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]

Personnel Upresa 6074

Dominique Lavenier [CR CNRS]

Laurent Perraudeau [maître de conférences, université de Rennes 1]

Patrice Quinton [professeur, université de Rennes 1]

Charles Wagner [IR CNRS (Atelier)]

Chercheurs doctorants

Steven Derrien [bourse MENESR]

Ferry Djieya [bourse Inria/CIES]

Erwan Fabiani [bourse MENESR]

Anne-Claire Guilloux [bourse Inria]

Katel Morin-Allory [bourse MENESR à partir du 1er octobre]

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.

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

[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.

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 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

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

[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.

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.

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

[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.

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 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.

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

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.*

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

[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.

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.

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 :
Les réalisations au niveau logiciel du projet Cosi sont matérialisées par MMA*lpha pour la synthèse de haut niveau.*

5.2 MMAlpha

Participants : Fabien Quilleré, Patrice Quinton, Sanjay Rajopadhye, Tanguy Risset [correspondant].

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 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 s'appuie sur une bibliothèque écrite en C.

Le noyau de ce logiciel est la bibliothèque polyédrique développée par Hervé Le Verge et Doran Wilde [Wil93]. La bibliothèque 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 bibliothèque. Cette bibliothèque 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 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.

[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.

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 : 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 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 (Departement de génie éléctrique de l'université du Quebec 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, Patrice Quinton, Fabien Quilleré, Sanjay Rajopadhye, Tanguy Risset.

Résumé : *Notre effort est concentré sur le développement du logiciel* MMA*lpha (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 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 2000

Nos recherches en 2000 ont porté sur les aspects suivants :

- 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'étude du pavage oblique optimal d'espaces d'itération à deux dimensions ;

- l'utilisation de MMAlpha en temps 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 ;
- la compilation d'Alpha sur une machine à usage général en optimisant la taille mémoire utilisée, et la génération du code impératif associé ;
- l'intégration de méthodes formelles pour la démonstration de propriétés de systèmes Alpha ;
- l'étude algorithmique du problème du chemin algébrique.

Résultats

- Le problème de la génération d'interfaces passe par la définition d'une architecture matérielle générique, afin de pouvoir ensuite automatiser la production d'interfaces [24].
- 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 [17, 16], et donne lieu à une validation expérimentale sur carte SPYDER-X2. D'autre part, l'utilisation de techniques de *retiming* permet d'aller vers un parallélisme à grain fin des processeurs produits par MMAlpha [34].
- L'étude du pavage oblique a pour but de minimiser le temps d'exécution total d'une machine parallèle à mémoire distribuée. Si les dépendances sont uniformes, et si l'on se restreint à des tuiles dont l'un des bords est parallèle à un bord du domaine considéré, il est possible de formuler une solution analytique par résolution d'un problème d'optimisation discrète non linéaire. Sur certaines classes de problèmes (comme par exemple l'alignement de séquences), le pavage oblique optimal donne une accélération quasi-linéaire sur IBM-SP2, ce qui semble être un nouveau résultat [27].
- 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. Les exemples traités ont été un filtre adaptatif (DLMS) et un algorithme de rétropropagation dynamique (avec application aux réseaux de neurones) [20, 19]. Ce travail a été réalisé en collaboration avec l'Université du Québec à Trois-Rivières.
- 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 utilisation de l'algèbre (*max, plus*). 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.

- La compilation de systèmes Alpha sous forme de nids de boucles impératifs avec optimisation de la mémoire utilisée a fait l'objet d'une implémentation dans le module CodeGen, qui génère du code C. Ce module a été mis sous licence GNU et intégré dans la suite MMAlpha.

- 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 [28].

- La visite de B. Sinha (ISI) a été l'occasion d'étudier l'expression de programmes non affines en Alpha (tris, FFT, etc.).

Perspectives

Les travaux futurs à court terme concernent les points suivants :

- l'expression au niveau matériel de systèmes avec ordonnancement multidimensionnel ;

- la génération de VHDL et de code impératif pour des programmes Alpha structurés ;

- la poursuite de la définition et de l'implémentation de méthodes de preuve formelle pour des programmes Alpha structurés ;

- l'implémentation dans MMAlpha des $\mathcal{Z}$-polyèdres ;

- l'intégration des outils vers les FPGA : interfaçage et optimisation globale (combinaison des optimisations liées au pavage et au *retiming*) ;

- l'étude de l'interaction entre estimation du routage et retiming (dans le cadre des FPGA).

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, 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.

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 soutenue en 1999.

Notre effort est concentré sur le développement de l'environnement Calife/Armor pour la génération de code flexible pour Asip. Les travaux de recherche en 2000 ont porté sur les aspects suivants :

- l'étude de l'extension du langage Armor pour la prise en compte de caractéristiques architecturales particulières ;
- la modélisation en Armor de processeurs de traitement de signal (OakDSPCore, Sapphire), d'un processeur de traitement de cellules ATM (ATMizerII+), d'un processeur de style RISC (RM7000) ;
- l'étude d'une passerelle entre le langage Armor et le langage VHDL ;
- le développement de modules de production de code ;
- l'étude de l'estimation de performance logicielle d'un code optimisé s'appuyant sur l'environnement Armor/Calife.

L'environnement Calife/Armor actuellement en développement est basé 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 ([9]).

S'agissant d'exploration architecturale où le problème est tout d'abord de faire le choix du processeur cible parmi l'ensemble des processeurs candidats, l'approche étudiée repose sur l'utilisation de modules d'estimation de performance d'un code optimisé. De tels modules fonctionnent de façon similaire à des modules permettant de concevoir des flots de compilation, à la différence près qu'ils ne contribueront pas directement à la génération de code – ils sont paramétrés par une modélisation du processeur cible dans le langage Armor. L'estimation va permettre de connaître assez précisément la performance de l'application sur le processeur cible. Le résultat de l'estimation de performance va ensuite pouvoir être utilisé pour aider à la définition du flot de compilation pour le processeur cible.

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.

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.*

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 deux 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.

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.

L'interfacage 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.

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 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 le combinant avec la propriété de périodicité et une représentation creuse, la dominance conduit à une réduction importante de temps. Les résultats sont démontrés par une expérimentation significative [8].

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 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.

Les travaux réalisés durant cette année concernent, d'une part, la modélisation Armor de processeurs, et d'autre part, la réalisation de passes d'estimation de performance dans l'environnement Calife/Armor. L'utilisation de Calife comme estimateur de performance dans un processus de conception de haut niveau requiert un pont entre les spécifications de haut niveau décrites à l'aide de Ptolemy et les estimations produites par Calife. Une passerelle a été réalisée entre Ptolemy et Calife. Elle permet, d'une part d'injecter les informations dynamiques collectées lors des simulations fonctionnelles avec Ptolemy dans le flot d'estimation, et d'autres part de récupérer les estimations produites pour annoter les primitives Ptolemy.

7.2 Asia, (Accelerated Signalling of Internet over ATM), 1 98 C 531

Participants : François Charot, Ferry Djieya, Charles Wagner.

Résumé : *Il s'agit d'un projet RNRT précompétitif, d'une durée de deux ans (01/01/1999 - 31/12/2000). Nos partenaires dans le projet sont le Cnet Lannion (chef de file), Airtria (PME de Lannion) et MET (Matra Ericsson Telecom).*

Le but de ce projet est la conception et la réalisation d'un réseau Internet haute qualité sur ATM. Pour faire face à l'explosion du trafic engendré par l'Internet ainsi qu'à l'émergence sur ce réseau de nouvelles applications multimédia exigeant une haute qualité de service, il devient primordial pour les opérateurs de télécommunications de concevoir un réseau capable non seulement d'écouler le trafic mais aussi d'améliorer sensiblement la qualité de transfert dans

l'Internet actuel. Granularité fine des débits, puissance de commutation, flexibilité d'allocation de ressources multidébits et capacité à garantir des objectifs de qualité de service sont les caractéristiques naturelles du réseau ATM, qui en font un réseau fédérateur apte à relever ce défi. Cependant, on constate que l'Internet n'exploite ces potentialités que de manière très limitée. C'est pour améliorer la synergie entre ATM et Internet que le projet Asia se propose, via des concepts de signalisation allégée, de réaliser un couplage efficace entre l'ATM et les protocoles de l'Internet, sans aucune modification de ces derniers. La preuve de ces concepts sera effectuée sur un matériel industriel enrichi de capacités de signalisation allégée et leurs performances seront analysées mathématiquement.

Il s'agit pour nous d'étudier l'évolution architecturale des organes de traitement de signalisation utilisés dans ce projet. Notre travail porte principalement sur la réalisation d'un état de l'art en matière de processeurs de réseaux dans le but de poser les bases de ce que pourrait être une architecture pour le traitement de signalisation.

7.3 CORCoP: Compilation and Optimization for Reconfigurable Co-processors

Participants : Sanjay Rajopadhye, Tanguy Risset, Steven Derrien.

Résumé : *Le projet CORCoP implique le projet Cosi et le Indian Statistical Institute, Calcutta, Inde. Ce projet de trois ans a débuté en février 1999. Il vise à explorer les méthodes de conception d'un système embarqué complet.*

Une large classe d'applications dans le domaine du traitement de l'image ou du signal, du multimédia, du calcul scientifique, nécessite une implémentation matérielle dédiée et parallèle. Même si les microprocesseurs à usage général s'améliorent constamment, certaines applications ne peuvent s'en satisfaire (parce que la demande en calcul est importante et croît plus vite que les performances des processeurs). Des co-processeurs à base de FPGA constituent un support intéressant pour de telles applications. Cependant, concevoir un matériel dédié à une application présente des difficultés :

- les applications cibles doivent être soigneusement choisies ;
- une solution complète et intégrée doit être conçue en incluant une interface adéquate entre le matériel et le logiciel ;
- la conception de matériel est une tâche longue et coûteuse qui doit être justifiée par un marché conséquent ;
- le matériel est figé et définitif. Il demande également des connaissances spécialisées et ne peut être conçu par un concepteur d'application typique.

Les co-processeurs reconfigurables se sont déjà distingués par certains succès. La machine Splash a été initialement conçue pour la comparaison de séquences génomiques et, plus tard, s'est révélée intéressante pour bien d'autres applications. Au laboratoire Parisien de DEC,

l'équipe de J. Vuillemin a développé les co-processeurs PeRLe-0 et PeRLe-1, qui possèdent des performances impressionnantes (aussi bien en temps de calcul qu'en coût), pour des applications telles que la cryptographie, la compression de données, la simulation d'équations calorifiques ou de réseaux de neurones, le traitement de l'image, etc.

Néanmoins, ces succès sont relatifs : ils n'ont pas été largement exploités pour d'autres domaines. En outre, ils ont été réalisés au prix d'une optimisation fine des algorithmes. Les problèmes à résoudre sont donc les suivants :

- dériver des mises en œuvre sur FPGA directement à partir de programmes de haut niveau ;
- s'assurer de la qualité des solutions à travers des outils et des méthodes d'optimisation ;
- permettre que ces outils puissent être réutilisés pour les prochaines générations de composants FPGA.

Tous ces problèmes sont à l'ordre du jour, mais aucune solution complète n'est encore disponible (par exemple, la page WEB de Xilinx mentionne explicitement la compilation parmi les problèmes ouverts qui frayeront la voie à une reconnaissance du calcul reconfigurable).

Les objectifs du projet CORCoP sont de relever les défis mentionnés précédemment en développant des méthodes de compilation et en associant 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 – qui peuvent permettre une séparation claire des intérêts.

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*.

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 aux GdR CAO, Isis (thème AAA) et ARP (thème HiPerf).

– 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).

– Le projet 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 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 bibliothèque 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 la compilation d'algorithmes de filtrage adaptatif (micro-action financée par le groupement de recherche HiPerf).

– 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 (mise à disposition pendant une année de D. Lavenier au LANL).

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.
- D. Lavenier a été membre du comité de programme de SympA (Symposium en Architectures de Machines).
- D. Lavenier a été membre du comité de programme de FPL (International Conference on Field Programmable Gate Array and Applications).
- D. Lavenier a été membre du du comité de programme de ERSA (International Conference on Engineering of Reconfigurable Systems and Algorithm).
- D. Lavenier a été éditeur d'un numero special de la revue TSI (Technique et Science informatiques, No 18, Vol 18).
- F. Charot et D. Lavenier sont membres du comité de lecture de la revue Traitement du Signal.
- P. Quinton et S. Rajopadhye sont membres du comité de lecture de *Integration : the* VLSI *Journal*.
- 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, Univ. Rennes 1, et d'un cours de ce DEA.
- D. Lavenier donne un cours d'informatique appliquée a 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 : Jean-Luc Binet, Aymeric Leperoux, Franck Morel, Fabrice Pokam (Diic), Mikaël Le Bozec (Enssat), Gautam Gupta (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'00 et membre du comité exécutif de DATE'2001.

– D. Lavenier a été invité au séminaire de Dagstuhl (Allemagne), Dynamically Reconfigurable Architectures, juin 2000.

– D. Lavenier a été invité au 10e Forum ORAP / 28th SPEEDUP Workshop (Genève), octobre 2000, - exposé invité intitulé : Reconfigurable Computing.

– F. Charot a été membre du conseil scientifique et du comité d'organisation de l'école thématique *Conception d'architectures de systèmes informatiques dédiés à des applications scientifiques de type enfoui* (Seix), novembre 2000 (http://www.ens-lyon.fr/LIP/Ecoles_Archi/).

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] T. Risset, *Contribution à la compilation de nids de boucles sur silicium*, Habilitation à diriger des recherches, Université de Rennes 1, novembre 2000.

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 123*, 2, juin 2000, p. 394–407.

[9] F. Charot, V. Messé, «La compilation reciblable au service de la définition interactive d'ASIP», *Technique et Science Informatiques*, 2000, à paraître.

[10] C. Djamegni, P. Quinton, R. Rajopadhye, T. Risset, «Derivation of Systolic Algorithms for the Algebraic Path Problem by Recurrence Transformations», *Parallel Computing*, 2000, à paraître.

[11] D. Lavenier, «An FPGA Systolic Array Using Pseudo Random Bit Generator for Computing Goldbach Partitions», *Integration, The VLSI Journal 30*, 1, 2000.

[12] E. Mémin, T. Risset, « On the Study of VLSI Derivation for Optical Flow Estimation », *International Journal of pattern recognition and Artificial Intelligence (IJPRAI) 14*, 4, juin 2000, p. 441–462, `http://www.irisa.fr/cosi/HOMEPAGE/Risset/papers/alpha/ijprai.ps`.

[13] F. Quilleré, S. Rajopadhye, D. Wilde, «Generation of Efficient Nested Loops from Polyhedra», *International Journal of Parallel Programming 28*, 5, octobre 2000.

[14] F. Quilleré, S. Rajopadhye, «Optimizing Memory Usage in the Polyhedral Model», *ACM Transactions on Programming Languages and Systems*, 2000, à paraître.

Communications à des congrès, colloques, etc.

[15] R. Andonov, P.-Y. Calland, S. Rajopadhye, N. Yanev, «First Steps Towards Optimal Oblique Tile Sizing», *in : CPS 2000: Compilers for Parallel Computers*, A. Darte, Y. Robert (éditeurs), p. 353–368, Aussois, France, janvier 2000.

[16] S. Derrien, S. Rajopadhye, « FCCMs and the Memory Wall », *in : IEEE Symposium on FPGAs for Custom Computing Machines*, avril 2000.

[17] S. Derrien, S. Sur Kolay, R. S., «Optimal Partitioning for FPGA based Arrays Implementation», *in : IEEE Conference on Parallel Computing in Electrical Engineering*, p. 155–159, août 2000.

[18] E. Fabiani, D. Lavenier, «Placement of Linear Arrays», *in : FPL 2000, 10th International Conference on Fields Programmable Logic and Applications*, Villach, Austria, août 2000.

[19] A.-C. Guillou, P. Quinton, T. Risset, D. Masicotte, «Automatic Design of VLSI Pipelined LMS Architectures», *in : IEEE Conference on Parallel Computing in Electrical Engineering*, p. 144–149, août 2000.

[20] A.-C. Guillou, P. Quinton, T. Risset, « De la spécification au silicium avec MMAlpha : application aux filtres adaptatifs LMS », *in : IIIe Journées Nationnales du Réseau Doctoral de Microélectronique*, Montellier, France, mai 2000.

[21] D. Lavenier, Y. Solihin, K. Cameron, « Reconfigurable Arithmetic and Logic Unit », *in : SYMPA'6, 6eme Symposium en Architecture de Machines*, Besancon, France, juin 2000.

[22] D. Lavenier, J. Theiler, J. Szymanski, M. Gokhale, J. Frigo, « FPGA Implementation of the Pixel Purity Index Algorithm for Hyper-Spectral images », *in : SPIE Photonics East, Workshop on Reconfigurable Architectures*, Boston, MA, USA, novembre 2000.

[23] P. Quinton, C. Tadonki, M. Tchuente, « Un échéancier systolique et son utilisation dans l'ATM», *in : CARI'2000*, octobre 2000.

[24] T. Risset, D. Derrien, «Interfacing Compiled FPGA Programs: the MMAlpha Approach», *in : Second International Workshop on Engineering of Reconfigurable Hardware/Software Objects, Int. conf. on Parallel and Distributed Processing Techniques and Applications (PDPTA)*, A. Arabnia (éditeur), CSREA Press, juin 2000, `http://www.irisa.fr/bibli/publi/pi/2000/1331/1331.html`.

[25] J. Szymanski *et al.*, « Advanced processing for high-bandwith sensor systems », *in : SPIE International Conference on Optical Science and Technology*, San Diego, CA, USA, août 2000.

[26] J. Theiler, D. Lavenier, N. Harvey, S. Perkins, J. Szymanski, «Using blocks of skewers for faster computation of pixel purity index », *in : SPIE International Conference on Optical Science and Technology*, San Diego, CA, USA, août 2000.

Rapports de recherche et publications internes

[27] R. Andonov, S. Balev, S. Rajopadhye, N. Yanev, « Optimal Semi-oblique Tiling», *rapport de recherche*, Irisa, Campus de Beaulieu, Rennes, novembre 2000.

[28] D. Cachera, P. Quinton, S. Rajopadhye, T. Risset, «Proving Properties of Multidimensional Recurrences with Application to Regular Parallel Algorithms», *rapport de recherche n° 1362*, Irisa, Campus de Beaulieu, Rennes, novembre 2000, `http://www.irisa.fr/bibli/publi/pi/2000/1362/1362.html`.

[29] F. Charot, F. Djieya, V. Messé, C. Wagner, «Retargetable Compilation in the service of interactive ASIP design», *rapport de recherche n° 1373*, IRISA, Rennes, France, décembre 2000.

[30] E. Fabiani, D. Lavenier, «Using knapsack technique to place linear arrays on FPGA», *rapport de recherche n° 1335*, IRISA, juillet 2000.

[31] D. Lavenier, «FPGA Implementation of the K-means Clustering Algorithm for Hyper-Spectral Images», *rapport de recherche n° LA-UR 00-3079*, Los Alamos National Laboratory, juillet 2000.

[32] M. Manjunathaiah, S. Rajopadhye, T. Risset, « Uniformization Tool for Systolic Array Designs», *rapport de recherche n° 1350*, Irisa, 2000.

[33] S. Nookala, T. Risset, «A Library for Z-polyhedral Operations», *rapport de recherche n° 1330*, Irisa, 2000.

[34] S. Sur Kolay, S. Rajopadhye, S. Derrien, « Fine grain parallelism for regular arrays on FPGAs», *rapport de recherche*, Irisa, Campus de Beaulieu, Rennes, novembre 2000.

Divers

[35] F. Charot, F. Djieya, C. Wagner, «Rapport du lot 2 : spécification d'un contrôleur-espaceur atm dans l'environnement Ptolemy», Convention Cnet 97 1B 546. Méthodes de spécification pour la conception d'architectures de contrôle de trafic en atm, septembre 1999.

[36] F. Charot, V. Messé, S. Rajopadhye, «Sysil : du système au silicium», Rapport Convention Serics 98.2.93.0331, octobre 1999.

[37] P. Quinton, T. Risset, «MMAlpha. A Tool Box for Silicon Compilation», Demonstation at the University Booth, DATE'2000, Paris, mars 2000.