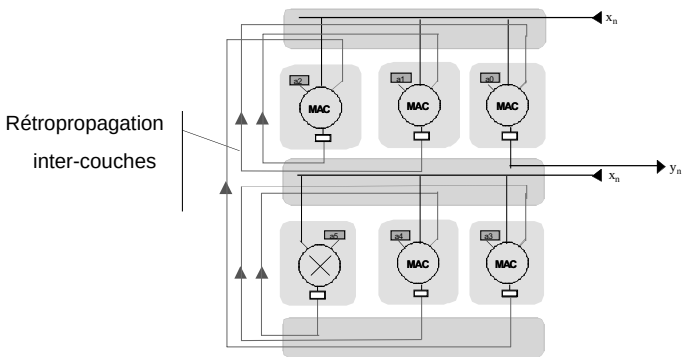

Figure 17. – Implantation pipelinée sur plusieurs couches.

3.5. synthèse

La structure exposée présente les caractéristiques suivantes pour une version à 8 *Dnodes* au total et 2 *Dnodes* par couche, en technologie CMOS 0.18 µm : puissance crête de 1600 MIPS à 200 MHz, bande passante de traitement de 3 Go/s (4 données sur 16 bits injectées par couche et par cycle d'horloge). Le tableau 3 (page suivante) expose le nombre de multiplieurs par million de λ^2 (unité de surface) en utilisant le critère de Dehon [DEHON]. Le résultat obtenu par le Systolic Ring (version placée-routée en technologie CMOS 0.35µm) souligne bien que le surcoût dû à l'utilisation de composants reconfigurables est moindre dans le cas du grain épais que du grain fin.

4. validations

Différents algorithmes multimédia ont été portés vers le Systolic Ring avec des performances souvent supérieures aux architectures dédiées. Nous présentons dans ce chapitre la mise en œuvre de la DCT, et exposons des résultats comparatifs.

Tableau 3. – Densités de traitement par unité de surface (multiplications 16 x16 et 8 x 8)

Architecture	Technologie	Surface et performance	Mult./Mλ^2 16×16	Mult./Mλ^2 8×8
Custom 16×16	0.63 µm	2.6 Mλ^2, 40 ns	0.38	0.38
Custom 8×8	0.80 µm	3.3 Mλ^2, 4.3 ns		0.3
Gate-Array	0.75 µm	26 Mλ^2, 30 ns	0.038	
FPGA (XC4000)	0.60 µm	1.25 Mλ^2/CLB		
		316 CLB, 26 ns	0.0025	
		220 CLB, 12.1 ns	0.0036	
DSP 16 bits	0.65µm	350Mλ^2, 50 ns	0.0028	
Systolic Ring (8 Dnodes/16b) Standard Cell	0.35µm	11.4 Mλ^2/Dnode	0.175	0.175

4.1. implantation de la DCT

La DCT, pour *Discrete Cosine Transform* est largement utilisée dans nombre d'applications relatives au traitement d'image [JPEG]. Le passage dans un espace transformé permet d'augmenter la corrélation des données et donc le taux de compression obtenu, par des méthodes classiques comme le codage de Huffmann. Le principe consiste à appliquer la DCT sur une image de taille quelconque. L'image considérée, de taille 64×64 pixels est décomposée en blocs carrés de 8 pixels de coté. Cette image est donc constituée de 64 blocs, eux-mêmes constitués de 64 échantillons chacun.

Chaque bloc de l'image subira indépendamment de ses voisins une DCT bidimensionnelle avant de subir les étapes de quantification et de codage entropique.

Le calcul d'une DCT uni-dimensionnelle sur 8 points implique le calcul suivant :

$$z_k = c(k) \sum_{n=0}^{7} x_n \cos \frac{\pi(2n+1)/k}{16} \qquad 0 \leqslant k \leqslant 7$$

avec x_n: échantillons, z_k : valeurs transformées

avec $c(k) = \sqrt{\frac{2}{N}}\alpha(k)$ et $\alpha(k) = \begin{cases} 1/\sqrt{2} \text{ pour } k = 0 \\ 1 \text{ sinon} \end{cases}$

Une des implantations les plus courantes est basée sur la décomposition en fréquences paires-impaires. La décomposition en fréquences paires-impaires divise les calculs en deux produits matriciels :

$$z = \sqrt{\frac{2}{N}} T(N) x \begin{cases} \begin{bmatrix} z_0 \\ z_2 \\ z_4 \\ z_6 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ \beta & \delta & -\delta & -\beta \\ \alpha & -\alpha & -\alpha & \alpha \\ \delta & -\beta & \beta & -\delta \end{bmatrix} \begin{bmatrix} x_0 + x_7 \\ x_1 + x_6 \\ x_2 + x_5 \\ x_3 + x_4 \end{bmatrix} \begin{cases} \alpha = \cos(\pi/4) \\ \beta = \cos(\pi/8) \\ \delta = \sin(\pi/8) \end{cases} \\ \begin{bmatrix} z_1 \\ z_3 \\ z_5 \\ z_7 \end{bmatrix} = \begin{bmatrix} \lambda & \gamma & \mu & \nu \\ \gamma & -\nu & -\lambda & -\mu \\ \mu & -\lambda & \nu & \gamma \\ \nu & -\mu & \gamma & -\lambda \end{bmatrix} \begin{bmatrix} x_0 - x_7 \\ x_1 - x_6 \\ x_2 - x_5 \\ x_3 - x_4 \end{bmatrix} \begin{cases} \lambda = \cos(\pi/16) \\ \gamma = \cos(3\pi/16) \\ \mu = \sin(\pi/16) \\ \nu = \sin(\pi/16) \end{cases} \end{cases}$$

Figure 18. – Décomposition en fréquences paires et impaires.

Comme le noyau de la DCT est séparable, il est possible d'effectuer la transformée 2D en deux étapes successives 1D : suivant les lignes puis les colonnes après transposition. Dans notre exemple, l'opération de transposition n'est pas traitée par l'architecture reconfigurable, et devrait, dans le cas d'une implantation matérielle, être assurée par une mémoire de transposition ou un processeur. Les échantillons et coefficients dans notre implantation sont également présentés par un processus externe (dans le cadre de notre évaluation par le *testbench* VHDL). La figure 19 détaille l'implantation de la DCT uni-dimensionnelle :

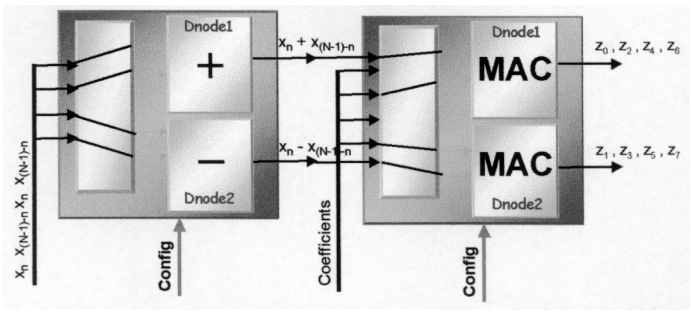

Figure 19. – Implantation de la DCT sur deux couches de *Dnodes*.

La première couche de *Dnodes* prend en charge les opérations d'addition et de soustraction (respectivement pour les fréquences paires et impaires). Une fois ces calculs effectués, les

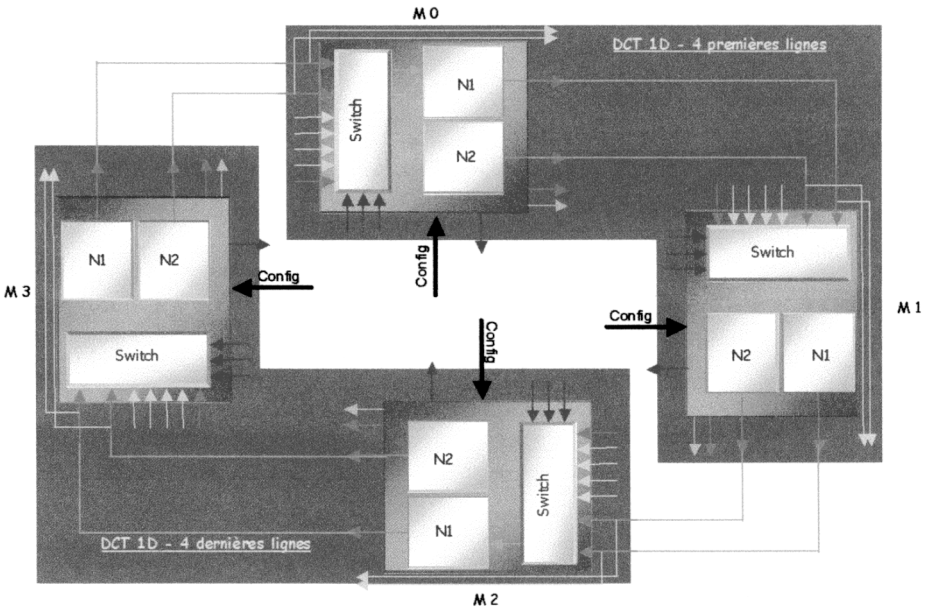

Figure 20. – Implantation de la DCT sur 8 lignes.

sommes et différences des échantillons sont transmis à la couche suivante qui se charge de réaliser les opérations de multiplication-accumulation (figure 19).

Ainsi, deux échantillons sont calculés tous les cinq cycles d'horloge. Une remise à zéro est effectuée durant le cinquième cycle dans le but de réinitialiser les registres d'accumulation des *Dnodes* de la seconde couche pour le calcul suivant. Le calcul d'une ligne entière (8 échantillons) nécessite 20 cycles d'horloge. Étant donné que 4 couches soit 8 *Dnodes* sont présents dans la structure utilisée, il est possible d'atteindre un degré de parallélisme de 2 par rapport à l'implantation exposée. La DCT suivant les quatre premières lignes est ainsi effectuée par les couches M0 et M1, alors que les quatre dernières le sont par les couches M2 et M3 (figure 20).

80 cycles sont nécessaires par dimension et par bloc, soit un total de 160 cycles pour le traitement complet d'une DCT 2D sur un bloc de 8x8 pixels. Le programme réalisant ce traitement et s'exécutant sur le contrôleur de configuration, effectue une transformée uni-dimensionnelle puis stoppe le traitement. Une fois que le processus externe effectuant l'opération de transposition est prêt à envoyer les échantillons pour le traitement suivant la seconde dimension (suivant les lignes puis les colonnes en alternance), il relance le contrôleur de configuration qui est sous son contrôle (remise à zéro). Il est à noter qu'il est également possible d'interrompre le traitement de la structure pendant un laps de temps en appliquant le signal HALT au contrôleur de configuration. La figure 21 expose le programme s'exécutant sur la structure pour le calcul parallèle de deux échantillons (un échantillon pair et un échantillon impair).

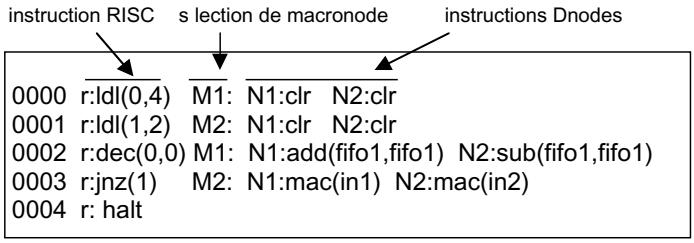

Figure 21. – Code de la DCT 1D.

Détails des opérations effectuées :

0000 – Chargement de #04h dans le registre 0 du contrôleur de configuration, et remise à zéro des 2 Dnodes de la couche 1 (macronode 1).

0001 – Chargement de #02h dans le registre 1 du contrôleur de configuration, et remise à zéro des 2 Dnodes de la couche 2 (macronode 2).

0002 – Décrémentation du registre 0 (registre compteur), puis stockage du résultat dans ce même registre. Configuration en mode global du Dnode 1 (addition) et du Dnode 2 (soustraction). Les arguments (fifo1, fifo1) servent à configurer le switch amont pour injecter les données provenant des entrées fifos.

0003 – Saut à l'adresse contenu dans le registre 1 du contrôleur de configuration tant que le résultat de la décrémentation n'est pas nul. Configuration en mode global des Dnodes 1 et 2 (MAC : multiplication-accumulation) de la couche 2.

0004 – Arrêt du traitement.

Les performances obtenues, pour le traitement d'un bloc de 8×8 échantillons (transformée 2D), sont exposées dans le tableau 4, et comparées à des implantations sur DSP de Texas Instruments [TEXAS] et à un cœur *logiciel* dédié [XILINX]. Il est à noter que deux familles d'implantations existent :

– Matrice : les calculs sont effectués simplement d'une façon matricielle, sans réarrangement des coefficients ; c'est l'implantation effectuée dans notre cas.

Tableau 4. – Performances des différentes implantations de la DCT.

	TMS 320C30	TMS320 C55	TMS320 C62	coeur Xilinx DCT	Ring-8
# cycles calcul	–	–	160	64	160
# cycles latence	–	–	48	75	16
# cycles	1316	1078	208	139	176
fréquence	20 MHz	200 MHz	300 MHz	80 MHz	200 MHz
Temps de traitement	65.8 µs	5.39 µs	690 ns	1,7 ns	880 ns
Algorithme	matrice	matrice	matrice	Signal-flow	matrice

– « *Signal-flow* » : étant donné le nombre restreint de coefficients différents, une matrice de réarrangement est appliquée pour éviter de ré-effectuer des calculs déjà réalisés. Cette implantation se traduit par un nombre de calculs largement diminué, et donc des performances améliorées, d'où les résultats obtenus sur DSP C62.

Une version à 8 *Dnodes* du Systolic Ring s'avère plus rapide que toutes les implantations sur DSP de famille Texas Instruments (le format des données manipulées par ces derniers est cependant de 32 bits). Néanmoins aucune modélisation de latence pour les accès mémoire nécessaires n'a été faite dans notre cas. La transposition des échantillons entre les deux dimensions n'est pas non plus gérée par le Systolic Ring et doit être prise en charge par un processus externe. Il est à noter que cette implantation est matricielle, c'est-à-dire qu'aucune optimisation n'a été effectuée sur le traitement des données. Par exemple, à chaque cycle de traitement pour le calcul des échantillons de fréquences paires, 4 cycles sont « perdus » pour effectuer des multiplications par 1. Une implantation de type « signal-flow », utilisant intensivement le mode local, donnerait des performances biens meilleures dans notre cas.

4.2. prototypage

Dans le but de valider fonctionnellement la structure développée, nous avons eu recours à une phase de prototypage sur une plate-forme à base de FPGA. En effet les simulations VHDL comportementales ne suffisent pas à la validation et l'étape de prototypage est nécessaire car essentielle à la réalisation d'un circuit intégré dédié, actuellement en cours de réalisation.

Un prototype d'une version à 8 *Dnodes* a été réalisé (figure 22), permettant d'affiner les résultats obtenus en simulation et ainsi de valider notre architecture d'un point de vue fonctionnel dans un contexte réel. Une version à 8 *Dnodes* ainsi qu'un contrôleur VGA ont été synthétisées et intégrés dans le FPGA de famille Altera APEX [ALTERA]. Les outils utilisés sont Synopsys FPGA Express [SYNOPSYS] pour la synthèse et Altera Quartus pour le placement/routage ainsi que le téléchargement sur la plate-forme. Une caméra CMOS a également été interfacée avec le prototype. L'architecture peut acquérir, traiter et afficher sur écran via le contrôleur VGA un flot vidéo monochrome issu de la caméra. L'ensemble à été validé avec succès sur un algorithme de détection de contours.

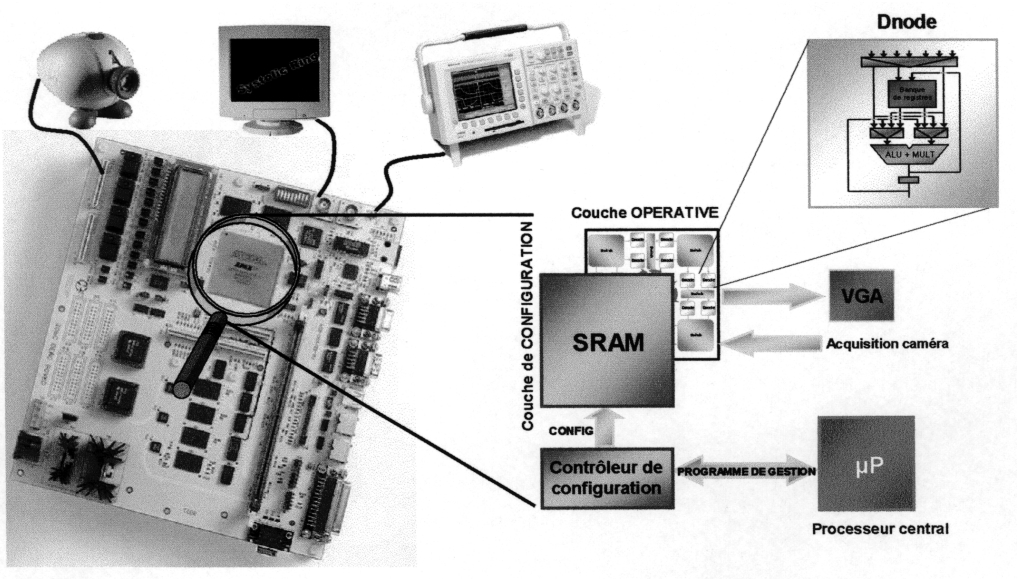

Figure 22. – Le prototype validé sur un algorithme de détection de contours.

Le tableau 5 expose les résultats d'implantation sur la plate-forme SOPC d'Altera. La fréquence maximale de fonctionnement est imposée par un chemin critique du multiplieur des *Dnodes*. L'utilisation d'un multiplieur pipeliné aurait probablement autorisé une fréquence de fonctionnement bien supérieure.

Tableau 5. – Résultats de synthèse sur plate-forme SOPC.

	RING-8	**Contrôleur VGA + logique de contrôle**
Fréquence maximale	*32 MHz*	*25 MHz (fréquence VGA)*
Taux d'occupation	*24 %*	*1 %*

La dernière validation, concerne un premier prototype sur silicium réalisé avec des cellules standards à partir d'une description VHDL. Une réalisation à 8 *Dnodes* (*4 switchs*) requiert une surface sur silicium de l'ordre de 4 mm^2 en technologie CMOS 0.18 µm autorisant une bande passante de traitement de l'ordre de 3 Gigaoctets par seconde à 200 MHz. La surface occupée par un *Dnode* est estimée à 0,8 mm^2 dans cette technologie, ce qui confirme le caractère extrapolable de l'architecture. Des réalisations de plusieurs centaines de *Dnodes* sont envisageables, et le caractère hautement régulier de la structure la rend particulièrement adaptée à une réalisation « full custom » du *Dnode* (conception au niveau masque), une opération qui double usuellement la densité sur silicium tout en accroissant les performances dans des proportions intéressantes.

5. outils de développement

5.1. logiciel d'assemblage

Le programme réalisant la génération du code objet à été écrit en C++ (3 500 lignes), il lit un fichier assembleur et crée le code objet prêt à s'exécuter sur le contrôleur de configuration. Ce programme effectue la configuration des *switchs* par résolution des opérandes des *Dnodes*, ce qui simplifie grandement la programmation de l'ensemble. Deux fichiers distincts sont générés :

– Le premier est destiné à la simulation VHDL. Le « testbench » utilise ce fichier, et chaque fois qu'un accès mémoire est rencontré, il renvoie au contrôleur de configuration la valeur du mot (instruction) correspondant à l'adresse lue.

– Le second est un fichier binaire dédié au contrôleur de configuration, gérant dynamiquement la configuration de l'ensemble de la structure.

5.2. environnements de compilation pour architectures reconfigurables

Les outils utilisés actuellement, quand ils existent, sont pour la plupart basés sur des méthodes de synthèse de haut niveau [HARTENSTEIN], ou de synthèse d'architecture utilisées pour les ASICs. Ces approches génèrent sous contraintes (performances et/ou surface) à partir d'une description donnée (HDL généralement) un graphe de flot de données en utilisant des méthodes d'allocation et ordonnancement. La génération de ce graphe doit cependant passer dans le cas d'une description haut niveau (C, java, etc.) par une phase d'interprétation du code durant laquelle des techniques propres aux compilateurs (élimination de code redondant, déroulage des boucles, propagations de constantes, etc.) doivent optimiser la description.

Les techniques de synthèse d'architecture permettent une génération rapide de matériel à partir d'une description de type HDL et d'une bibliothèque d'opérateurs (arithmétiques) assimilables aux blocs configurables du réseau concerné. Les architectures reconfigurables n'ayant pas tous les degrés de liberté autorisés lors de la réalisation d'un ASIC, ces techniques ne sont plus adaptées et doivent être repensées. Des approches par méthodes heuristiques telles que recuit simulé, algorithmes génétiques pour le placement et le routage sont proposées dans la littérature. Toutefois, le succès des différentes approches est intimement corrélé à l'architecture même du réseau reconfigurable (granularité, reconfiguration statique ou dynamique, sous-système de routage…).

D'autre part, les capacités de reconfiguration dynamique ajoutent une dimension au problème déjà non trivial de compilation, de ce fait d'autant plus difficile à appréhender de manière efficace.

5.3. approche retenue

Les techniques existantes de compilation ciblant des architectures reconfigurables sont basées sur l'assignation statique d'une description sur une bibliothèque d'opérateurs donnée [HARTENSTEIN]. Cette bibliothèque se résume souvent à des opérateurs arithmétiques simples, et l'implantation de flots de données complexes rend cette phase difficile et nécessite la génération d'un contrôleur conséquent non réalisable dans les architectures reconfigurables à granularité épaisse.

Notre approche en cours de développement utilise de façon optimisée la reconfiguration dynamique par l'emploi du mode local : celui-ci autorise la définition d'une bibliothèque de macro-opérateurs émulés. Ainsi, des opérateurs complexes de type filtres numériques, lignes à retard, opérateurs arithmétiques à latence variable sont directement identifiables par les outils de « compilation ». Une fois cette librairie définie et l'assignation réalisée, l'outil d'assemblage du code remplace les primitives

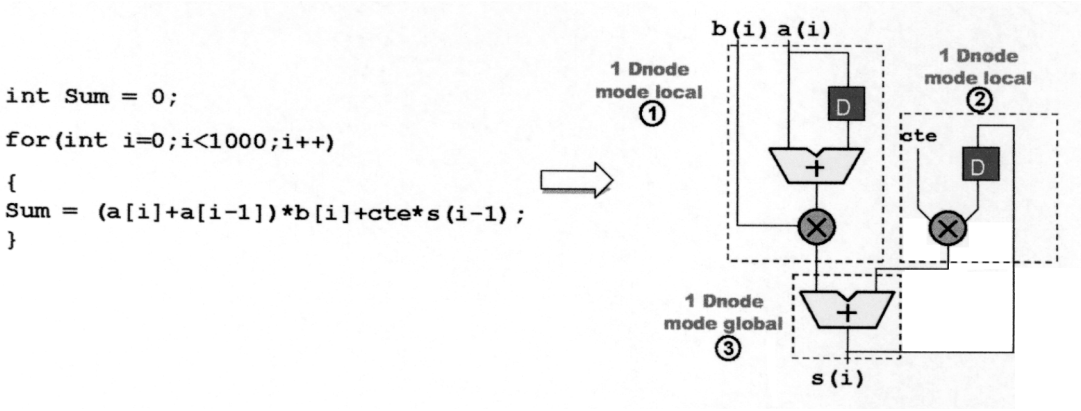

Figure 23. – Exemple de phase d'extraction du flot de données à partir de la description C.

complexes par une portion de code objet correspondante chargée dynamiquement dans chaque *Dnode* concerné par le contrôleur de configuration.

La figure 23 expose le principe de compilation ciblant le Systolic Ring à partir d'une description en langage C. À partir du code C, un graphe de flot de données est généré sous contraintes de latence minimale par une technique classique de synthèse d'architecture. Le graphe représentant le flot de données est ensuite partitionné par un algorithme le parcourant à la recherche de motifs issus de la bibliothèque de macro-opérateurs implémentables en mode local. Une fois ces sous-graphes identifiés, leurs paramètres propres sont extraits (latences, coefficients, etc.) puis passés au module d'assemblage qui réalise l'édition des liens du code objet final voué à s'exécuter sur le contrôleur de configuration. Le code objet de chaque *Dnode* assigné en mode local est encapsulé dans le code RISC, puis téléchargé dans la phase d'initialisation de la structure.

Sur l'exemple figure 23, trois sous-graphes correspondant chacun à un *Dnode* sont identifiés. Ils sont numérotés de 1 à 3, et à chacun est associé un jeu de paramètres propres (pour le sous-graphe 2 par exemple, un délai unitaire est codé ainsi que la valeur de la constante). Une fois la phase de compilation terminée, les premières instructions s'exécutant sur le contrôleur de configuration configurent les éléments statiques (injection des données à travers le *switch* Nord, renvoi par le *switch* Est et addition sur le *Dnode* 3) comme illustré figure 24. Les instructions suivantes téléchargent successivement les microcodes des *Dnodes* 1 et 2 puis déclenchent le traitement. À partir de ce moment le contrôleur de configurations n'a plus à intervenir dans le traitement des données et reste disponible pour d'autres traitements sur le reste de la structure.

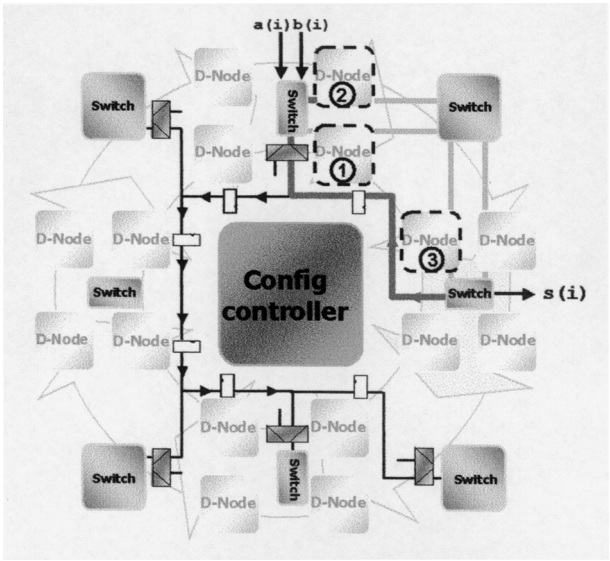

Figure 24. – Implantation dans l'anneau de l'algorithme de la figure 23.

6. conclusion

Nous avons ici exposé une alternative aux architectures traditionnelles. En lieu et place des solutions classiques consistant à utiliser un unique cœur de processeur ou DSP pour gérer l'ensemble des fonctionnalités au niveau numérique, nous proposons une approche coprocesseur avec un cœur dédié au traitement des algorithmes à chronophages. À chacun de ceux-ci correspond un programme s'exécutant sur le contrôleur de configuration et gérant dynamiquement la configuration de notre réseau. Dans un contexte d'application orientée téléphonie de troisième génération, on peut imaginer que le terminal intégrant cette architecture soit évolutive, ne se limitant donc pas à la bibliothèque de programmes initiale, et, consécutivement au lance-

ment d'une nouvelle application (visioconférence par exemple) se connecte directement à un centre serveur et télécharge le code objet correspondant. Notre réseau, dynamiquement reconfigurable dédié aux applications orientées données prouve donc son efficacité non seulement en termes de performances, mais aussi en termes de coût. Une version à 8 *Dnodes* de notre architecture à été synthétisée en technologie CMOS 0.18 µm, avec une surface sur Silicium de l'ordre de 0,7 mm^2, ce qui la situe clairement dans un contexte de système sur puce destiné aux applications fortement contraintes telles que celles de l'embarqué. Avec une approche système sur puce, cette architecture pourrait prendre place aux cotés d'un cœur de processeur éprouvé du commerce comme l'ARM7 ; l'ensemble constituerait une solution (figure 25), disposant des capacités de traitement optimisées du Systolic Ring jointe à l'utilisation d'un cœur de processeur apte à faire tourner un grand éventail de systèmes d'exploitation et d'applications, le tout à un coût raisonnable.

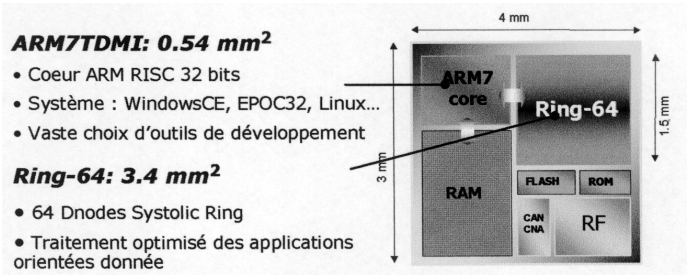

Figure 25. – Une architecture envisageable de SoC.

Plusieurs points restent néanmoins à traiter. Parmi ceux-ci nous pouvons citer :

- Le problème des bandes passantes mémoires [HARTENSTEIN] déjà présent dans les architectures mono-processeur conduit aujourd'hui les concepteurs à introduire des mémoires caches sur plusieurs niveaux. Celles-ci sont vouées à limiter le goulot d'étranglement que constituent les accès aux mémoires principales, bien plus lentes que le processeur lui-même. Dans le cas d'architectures reconfigurables comme le Systolic Ring où de multiples unités de traitement sont présentes ; ce problème est d'autant plus marqué que la bande passante de traitement résultante est importante. La solution réside certainement dans l'utilisation de multiples banques mémoires mais l'interfaçage optimal avec celles-ci est dépendant de l'application considérée. De ce fait, la flexibilité de l'architecture de communication est un facteur crucial influant sur les performances finales du système. Ces banques de mémoire distribuées (sur chaque couche de *Dnodes* par exemple) peut ainsi être vues comme une mémoire cache. Ceci ne dispense toutefois pas de gérer l'interfaçage avec une mémoire centrale, qui reste à définir. La logique reconfigurable à grain fin constitue probablement une solution de choix, permettant de synthétiser une architecture d'interfaçage optimale pour chaque application considérée. Cette solution que nous avons retenue lors des phases de prototypage utilisant un FPGA fait actuellement l'objet d'une étude approfondie.
- L'évolution de l'efficacité de la programmation relativement aux nombre de *Dnodes*. La réalisation de structures à fort nombre de *Dnodes* répondant aux principes énoncés dans cet article se heurte au difficile compromis scalabilité/reconfiguration dynamique. En effet une augmentation marquée du nombre de couches de *Dnodes* conduirait à une diminution du caractère dynamique de la structure : un nombre de cycles du contrôleur de configuration plus important serait nécessaire à la reconfiguration de la structure. Ce problème peut bien sur être adressé en augmentant les performances du contrôleur de configuration, en lui autorisant de reconfigurer un nombre plus important de *Dnodes* à chaque cycle. Ceci se ferait cependant au détriment de la scalabilité [BENOIT]: la réalisation de structures à fort nombre de *Dnodes* s'avérerait excessivement coûteuse en surface silicium.
- Le compromis nombre de *Dnodes* par couche/nombre de couches doit être analysé. En effet, l'utilisation d'un nombre important (N) de *Dnodes* par couche implique la réalisation de *Switchs* (interconnexion totale entre 2 couches) dont la taille croit en O(N^2), ce qui limite les capacités d'extrapolation (scalabilité) sur de nouvelles technologies silicium [BENOIT]. L'utilisation d'un nombre de couches important, en limitant le nombre de *Dnodes* par couche diminue le caractère dynamique de la structure : le nombre de cycles du contrôleur nécessaires à la reconfiguration de la structure croît ici de manière linéaire.
- Méthodologies de développement. Comme abordé en partie 5, la réalisation d'outils de portage est un problème difficile : le portage d'algorithmes n'est malheureusement pas aussi simple que dans le cas des processeurs où la phase se résume à une simple compilation (exécution temporelle et non spatiale). L'expérience capitalisée dans le cas de réalisation de compilateur pour processeurs CISC montre qu'il est de manière générale difficile d'exploiter efficacement des architectures à jeu d'instruction complexe (bibliothèque de macro-opérateurs dans le cas du Systolic Ring). Les prochaines expérimentations sur l'environnement de compilation en cours de développement devraient permettre de dégager des prospectives, et ainsi orienter la recherche vers des techniques issues soit de la compilation, soit de la synthèse d'architecture.

Manuscrit reçu le 13 novembre 2001

BIBLIOGRAPHIE

[ALTERA] http://www.altera.com

[ARVI] Arvind and Robert A. Iannucci, « A critique of multiprocessing von Neumann style », *10th International Symposium on Computer Architecture*, pages 426-436, 1983.

[BENOIT] Journées Francophones sur l'Adéquation Algorithme Architecture, « Caractérisation et comparaison d'Architectures Reconfigurables Dynamiquement. Un exemple : Le Systolic Ring », *à paraître dans JFAAA'2002 16-18 Décembre 2002*, Monastir, Tunisie.

[BROWN] Stephen Brown and J. Rose, « Architecture of FPGA and CPLDs: A Tutorial », *IEEE Design and Test of Computers*, Vol. 13, n°2, pp. 42-57, 1996.

[DEHON] André DeHon, Comparing Computing Machines, Configurable Computing: Technology and Applications, *Proceeding SPIE Vol. 3526*, 2-3 Novembre 1998.

[DEMIGNY] Didier Demigny, « Méthodes et architectures pour le TSI en temps reel » Traité IC2 – *Série Traitement du signal et de l'image*, ISBN : 2-7462-0327-8.

[HARDENBERGH] Hal W. Hardenbergh. « CPU performance: Where are we headed? Dr. Dobb's Journal », pp. 30-38, Janvier 1994.

[HARTENSTEIN] R. Hartenstein, H. Grünbacher (Editors): *The Roadmap to Reconfigurable computing* Proc. FPL2000, 27-30 Août 2000 ; LNCS,Springer-Verlag 2000.

[INTEL] Intel Application Notes for Pentium MMX, *http://developer.intel.com/*

[JPEG] ISO/IEC JTC1 CD 10918. *Digital compression and coding of continuous-tone still images – part 1, requirements and guidelines*, ISO, 1993 (JPEG).

[KRESS] R. Kress *et al.* : A Datapath Synthesis System for the Reconfigurable Data-path Architecture, *ASP-DAC'95*, Chiba, Japon, 29 Août-1 Sept. 1995.

[KUNG] Sun Yuan Kung, « VLSI Array processors », Prentice Hall Information and System Sciences Series, Thomas Kailath Editor, 1985. ISBN 0-13-942749-X.

[LEE] H. Lee, K. Nguyen-Phi, H. Alnuweiri and F. Kossentini, « Software-Only Real-time MPEG-2 Video Encoding on The C62x VLIW Processor », *DSPs FEST '99*, Houston, Texas, U.S.A., August 1999.

[MANGIONE] W. H. Mangione-Smith *et al.*, « Seeking Solutions in Configurable Computing », *IEEE Computer*, pp. 38-43, Décembre 1997.

[NAGELDINGER] U. Nageldinger *et al.* : KressArray Xplorer: A New CAD Environment to Optimize Reconfigurable Datapath Array Architectures; *ASP DAC*, Yokohama, Japan, Jan. 25-28, 2000.

[SHIVA] Sajjan G. Shiva, « Computer Design and Architecture », third edition, Marcel Dekker editor, *ISBN 0 8247 0368 5*, 2000.

[SYNOPSYS] http://www.synopsys.com

[TEXAS] http://www.ti.com

[XILINX] Xilinx, *the Programmable Logic Data Book*, 2000

[XPUTERS] Why reconfigurable computing, Department of Computer Science, Computer Structures Group http://xputers.informatik.uni-kl.de/

Manuscrit reçu le 13 novembre 2001

LES AUTEURS

Gaston CAMBON

Gaston CAMBON est Professeur à l'Institut ISIM de l'Université Montpellier II depuis 1976. Depuis cette date, il y a développé les enseignements de Conception Assistée par Ordinateur (CAO) des circuits intégrés. De 1985 à 1995, il a mis en place et fait fonctionner un service national de diffusion et de support d'outils de CAO industriels dans le cadre du réseau « Comité National de Formation en Micro-électronique » (CNFM). De 1991 à 2001 il a été le (premier) directeur du laboratoire de recherche LIRMM (Unité mixte CNRS/Université Montpellier).

Au début de sa carrière (années 70) il a mené des recherches sur les composants électroniques pour le traitement du signal (ondes acoustiques de surface). Dans les années 80-90, il a mené des recherches sur la modélisation des circuits intégrés ASIC. Actuellement , il s'intéresse plus particulièrement aux méthodes de conception et de validation des systèmes sur puce (SOC) ; il est directeur du réseau national « System On Chip » qu'il a mis en place pour le département STIC (Sciences et Technologies de l'Information et de la Communication) du CNRS. Ce réseau rassemble environ 150 chercheurs de 30 laboratoires.

Camille DIOU

Camille Diou a obtenu son doctorat en Micro-électronique en 2000 à l'université Montpellier 2. Depuis 2001, il est Maître de Conférences à l'université de Metz où il enseigne la CAO micro-électronique et les systèmes numériques. Les travaux de recherche qu'il effectue au Laboratoire Interfaces Capteurs et Micro-électronique de Metz (LICM) dans le domaine des systèmes électroniques pour les télécommunications, portent sur la conception d'architectures fortement parallèles de circuits et systèmes pour la transmission d'information à haut débit et la modulation numérique.

Jérôme GALY

Jérôme GALY a obtenu son doctorat en Traitement du Signal en 1998 à l'université Paul Sabatier de Toulouse. Depuis 1998 il est Maître de Conférences à l'université Montpellier 2 et il effectue sa recherche au Laboratoire d'Informatique, de Robotique et de Micro-électronique de Montpellier. Ses recherches au sein du département micro-électronique concernent principalement les aspects traitement et transmission du signal pour des réseaux mobiles ou optiques.

Michel ROBERT

Michel Robert, 44 ans, est Professeur à l'université Montpellier2. Il enseigne la CAO des circuits et des systèmes intégrésmicro-électroniques à l'ISIM (Institut des Sciences de l'Ingénieur de Montpellier). Après une expérience industrielle d'ingénieur d'étude à Texas Instruments, il débute sa carrière d'enseignant-chercheur à l'université Montpellier 2, en octobre 1984 (thèse de doctorat soutenue en 1987). Il effectue ses travaux de recherche au LIRMM (laboratoire d'informatique, de robotique et de micro-électronique) dans le domaine de la conception et de la validation des systèmes intégrés micro-électroniques, la synthèse électrique et topologique des circuits intégrés numériques CMOS, et l'adéquation algorithme-architecture en traitement d'images temps réel. Il est auteur ou co-auteur de plus de 150 publications dans ces domaines. Nommé membre de l'Institut Universitaire de France en 1997, il est Vice Président du CNFM (Comité National de Formation en Micro-électronique) depuis 1999, conseiller au Ministère de la Recherche (membre du Conseil Supérieur de la Recherche et de la Technologie) depuis 1999, chargé de mission au département STIC du CNRS depuis 2001 et directeur de l'école doctorale I2S (Information Structures et Systèmes) depuis 2001.

Lionel TORRES

Lionel Torres a obtenu son doctorat en Micro-électronique en 1996 à l'université Montpellier 2. De 1996 à 1997, il a travaillé au sein de la société ATMEL en tant qu'ingénieur de conception. Depuis 1998 il est Maître de Conférences à l'université Montpellier 2 et il effectue sa recherche au Laboratoire d'Informatique, de Robotique et de Micro-électronique de Montpellier. Ses recherches au sein du département micro-électronique concernent les architectures micro-électroniques pour le traitement du signal et des images, et les architectures reconfigurables dynamiquement.

Pascal BENOIT

Pascal BENOIT, 25 ans, a obtenu son DEA Systèmes Automatiques et Micro-électroniques, option Micro-électronique, en 2001, à l'Université Montpellier 2. Actuellement en 2ème année de thèse au LIRMM, ses travaux de recherche concernent la conception d'architectures reconfigurables dynamiquement dédiées au traitement du signal et de l'image.

Gilles SASSATELLI

Gilles SASSATELLI a obtenu son doctorat en Micro-électronique en 2002 à l'université Montpellier 2. Il a par la suite été professeur assistant au laboratoire MES (MicroElectronics Systems) de l'université de Darmstadt en Allemagne. Il est actuellement chercheur CNRS au sein du Laboratoire d'Informatique, de Robotique et de Micro-électronique de Montpellier (LIRMM) où il poursuit ses travaux de recherche sur les architectures reconfigurables dynamiquement dédiées au traitement du signal et des images.