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

Responsable scientifique

Sanjay Rajopadhye [CR CNRS]

Assistante de projet

Maryse Auffray [AA Inria]

Personnel Inria

François Charot [CR Inria]
Tanguy Risset [CR Inria]
Vincent Messé [ingénieur expert à partir du 1er mars]

Personnel Upresa 6074

Dominique Lavenier [CR CNRS]
Laurent Perraudeau [maître de conférences université de Rennes 1]
Patrice Quinton [professeur, à l'université de Rennes 1]
Charles Wagner [IR CNRS (Atelier)]

Chercheurs doctorants

Franck Bardoult [bourse Inria jusqu'au 30 Septembre 1999]
Steven Derrien [bourse MENESR]
Ferry Djieya [bourse Inria/CIES]
Erwan Fabiani [bourse MENESR]
Anne-Claire Guilloux [bourse Inria à partir du 1er octobre 1999]
Fabien Quilleré [bourse MENESR jusqu'au 10 septembre 1999]

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[1]. Le plus souvent possible, la conception d'un système dédié

1. Asip signifie *Application Specific Instruction-set Processor* et se dit d'un processeur à jeu d'instructions spécifique conçu pour exécuter le plus efficacement possible un petit nombre d'algorithmes.

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.

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 le 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 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 prototype 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 méthodes peuvent trouver des applications dans plusieurs domaines applicatifs. Comme il est essentiel que nos techniques et méthodes soient validées sur des applications réelles, nous choisissons certains volets spécifiques. Sachant que chaque volet nécessite un investissement lourd pour arriver à des résultats concrets et significatifs, nous choisissons ces domaines en forte collaboration avec d'autres chercheurs ou projets, ou en réponse aux besoins des partenaires contractuels.*

Dans ce contexte, nos domaines applicatifs actuels sont le traitement d'image 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

Mots clés : ASIP, compilation, optimisation de code.

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 pour le traitement bas niveau d'images.*

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.

[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.

L'étude d'applications dans le domaine de la compression d'image, menée dans le projet depuis maintenant quelques années, a conduit à définir des architectures semi-spécialisées réalisées à partir de briques de base matérielles et logicielles et permettant la mise en œuvre rapide d'applications, pour les besoins de la simulation. Les travaux réalisés concernent les aspects suivants :

- l'étude de l'utilisation des réseaux de processeurs SIMD pour la réalisation de simulateurs temps réel d'algorithmes de compression de séquences d'images animées. Cette étude a abouti à la spécification du circuit Movie ainsi que son environnement de programmation;
- l'étude d'architectures spécialisées pour une nouvelle classe d'algorithmes d'estimation de mouvement, dits «bloc récursifs» tant du point de vue de leurs paramètres pertinents que de leur mise en œuvre dans le silicium.

Nous poursuivons l'investissement réalisé dans ce domaine d'application, et des algorithmes du domaine servent de support d'étude pour l'expérimentation des outils de compilation pour Asip.

4.3 Conception d'architectures pour les télécommunications

Mots clés : conception par codesign, spécification, flot de données, ATM.

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.

5 Logiciels

5.1 Panorama

Résumé : *Pour le projet Cosi, cette section serait mieux vue sous le titre «Réalisations». Elle inclut les réalisations de machines spécifiques, en plus des logiciels de conception d'architectures.*

Les réalisations au niveau logiciel du projet Cosi sont matérialisées par MMAlpha *pour la synthèse de haut niveau, et C-stolic, pour la programmation de réseau systoliques linéaires. La réalisation de l'accélérateur systolique Samba, dédié à la comparaison de séquences biologiques est un exemple d'application de techniques systoliques.*

5.2 MMAlpha

Participants : Franck Bardoult, Fabien Quilleré, Patrice Quinton, Sanjay Rajopadhye, Tanguy Risset.

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

Résumé : *Le logiciel* MMAlpha *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 la réseaux systoliques.*

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

Le noyau de ce logiciel est la librairie polyédrique développée par Hervé Le Verge et Doran Wilde [Wil93]. La librairie polyédrique 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 librairie. Cette librairie est actuellement utilisée (indépendamment de MMAlpha) par plusieurs organismes de recherche (en Angleterre, États Unis, ainsi qu'en France).

Les transformations de programmes Alpha sont implémentées en utilisant les possibilités de Mathematica et de la librairie 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 exécuté par MMAlpha, ce qui 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/api/ALPHA/, correspondant : risset@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 distribué à quelques équipes de recherche dans le cadre de collaborations précises. 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.

[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.

[Wil93] D. Wilde, « A library for doing polyhedral operations », *rapport de recherche n° 785*, IRISA, Rennes, France, 1993.

5.3 C-stolic

Participants : Dominique Lavenier, Gwendal Le Fol.

Mots clés : algorithme systolique, simulation, parallélisme.

Résumé : *Cstolic est un langage adapté à la programmation d'algorithmes systoliques. Il se base sur un mécanisme fondamental : l'affectation systolique. Cette opération, associée à une nouvelle classe, baptisée «systolic», explicite tous les mouvements de données au sein d'une architecture systolique.*

Le compilateur génère un code portable sur différentes machines parallèles ainsi qu'une version séquentielle, disponible dans le domaine public, pour exécuter et valider la parallélisation d'algorithmes systoliques.

Cstolic est un langage à parallélisme de données qui reprend la syntaxe du langage C, avec deux extensions principales. La première est la définition d'une nouvelle classe d'objets, la classe *systolic*, qui représente les objets (les variables) manipulés sur le réseau systolique linéaire. La seconde extension est l'introduction de l'opérateur d'*affectation systolique* qui décrit, à la fois, les transferts de données au sein du réseau, et les communications avec l'extérieur du réseau.

La compilation d'un programme C-stolic génère deux codes (en C). L'un est associé au traitement externe (alimentation en données, récupération et traitement des résultats), l'autre représente la tâche exécutée par le réseau systolique. Ces deux codes sont synchronisés par des primitives de communication qui dépendent de la machine sur laquelle s'exécute le programme C-stolic.

Une version «allégée» du compilateur est disponible dans le domaine public. Cette version compile et exécute un programme C-stolic uniquement sur une machine séquentielle. Cet outil valide par simulation l'exactitude d'une description systolique d'un algorithme (voir `http://www.irisa.fr/CSTOLIC/`, correspondant : lavenier@irisa.fr).

5.4 Samba

Participants : Dominique Lavenier, Charles Wagner.

Mots clés : biologie moléculaire, architecture systolique, comparaison de séquences.

Résumé : *Samba (Systolic Accelerator for Molecular Biological Applications) est un accélérateur matériel dédié à un traitement de base de la biologie moléculaire : la comparaison de séquences génétiques. Le cœur de l'accélérateur est un réseau linéaire de 128 processeurs. Connecté à un ordinateur hôte, l'accélérateur divise les temps de calcul liés aux traitements des séquences par un facteur 100. Dominique Lavenier a reçu le prix Seymour Cray France 1996 pour cette architecture.*

La biologie moléculaire doit faire face à une croissance exponentielle des banques de données (banques de séquences nucléotidiques et protéiques). Les traitements informatiques associés, en dépit d'ordinateurs toujours plus performants, deviennent de plus en plus longs. Samba est une solution pour réduire fortement les temps de calcul dans ce domaine.

Le prototype réalisé dans l'équipe est bâti autour d'un réseau systolique linéaire de 128 processeurs (32 puces de 4 processeurs). Les processeurs sont des circuits *full custom* conçus spécialement pour accélérer une famille d'algorithmes propre à la comparaison de séquences génétiques. Ils sont donc *paramétrables* pour adapter le traitement à toutes les applications de la biologie moléculaire qui manipulent des séquences de manière intensive.

La programmation de l'accélérateur Samba fait appel à une bibliothèque de procédures et de fonctions que l'on inclut dans un programme C. Cette approche autorise l'élaboration de nouvelles applications de manière aisée et rapide.

L'accélération dépend de l'application. Typiquement, l'exploration d'une banque de séquences est 50 fois plus rapide que l'usage de programmes standard tels que sSearch ou Bestfit, programmes réputés pour la qualité des résultats produits, mais aussi par le temps de calcul excessif. Pour des applications encore plus coûteuses, en terme de quantité de calculs, comme la comparaison banque à banque, les performances de Samba sont optimales : il y a peu d'entrées/sorties (relativement au calcul) et le réseau systolique fonctionne à plein régime. Des accélérations de l'ordre de 200 et plus peuvent être obtenues.

L'accélérateur Samba peut être testé grâce à un site internet (correspondant : Dominique Lavenier, http://www.irisa.fr/SAMBA, lavenier@irisa.fr).

6 Résultats nouveaux

6.1 Synthèse de très haut niveau

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

Participants : Franck Bardoult, Patrice Quinton, Fabien Quilleré, Sanjay Rajopadhye, Tanguy Risset, Dominique Lavenier, Erwan Fabiani.

Résumé : *Le langage Alpha offre la possibilité de développer une plate-forme de synthèse de haut niveau pour des algorithmes réguliers (traitement du signal, algèbre linéaire par exemple), jusqu'au niveau architectural (ou niveau transfert de registres). Cette description architecturale peut alors être implementé en* VLSI *ou en* FPGA. *Le processus de conception comprend divers traitements complexes (parallélisation, description d'architecture par exemple) qui constituent autant de sujets de recherche. La validation des techniques est faites grâce au logiciel* MMA*lpha (voir section 5.2).*

Notre effort est concentré sur le développement du logiciel MMAlpha (voir section 5.2) 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 Alfa 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.

Nos recherches en 1999 ont porté sur les aspects suivants :

– la mise en place d'une chaîne de compilation Alfa →FPGA ;

- l'étude de la génération d'interfaces entre un système matériel généré par le système MMAlpha et le système (logiciel) qui controlera ce matériel ;
- la compilation d'Alfa sur une machine à usage général en optimisant la taille mémoire utilisée ;
- l'extension de la librairie polyèdrique aux calculs sur les $\mathcal{Z}$-polyèdres ;
- l'étude de l'utilisation de MMAlpha en temps que plate-forme de prototypage rapide pour différents algorithmes (estimation de mouvement, filtre de Kalman) ;
- le portage de MMAlpha sur plate-forme WindowsNT et la distribution du logiciel MMAlpha ;
- l'étude algorithmique du problème du chemin algébrique.

Les premières expériences utilisant le traducteur Alfa vers VHDL ont prouvé deux choses importantes :

- d'une part, les outils de synthèse VLSI ont atteint un tel degré de complexité qu'il devient très difficile de rivaliser avec eux en imposant des stratégies particulières de synthèse. En pratique, cela veut dire qu'il est difficile d'obtenir des circuits meilleurs en utilisant les informations dérivées du système MMAlpha : régularité des calculs, structure de la hiérarchie des cellules, etc. Ces informations se révèlent utiles lorsque les circuits atteignent une taille importante (donc un coût important), or ces gros circuits doivent être optimisés à la main, ce qui diminue l'intérêt de l'utilisation de MMAlpha. Cette constatation nous a poussé à développer plus intensément la génération de code pour FPGA, puisque très peu de tels outils existent pour cette technologie ;
- d'autre part, les études menées sur l'interfaçage entre Alpha et d'autres langages ont fait apparaître clairement le caractère critique de l'interface. Par exemple, dans l'optique de compilation de code pour FPGA, le temps d'acheminement des données entre le processeur hôte et le FPGA est le goulot d'étranglement qui ralentit tout le calcul. Nous avons donc étudié plus spécifiquement ce problème d'interfaçage (problème de co-synthèse matériel-logiciel).

La mise en place d'une plate-forme de génération de code pour FPGA aurait pu être réalisée par l'intermédiaire du traducteurs Alfa vers VHDL déjà disponible, mais il s'est avéré que les synthétiseurs VHDL ne prennent pas en compte efficacement les spécificités des FPGA, d'où l'idée de développer le langage Spray et une plate-forme de synthèse spécifique pour FPGA (voir section 6.3).

Un traducteur Alfa vers Spray a été réalisé permettant un premier chemin entre Alpha et FPGA, en passant successivement par Spray (pour le placement régulier), JHDL (pour les outils de synthèse) puis EDIF (format standard à donner aux outils Xilinx). Ce traducteur a pu être réalisé grâce au package *Meta* qui permet de développer rapidement des traducteurs de Alfa vers d'autre formats.

Parallèlement à ce travail concernant la programmation du FPGA, nous avons cherché à réaliser automatiquement la programmation de l'interface entre le FPGA et le processeur hôte. L'équipe LSL de L'EPFL (Lausanne) nous a prêté la carte *Labomat3* qu'elle avait développée pour l'enseignement de la conception conjointe. Cette carte comprenant deux FPGA (un XC4013 et un XC6216), un processeur (Motorola 68356) et plusieurs types de mémoire, permet de prototyper différents types d'interface entre le processeur et les FPGA. Une étude est en cours pour déterminer quelle est la meilleure façon d'acheminer les données sur ce type de machine.

Suite aux études menées les années précédentes sur les $\mathcal{Z}$-polyèdres (intersection de polyèdres avec des treillis ou réseaux euclidiens), nous avons entrepris l'implémentation dans la librairie polyédrique des calculs sur les Z-polyèdres. Nous avons en parallèle proposé une extension du langage Alfa qui permettra de traiter de nouvelles applications. Il reste maintenant à adapter l'ensemble du système MMAlpha à la manipulation de cette nouvelle structure de données.

Un des obstacles à l'efficacité du code généré à partir d'Alpha était la grande quantité de mémoire utilisée à cause de la règle d'assignation unique. Une étude approfondie sur la réutilisation de mémoire a été menée. Nous avons proposé une méthode permettant de déterminer automatiquement, pour un ordonnancement donné, une allocation de mémoire optimale pour chaque variable. Ce travail a été présenté à la conférence Renpar99, et est mis en œuvre dans le compilateur Alpha disponible sous licence GPL.

La chaîne de synthèse a continué d'être expérimentée sur plusieurs exemples, dans le domaine du traitement du signal et de l'image (estimation de mouvement, filtrage de Kalman). Enfin, le développement de MMAlpha s'est poursuivi par, en particulier, le portage sur plate-forme Windows-NT qui permet à de nombreuses équipes (Inde, Afrique, Québec) de pouvoir l'utiliser.

6.2 Compilation pour processeurs spécialisés programmables

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

Participants : François Charot, Vincent Messé, 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.*

De plus en plus souvent, la conception de systèmes spécialisés fait appel à des cœurs de processeurs - DSP en particulier, ou Asip - qui sont optimisés et spécialisés pour tenir compte des contraintes très fortes d'utilisation du système et qui représentent un compromis entre efficacité - les ressources utilisées sont limitées - et flexibilité - la programmation permet des modifications de l'algorithme.

La principale difficulté liée à l'utilisation de processeurs programmables spécialisés réside dans la définition de leur architecture interne ; il est en effet difficile d'estimer a priori l'adéquation entre un processeur et un domaine d'application. Une solution consiste à compiler des portions significatives du code de l'application sur de nombreuses architectures candidates, puis à évaluer les résultats obtenus. La mise en œuvre de cette exploration nécessite des compilateurs flexibles, capables d'être adaptés rapidement à une grande variété de processeurs.

Les travaux de recherche conduits en 1999 ont porté sur les points suivants :

- le langage de modélisation de processeurs ;
- la mise en place d'une plate-forme de compilation flexible ;

– l'étude de passes de compilation spécifiques à l'architecture des processeurs cibles.

La modélisation de l'architecture du processeur cible est la base des techniques de compilation reciblable. En effet, chaque composante d'une chaîne de compilation a besoin d'informations sur l'architecture du processeur cible et l'écriture d'une description de l'architecture propre à chaque outil représente un travail considérable, les risques d'incohérence entre les différentes descriptions étant par ailleurs importants. Cette grande palette d'outils suggère l'emploi d'une description unique, à partir de laquelle sont extraites les informations utiles aux différents outils. 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, décrit dans la thèse de Vincent Messé [7].

La plate-forme de compilation flexible actuellement en développement est basée sur une bibliothèque de modules de production et d'optimisation de code. 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. La combinaison de ces deux niveaux de flexibilité autorise l'évaluation d'une large gamme de processeurs et s'inscrit dans une démarche d'exploration architecturale [13].

Nombre de mécanismes architecturaux présents dans les processeurs spécialisés justifient le besoin de nouvelles techniques d'optimisation de code. C'est en particulier le cas des unités de calcul d'adresse couramment employés dans les processeurs DSP et de leurs modes d'adressage postincrémentés pour lesquels des modules de bibliothèque ont été développés.

reconfigurables

6.3 Outils de CAO pour architectures reconfigurables

Mots clés : composant FPGA, architecture reconfigurable, partition de Goldbach.

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

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 Alfa nous automatisons le processus d'implantation sur une plate-forme reconfigurable.*

Par architecture régulière, nous entendons un tableau linéaire ou bidimensionnel de cellules élémentaires. Dans un premier temps, nous ne considérons que les réseaux linéaires. Cette année, nos efforts se sont concentrés sur deux points :

1. Définition d'un format intermédiaire pour décrire une architecture régulière au niveau transfert de registre. Ce format, appelé Spray, est adapté à la technologie FPGA dans la mesure où un placement de l'architecture peut être spécifié.
2. Définition d'une stratégie d'implémentation des architectures régulières sur plate-formes reconfigurables. Cette stratégie est mise en œuvre par l'outil Snake.

Spray (Specify and Place Regular fpgA arraY) est un format concis pour décrire une architecture régulière. Il est compréhensible et peut être facilement généré par des outils de synthèse tels qu'Alfa. L'objectif est que ce format serve de pivot pour divers outils de placement et qu'il soit indépendant des différentes technologies FPGA. Le format est défini et un «parser» a été développé.

L'outil Snake prend en entrée une description Spray non placée d'une architecture linéaire et produit une description Spray placée de cette même architecture. La 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. Cet outil est en cours d'élaboration.

6.4 Algorithmique fondamentale

Mots clés : algorithmique fondamentale, programmation dynamique, sac à dos non borné, tri, VLSI, ATM.

Participants : Patrice Quinton, Sanjay Rajopadhye.

Résumé : *En collaboration avec R. Andonov et V. Poirriez de l'université de Valenciennes, nous avons développé un algorithme pour le sac à dos non borné. Il utilise une généralisation et extension de la relation de dominance.*

Nous avons également, en collaboration avec M. Tchuenté et C. Tatonki des universités de Dschang et Yaoundé, développé un nouvel algorithme de tri qui se prête bien à une réalisation VLSI. *Cet algorithme a fait l'objet d'un dépôt de brevet.*

Nous avons développé un algorithme efficace de programmation dynamique pour le problème du sac à dos non borné. Il est basé sur la notion de *dominance de seuil* entre les solutions partielles des sous problèmes. La dominance de seuil est une généralisation stricte de toutes les dominances connues dans la littérature. Nous avons montré que, en combinaison avec la propriété de périodicité et une représentation creuse, la dominance conduit à une réduction importante de temps. Les résultats sont supportés par une expérimentation significative [8].

Le 15 janvier 1999, l'université de Rennes 1, pour P. Quinton, M. Tchuenté et C. Tatonki, a déposé une demande de brevet (n° 99 0540) sous le titre *Dispositif échéancier et dipositif de gestion correspondant.* Le dispositif breveté est un échéancier systolique permettant de trouver en temps réel le minimum d'un ensemble de valeurs tout en assurant l'insertion de nouvelles valeurs. Les opérations d'insetion et d'extraction se font en temps constant, et le nombre de processeurs de comparaison pour traiter n valeur est $\log n$. L'application naturelle de ce dispositif est l'ordonnancement de cellules ATM.

7 Contrats industriels (nationaux, européens et internationaux)

d'architectures de contrôle de trafic en ATM, (CTI Cnet 97 1B 546)

7.1 SPART, Méthodes de spécification pour la conception d'architectures de contrôle de trafic en ATM, (CTI Cnet 97 1B 546)

Participants : François Charot, Ferry Djieya, Patrice Quinton, 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.

La mise en place d'un réseau ATM offrant à un prix abordable des services à qualité garantie, requiert la mise en œuvre, à l'accès et aux nœuds du réseau, de mécanismes de contrôle de trafic et de gestion de ressources ; le contrôle de trafic a pour but de garantir la Qualité de Service (QoS) et la gestion de ressources d'optimiser l'utilisation de la bande passante.

La mise au point des mécanismes de contrôle de trafic et de gestion de ressources nécessite des expérimentations en situation réelle et le développement de démonstrateurs. De tels démonstrateurs peuvent être conçus entièrement à partir de matériels spécifiques pour en garantir les performances ou être réalisés en logiciel pour satisfaire la souplesse d'utilisation. Aucune de ces solutions n'est satisfaisante car ces algorithmes ont des contraintes temporelles extrêmement fortes et des spécifications qui évoluent en permanence au fur et à mesure de la mise au point des protocoles. La solution logicielle conduit à des mises en œuvre peu efficaces et les temps de conception d'une solution matérielle sont trop longs du fait des méthodes de conception classiques actuelles.

Une solution réside dans la réalisation de systèmes mélangeant plusieurs technologies de mise en œuvre : logiciel pour des processeurs programmables généraux ou des processeurs de traitement du signal, processeurs programmables spécialisés, circuits «câblés» spécialisés, etc.

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.

Il s'agit donc dans un premier temps de disposer d'une chaîne de simulation de niveau système, pour ensuite étudier, en utilisant des outils existants, les problèmes de modélisation au niveau système, de co-simulation logicielle/matérielle, les aspects de partitionnement matériel/logiciel et les liens avec les outils de synthèse de matériel et de logiciel et ainsi montrer la faisabilité de réalisation de démonstrateurs mélangeant plusieurs technologies de réalisation. Le travail a porté sur l'expérimentation des outils SPW, Ptolemy et Bones pour la spécification d'un contrôleur-espaceur.

C 021 31312 01 1)

7.2 Sysil : du système au silicium (ref. 2 99 C 021 31312 01 1)

Participants : François Charot, Charles Wagner, Vincent Messé.

Résumé : *Le projet Sysil implique la division* PPG *(Programmable Products Group) de* STM*icroelectronics, Grenoble et le projet Cosi de l'Inria Rennes. Sysil est un projet annuel, reconductible deux fois. Il vise à explorer les méthodes de conception d'un système embarqué complet.*

Ce projet se situe dans le contexte où STMicroelectronics cherche à définir une architecture parallèle pour ses générations futures de circuit vidéo et son. Parce que le temps de conception d'un circuit est long et qu'un circuit ASIC («Application Specific Integrated Circuit») n'est pas facilement modifiable, on souhaite que la définition du système soit logicielle (c'est-à-dire, la spécification reste sous forme de code exécutable) autant que possible. Pour réaliser cette étude, on se base concrètement sur un codec vidéo et son pour le DVD («digital video disc»).

Une première génération du codec est réalisée par STMicroelectronics basée sur une implementation multiprocesseur à base de DSP950 (un proceseur de traitement de signal de STMicroelectronics). L'Inria fait une étude en amont pour guider l'évolution architecturale de cette réalisation, et en général, d'un système de traitement de la vidéo et du son. Cette étude se focalise sur la partie de calcul intensif, et plus précisément sur le problème suivant :

Étant donné un certain nombre d'algorithmes (programmes) dominés par des boucles, et une surface de silicium fixée par une technologie, déterminer systématiquement la meilleure architecture «spécialisée» pour ces programmes.

Dans un premier temps, le projet Sysil a comme architecture cible des Processeurs programmables dont les caractéristiques architecturales sont à définir en fonction de l'application : des ASIP («Application Specific Instruction-set Processors»).

8 Actions régionales, nationales et internationales

8.1 Actions régionales

- Le projet Cosi collabore avec l'unité U435 de l'Inserm sur un projet de gestion de base de données de séquences d'ADN.
- Le projet Cosi entretient des relations avec l'Ubo, le Lip et participe au Club d'Architecture de l'Ouest (Enssat, ENST, Ireste, UBS, Ubo).
- Le projet Cosi collabore avec l'équipe Recombinaison Génétique (UPR41 du CNRS) sur le thème de la cartographie génétique à haute densité du chromosome de *Rhizobium meliloti*.
- Le projet Cosi participe activement au montage d'un pôle bio-informatique en collaboration avec divers laboratoires de biologie du CNRS (UPR 41), de l'Inserm et de l'Inra. Ce pôle s'articule autour d'une activité enseignement-recherche : (mise en place de modules bio-info en maîtrise de biologie et proposition d'un DEA Génome et Informatique dont D. Lavenier est responsable) et mise en place d'une activité «calcul intensif».

8.2 Actions nationales

- La projet Cosi collabore avec les projet A3 (Rocquencourt/Versaille), CRI (ENSMP), ICPS (Strasbourg), ReMap (Lyon), LIFL (Lille) sur la parallélisation automatique et les fondements du modèle polyédrique.

- La projet Cosi participe aussi au GDR CAO et Isis (thème AAA).
- Le projet Cosi collabore avec l'Institut Pasteur (encadrement de stagiaire).
- Le projet Cosi collabore 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

- Le projet Cosi collabore 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).

8.3.2 Pacifique et Asie du Sud

- Le projet Cosi collabore avec l'université de Nouvelle Angleterre, Australie sur le thème des architectures régulières à horloges multiples (réseaux polychrones).
- Le projet Cosi collabore avec l'Electronics Unit, Indian Statistical Institute, Calcutta, Inde (projet Corcop, B. Sinha et S. Sur-Kolay) sur la compilation et optimisation pour des FPGA (financé par un projet Cefipra).
- Une coopération Inria se met en place avec les instituts indiens de technologie (IIT). Dans ce cadre, S. Rajopadhye a organisé l'accueil de dix étudiants en stage d'été dans divers laboratoires français.

8.3.3 Afrique

- Dans le cadre du projet Cari, Cosi coopère avec l'université de Yaoundé et de Dschang (Cameroun) sur l'algorithmique systolique (Co-encadrement de la thèse de C. Tadonki).

8.3.4 Amérique du Nord

- Le projet Cosi collabore avec l'Electrical Engineering Department, Brigham Young University, USA (D. Wilde). Développement du logiciel MMAlpha, et extensions de la librairie Polylib pour manipuler des $\mathcal{Z}$-polyèdres (financé par un projet NSF-Inria).
- Le projet Cosi collabore avec le département de Génie Électrique, Université du Québec à Trois Rivières, Canada (D. Massicotte) sur l'utilisation du système MMAlpha pour la conception d'architectures pour les filtres de Kalman (financé par un projet Horizon Le Monde).

8.4 Accueils de chercheurs étrangers

- Susmita Sur-Kolay (Isi, Calcutta) a passé un mois à l'Irisa.
- Claude Tadonki (université de Yaoundé, Cameroun) a effectué un séjour de six mois à l'Irisa.

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.
- S. Rajopadhye est co-responsable du thème HiPerf du pôle Parallélisme du GdR ARP.
- P. Quinton et S. Rajopadhye sont membres du comité de lecture de *Integration : the* VLSI *Journal*.
- P. Quinton est éditeur de la revue *Parallel Processing Letters*.
- F. Charot fait partie du comité de lecture de la revue *Traitement du signal*.
- S. Rajopadhye est membre de l'«advisory board» de la conférence Europar.
- S. Rajopadhye a été co-organisateur du workshop "Compilation et parallélisation Automatique" (Domaine Saint-Jacques, octobre 1999).

9.2 Enseignement universitaire

- D. Lavenier est chargé de cours à l'université de Bretagne sud et enseigne un module de bio-informatique.
- 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 : Vincent Deshaye (DESS), Tristan Boyer, Rodolphe Beaugeard (Diic), Sunder Nookala, Dibyapran Sanyal (Univ. Pune, Inde), Mukul Kumar (IIT, Inde).
- S. Rajopadhye et T. Risset sont responsables d'une option du DEA d'informatique (Rennes) sur l'algorithmique parallèle (module Alpa).

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.
- Un cours sur les architectures spécialisées du type DSP est donné par F. Charot à l'ISMRA de Caen.

9.4 Participation à des colloques, séminaires, invitations

- P. Quinton a été membre du comité de programme de la conférence ISCIS'99, président du comité d'organisation de SPAA'99, et membre du comité d'organisation et de programme de Renpar 99.
- S. Rajopadhye a été co-organisateur d'un panel sur les Benchmarks (LPC'99).
- D. Lavenier a été membre du comité d'organisation et de programation du 5eme Symposium en Architecture de Machines (SympA'5 - Rennes, juin 1999)
- S. Rajopadhye et F. Quillere ont été invités au séminaire à Dagstuhl (Allemagne) sur le thème *Instruction-Level Parallelism and Parallelizing Compilation*

– S. Rajopadhye a été invité au séminaire à Dagstuhl (Allemagne) sur le thème *High Level Parallel Programming: Applicability, Analys and Performance*

10 Bibliographie

Ouvrages et articles de référence de l'équipe

[1] F. Charot, G. Le Fol, C. Wagner, « Approche de conception du processeur vidéo programmable movie », *Traitement du signal 14*, 6, 1997.

[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.

Thèses et habilitations à diriger des recherches

[7] V. Messé, *Production de compilateurs flexibles pour la conception de processeurs programmables spécialisés*, thèse, Université de Rennes 1, mars 1999.

Articles et chapitres de livre

[8] R. Andonov, V. Poirriez, S. Rajopadhye, « Efficient Dynamic Programming for the Unbounded Knapsack Problem », *European Journal of Operations Research*, à paraître.

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

[10] C. T. Djamegni, P. Quinton, S. Rajopadhye, T. Risset, « Derivation Of Systolic Algorithms For The Algebraic Path Problem By Recurrence Transformations », *Parallel Computing*, 1999, à paraître.

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

[12] P. Quinton, « Architectures spécialisées », *TSI*, 1999, à paraître.

Communications à des congrès, colloques, etc.

[13] F. Charot, V. Messé, « A Flexible Code Generation Framework for the Design of Application Specific Programmable Processor », *in : 7th international workshop on Hardware/Software Codesign*, p. 27–31, Rome, mai 1999.

[14] D. Lavenier, « Calcul, Architectures et Circuits Reconfigurables », *in : Colloque CAO de circuits intégrés et systèmes*, Aix en Provence (Fuveau), mai 1999.

[15] V. Messé, F. Charot, « Définition interactive d'Asip », *in : 5ème Symposium en architectures nouvelles de machines*, Rennes, juin 1999.

[16] A. Mozipo, D. Massicote, P. Quinton, T. Risset, « A Parallel Architecture for Adaptative Channel Equalization Based On Kalman Filter Using MMAlpha », *in : 1999 IEEE Canadian Conference on Electrical & Computer Engineering*, Calgary, Canada, mai 1999.

[17] A. Mozipo, D. Massicote, P. Quinton, T. Risset, « Automatic Synthesis of a Parallel Architecture for Kalman Filtering using MMAlpha », *in : 1999 IEEE Canadian Conference on Electrical & Computer Engineering*, Edmonton, Canada, mai 1999.

[18] S. Rajopadhye, T. Risset, C. Tadonki, « The Algebraic Path Problem Revisited », *in : Fifth International Euro-Par Conference*, p. 698–707, Toulouse, France, August/September 1999.

Rapports de recherche et publications internes

[19] F. Dupont de Dinechin, P. Quinton, S. Rajopadhye, T. Risset, « First Steps in Alpha », *PI n° 1244*, Irisa, 1999.

[20] F. Bardoult and P. Quinton and S. Rajopadhye and Tanguy Risset, « Synthesis of data-flow interfaces for regular parallel programs », *PI n° 1260*, Irisa, septembre 1999.

[21] Étienne Mémin, T. Risset, « Hardware driven considerations for energy based applications », *PI n° 1220*, Irisa, 1999.