INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Projet COSI

Conception de systèmes sur silicium

Rennes

THÈME 1A

Table des matières

1 Composition de l'équipe **3**

2 Présentation et objectifs généraux **3**

3 Fondements scientifiques **5**
3.1 Panorama . . . 5
3.2 Synthèse à partir d'équations récurrentes . . . 5
3.3 Conception d'architectures parallèles intégrées . . . 7

4 Domaines d'applications **8**
4.1 Panorama . . . 8
4.2 Conception de processeurs programmables spécialisés et leurs outils de compilation . . . 8
4.3 Conception d'architectures pour les télécommunications . . . 10
4.4 Biologie Moléculaire . . . 10
4.5 Filtrage adaptatif . . . 11

5 Logiciels **11**
5.1 Panorama . . . 11
5.2 PolyLib . . . 12
5.3 MMAlpha . . . 12

6 Résultats nouveaux **13**
6.1 Synthèse de très haut niveau . . . 13
6.2 Compilation pour processeurs spécialisés programmables . . . 15
6.3 Outils de CAO pour architectures reconfigurables . . . 16
6.4 Algorithmique fondamentale . . . 17

7 Contrats industriels (nationaux, européens et internationaux) **18**
7.1 SPART, Méthodes de spécification pour la conception d'architectures de contrôle de trafic en ATM, (CTI Cnet, 197C9350031312012) . . . 18

8 Actions régionales, nationales et internationales **19**
8.1 Actions régionales . . . 19
8.2 Actions nationales . . . 19
8.3 Relations bilatérales internationales . . . 19
8.3.1 Europe . . . 19
8.3.2 Pacifique et Asie du Sud . . . 19
8.3.3 Afrique . . . 20
8.3.4 Amérique du Nord . . . 20
8.4 Accueil de chercheurs étrangers . . . 20

9 Diffusion de résultats **20**
9.1 Animation de la communauté scientifique 20
9.2 Enseignement universitaire 21
9.3 Autres enseignements 21
9.4 Participation à des colloques, séminaires, invitations 21

10 Bibliographie **22**

1 Composition de l'équipe

Responsable scientifique

Sanjay Rajopadhye [CR CNRS, en disponibilité à l'université d'État du Colorado (Fort Collins, USA) depuis le 1/09/2001]

Assistante de projet

Maryse Auffray [AA Inria]

Personnel Inria

François Charot [CR Inria, responsable permanent du projet depuis le 1/09/2001]
Olivier Sentieys [CR Inria, en détachement dans le projet depuis le 1/09/2001]
Tanguy Risset [CR Inria, muté à l'UR Inria-Rhône-Alpes au 01/09/01]
Sorin Olaru [ingénieur associé Inria depuis le 15/10/2001]

Personnel CNRS

Dominique Lavenier [CR CNRS]
Charles Wagner [IR CNRS (Atelier)]

Personnel Université de Rennes 1

Laurent Perraudeau [maître de conférences, université de Rennes 1]
Steven Derrien [ATER, université de Rennes 1 depuis le 1/10/2001]
Fabien Quilleré [ATER, université de Rennes 1, jusqu'au 30/08/2001]
Patrice Quinton [professeur, université de Rennes 1]

Chercheurs doctorants

Steven Derrien [bourse MENESR jusqu'au 30/09/2001]
Ferry Djieya [bourse CIES, jusqu'au 15/02/2001]
Erwan Fabiani [bourse MENESR, jusqu'au 31/08/2001]
Anne-Claire Guillou [bourse Inria]
Gautam Gupta [bourse CIES en co-tutelle]
Katell Morin-Allory [bourse MENESR]

Collaborateur extérieur

David Cachera [maître de conférences ENS-cachan]

2 Présentation et objectifs généraux

Résumé : *Le projet Cosi vise à développer des outils et des méthodes pour la mise en œuvre de systèmes complets sur silicium. Ces méthodes doivent pouvoir s'adapter à différentes technologies de réalisation : circuits* VLSI, FPGA, *implémentations hybrides logicielles-matérielles, etc.*

Le projet Cosi met l'accent sur trois thèmes principaux : (1) la synthèse de très haut niveau de systèmes dédiés, (2) la compilation et l'optimisation pour des processeurs spécialisés programmables, et (3) la conception et la réalisation de circuits réguliers.

En parallèle, le projet s'appuie sur des applications pour obtenir un retour sur les méthodes et les outils, mais aussi pour développer des architectures nouvelles pour ces applications. Ces applications incluent le traitement du signal et de l'image,

les télécommunications, le calcul haute performance, la comparaison de séquences génétiques et les réseaux ATM.

Enfin, le projet travaille aussi sur l'algorithmique fondamentale (parallèle et séquentielle) pour des problèmes comme le sac-à-dos, le tri, les files de priorité, le problème du chemin algébrique, etc.

Le projet Cosi propose de développer des outils et des méthodes pour la mise en œuvre de systèmes sur silicium. Ces méthodes doivent pouvoir s'adapter à différentes technologies de réalisation : circuits VLSI, FPGA, co-processeurs reconfigurables, implémentations hybrides logicielles-matérielles, etc.

Le concepteur de systèmes dédiés doit faire face au défi suivant : concevoir *rapidement* des systèmes *corrects*, de complexité *croissante*, incluant une partie *logicielle* importante, à partir de spécifications *changeantes*, en visant un ensemble de technologies de réalisation en *évolution* extrêmement rapide.

En réponse, l'outil de conception idéal doit permettre une évolution rapide des spécifications, un redéploiement rapide vers de nouvelles technologies cibles, une modification des contraintes de coût et de performances et une dérivation sûre des solutions optimales. Le processus de conception peut ainsi être vu comme une série de raffinements d'une *spécification exécutable*. Chaque étape du raffinement est réalisée soit manuellement (éventuellement justifiée par une preuve formelle ou des tests et des simulations) ou automatiquement en utilisant un logiciel. Si cela est fait avec un outil logiciel, le *choix* du raffinement à appliquer peut être automatique (compilation presse-bouton) ou au contraire donné par l'utilisateur (exploration systématique de l'espace de conception). Les outils (logiciels ou autres) pour cette activité peuvent ainsi être vus comme une *plate-forme ouverte de conception*.

Nos recherches antérieures ont contribué au développement d'un modèle de calculs massivement parallèles – le modèle polyédrique – et ont aussi donné lieu à la réalisation de plusieurs prototypes de machines et de circuits. Nous mettons l'accent sur trois thèmes, complétés par la mise en œuvre d'applications concrètes ainsi que des études sur l'algorithmique.

Le premier thème est la ***synthèse de systèmes dédiés complets à partir de spécifications de haut niveau***. Aujourd'hui, la technologie des circuits intégrés permet de réaliser des systèmes entiers sur silicium, et c'est donc la maîtrise de la conception de tels systèmes qu'il faut rechercher. Pour aborder ce thème, Cosi s'appuie sur les recherches menées sur le langage Alpha et son environnement de développement MMAlpha. Le modèle polyédrique qui sous-tend Alpha constitue une base pour poursuivre les recherches visant le partitionnement d'un système, l'expression de calculs irréguliers, et l'interfaçage avec des formalismes de flots de données synchrones.

Le second thème est la ***compilation optimisée pour des processeurs spécialisés programmables***, appelés des Asip *(Application Specific Instruction-set Processor)*. Le plus souvent possible, la conception d'un système dédié fait appel à des «cœurs» de processeurs – processeur Risc ou DSP – qui sont optimisés et spécialisés pour tenir compte des contraintes d'utilisation du système. La compilation pour des Asip est un défi motivant : il s'agit de produire, pour une application particulière, à la fois l'architecture et le compilateur permettant d'atteindre les performances visées par cette application. Cette technique est l'une des clés de la réalisation de systèmes dédiés aux télécommunications. Les recherches s'appuient sur le

langage Armor pour la modélisation comportementale des processeurs.

Le troisième thème abordé concerne les ***architectures configurables*** à base de circuits FPGA. Cette technologie très prometteuse constitue un axe de recherche à long terme. Elle possède de très fortes potentialités mais de nombreux problèmes doivent être encore résolus, notamment le manque d'outils de programmation, compilation et optimisation.

3 Fondements scientifiques

3.1 Panorama

Mots clés : synthèse d'architecture, CAO, ASIC, architecture parallèle, régularité, circuit intégré, silicium, méthodologie de conception.

Résumé : *La synthèse de circuits se fait aujourd'hui à partir de spécifications de plus en plus haut niveau. La spécification de programmes effectuant des calculs réguliers sous forme d'équations récurrentes permet des analyses statiques puissantes et des transformations de programmes pour la dérivation d'architectures régulières. Ce type de spécification est également applicable pour la compilation et la parallélisation des boucles, permettant ainsi de traiter la conception conjointe (co-design) sous un unique formalisme.*

3.2 Synthèse à partir d'équations récurrentes

Le développement des systèmes d'équations récurrentes (SER) commence vers la fin des années 60 avec les travaux de Karp, Miller et Winograd [KMW67] qui proposent l'expression d'algorithmes itératifs comme des systèmes d'équations récurrentes uniformes (SERU). Ensuite, Lamport utilise les mêmes concepts dans le domaine de la parallélisation [Lam74]. À la fin des années 70, apparaissent les réseaux systoliques [KL80], architectures spécialisées synchrones régulières possédant un contrôle décentralisé. Au début, de telles architectures sont conçues «à la main» par des concepteurs spécialistes, souvent avec des astuces remarquables. Puis, plusieurs recherches indépendantes montrent que le formalisme des équations récurrentes — au départ, une seule équation uniforme (ERU), puis des systèmes uniformes (SERU), ensuite des équations *affines* (ERA) et enfin des systèmes d'équations *affines* (SERA) s'adaptent bien à la

[KMW67] R. Karp, R. Miller, S. Winograd, « The Organization of Computations for Uniform Recurrence Equations », *Journal of the ACM 14*, 3, juillet 1967, p. 563–590.

[Lam74] L. Lamport, « The Parallel Execution of DO Loops », *Communications of The ACM 17*, 2, février 1974, p. 83–93.

[KL80] H. Kung, C. Leiserson, « Systolic arrays for VLSI », *in : in Introduction to* VLSI *systems*, C. Mead, L. Conway (éditeurs), Addison-Wesley, 1980.

synthèse de tels réseaux [Qui84,QR89,RPF86,QV89]. Ce formalisme et ses extensions donnent lieu à ce qui est communément appelé le modèle polyédrique, base du langage Alpha développé dans API. Le choix du modèle polyédrique est motivé par plusieurs raisons.

- Le modèle permet une analyse statique puissante avec un modèle de quantification de performances sous jacent (formulation et résolution des problèmes d'une implémentation optimale).
- Le problème d'ordonnancement des SERU est souvent résolu par la programmation linéaire (les solutions appartiennent donc à un polyèdre) utilisant une représentation compacte. Cette méthode repose sur le fait que les dépendances entre calculs peuvent être représentées par un nombre fini de vecteurs (indépendamment de la taille du problème). Pour des équations affines, le problème d'ordonnancement est plus difficile car il n'y a pas un nombre fini de dépendances. Néanmoins, dans le cas où les domaines (les ensembles d'indices où les équations sont définies) sont des polyèdres, on peut profiter d'une représentation compacte de ces polyèdres (un nombre fini de sommets et de rayons, par exemple) pour formuler les contraintes d'ordonnancement par un programme linéaire.
- Une troisième motivation vient du fait qu'un SERA peut être vu comme un programme fonctionnel. Les manipulations classiques de synthèse systolique (notamment l'ordonnancement et l'allocation) peuvent être vues comme des transformations géométriques (aussi appelées transformations espace-temps). Ces transformations peuvent être appliquées de manière automatique si les systèmes d'équations sont des SERA avec des domaines polyédriques. Le langage Alpha est basé sur ces fondements.
- Il y a d'autres champs d'application que la synthèse systolique (le domaine de la parallélisation automatique des boucles par exemple). P. Feautrier a montré [Fea91] que l'analyse exacte de flot de données d'un programme composé de boucles à contrôle statique (correspondant à des boucles dont les bornes d'indices et les accès aux tableaux sont des fonctions affines), donne un SERA dont les domaines sont polyédriques. Le paralléliseur automatique de Fortran (PAF) développé dans son équipe utilise donc une généralisation et une extension des techniques de synthèse systolique (ordonnancement multidimensionnnel, placement et allocation de mémoire pour réduire la communication dans des machines à usage général) pour générer du code parallèle. Du fait de l'évolution des circuits, et parce que l'on trouve de plus en plus de composants programmables, le modèle polyédrique est donc un modèle fondamental pour la conception conjointe.

[Qui84] P. QUINTON, « Automatic Synthesis of Systolic Arrays from Uniform Recurrent Equations », *in : The 11th Annual International Symposium on Computer Architecture*, IEEE Computer Society Press, Ann Arbor, Michigan, juin 1984.

[QR89] P. QUINTON, Y. ROBERT, *Systolic Algorithms and Architectures*, Prentice Hall and Masson, 1989.

[RPF86] S. V. RAJOPADHYE, S. PURUSHOTHAMAN, R. M. FUJIMOTO, « On Synthesizing Systolic Arrays from Recurrence Equations with Linear Dependencies », *in : Proceedings, Sixth Conference on Foundations of Software Technology and Theoretical Computer Science*, Springer Verlag, LNCS 241, p. 488–503, New Delhi, India, décembre 1986.

[QV89] P. QUINTON, V. VAN DONGEN, « The Mapping of Linear Recurrence Equations on Regular Arrays », *Journal of VLSI Signal Processing 1*, 2, 1989, p. 95–113.

[Fea91] P. FEAUTRIER, « Dataflow analysis of array and scalar references », *Int. J. Parallel Programming 20*, 1, 1991, p. 23–51.

Il existe de nombreux prototypes d'environnements académiques pour la synthèse automatique d'architectures spécialisées (par exemple, Diastol, Presage, Hifi, Cathedral, Sade, PEI et MMAlpha) ainsi que certains outils commerciaux tel que Pico développé à HPlabs, Palo Alto. Alpha [Mau89] et MMAlpha ont évolué à partir de Diastol et constituent aujourd'hui un environnement pratique pour la manipulation d'équations récurrentes affines et la synthèse (dite de *très haut niveau*) d'architectures spécialisées.

La synthèse de haut niveau à partir d'Alpha se fait par transformations successives de programmes pour aboutir, par exemple, au format quasiment normalisé de VHDL synthétisable, ce qui permet de s'affranchir de la partie «basse» de la synthèse qui a été largement étudiée depuis de nombreuses années. Pour certaines technologies récentes, les FPGA par exemple, il est nécessaire que l'on descende plus bas dans la synthèse. Néanmoins, le modèle polyédrique, le langage Alpha et l'environnement MMAlpha constituent les fondements pour la synthèse de très haut niveau des systèmes spécialisés, soit en logiciel soit en matériel.

Les principales limitations de ce modèle proviennent du fait qu'il ne traite que difficilement le problème de partitionnement sous contraintes de ressources et qu'il est difficile d'exprimer un contrôle dynamique du programme.

Pour plus de détails, voir `http://www.irisa.fr/api/Rajopadhye/HiPC96.html`, un tutoriel «Why Systolic Arrays : the real answer» présenté à HiPC 96 par Quinton et Rajopadhye, ainsi que [LQR99]. Les détails du langage Alpha et son environnement de programmation et de transformation sont disponibles à `http://www.irisa.fr/api/ALPHA`.

3.3 Conception d'architectures parallèles intégrées

Résumé : *La conception d'architectures parallèles intégrées exploite la régularité des traitements que l'on implante dans le silicium, de la synthèse de haut niveau jusqu'à la vérification physique des dessins de masques. Cette activité inclut également la conception des mécanismes d'interface pour contrôler, initialiser et alimenter efficacement l'architecture parallèle.*

La conception d'un circuit intégré est l'activité qui consiste à produire le dessin de masques des différentes couches technologiques nécessaires à la fabrication de la puce de silicium, à partir de ses spécifications. Le processus requiert de nombreuses étapes dont l'enchaînement constitue la méthodologie de conception. L'objectif est de produire un circuit correct (i.e. conforme aux spécifications) dans un laps de temps le plus court possible.

L'intégration d'un calcul régulier est synonyme d'architecture parallèle. Les propriétés du traitement (la régularité) sont exploitées, d'une part, pour en dériver un mécanisme matériel performant (une architecture parallèle) et, d'autre part, pour faciliter la conception du circuit [Kun88].

La conception d'une architecture parallèle trouve d'abord sa source dans la formulation concise du traitement. Cette concision est ensuite reportée à toutes les étapes du processus de

[Mau89] C. Mauras, *Alpha: un langage équationnel pour la conception et la programmation d'architectures parallèles synchrones*, thèse de doctorat, Université de Rennes 1, IFSIC, décembre 1989.

[LQR99] D. Lavenier, P. Quinton, S. Rajopadhye, *Advanced Systolic Design, Signal Processing Series*, Marcel Dekker, 1999, ch. 23, p. 657–692.

[Kun88] S. Kung, *VLSI array processors*, Prentice-Hall, 1988.

conception. Avant tout, ce que l'on retient, c'est la structure du circuit : le nombre d'éléments qui le composent, par exemple, est secondaire alors que la fonctionnalité de ces mêmes éléments et leur schéma d'interconnexion sont primordiaux. On peut alors travailler sur des structures réduites, plus faciles à étudier, ou directement sur des structures paramétrables comme le proposent les outils de synthèse de haut niveau.

Mais au delà de l'élaboration de l'architecture parallèle proprement dite, se pose le problème plus général de son intégration dans un environnement donné. Ces architectures sont extrêmement performantes et exigent d'être pourvues en données à un rythme très élevé. Les mécanismes d'interfaçage, pour être efficaces, doivent alors être imaginés de concert avec le cœur du circuit et intégrés sur la même puce de silicium.

4 Domaines d'applications

4.1 Panorama

Mots clés : traitement d'image, télécommunication, biologie moléculaire.

Résumé : *Notre thématique est de développer des méthodes systématiques pour l'accélération des applications sur matériel dédié. Deux activités plus fondamentales dans le projet Cosi, la compilation pour processeurs spécialisés programmables et la conception et réalisation d'architectures parallèles intégrées sont elles mêmes des domaines applicatifs importants.*

Nos domaines applicatifs actuels sont le traitement d'image, le filtrage adaptatif, et la biologie moléculaire. Nous développons aussi une activité autour des algorithmes et architectures pour les protocoles ATM.

4.2 Conception de processeurs programmables spécialisés et leurs outils de compilation

Résumé : *La conception d'un système matériel fait de plus en plus souvent appel à des «cœurs» de processeurs qui sont optimisés et spécialisés pour tenir compte des contraintes d'utilisation du système. Il est souvent nécessaire de produire, pour une application particulière, à la fois l'architecture et le compilateur permettant d'atteindre les performances visées par cette application. Cette technique est l'une des clés de la réalisation de systèmes dédiés aux traitement d'images et aux télécommunications.*

Parmi les diverses technologies possibles comme les circuits spécialisés (Asic), les co-processeurs reprogrammables (basés sur des FPGA*), les processeurs programmables spécialisés (des* DSP *ou des Asip) nous avons choisi de nous focaliser sur les Asip.*

L'utilisation de *logiciels* embarqués, implantés sur des dispositifs programmables intégrés dans un circuit VLSI, est une tendance inéluctable dans les systèmes dédiés. Ces dispositifs ou cœurs de processeurs peuvent être de trois types : processeurs à *usage général*, processeurs *paramétrables* et processeurs *spécifiques*. Des instances de processeurs à usage général

sont maintenant disponibles sous forme de composants de base dans les bibliothèques des concepteurs de systèmes VLSI. Pour les processeurs paramétrables, le concepteur peut agir sur certains paramètres de l'architecture comme le nombre de registres, la largeur des bus, la présence d'unités fonctionnelles optionnelles et choisir l'instance la plus appropriée pour son application (processeurs de traitement du signal par exemple). Bien que les cœurs de processeurs existants en bibliothèque (à usage général ou paramétrable) permettent de prototyper rapidement un système, ils ne satisfont généralement pas les contraintes imposées par les applications en terme de temps d'exécution, de surface de silicium, de consommation et l'utilisation d'Asip est très souvent nécessaire.

Aujourd'hui les dispositifs programmables sont généralement programmés en langage d'assemblage, ce qui est très fastidieux et provoque des erreurs [PCL+96], et donc augmente fortement le temps de conception. Dans un avenir proche, il n'est pas raisonnable d'imaginer que toutes les applications enfouies seront réalisées avec des processeurs standards au moyen de compilateurs universels. Les Asip souffrent d'un manque évident d'outils logiciels, tels que compilateurs et simulateurs de jeu d'instructions [GPV+97]. C'est plus particulièrement vrai pour les processeurs dont l'architecture n'est pas connue à l'avance.

La raison d'être des Asip étant leur spécialisation et leur adaptation à une application donnée, il est primordial que les compilateurs atteignent des performances très proches du code produit manuellement, ce qui place la barre très haut. De plus les compilateurs doivent être *paramétrés par l'architecture visée*, parce qu'il est hors de question de redévelopper le compilateur pour chaque changement architectural. Il faut par conséquent pouvoir adapter très rapidement les compilateurs pour ces processeurs, ce qui pose des problèmes de recherche non résolus et très ardus. En outre, les architectures visées incluent les processeurs de traitement du signal, pour lesquels on sait que la production de bons compilateurs est difficile.

Les problèmes posés par l'utilisation des Asip auxquels nous nous intéressons sont de deux ordres : la définition de l'architecture d'un Asip, et sa programmation. La définition d'un Asip nécessite des méthodes de conception et des outils permettant d'organiser une architecture avec pertinence : choix des unités fonctionnelles, des unités de mémorisation, de la structure de registres, des interconnexions, du type de contrôle, etc. La génération de code nécessite des outils de compilation adaptés à l'architecture paramétrés par une description unique, à partir de laquelle sont extraites les informations utiles. Le formalisme doit permettre une modélisation rapide de l'architecture, qui soit lisible, compréhensible par un concepteur de processeur et exploitable par les différentes passes du compilateur. L'absence de formalisme adapté au contexte de l'exploration architecturale nous a conduit à définir le langage Armor.

Des algorithmes issus des domaines du traitement de l'image et du signal servent de support d'étude pour l'expérimentation des outils de compilation pour Asip.

[PCL+96] P. Paulin, M. Cornero, C. Liem, F. Nabaçal, C. Donawa, S. Sutarwala, T. May, C. Valderrama, « Trends in Embedded Systems Technology: An Industrial Perspective », *in: Hardware/Software Co-Design*, Kluwer Academic Publishers, 1996.

[GPV+97] G. Goossens, P. Paulin, J. Van Praet, D. Lanneer, W. Geurts, A. Kifli, C. Liem, « Embedded Software in Real-Time Signal Processing Systems: Design Technologies », *Proceedings of the IEEE (Special Issue on HW/SW Co-Design)*, 1997.

4.3 Conception d'architectures pour les télécommunications

Résumé : *Cette activité concernant les architectures de contrôle de trafic pour les réseaux à infrastructure* ATM *est très récente. Nous nous intéressons à l'expérimentation de méthodes de spécification dans le but d'améliorer le processus de conception de prototypes, pour des architectures de contrôle de trafic. Les méthodes envisagées reposent sur les recherches développées dans le projet et l'utilisation de formalismes de type flot de données.*

Les prochaines générations de commutateurs ATM intégreront des algorithmes de contrôle et de lissage du trafic, de gestion de ressources, dont la complexité et les contraintes temporelles nécessiteront des architectures matérielles très performantes mais également très flexibles. La conception de ces architectures, mélangeant processeurs programmables et circuits spécialisés, motive des approches de conception de haut niveau, qui grâce à l'utilisation de synthèse comportementale, de génération de code reciblable, permettront d'explorer rapidement différentes architectures et d'estimer de manière précise leurs performances.

Les méthodes de spécification basées sur le mécanisme flot de données présentent un certain nombre d'avantages (analyse statique, mise en œuvre avec des propriétés certifiées) qui facilitent le partitionnement et la génération des interfaces entre les partitions. Ces méthodes sont maintenant couramment employées dans des environnements de conception de systèmes tels que SPW de Cadence, Cossap de Synopsys, pour les versions industrielles, Ptolemy pour le domaine public. Ils permettent de décrire les fonctionnalités d'un système tout en faisant abstraction des détails d'implémentation et de rester indépendant de la technologie de réalisation. L'expérimentation de ces méthodes fait l'objet d'une collaboration avec le Cnet dans le cadre d'une convention CTI.

4.4 Biologie Moléculaire

Résumé : *La comparaison de séquences biologiques est un domaine d'application de la biologie moléculaire qui s'inscrit dans une thématique plus générale, le traitement des chaînes de caractères (string processing), sur laquelle nous travaillons depuis des années. Cela nous permet de tester nos stratégies de conception et nos architectures sur des problèmes concrets et réels.*

La biologie moléculaire est en pleine expansion et produit un volume de données phénoménal. Par exemple, en 1991, à l'institut Pasteur, l'ensemble des séquences biologiques tenait sur un CD-ROM (soit 650 Mo). En 1997, l'ensemble des banques occupe plus de 28 Go, réparti en 20 000 fichiers. Cette croissance, exponentielle, se traduit par des temps de calcul de plus en plus longs lorsqu'il s'agit de manipuler ces données ; par exemple pour la comparaison de ces séquences, la mise à jour des banques, la gestion de leur cohérence, l'élimination des redondances, l'interrogation flexible, etc. Ces problèmes restent dans le cadre de nos compétences (le traitement des chaînes de caractères) et sont des sources d'inspiration pour nos travaux.

Mais le problème de la comparaison de séquences biologiques tel que nous l'avons traité jusqu'à présent (machine Samba) est loin d'être résolu. La production automatique des textes des séquences d'ADN ou le séquencement de génomes complets, par exemple, demandent de nouvelles puissances de calcul pour traiter (comparer) ces données. Aujourd'hui les banques

sont constituées de centaines de milliers de séquences (pour simplifier des gènes) de quelques milliers de caractères chacune. Demain elles contiendront des génomes complets dont la taille est 1000 fois plus importantes (quelques millions de caractères). Il est alors fort probable que les biologistes souhaiteront manipuler ces entités (les génomes) comme ils manipulent actuellement les gènes. Les machines spécialisées, et notamment les architectures reconfigurables, ont donc d'intéressantes perspectives.

4.5 Filtrage adaptatif

Résumé : *De nombreuses applications liées aux télécommunications requièrent l'utilisation de circuits spécialisés parallèles. Les algorithmes de filtrage adaptatif, en particulier, se prêtent bien à ce type de réalisation. En collaboration avec l'université de Trois Rivières au Québec, des architectures de filtres sont étudiées et synthétisées pour être mises en œuvre sur des circuits reconfigurables ou des circuits intégrés. Ces architectures ont des applications pour l'égalisation de canal en télécommunication, la robotique, ou les mesures en milieu bruité par exemple.*

De nombreuses applications liées aux télécommunications requièrent l'utilisation de circuits spécialisés parallèles. Les algorithmes de filtrage adaptatif, en particulier, se prêtent bien à ce type de réalisation. En collaboration avec l'université de Trois Rivières au Québec, des architectures de filtres sont étudiées et synthétisées pour être mises en œuvre sur des circuits reconfigurables ou des circuits intégrés.

Deux algorithmes de filtrage adaptatif particuliers sont étudiés : un algorithme de filtrage avec adaptation retardée par la méthode des moindres carrés (DLMS), et un algorithme de filtrage avec adaptation par réseau de neurones. Dans les deux cas, les recherches effectuées consistent à synthétiser une architecture à l'aide du logiciel MMAlpha, et à comparer l'architecture obtenue avec une version conçue avec des outils standard. La synthèse met en évidence des limitations du logiciel MMAlpha qui conduisent à en étendre les fonctions, faisant ainsi progresser les techniques de synthèse.

Dans le cas du DLMS, ces recherches ont conduit à la réalisation d'un bloc flexible IP (*Intellectual Property*) qui a été présenté au concours *Design Contest* de la conférence Date'01, et a remporté le second prix.

Le circuit de réseaux de neurones est en cours d'étude, et sera comparé à une version analogique.

Ces applications sont étudiées dans le cadre d'une collaboration à long terme avec l'université du Québec à Trois Rivières, collaboration soutenue au cours du temps par divers contrats.

5 Logiciels

5.1 Panorama

Résumé :

Les réalisations au niveau logiciel du projet Cosi sont matérialisées par PolyLib, bibliothèque open source *de calcul sur les polyhèdres et* MMA*lpha pour la synthèse de haut niveau.*

5.2 PolyLib

Participants : Sorin Olaru, Patrice Quinton [correspondant], Tanguy Risset.

Mots clés : bibliothèque, calcul polyhèdre.

Résumé : *La bibliothèque polyédrique (initialement développée par Hervé Le Verge et Doran Wilde) permet la manipulation de polyèdres et de fonctions affines. La manipulation des domaines utilisés dans les équations récurrentes ou des espaces d'indices décrits par les boucles imbriquées justifie l'emploi d'une telle bibliothèque.*

PolyLib. La *bibliothèque polyédrique* PolyLib[1] ou `http://icps.u-strasbg.fr/PolyLib/` est une bibliothèque C, *open source* de calcul sur les polyèdres convexes. Elle a été développée initialement par Hervé Le Verge et Doran Wilde à l'Inria Rennes. La manipulation des domaines utilisés dans les équations récurrentes ou des espaces d'indices décrits par les boucles imbriquées justifie l'emploi d'une telle bibliothèque. Cette bibliothèque est actuellement utilisée (indépendamment de MMAlpha) par plusieurs organismes de recherche (en Angleterre, États Unis, ainsi qu'en France).

La bibliothèque de calcul polyédrique PolyLib (correspondants : olaru@irisa.fr, quinton@irisa.fr) fait l'objet d'une opération de développement logiciel (ODL) démarrée en octobre 2001. Dans cette ODL, nous proposons de consolider l'implémentation de PolyLib grâce à la mise en place de tests systématiques et complets, l'écriture d'une véritable documentation utilisateur, la mise en place d'une interface conviviale ainsi que la diffusion élargie du logiciel.

5.3 MMAlpha

Participants : David Cachera, Anne-Claire Guillou, Fabien Quilleré, Patrice Quinton [correspondant], Sanjay Rajopadhye, Tanguy Risset.

Mots clés : synthèse d'architecture, CAO, ASIC, programmation fonctionnelle, parallélisme de données, parallélisation automatique.

Résumé : *Le logiciel* MMA*lpha est une plate-forme écrite en Mathematica et C permettant de manipuler des programmes Alpha dans le double but de générer soit des architectures régulières à partir de spécifications de haut niveau, soit du code pour des machines programmables. Les techniques utilisées sont celles de la parallélisation automatique et de synthèse de réseaux systoliques.*

MMAlpha. MMAlpha est un logiciel qui implémente des transformations sur le langage Alpha. Le langage Alpha a été proposé par Christophe Mauras lors de sa thèse en 1989 [Mau89].

[1] `http://www.ee.byu.edu/~wilde/polyhedra.html`

[Mau89] C. MAURAS, *Alpha : un langage équationnel pour la conception et la programmation d'architectures parallèles synchrones*, thèse de doctorat, Université de Rennes 1, IFSIC, décembre 1989.

L'implémentation est écrite dans les langages Mathematica (d'où le nom MMAlpha) et s'appuie sur une bibliothèque écrite en C.

Les transformations de programmes Alpha sont implémentées en utilisant les possibilités de Mathematica et de la bibliothèque polyédrique. Le principe d'utilisation de ces transformations est de dériver soit une architecture, soit du code séquentiel ou parallèle à partir d'une spécification algorithmique d'un traitement. Ces transformations sont semi-automatiques, c'est-à-dire que les actions à effectuer sont indiquées par l'utilisateur mais la transformation elle-même est exécutée par MMAlpha. Ceci permet de limiter les erreurs lors de la manipulation manuelle de programmes. Il est possible d'effectuer une dérivation automatique par défaut mais l'expérience montre que l'espace de conception est si important que cela est rarement satisfaisant.

La méthodologie de conception est héritée de la méthode de synthèse de réseaux systoliques. Ce domaine a été longuement étudié du point de vue théorique et l'environnement MMAlpha permet de tester les différentes stratégies de synthèse existantes, d'étudier différentes possibilités de parallélisation et de générer une description architecturale d'un circuit grâce au format AlpHard (sous-ensemble du langage Alpha). La communication avec les outils de synthèse logique se fait grâce à une traduction automatique du format Alphard vers VHDL.

Le logiciel MMAlpha (`http://www.irisa.fr/cosi/ALPHA/`, correspondant : `quinton@irisa.fr`) est déposé à l'association de protection des programmes, et a été mis sous licence GNU. Il a été le support d'implémentation de nombreuses thèses réalisées à l'Irisa. Il est utilisé par quelques équipes de recherche dans le cadre de collaborations avec Cosi. Actuellement c'est un des seuls outils permettant de décrire un algorithme et son implémentation matérielle dans le même langage et de déduire cette implémentation avec des transformations sûres.

MMAlpha est actuellement utilisé à Oxford (Oxford University Computing Laboratory), à Trois Rivières (Département de génie électrique de l'université du Québec à Trois Rivières) et à Calcutta (Indian Statistical Institute).

6 Résultats nouveaux

6.1 Synthèse de très haut niveau

Mots clés : synthèse d'architecture, CAO, ASIC.

Participants : David Cachera, Steven Derrien, Anne-Claire Guillou, Gautam Gupta, Katell Morin-Allory, Sorin Olaru, Patrice Quinton, Fabien Quilleré, Sanjay Rajopadhye, Tanguy Risset, Charles Wagner.

Résumé : *Notre effort est concentré sur le développement du logiciel* MMA*lpha (voir section 5.3) pour (1) la synthèse d'architectures régulières – notamment systoliques – et (2) la génération de code séquentiel ou parallèle à l'aide de transformations interactives. Le langage Alpha est la base de ce logiciel. Il autorise à la fois l'expression d'un algorithme parallèle régulier, et la description d'une architecture synchrone qui en supporte l'exécution.*

Recherches menées au cours de l'année 2001

Nos recherches en 2001 ont porté sur les aspects suivants :
- la poursuite de la mise en place d'une chaîne de compilation Alpha→FPGA ;
- la génération d'interfaces entre un système matériel généré par le système MMAlpha et le système (logiciel) qui contrôlera ce matériel ;
- l'étude de méthodes de pavage et de *retiming* appliquées aux co-processeurs FPGA ;
- l'utilisation de MMAlpha en tant que plate-forme de prototypage rapide pour différents algorithmes ;
- l'étude de méthodes pour déterminer automatiquement la largeur du chemin de données dans une architecture ;
- l'utilisation de méthodes formelles pour la démonstration de propriétés de systèmes Alpha ;
- l'étude algorithmique du problème du chemin algébrique.

Résultats
- En ce qui concerne la génération d'interfaces, le travail a porté sur l'interfacage des architectures générées par les descriptions Alpha à travers le bus qui relie le FPGA au processeur. Un traducteur a été écrit qui, à partir de n'importe quel programme AlpHard représentant un réseau systolique linéaire, génère le code VHDL de l'interface. D'autre part, une interface a été écrite pour l'application Samba, basée sur un protocole qui sérialise les données pour les faire passer sur le bus puis les parallélise à l'entrée de l'architecture systolique.
- L'application des méthodes de pavage aux FPGA consiste en la détermination du meilleur compromis entre taille de la mémoire locale et parallélisme, sous contraintes de bande passante. Une solution analytique a été donnée, et donne lieu à une validation expérimentale sur la carte SPYDER-X2 [15]. D'autre part, l'utilisation de techniques de *retiming* permet d'aller vers un parallélisme à grain fin des processeurs produits par MMAlpha [14].
- Le flot de synthèse de MMAlpha a été validé par des exemples d'applications de « taille réelle ». À partir d'une description Matlab de l'application, un code Alpha est généré puis raffiné jusqu'à une description architecturale en VHDL synthétisable, qui peut ensuite être simulée. Nous avons d'abord étudié des algorithmes de filtrage adaptatifs comme celui du filtre DLMS (*Delayed Least Mean Square*) qui implémente la méthode des moindres carrés, et des algorithmes de la même famille comme le *Look-Ahead DLMS* [8, 28]. L'architecture obtenue et validée par une simulation post-synthèse nous sert de base pour expérimenter les autres aspects de MMAlpha tels que l'interfaçage entre l'architecture implantée sur le FPGA de la carte SPYDER et une station hôte. Nous poursuivons ce travail avec l'étude d'un algorithme de *rétro-propagation dynamique* (RPD) basé sur un réseau de neurones, utilisé pour l'identification de systèmes dynamiques non-linéaires. Ce travail a été réalisé en collaboration avec l'Université du Québec à Trois-Rivières.
- L'utilisation de systèmes structurés d'équations récurrentes nécessite la mise au point de nouvelles techniques d'ordonnancement. Une nouvelle méthode d'ordonnancement pour de tels systèmes a été proposée et expérimentée sur plusieurs exemples. Un des buts est de quantifier la réduction de complexité que de telles méthodes apportent par rapport

à une « mise à plat » des systèmes et une utilisation des méthodes d'ordonnancement classiques. Pour un exemple de système implantant un réseau de neurones, le temps nécessaire au calcul de l'ordonnancement a été divisé par dix [23].
- La détermination de la largeur du chemin de données passe par une traduction d'un programme Alpha vers une forme abstraite (exprimée également en Alpha) qui consiste en un système d'équations sur les tailles des mots codant chaque variable. Dans certains cas, il est possible de résoudre ces équations par un appel au solveur de Mathematica, ou par l'utilisation de calculs algébriques [27]. La détermination de certaines largeurs doit faire appel à l'utilisateur, en particulier lorsque celles-ci sont liées à des propriétés intrinsèques de l'algorithme (convergence par exemple). L'implémentation de ces méthodes dans l'environnement MMAlpha est en cours, via l'enrichissement du système de typage de Alpha.
- Le flot de dérivation MMAlpha qui part d'une spécification de très haut niveau pour arriver à une description architecturale garantit théoriquement la correction de l'architecture finale. Ceci est dû au fait que les transformations utilisées conservent la sémantique des systèmes. Cependant, il est parfois nécessaire de valider certaines transformations (qui ne préservent pas intégralement la sémantique), ou de vérifier certaines propriétés non triviales de spécifications complexes. Dans ce but, nous avons défini et implémenté un ensemble de méthodes de vérification formelle qui font appel à la fois à MMAlpha et au démonstrateur de théorèmes PVS [12]. Par ailleurs, nous avons commencé à explorer de nouvelles techniques de preuve faisant directement appel au modèle polyédrique : plutôt que de faire appel à des outils externes de vérification, nous développons des algorithmes de vérification qui utilisent la librairie de calcul polyédrique. Cette méthode est utilisée dans le cadre de la vérification de propriétés de contrôle.

6.2 Compilation pour processeurs spécialisés programmables

Mots clés : ASIP, exploration architecturale, compilation reciblable, modélisation d'architecture, génération de code.

Participants : François Charot, Ferry Djieya, Olivier Sentieys, Charles Wagner.

Résumé : *Les cœurs de processeurs spécialisés programmables sont une technologie de réalisation de systèmes sur silicium. Leur conception requiert des outils d'exploration architecturale s'appuyant sur des techniques de compilation flexible pilotées par une modélisation du processeur cible dans le langage Armor.*

Recherches menées au cours de l'année 2001

Notre effort est concentré sur le développement de l'environnement Calife/Armor pour la génération de code flexible pour Asip. Les travaux de recherche ont plus particulièrement porté sur les aspects suivants :
- l'étude de l'estimation de performance logicielle d'un code optimisé s'appuyant sur l'environnement Armor/Calife, et le développement de passes spécifiques à l'estimation ;

– l'étude et le développement d'une passerelle entre le langage Armor et le langage VHDL, pour la partie chemin de données du processeur.

Résultats

– L'environnement Calife/Armor est basé sur une bibliothèque de modules de production et d'optimisation de code [7]. La flexibilité est à deux niveaux : flot de compilation et module de la bibliothèque. Un flot de compilation consiste en un agencement de modules choisis dans la bibliothèque, le flot de compilation le plus adapté à l'architecture du processeur cible peut ainsi être spécifié. Les modules sont paramétrés automatiquement à partir de la description du processeur cible en langage Armor.
– Des passes spécifiques à l'estimation de performances réalisant la sélection d'instructions et l'ordonnancement ont été développées [13]. Elles ont permis la réalisation d'un estimateur pour le processeur OakDSPCore, un processeur de traitement de signal très largement utilisé dans des applications en télécommunication. Cet estimateur a été expérimenté sur des noyaux d'applications de traitement de signal typiques. Il montre des performances très proches de celles du code assembleur développé manuellement.

6.3 Outils de CAO pour architectures reconfigurables

Mots clés : composant FPGA, architectures reconfigurables.

Participants : Steven Derrien, Erwan Fabiani, Dominique Lavenier, Laurent Perraudeau, Sanjay Rajopadhye, Tanguy Risset, Charles Wagner.

Résumé :

Les outils de CAO *pour* FPGA *que nous développons se situent en aval du processus de conception d'architectures régulières. À partir de la description d'une architecture (niveau transfert de registres) synthétisée par Alpha nous automatisons le processus d'implantation sur une plate-forme reconfigurable.*

Recherches menées au cours de l'année 2001

Par architecture régulière, nous entendons un tableau linéaire ou bidimensionnel de cellules élémentaires. Ce tableau est connecté à un processeur hôte qui reçoit et émet des données de/vers cette structure. Dans un premier temps, nous ne considérons que les réseaux linéaires. Nos efforts se concentrent sur trois points particuliers.

– La définition d'une stratégie d'implémentation des architectures régulières sur plate-formes reconfigurables. Cette stratégie est mise en œuvre par l'outil Snake.
– Le protocole d'interfaçage entre la plate-forme reconfigurable et le processeur hôte.
– Des transformations basées sur le pavage de boucles qui permettent une meilleure adaptation aux systèmes de mémoires hiérarchiques.

Résultats

- L'outil Snake prend en entrée une description non placée d'une architecture linéaire et produit une description placée de cette même architecture. L'idée principale repose sur le fait que les outils de CAO des constructeurs ne prennent pas en compte la régularité des structures que nous visons, ce qui se traduit à la fois par des temps de placement et de routage relativement longs (plusieurs dizaines de minutes, voire plusieurs heures) et un placement final non satisfaisant. Notre stratégie de placement consiste d'abord à évaluer les divers placements possibles d'une cellule. La seconde phase construit un serpentin dont le placement est optimisé en fonction des ressources reconfigurables disponibles. L'outil Snake est en cours d'élaboration. Les premières expérimentations montrent que pour un placement qui respecte la structure des architectures, nous obtenons un gain de 3,5 sur les temps de calcul. La thèse d'Erwan Fabiani qui porte sur ce sujet sera soutenue fin 2001.
- L'interfaçage de l'architecture régulière avec le reste du système, en particulier avec le processeur hôte, est très délicat. D'une part il conditionne fortement les performances du système et, d'autre part, il est souvent difficile à mettre au point manuellement. Notre stratégie consiste à réaliser les échanges via un modèle unique de file d'attente. Vu du processeur hôte, le programmeur dispose d'un nombre restreint de primitives pour émettre ou recevoir de l'information vers l'architecture reconfigurable. Vu du circuit, on dispose d'un protocole simple à partir duquel les échanges s'établissent. À partir de ces briques de base, une spécification d'échange de haut niveau génère la partie programme sur le processeur hôte et la partie matérielle sur la plate-forme reconfigurable.
- Nous avons proposé un ensemble de transformations architecturales permettant aux processeurs du réseau de tirer parti d'un chemin de données très fortement pipeliné [14]. Ces transformations formelles préservent le fonctionnement du circuit tout en en améliorant les performances. Nos résultats expérimentaux sont très prometteurs : pour certaines applications, nous avons pu observer une amélioration de la fréquence de fonctionnement par des facteurs allant jusqu'à un ordre de grandeur, pour un surcoût en surface minimal (toujours inférieur à 100%).
- Le principal facteur limitant les performances des accélérateurs matériels reconfigurables est la bande passante entre le circuit FPGA et la mémoire principale du système. Nous avons donc adapté des transformations basées sur le pavage de boucles [15], qui permettent une meilleure adaptation aux systèmes de mémoires hiérarchiques disponibles sur la plupart des accélérateurs. Ces tranformations ont été associées à un modèle de performance analytique qui permet de déterminer les paramètres de pavage optimaux. Une validation expérimentale basée sur une application de comparaison banque à banque de séquences génomiques a permis de confirmer les résultats théoriques.

6.4 Algorithmique fondamentale

Mots clés : Fermeture transitive, plus courts chemins, inversion de matrices, synthèse systolique, équations récurrentes, machines à mémoire distribuée, démonstrateurs de théorèmes..

Participants : David Cachera, Tanguy Risset, Sanjay Rajopadhye.

Résumé : *Notre travail a porté sur le problème du chemin algébrique. Nous avons défini l'implémentation d'un algorithme résolvant ce problème sous la forme d'une architecture SIMD linéaire efficace.*

En collaboration avec C. Tatonki de l'université de Yaoundé, nous avons travaillé sur le problème du chemin algébrique. Notre but était d'implémenter un algorithme permettant de résoudre ce problème sous la forme d'une architecture SIMD linéaire efficace. Pour un graphe avec n nœuds, notre réseau a n processeurs, chacun d'eux disposant de n cellules mémoire et calcule le résultat en $2n^2 + n - 2$ étapes. Notre réseau est parfaitement adapté à une synthèse VLSI car le contrôle est assez simple et la mémoire est implémentée par des files. Seuls les processeurs situés à chaque extrémité de ce réseau linéaire communiquent avec l'hôte. Nous avons donné une dérivation formelle de ce réseau, qui s'appuie sur le formalisme des équations récurrentes, et la validité de l'ordonnancement et de la stratégie d'allocation de la mémoire a été prouvée, en partie au moyen d'un démonstrateur de théorèmes. Le réseau peut immédiatement être adapté pour fonctionner en plusieurs passes, et, de plus, cette version améliore l'efficacité en termes de travail. Pour tout entier positif a, le temps de calcul sur $\frac{n}{a}$ processeurs est inférieur à $(\frac{a+1}{a})n^2$. Le travail est inférieur à $(1 + \frac{1}{a^2})n^3$, et peut être approché de n^3 autant que désiré. Finalement, nous avons proposé une version par blocs de notre ordonnancement et traité le problème du partitionnement optimal. Les résultats ont été validés expérimentalement sur un Cray T3E. Ce travail a fait l'objet d'une publication interne [26] et d'une soumission à IEEE TPDS.

7 Contrats industriels (nationaux, européens et internationaux)

7.1 SPART, Méthodes de spécification pour la conception d'architectures de contrôle de trafic en ATM, (CTI Cnet, 197C9350031312012)

Participants : François Charot, Ferry Djieya, Sanjay Rajopadhye, Charles Wagner.

Résumé : *Le projet* SPART *(décembre 1997 - juin 2001) est une convention* CTI *du* CNET *et de France-Télécom.* SPART *concerne l'expérimentation de méthodes de spécification pour la conception d'architectures de contrôle de trafic en* ATM.

Le travail de recherche effectué dans SPART concerne l'expérimentation des environnements de conception de systèmes hétérogènes existants et utilisés couramment en traitement de signal (tels que Cossap, SPW et Ptolemy), dans le but d'améliorer le processus de conception d'architectures supportant les algorithmes de contrôle et de lissage de trafic, de gestion de ressources, etc.

Les résultats de ces travaux concernent les aspects suivants.

- L'étude des algorithmes de contrôle de trafic et de gestion de ressource en ATM a montré que ceux-ci pouvaient être spécifiés dans le modèle de calcul de type flot de données synchrone, SDF. Celui-ci est supporté par des outils de spécification de systèmes tels que Ptolemy, SPW, Cossap.
- L'expérimentation des environnements de conception SPW et Ptolemy pour la spécification d'algorithmes de contrôle de trafic et de gestion de ressources en ATM a été concré-

tisée par le développement d'une bibliothèque pour l'environnement Ptolemy. Cette bibliothèque a permis la spécification d'un contrôleur-espaceur.
- L'étude de techniques d'estimation de performance d'un code optimisé, basée sur la construction de flots d'estimation dans l'environnement Calife, flots paramétrés par une description du processeur candidat dans le langage de modélisation de processeurs Armor.

Les perspectives de ces travaux concernent la mise en place d'un ensemble de techniques (synthèse d'architecture, *profiling*, compilation flexible, estimation flexible, etc.) contribuant par raffinements successifs aux choix d'implémentation d'une application sur des plates-formes matérielles mélangeant différentes technologies de réalisation.

8 Actions régionales, nationales et internationales

8.1 Actions régionales

- Relations avec l'UBO, le LIP et participation au Club d'Architecture de l'Ouest avec Enssat, l'ENST, l'Ireste, l'UBS, et l'UBO.

8.2 Actions nationales

- Collaboration avec les projet A3 (Rocquencourt/Versailles), CRI (ENSMP), ICPS (Strasbourg), ReMap (Lyon), LIFL (Lille) sur la parallélisation automatique et les fondements du modèle polyédrique.
- Participation aux GdR CAO, Isis (thème AAA) et ARP (thème HiPerf).
- Collaboration avec le Lamim de l'université de Valenciennes sur les méthodes d'optimisation dans la parallélisation.

8.3 Relations bilatérales internationales

8.3.1 Europe

- Collaboration avec l'Imec (Interuniversity Microelectronics Center), Louvain, Belgique (F. Catthoor) sur le partitionement et l'optimisation de la mémoire pour des systèmes multimédia (co-encadrement de la thèse de Thierry Omnès).
- Cosi est partenaire extérieur du réseau de recherche PACT (Program acceleration by Application-driven & architecture-driven Code Transformations) organisé par le fonds de la recherche scientifique flamand sur la conception d'architectures.

8.3.2 Pacifique et Asie du Sud

- Le projet CORCoP (Compilation and Optimization for Reconfigurable Co-processors, financé par un projet Cefipra, 198C5320031312005)), implique le projet Cosi et le Indian Statistical Institute, Calcutta, Inde (B. Sinha et S. Sur-Kolay). Ce projet de trois ans a débuté en février 1999. Les objectifs de CORCoP sont le développement de méthodes de compilation incluant des techniques d'optimisation adaptées aux co-processeurs

reconfigurables (bâtis avec des composants FPGA). Notre contribution porte sur le développement de modèles quantitatifs à chaque niveau du processus de compilation – parallélisation de boucles, optimisation de code, spécification de matériel et optimisation au niveau des circuits.
- Participation à la mise en place de la coopération Inria – Instituts indiens de technologie (IIT). Dans ce cadre, S. Rajopadhye a organisé l'accueil de plusieurs étudiants en stage d'été dans divers laboratoires français.

8.3.3 Afrique

- Dans le cadre du projet Cari et du programme FICU de l'Agence Universitaire pour la Francophonie, Cosi coopère avec l'université de Yaoundé (Cameroun) sur l'algorithmique systolique.
- Cosi coopère avec l'université de Tananarive à Madagascar, pour la formation à la recherche d'enseignants–chercheurs.

8.3.4 Amérique du Nord

- Avec l'Electrical Engineering Department, Brigham Young University, USA (D. Wilde), développement du logiciel MMAlpha, et extensions de la bibliothèque Polylib pour manipuler des $\mathcal{Z}$-polyèdres (financé par un projet NSF-Inria).
- Avec le département de Génie Électrique, Université du Québec à Trois Rivières, Canada (D. Massicotte) : compilation d'algorithmes de filtrage adaptatif. Les travaux font l'objet d'une micro-action financée par le groupement de recherche HiPerf, et d'un projet FICU (contrat Agence universitaire pour la Francophonie, 11C03670031326005), de deux ans qui a débuté en janvier 2001.
- Le projet Cosi coopère avec le Los Alamos National Laboratory, Nouveau Mexique, USA sur le thème des architectures reconfigurables pour le traitement d'images hyperspectrales.

8.4 Accueil de chercheurs étrangers

- Susmita Sur-Kolay (Isi, Calcutta) : un mois
- Claude Tadonki (université de Yaoundé, Cameroun) : six mois
- Daniel Massicote (université du Québec à Trois Rivières, Québec) : une semaine
- Doran Wilde (Brigham Young University, Etats-Unis) : une semaine

9 Diffusion de résultats

9.1 Animation de la communauté scientifique

- D. Lavenier est responsable du thème Architectures Spécialisées du pôle Architecture du GdR ARP.
- F. Charot est membre du comité de pilotage du Programme Thématique Pluridisciplinaire "Systèmes Complexes Intégrés sur Puces (SOC)", mis en place au département STIC

du CNRS. F. Charot est co- animateur de l'une des actions spécifiques de ce programme thématique, l'action "Architectures reconfigurables dynamiquement".
- S. Rajopadhye est co-responsable du thème HiPerf du pôle Parallélisme du GdR ARP.
- P. Quinton a été membre du comité de programme de la conférence ISCIS'01 et membre du comité exécutif de DATE'2001. Il est membre du comité exécutif de DATE'2002.
- D. Lavenier a été membre du comité de programme de SympA (Symposium en Architectures de Machines), FPL (International Conference on Field Programmable Gate Array and Applications, et ERSA (International Conference on Engineering of Reconfigurable Systems and Algorithm).
- F. Charot et D. Lavenier sont membres du comité de lecture de la revue Traitement du Signal.
- S. Rajopadhye est membre de l'«advisory board» de la conférence Europar.

9.2 Enseignement universitaire

- D. Lavenier est responsable du DEA Génomique et Informatique, Université de Rennes 1, et d'un cours de ce DEA.
- D. Lavenier donne un cours d'informatique appliquée à la biologie en maîtrise de biochimie.
- F. Charot est responsable d'un cours sur les applications de l'architecture dans les télécommunications en DIIC.
- Le projet Cosi a accueilli des stagiaires : Touria Arich (4 mois), Inderaj Sing Bains (6 mois), Monojit Choudhury (2 mois), Fatima Hadjam (1 mois), Aditya Gupta (6 mois), Rahul Tripathi (6 mois).
- P. Quinton est directeur de l'Ifsic. Il est responsable du cours algorithmique parallèle (module Alpa) dans le DEA d'informatique de l'université de Rennes 1, enseigne en Deug Sciences, mention SM et STPI, ainsi qu'en DIIC (seconde et troisième année).
- L. Perraudeau est directeur adjoint de l'Ifsic (responsable du budget et des équipements informatiques). Il est responsable d'un cours sur les langages objet dans le DESS Isa (Informatique et ses applications) de l'université de Rennes 1, enseigne la conception de circuits intégrés en DIIC deuxième année), et intervient en Licence d'informatique, en Deug Sciences, mention SM et STPI ainsi qu'en formation continue.

9.3 Autres enseignements

- Un cours sur les architectures parallèles spécialisées est donné par T. Risset à Supelec (Paris) dans le cadre de la formation permanente.

9.4 Participation à des colloques, séminaires, invitations

- P. Quinton a été invité au Workshop System Architecture MOdelling and Simulation, en juillet 2001 à Samos, Grèce.
- A.-C. Guillou a passé un mois à l'Université de Québec à Trois-Rivières dans le cadre de la coopération franco-québécoise.

– T. Risset a été invité pour un séminaire à l'OUCL (Oxford Univ. Computing Laboratory, UK).

10 Bibliographie

Ouvrages et articles de référence de l'équipe

[1] F. Charot, G. Le Fol, P. Lemonnier, C. Wagner, C. Bouville, R. Barzic, « Towards Hardware Building Blocks for Software-Only Real Time Video Processing : the MOVIE Approach », *IEEE Transactions on Circuits and Systems for Video Technology 9*, 6, September 1999.

[2] P. Guerdoux-Jamet, D. Lavenier, « SAMBA : Hardware Accelerator for Biological Sequence Comparison », *CABIOS 13*, 6, décembre 1997.

[3] C. Mauras, *Alpha : un langage équationnel pour la conception et la programmation d'architectures parallèles synchrones*, Thèse, Université de Rennes 1, IFSIC, décembre 1989.

[4] P. Quinton, V. V. Dongen., « The mapping of linear recurrence equations on regular arrays », *Journal of VLSI Signal Processing 1*, 1989, p. 93–113.

[5] P. Quinton, Y. Robert, *Systolic Algorithms and Architectures*, Prentice Hall and Masson, 1989.

[6] S. V. Rajopadhye, S. Purushothaman, R. M. Fujimoto, « On Synthesizing Systolic Arrays from Recurrence Equations with Linear Dependencies », *in : Proceedings, Sixth Conference on Foundations of Software Technology and Theoretical Computer Science*, Springer Verlag, LNCS 241, p. 488–503, New Delhi, India, décembre 1986.

Articles et chapitres de livre

[7] F. Charot, V. Messé, « La compilation reciblable au service de la définition interactive d'ASIP », *Technique et Science Informatiques 20*, 2, février 2001.

[8] A.-C. Guillou, P. Quinton, T. Risset, D. Massicotte, « Automatic Design of VLSI Pipelined LMS Architecture », *IEEE Transaction on Parallel and Distributed Systems*, 2001, à paraître.

[9] A. Mozipo, D. Massicotte, P. Quinton, T. Risset, « Structured Design of Parallel VLSI Architectures for Kalman Filters using a Declarative Language », *soumis pour publication à IEEE Transactions on VLSI Systems*, 2001.

[10] S. Rajopadhye, T. Risset, C. Tadonki, « Le chemin algébrique sur réseaux linéaires », *Technique et Science Informatique 20*, 5, 2001, p. 655–676.

Communications à des congrès, colloques, etc.

[11] D. Cachera, A.-C. Guillou, F. Quilleré, P. Quinton, S. Rajopadhye, T. Risset, « Hardware Design Methodology with the Alpha Language », *in : FDL'01*, Lyon, France, septembre 2001.

[12] D. Cachera, P. Quinton, S. Rajopadhye, T. Risset, « Proving Properties of Multidimensional Recurrences with Application to Regular Parallel Algorithms », *in : 6th International Workshop on Formal Methods for Parallel Programming : Theory and Applications (FMPPTA)*, San Francisco, avril 2001.

[13] F. Charot, F. Djieya, « Approche d'estimation flexible de performances pour DSP », *in : 7ième Symposium en Architectures nouvelles de machines*, avril 2001.

[14] S. Derrien, S. Rajopadhye, S. Sur-Kolay, « Combining Instruction and Loop Level Parallelism for Array Synthesis on FPGAs », *in : International Symposium on System Synthesis (ISSS'01), Montréal*, 2001.

[15] S. Derrien, S. Rajopadhye, « Loop Tiling for Reconfigurable Accelerators », *in : International conference on Field Programmable Logic (FPL'01), Belfast*, 2001.

[16] E. Fabiani, D. Lavenier, « Experimental Evaluation of Place-and-Route of Regular Arrays on Xilinx Chips », *in : International Conference on Engineering of Reconfigurable Systems and Algorithms*, Las Vegas, Nevada, USA, 2001.

[17] E. Fabiani, D. Lavenier, « Stratégie de placement de réseaux linéaires sur circuits FPGA », *in : 7ème Symposium en Architectures Nouvelles de Machines*, Paris, France, 2001.

[18] J. Frigo, M. Gokhale, D. Lavenier, « Evaluation of the Streams-C C-to-FPGA Compiler : an Application Perspective », *in : 9th ACM International Symposium on Field-Programmable Gate Arrays*, Monterey, CA, USA, 2001.

[19] M. Gokhale, J. Frigo, K. McCabe, J. Theiler, D. Lavenier, « Early Experience with a Hybrid Processor : K-Means Clustering », *in : International Conference on Engineering of Reconfigurable Systemsand Algorithms*, Las Vegas, Nevada, USA, 2001.

[20] L. Lagadec, D. Lavenier, E. Fabiani, B. Pottier, « Placing, Routing and Editing Virtual FPGAs », *in : FPL 2001, 11th International Conference onField Programmable Logic and Applications*, Belfast, Northern Ireland, U.K., 2001.

[21] D. lavenier, E. Fabiani, S. Derrien, C. Wagner, « Systolic Array for Computing the Pixel Purity Index (PPI) Algorithm on Hyperspectral Images », *in : SPIE Conference on Imaging Spectrometry*, San Diego, CA, USA, 2001.

[22] M. Manjunathaiah, G. Megson, T. Risset, S. Rajopadhye, « Uniformization of Affine Dependence Programs for Parallel Embedded System Design », *in : Internationnal Conference on Parallel Processing*, L. Ni, M. Valero (éditeurs), p. 205–213, 2001.

[23] P. Quinton, T. Risset, « Structured Scheduling of Recurrence Equations : Theory and Practice », *in : Proc. of the System Architecture MOdelling and Simulation Workshop, Lecture Notes in Computer Science*, Springer Verlag, Samos, Greece, 2001. À paraître.

[24] Y. Solihin, K. Cameron, Y. Luo, D. Lavenier, M. Gokhale, « Mutable Functional Units and their Applications on Microprocessors », *in : ICCD 2001, International Conference on Computer Design*, Austin, Texas, USA, 2001.

[25] Y. Solihin, K. Cameron, Y. Luo, D. Lavenier, M. Gokhale, « Mutable Functional Units : Initial Results », *in : IEEE Symposium on Field-Programmable Custom Computing Machines*, Rohnert Park, CA, USA, 2001.

Rapports de recherche et publications internes

[26] D. Cachera, S. Rajopadhye, T. Risset, C. Tadonki, « Parallelization of the Algebraic Path Problem on Linear SIMD/SPMD Arrays », *Publication interne n°1409*, IRISA, jul 2001, soumis pour publication à IEE Transactions on Parallel and Distributed Systems.

[27] D. Cachera, T. Risset, D. Zegaoui, « Formal Bit Width Determination for Nested Loop Programs », *Publication Interne n°1389*, Irisa, janvier 2001.

[28] A. Guillou, P. Quinton, T. Risset, D. Massicotte, C. Wagner, « High Level Design of Digital Filters in Mobile Communications », *Publication Interne n°1405*, Irisa, juin 2001.

Divers

[29] F. Charot, F. Djieya, « Rapport de contrat : Estimation de performances logicielles avec Calife/Armor », Convention Cnet 97 1B 546. Méthodes de spécification pour la conception d'architectures de contrôle de trafic en ATM, septembre 2001.

[30] A.-C. Guillou, P. Quinton, T. Risset, D. Massicotte, C. Wagner, « High Level Design of Digital Filters in Mobile Communications », DATE Design Contest 2001, mars 2001, Second place.