Exploration de l'espace de conception de SOC, de l'asservissement à la coopération

Jean-Philippe Diguet

Habilitation à Diriger des Recherches

Université De Bretagne Sud

par

Jean-Philippe Diguet

Exploration de l'espace de conception de SOC, de l'asservissement à la coopération

soutenue le 20 septembre 2005 devant la commission d'examen composée de :

M. :	D. Lavenier	Directeur de recherche CNRS, IRISA, Rennes	Rapporteur
M. :	M. Paindavoine	Professeur, Université de Bourgogne, Dijon	Rapporteur
M. :	O. Sentieys	Professeur, ENSSAT, Université de Rennes I, Lannion	Rapporteur
M. :	K. Bertels	Professeur, Delft University of Technology, Netherlands	Rapporteur
M. :	M. Auguin	Directeur de recherche CNRS, I3S, Sophia-Antipolis	Examinateur
M. :	E. Martin	Professeur, Président de l'Université de Bretagne Sud, Lorient	Examinateur
M. :	J-L. Philippe	Professeur, Université de Bretagne Sud, Lorient	Examinateur

Laboratoire d'Électronique des Systèmes TEmps Réel

Remerciements

Cette Habilitation à diriger les recherches marque en quelque sorte un rite de passage dans le métier de la recherche, en passant ce seuil symbolique j'ai une pensée pour l'ensemble des personnes qui m'ont permis de l'exercer.

Avant tout chose, je souhaite témoigner ma gratitude à l'ensemble du jury pour le temps qu'ils acceptent de m'accorder en jugeant ce travail. Je transmets mes sincères remerciements à Dominique Lavenier, Olivier Sentieys, Koen Bertels et Michel Paindavoine qui me font la faveur d'être rapporteurs malgré la période estivale qui s'annonce.

Ces années de recherche n'auraient pu aboutir a quoi que soit sans l'aide des doctorants, postdocs, DEA que j'ai eu la chance d'encadrer dans la bonne humeur, que Yannick Le Moullec, Sebastien Bilavarn, Lilian Bossuet, Sven Wuytack, Samuel Évain, Abedenour Azzedine, Samuel Rouxel, Belgacem Bouallegue, Nader Ben Amor, Hedi Tmar et Yvan Eustache aient l'assurance de ma sympathie et de ma reconnaissance.

Je souhaite profiter également de cette occasion pour saluer ceux qui m'ont accompagné tout au long de ces années de recherche. Je pense tout d'abord à Jean-Luc Philippe avec qui j'ai eu le plaisir de travailler de manière quasi ininterrompue de puis la thèse, je conserverai de cette collaboration l'idée selon laquelle on peut être compétent et efficace sans perdre le sens de la décontraction et du juste détachement. Je tiens également à transmettre un salut amical à Olivier Sentieys avec qui, depuis mon séjour à Lannion je pense conserver une certaine idée de la recherche où l'humour à sa place. Le projet Design Trotter n'aurait pas fonctionné sans la connivence établie avec Guy Gogniat que je remercie entre autres pour son enthousiasme. Je ne peux omettre monsieur le Président Eric Martin bien sûr de qui j'ai appris beaucoup et notamment l'assurance que la ténacité finit par être récompensée. J'aurai également une pensée pour F.Catthoor dont la rigueur scientifique restera pour moi un modèle. Enfin, je souhaite transmettre à Michel Auguin toute ma reconnaissance, car sans son aide et sa compréhension je ne serais sans doute pas devenu chercheur au CNRS et n'aurais donc pas la chance de rédiger ce document.

Pour finir je souhaite témoigner aux aventuriers de dixip mon amitié et mon soutien.

Résumé

Le domaine de l'électronique embarquée est une dimension essentielle des technologies de l'information et de la communication. Le terme systèmes enfouis désigne son intégration sous forme de composants d'un système plus complexes issus des domaines de l'avionique, de l'automobile, des objets mobiles communiquants, du multimédia etc. Leur réalisation sous la forme de systèmes sur silicium (SOC) souligne la complexité et l'hétérogénéité qui les caractérisent désormais. La maîtrise de la conception des SOC représente un enjeu économique majeur à la hauteur de la place qu'ils occupent dans tous les secteurs d'activités (industriel, loisirs, domestique). Les outils et méthodes pour la conception de SOC constituent un domaine de recherche multi-formes dont le but global est de concevoir rapidement des systèmes qui soient fiables, performants et efficace d'un point de vue énergétique.

Ce document est une synthèse de mes recherches effectuées dans le domaine général des outils et méthodes de conception de SOC. Plus précisément, les travaux détaillés ici traitent des différents aspects d'un domaine unique à savoir l'exploration de l'espace de conception des SOC éventuellement reconfigurables. Ces travaux de recherche s'articulent principalement autour de six projets menés depuis la thèse de doctorat. Il s'agit de l'exploration de la hiérarchie mémoire, du projet Design Trotter pour l'exploration des solutions architecturales de la spécification algorithmique jusqu'au niveau tâche au sens temps réel. Dans le domaine de la gestion des entrées/sortie les projets présentés traitent d'un exemple d'interface réseau / flux multimédia et d'un environnement μSpider de synthèse et de dimensionnement de Network On Chip. Enfin, le document présente le projet en cours dans le domaine des architectures auto-reconfigurables.

Table des matières

Chapitre 1

Introduction

1.1 Avant Propos

La place des systèmes informatiques dans la vie quotidienne a considérablement évolué depuis les années cinquante, ces changements se sont traduits par des bouleversements aux niveaux sociétal et économique, la figure 1.1 -A en propose une synthèse. L'informatique est née de la nécessité de soulager les scientifiques de fastidieux calculs (trajectographie, chiffrement). Elle est restée longtemps l'apanage d'un petit nombre de spécialistes qui, pour bénéficier de la puissance de traitement, devaient s'astreindre à user de procédures complexes. À cette époque l'individu ou plutôt le pionnier doit s'adapter au langage restreint que lui permet la machine qu'il a lui même conçu. De ce point de vue l'usage à présent courant du terme *cyber*, qui provient du mot grec "gouverner", souligne avec acuité l'ambiguïté qui perdure sur l'identité des véritables maîtres du jeu.

La deuxième époque a débuté avec l'arrivée d'interfaces de communication et de moyens de programmation plus évolués. Le cercle des personnes concernées par le dialogue avec la machine s'est élargi. Ces derniers demeuraient cependant des experts suivis peu à peu par une communauté encore restreinte d'initiés passionnés. De fait l'impact économique restait limité. À cette époque l'électronique embarquée fait ses premiers pas sous la forme d'automates capables de contrôler de manière précise et rapide des tâches répétitives en milieu industriel.

Peu à peu le dialogue entre la machine et l'homme s'équilibre pour finalement produire un nouvel espace de communication en s'ouvrant sur les réseaux. Le véritable bouleversement survient lorsque l'évolution des interfaces (IHM conviviale, réseau haut débit) permet enfin à l'informatique de faire son entrée dans l'espace domestique. L'accès à des moyens de communication rapides et l'échange d'informations sous la forme de copies numériques transforment peu à peu l'organisation des sociétés. Ce mouvement est toujours en progression. La place de ce qu'on appelle de manière un peut vague les nouvelles technologies dans l'économie devient prépondérante et impose des changements dans la manière de concevoir les machines. Dans le même temps, les progrès d'intégration établissent les conditions de l'essor des systèmes embarqués. Ceux-ci deviennent enfouis et s'insinuent de manière définitive dans des secteurs de masse tels que l'électro-ménager ou l'automobile. Cette intrusion ouvre des perspectives considérables et les méthodes de fabrication commencent à évoluer.

La suite de l'histoire et des progrès va dans le sens de l'équilibrage du rapport de forces. La mobilité, maître mot de la troisième phase, délivre l'usager de l'asservissement à son pupitre. En étendant le temps et les possibilités de connexion, l'usager s'habitue à la présence permanente des espèces électroniques dans son environnement. Les systèmes embarqués s'enfouissent de plus en plus au point de devenir des composants indispensables de leurs hôtes (1/3 du prix d'une automobile en 2006). La dépendance vis à vis des systèmes embarqués croît de manière quasi irréversible (660M de téléphones mobiles [1] ont été vendus en 2004 dont 80M avec appareil

photo intégré). Cette fois, les contraintes économiques aux niveaux des temps de conception et de prix du produit final imposent de nouvelles méthodes de conception proches de celles mises au point dans d'autres secteurs : automatisation, réutilisation, innovation. Le modèle économique a radicalement basculé au point que la contrainte du prix d'achat s'estompe avec l'effondrement de celui des processeurs qui présage d'un PC a 100$ [2]. On observe par ailleurs que cette époque est marquée par un événement qui en souligne la maturité : les performances offertes par les systèmes dépassent peu à peu, suivant les domaines, les exigences des utilisateurs. Au terme de cette quatrième phase les verrous principaux, outre le temps de conception, sont la dissipation et le stockage de l'énergie.

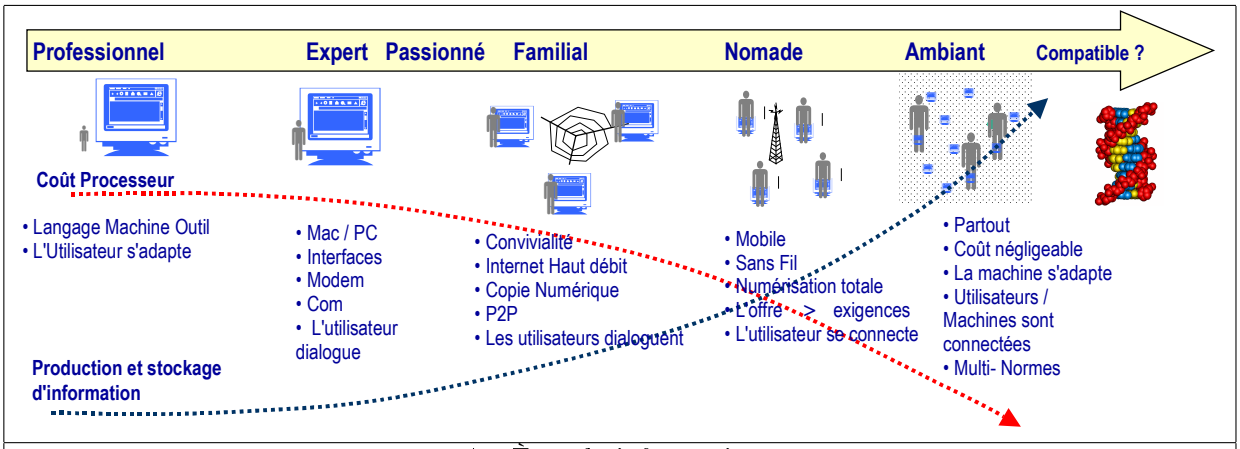
Nous sommes à présent au début de la cinquième ère, celle du système enfouis devenu banal. De leurs côté les machines gagnent un degré d'autonomie supplémentaire en accédant au contrôle de leurs communications propres. Une deuxième révolution est en marche, celle de l'électronique naturelle où l'individu est tout simplement connecté de manière inconsciente et où l'électronique se fond dans son environnement en s'adaptant aux requêtes que le hasard porte à sa connaissance. En supposant la question de l'énergie ramenée à un niveau raisonnable, le véritable frein à l'expansion de cette quatrième phase sera la portabilité et la sécurité à assurer dans un environnement hétérogène distribué.

L'évolution des outils et méthodes de conception a suivi naturellement celle de l'informatique et des rapports entre l'homme et la machine. La figure 1.1-B est une tentative de synthèse pour résumer l'évolution comparable des méthodes de conception de circuits puis de SOC en considérant quatre acteurs. Il s'agit de la machine (ou méthode ou CAO), du concepteur, du circuit (ou SOC) produit et utilisé pour produire à son tour et enfin l'usager du circuit. Les flèches indiquent l'ordre dominant des contraintes, les positions horizontales relatives traduisent le degré de servitude dans le temps.

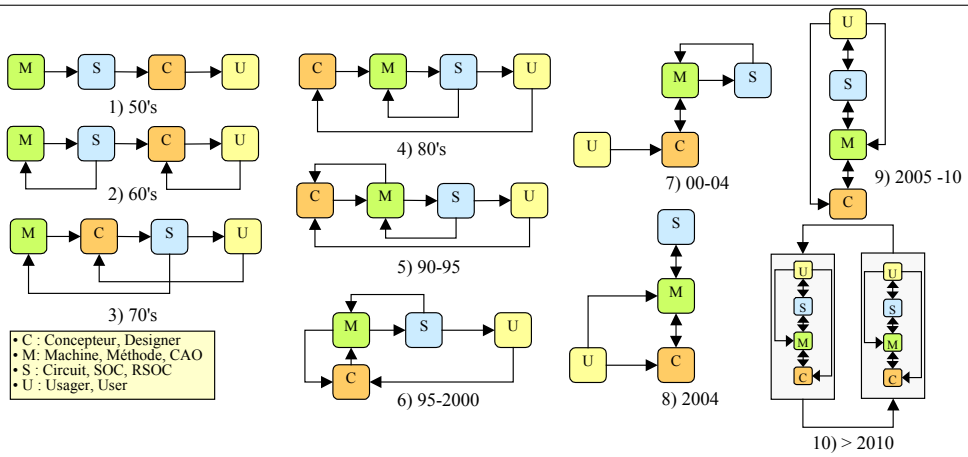
1. La méthode prime et produit un circuit que le concepteur tente d'utiliser pour un usage limité.
2. Les machines gagnent en puissance et en flexibilité grâce au processeur produit, le concepteur adapte son savoir faire aux besoins exprimés par l'usager potentiel.
3. Le concepteur prend simultanément le pouvoir et conscience du potentiel qui s'offre à lui, il contrôle mieux le choix du circuit produit.
4. Le concepteur conçoit l'outil de CAO pour produire automatiquement le circuit. C'est l'ère des simplifications logiques.
5. Des librairies issues de la synthèse logique sont disponibles, le niveau d'abstraction de l'outil réalisé par le concepteur s'élève, la synthèse d'architecture (ou comportementale puis fonctionnelle) affiche l'ambition, par une approche "presse bouton", de produire automatiquement un processeur dédié.
6. La complexité des circuits a cru et les méthodes automatiques ne peuvent suivre le rythme, les approches coopératives concepteur/outil (C/M) sont privilégiées. Le codesign logiciel/matériel élève encore le niveau et débute en réitérant l'erreur de l'automatisation complète avant d'être à son tour corrigé. La consommation devient le critère dominant.
7. Les SOC dédiés aux classes d'applications se généralisent, l'usage et donc l'économie priment. L'approche par exploration et affinements successifs s'impose. Les architectures reconfigurables s'ouvrent aux SOC. L'incertitude, sur les délais de transmission des données, devient une contrainte à considérer.
8. Les architectures reconfigurables à grain élevé ou épais sont matures, l'auto-reconfiguration des circuits devient accessible. Le (R)SOC gagne en autonomie, le concepteur lui a délégué une partie des décisions et l'utilisateur par ses choix agit sur la configuration du SOC.

9. Le SOC devient intelligent et conscient de son état, il s'adapte et inter-agit avec l'utilisateur au travers de capteurs et de capacités propres d'adaptation.
10. Les connexions et communications entre systèmes embarqués (SOC) se généralisent. Un système global distribué et fonctionnant sur un mode coopératif apparaît, il peut s'auto-configurer afin de rendre les service souhaités par la communauté d'utilisateurs et de systèmes.

Cette synthèse ne prétend pas être formelle ni exhaustive, elle met simplement en lumière comment les dépendances croisées entre les usages, les méthodes et les circuits ont transformé la façon d'appréhender le domaine des méthodologies de conception. Ces constats et perspectives ont guidé le travail que résume ce manuscrit. Le reste de ce document est organisé autour des différents projets que j'ai pu mener depuis la thèse jusqu'aux perspectives de recherche que je dresse actuellement.



A : Ères de informatique



B : Évolution des rapports entre espèces : Utilisateur, Concepteur, Machine, SoC

FIG. 1.1 – Évolution de l'informatique et de la CAO

1.2 Parcours recherche

1.2.1 Parcours universitaire & recherche

- 1992, Ingénieur Electronicien Informaticien, Ecole Supérieure d'Electronique de l'Ouest, (ESEO, Angers). Stage puis vacations au Centre de Recherche en Physique de l'Environnement (CNET/CNRS). Participation au projet TRMM de satellite d'observation du climat [3].
- 1993 : DEA STIR (Signal, Télécoms, Image, Radar), Université de Rennes I. Stagiaire au Centre de recherche de Thomson à Rennes. Synthèse comportementale de filtres optimisés.
- 93/96, Thèse de doctorat au LASTI (Lannion, à présent CNRS R2D2/IRISA), Université de Rennes I, Mention Traitement du signal et télécommunications, domaine d'étude : la synthèse d'architectures (estimation de complexité et transformations algorithmiques).
- 96/97, Post Doc à l'IMEC, Leuven, Belgique, laboratoire majeur au niveau mondial dans le domaine de la micro-électronique et de la conception de systèmes sur silicium. Projet : méthode de conception de hiérarchies mémoires pour la minimisation de la consommation.
- Sep. 97/ oct. 02 : Maître de conférence, UBS, Lorient. Membre du LESTER, responsable du projet Design Trotter pour l'exploration de l'espace de conception. PEDR 2002/2003.
- Oct. 02/Déc.03 : En disponibilité de l'UBS pour transfert technologique. Lauréat concours Anvar, "Ecrans Micro-électromécaniques non lumineux, très faible consommation et Supports de Communication de Proximité ".
- Co-fondateur de l'entreprise dixip en mars 2004.
- Depuis Janv. 04, chargé de recherche CR1 CNRS. Responsable du projet RaaR portant sur les architectures intelligentes capables de s'auto-configurer en fonction de l'environnement.

1.2.2 Résumé des travaux de recherche

À ce jour mes activités de recherche ont principalement traité des différents aspects d'un domaine unique à savoir l'exploration de l'espace de conception des systèmes sur silicium (SOC) éventuellement reconfigurables. Celles-ci se décomposent en quatre phases.

La première est le travail de thèse réalisé au LASTI (R2D2/IRISA) à Lannion dans le domaine de la synthèse d'architecture et consacré à l'estimation de complexité d'intégration d'algorithmes de type calcul intensif (ex. filtrage adaptatif pour les télécoms). Cette première phase a abouti, dans le cadre de l'environnement de synthèse comportementale GAUT, à l'implantation de deux méthodes originales d'estimation [4, 5].

La seconde est liée à mon année post-doctorale à l'IMEC (Leuven, Belgique) dans l'équipe SEMP *system exploration for memory and power* pilotée par Francky Catthoor, et traite de l'estimation de complexité d'intégration d'applications dominées par les transferts de données (ex. MPEG). Le résultat obtenu est la définition de l'étape de sélection de la hiérarchie mémoire pour la réduction de la consommation, celle-ci a donné lieu à un dépôt de brevet aux USA dans le cadre de l'industrialisation de l'environnement ATOMIUM.

La troisième menée au LESTER entre 1997 et 2002 est dédiée à l'estimation de complexité d'intégration de systèmes hétérogènes (logiciel / matériel) de granularité variable incluant des fonctions orientées mémoire, traitement et contrôle. Cette étape a abouti à un flot de conception matérialisé par un de environnement logiciel ouvert Design Trotter.

La quatrième étape correspond à ma prise de fonction en tant CR CNRS qui me permet depuis janvier 2004 de mettre en place un nouveau projet de recherche au LESTER. Il s'agit d'outils et de méthodes pour la conception d'architectures dont les reconfigurations algorithmique et architecturale sont régulées en ligne (architectures intelligentes). Ce dernier s'appuie sur la consolidation de mes précédents travaux en matière d'exploration.

1.2.3 Transfert technologique

En 2001, j'ai été lauréat du concours Anvar dans la catégorie émergence. En Oct. 2002, je me suis mis en disponibilité de l'université pour effectuer un transfert technologique tout en continuant l'encadrement des thèses en cours. Ce transfert repose sur deux dépôts de brevets (2001, 2003) distincts dans deux domaines différents. Le premier est la démonstration théorique puis expérimentale d'un micro-moteur électro-statique inédit pour la réalisation de pixels et donc d'écrans très faible consommation. Le second est un procédé efficace de codage et d'échange de données (initialement : communications sans fil entre les usagers et les écrans précédants en milieu urbain) reposant non pas sur le contenu mais sur le format et offrant des propriétés de mesure du taux de pertinence. Ce dernier aspect a abouti à la création de l'entreprise dixip à Ploemeur, celle-ci emploie à ce jour quatre ingénieurs.

1.2.4 Organisation du document

Après un retour sur le postdoc, je présente une vue globale du projet de recherche que j'ai mené au LESTER. Ensuite, chaque chapitre est dédié à un des aspects de ce projet. Le second chapitre est le plus dense en raison de son ancienneté et du nombre de participants, il est consacré aux deux niveaux de l'environnement Design Trotter (DT au niveau fonction et RTDT au niveau tâche). Le troisième chapitre traite des différents travaux effectués autour de la problématique des entrées / sorties d'un SOC. Le quatrième chapitre, qui s'appuie sur les leçons tirés des travaux précédents, détaille l'état d'avancement de mon projet actuel dans le domaine des systèmes auto-reconfigurables. Le chapitre suivant est une synthèse de mes diverses responsabilités. Le sixième chapitre est une conclusion qui résume l'état de mes réflexions à ce jour. Enfin, le dernier chapitre est la liste de mes publications dont deux sont fournies en annexes.

1.3 Post-Doc : Hiérarchie mémoire et consommation

Les applications de type multi-média se traduisent par des algorithmes où les transferts de données multi-dimensionnelles dominent le reste des traitements notamment au niveau de la consommation. Dans ce contexte les choix réalisés au niveau de la hiérarchie mémoire sont déterminants et reposent en grande partie sur l'exploitation de la localité des données. Le projet que j'ai mené dans le groupe SEMP à l'IMEC a consisté à formaliser avant toute décision architecturale, l'exploration de l'espace de conception. Celui-ci a été réalisé en collaboration avec Sven Wuytack, il constitue une extension de son travail de thèse. Le projet a abouti à la spécification de l'ensemble des hiérarchies mémoire possibles pour une application donnée et à l'extraction des solutions offrant le meilleur compromis en termes de surface et de consommation.

L'extraction des informations relatives à la réutilisation des données entres les différents niveaux d'un nid de boucles constitue la base de la méthode. Le type de représentation utilisé est le graphe polyhédrique qui offre un modèle analytique efficace pour la manipulation de données multi-dimensionnelles. Historiquement ce choix provient d'une collaboration entre l'IMEC et l'IRISA (INRIA) dans la cadre du projet européen NANA. La méthode développée se situe au niveau algorithmique et construit, sur la base des taux de réutilisation de données issues de la représentation polyhédrique, un *copy candidates graph* représentant efficacement un espace des solutions qui peut s'avérer être de taille très importante. Nous avons montré, notamment sur l'exemple de l'estimation de mouvement (MPEG), que des gains extrêmement importants pouvaient être obtenus par la réalisation de hiérarchies mémoire ad hoc à destination de systèmes embarqués.

L'article [6] joint en annexe précise les détails de la méthode.

1.4 Vue globale du projet de recherche

Au fur et à mesure des projets et collaborations, je me suis attaché à construire, au sein du groupe AAS (adéquation application système) du LESTER, un projet dont les diverses ramifications conduisent progressivement vers un schéma global cohérent. Le but n'est pas d'atteindre l'outil presse-bouton universel. L'objectif est d'offrir au concepteur, en collaboration avec d'autres outils existants ou en devenir, les différents niveaux d'analyse dont il a besoin pour la réalisation d'un système sur puce répondant aux critères fixés par l'usage (performance, consommation, QoS). L'environnement d'exploration et de conception de SOC logiciel/matériel Design Trotter constitue le noyau autour duquel se développent tous les projets de l'équipe. Il vise à guider la conception de systèmes sur silicium disposant éventuellement de plusieurs processeurs et de ressources matérielles reconfigurables.

Cette problématique englobe un certain nombre de sous-thèmes de recherche liés, comme ceux de la spécification d'un système aux niveaux architectural et applicatif (Design Trotter, A3S), l'estimation en ressources et performance(Design Trotter : DT), la décision de répartition logiciel matériel (Design Trotter : RTDT), la reconfiguration architecturale (RaaR), la modélisation d'architectures et la liaison avec les outils de compilation logiciel/matériel (A3S), la gestion des entrées/sorité (NOC/IP, *muSpider*). Mes activités et celle de l'équipe abordent ces différents aspects précisés sur le schéma du haut de la figure 1.2.

Le point de départ de ce travail est le constat établi après la thèse et le postdoc. Comment aborder le problème de l'exploration de l'espace de conception pour un système complexe et potentiellement hiérarchique, composé de tâches pouvant être en partie dominées par les traitements, les transferts mémoire ou le contrôle ? Les modèles utilisés où disponibles à l'époque ne correspondaient pas à nos besoins, nous avons donc entrepris de construire le notre. L'objectif initial du problème était la transformation d'un graphe de tâches composées de fonctions définies de manière hiérarchique vers un graphe représentant des processeurs hétérogènes inter-connectés.

Ces projets ont débuté avec l'exploration architecturale au niveau fonction (Design Trotter, DT : thèses de Y.Le Moullec, S.Bilavarn, N.ben Amor), puis au niveau des systèmes monoprocesseur multi-tâches augmentés d'accélérateurs et co-processeurs (PACM) dans le cadre de l'outil (Design Trotter, RTDT : thèse de A.Azzedine).

Les architectures multi-processeurs ont ensuite été considérées à travers une étape de partitionnement préalable qui crée des clusters de tâches selon une approche multi-critères (thèse en cours de I.Maalej). La notion de gestion de la qualité de service lors du partitionnement et de l'ordonnancement a ensuite été introduite (thèse en cours de H.Tmar), ce travail repose sur un modèle probabiliste des temps d'exécution. La dimension suivante est celle de leur mise en réseau au travers d'un *Network on Chip* (μSpider : thèse en cours de S.Évain), ce qui se traduit également par une étude de l'interface réseau (thèse de B.Bouallegue) et enfin leur auto-reconfiguration en fonction de la situation c'est à dire des choix de l'utilisateur et de l'environnement du système (Batterie, SNR, %CPU, %Buffer, ..) dans le cadre du projet RaaR (thèse en cours de Y.Eustache).

La spécification haut niveau, en l'occurence par UML, a été développée et intégrée dans le cadre du projet A3S, selon l'approche MDA (*Model Driven Architecture*) dans le contexte de la radio-logicielle (Thèse en cours de S.Rouxel). Dans ce contexte des composants logiciels et matériels sont définis et associés, le travail en cours consiste à intégrer les services et ressources des OS disponibles sur la cible.

Les interactions entre ces domaines ainsi que les différents travaux qu'ils intègrent sont précisés sur le schéma du bas de la figure 1.2. Les points rouges correspondent aux thèses que j'ai co-encadrées ou co-encadre actuellement.

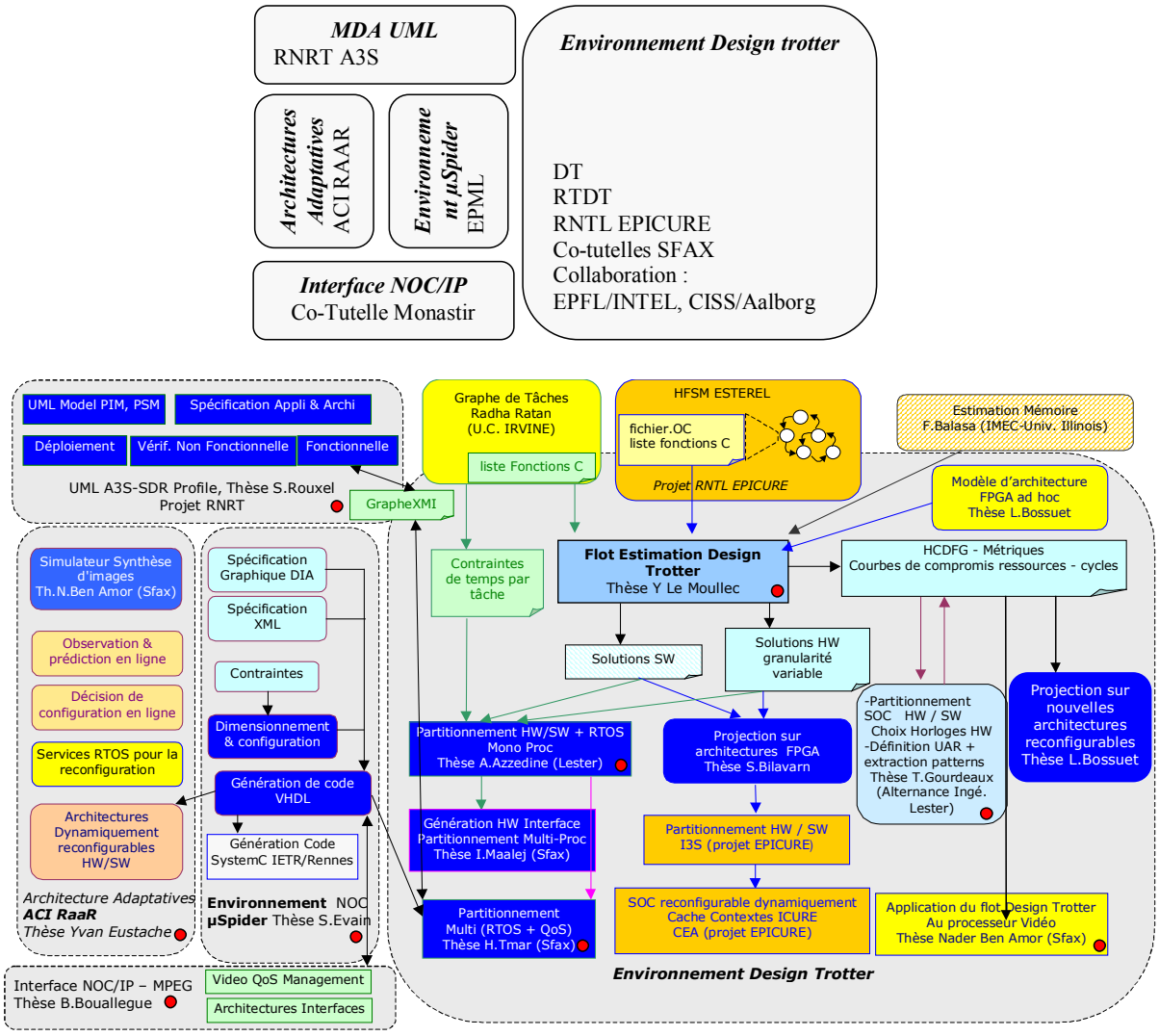


FIG. 1.2 – Vision globale des projets

1.5 Intégration dans le projet global du LESTER

Comme précisé plus haut, nous ne cherchons pas à embrasser l'ensemble des niveaux du flot de conception. De ce point de vue, celui s'intègre dans la stratégie globale du Lester et s'interface avec les projets des autres équipes du laboratoire. L'exploration précède la synthèse d'architecture en termes de choix d'architecture et de paramétrage mais peut également exploiter une libraire d'IP prédéfinis et caractérisés (équipe IP : E.Martin, E.Casseau, P.Coussy,A.Baganne). L'exploration s'appuie sur des modèles d'architectures éventuellement reconfigurables (équipe AS : B.Pottier, L.Lagadec, C.Dezan, E.Fabiani). L'exploration nécessite des mesures ou estimations de consommation pour les composants manipulés (équipe Consommation : N.Julien, E.Senn, J.laurent). La conception d'architectures adaptatives nécessitent une expertise dans le domaine d'application visé, il s'agit notamment de la radio-logicielle (équipe Communications Numériques : E.Boutillon, C.Roland) ou celui de la sécurité (G.Gogniat). Enfin la modélisation de l'espace de reconfiguration s'enrichit de concepts et de méthodes similaires étudiés dans d'autres domaines tel que celui de la transitique où la notion de machines et de convoyeurs rejoint celles des processeurs et bus (P.Berruet, A.Rossi).

Chapitre 2

Design Trotter : atelier pour l'exploration de l'espace de conception

2.1 Positionnement

Cette partie positionne les travaux, effectués principalement dans le cadre de l'environnement Design Trotter, selon les différentes assertions du domaine de recherche dit de l'exploration de l'espace de conception. La notion d'exploration revêt plusieurs significations qui peuvent être classées suivant quatre axes indiqués sur la figure 2.1 et détaillés ci-après.

2.1.1 Les domaines architecturaux visés

Un outil d'exploration idéal devrait adresser l'ensemble des domaines suivants : choix et nombre de processeurs, dimensionnement et paramétrage de NOC *Network On Chip* ou de hiérarchie de bus, hiérarchie de mémoires, sélection de modules matériels dédiés ou reconfigurables, le choix de périphériques, la personnalisation des RTOS (*Real-Time Operating System*), enfin la spécification du jeu d'instructions et de coprocesseurs dans le cas de cœurs de processeurs configurables. En pratique les outils et méthodes d'exploration ne traitent que partiellement l'espace de conception.

Notre approche se situe à plusieurs niveaux. Design Trotter (DT, section 2.2) explore le parallélisme avant le choix de la cible et peut ensuite être spécialisé pour une implantation de type IP matérielle dédiée ou configurable ou cœur de processeur configurable.
Le second niveau RTDT (section 2.3) a été conçu initialement pour le partitionnement logiciel/matériel à granularité variable sur des architectures de type processeur + coprocesseurs + IP matérielles communiquant à travers un bus. Les différentes solutions issues de l'analyse par DT alimentent les alternatives pour chacune des tâches considérées. Dans le cadre du projet RNTL A3S, le modèle a évolué pour intégrer des composants de type DSP et s'étend à présent vers les architectures multi-processeurs.
Enfin l'outil μSpider (section 3.4) a été développé pour spécifier mais également dimensionner un NoC en fonction de contraintes applicatives.

2.1.2 Les modèles de spécification utilisés

Le modèle de spécification dépend du domaine d'application lequel peut nécessiter des caractéristiques de contrôle, de synchronisation, d'exhibition de parallélisme et de granularité qu'offrent de manière inégale les différents modèles existants. Ainsi une application hétérogène nécessite un modèle hybride.

Notre vision de cette question est pragmatique, l'outil DT repose sur un modèle de type flot de données et de contrôle hiérarchique (HCDFG, cf section 2.2.3) qui est utilisé pour l'extraction

du parallélisme et des structures de contrôle au sein d'une tâche initialement décrite par un langage procédural (e.g. C). Ce modèle HCDFG est encapsulé dans un graphe de tâches (cf. section 2.3) au niveau système, ce dernier est utilisé lors de l'exploration logicielle/matérielle. La séparation entre les modèles flots de données et contrôle (GT)relève de la décision de l'utilisateur.

2.1.3 Le type d'exploration

Le terme même d'exploration exprime un parcours au sein d'un ensemble de solutions, cependant il est généralement employé en pratique pour désigner une plate-forme logicielle permettant de spécifier un système. L'exploration peut ainsi, se décomposer en plusieurs niveaux suivant un modèle comparable à celui régissant les problèmes NP complets qui se déclinent en trois phases Optimisation, Décision, Validation.

En l'espèce il s'agit de i) la validation non fonctionnelle (ex. contraintes temps réels, cohérence de types, espace mémoire), ii) de la validation fonctionnelle au travers de la (co-)simulation de systèmes hétérogènes et iii) du partitionnement d'une application sur une plate-forme plus ou moins figée ou générique. Les deux premiers points sont associés à l'exploration dans le sens où ils la rendent possible dans une certaine mesure. Ils offrent en effet au concepteur un niveau d'abstraction permettant de tester rapidement une configuration parmi d'autres. Le partitionnement par contre s'accompagne d'algorithmes de parcours d'espace de solutions (algorithmes évolutifs, branch & bound, gloutons etc.) qui font évoluer les caractéristiques du système visé (choix des cibles processeurs et/ou DSP et/ou IP matériels éventuellement reconfigurables, choix de la granularité des composants à disposer, etc.). Ce type d'exploration repose généralement sur une décision d'ordonnancement à grain habituellement élevé (tâche) dans le domaine du codesign logiciel/matériel. Cependant la question de l'ordonnancement constitue une dimension de l'exploration en soi, dans ce cas celle-ci consiste à allouer un budget de cycles variable aux différentes fonctions composant une application et produit des courbes de compromis dynamiques exprimant le nombre de ressources en fonction du nombre de cycles à tous les niveaux de granularité. Enfin, le dimensionnement des composantes d'un SoC représente une dernière dimension de l'exploration qui consiste à faire varier des paramètres tels que la taille de cache, le jeu d'instructions, les types d'accès mémoire, etc.).

Les travaux que j'ai encadrés adressent différents types d'exploration. Le projet RNRT A3S, qui utilise l'outil RTDT, effectue des vérifications non-fonctionnelles et fonctionnelles à partir de descriptions UML de l'application et de la l'architecture. RTDT effectue une exploration de type partitionnement logiciel / matériel à partir d'un modèle de spécification de type graphe de tâches. Ces tâches sont spécifiées avec différents niveaux de granularité possible. DT effectue une exploration qui repose sur des combinaisons d'ordonnancements multiples et produit des courbes de compromis pour tous les niveaux de granularité issus de la spécification de type HCDFG. Enfin, outre la spécification et la génération de code, μSpider effectue un dimensionnement du NoC en fonction des contraintes de l'utilisateur.

2.1.4 Le niveau de raffinement

Une clef de l'exploration est sa complexité, celle-ci croît avec le niveau de raffinement des architectures et applications. Une grande diversité de formalismes est utilisée dans la littérature pour désigner les différents niveaux de raffinement. Cependant une sorte de consensus semble émerger du domaine de la co-simulation, celui-ci est reporté sur le schéma de la figure 2.1. La terminologie employée peut cependant varier, on le constate par exemple en comparant l'approche de Gajski [7] avec celle de MASIC [8]. On distingue cependant deux axes communs, ceux de la communication et du traitement pour lesquels trois niveaux sont globalement définis de la manière suivante :

Le niveau fonctionnel ou *untimed* correspond à une spécification (Matlab, C, ..) pour la simulation fonctionnelle sans notion de temps, le niveau *approximate time* confère à une suite

d'opérations de grain généralement élevé (thread, fonction) une latence estimée, enfin le niveau *cycle accurate* tient compte des délais à un niveau très fin (ex. *Clock'event* en RTL). Il apparaît que deux niveaux supplémentaires sont nécessaires si l'on considère l'intégration, le premier se situe après le niveau *cycle accurate* et prend en compte le placement et le routage des IP sur SoC. Le second nous concerne et se situe entre les niveaux fonctionnel et temps approximé (zone DT1 sur la figure 2.1). Il s'agit de celui des cycles abstraits qui permet d'explorer l'espace de conception avant d'avoir défini une architecture tout en prenant en compte l'aspect temporel. Ce dernier est utilisé par l'attribution de cycles abstraits aux différents types d'opérations présents dans l'application.

Notre approche consiste à considérer que l'exploration prend son sens à un haut niveau d'abstraction afin d'évaluer un large spectre de solutions. Comme indiqué précédemment les niveaux précis relèvent de la vérification et/ou de la simulation. Aussi, l'exploration de type DT est effectuée au niveau algorithmique à partir d'un code C sans a priori sur l'implantation. L'objectif est de rapidement fournir au concepteur les clefs lui permettant de prévoir les gains potentiels pouvant être obtenus sur une architecture disposant du parallélisme exhibé par l'exploration. Cependant la projection physique, qui consiste à préciser le modèle d'architecture (prioritairement sur FPGA), offre un niveau de raffinement supplémentaire du type *approximate time* qui permet d'estimer les temps de traitements et d'accès mémoire ainsi que la surface de l'architecture (zone DT 2 sur la figure 2.1).

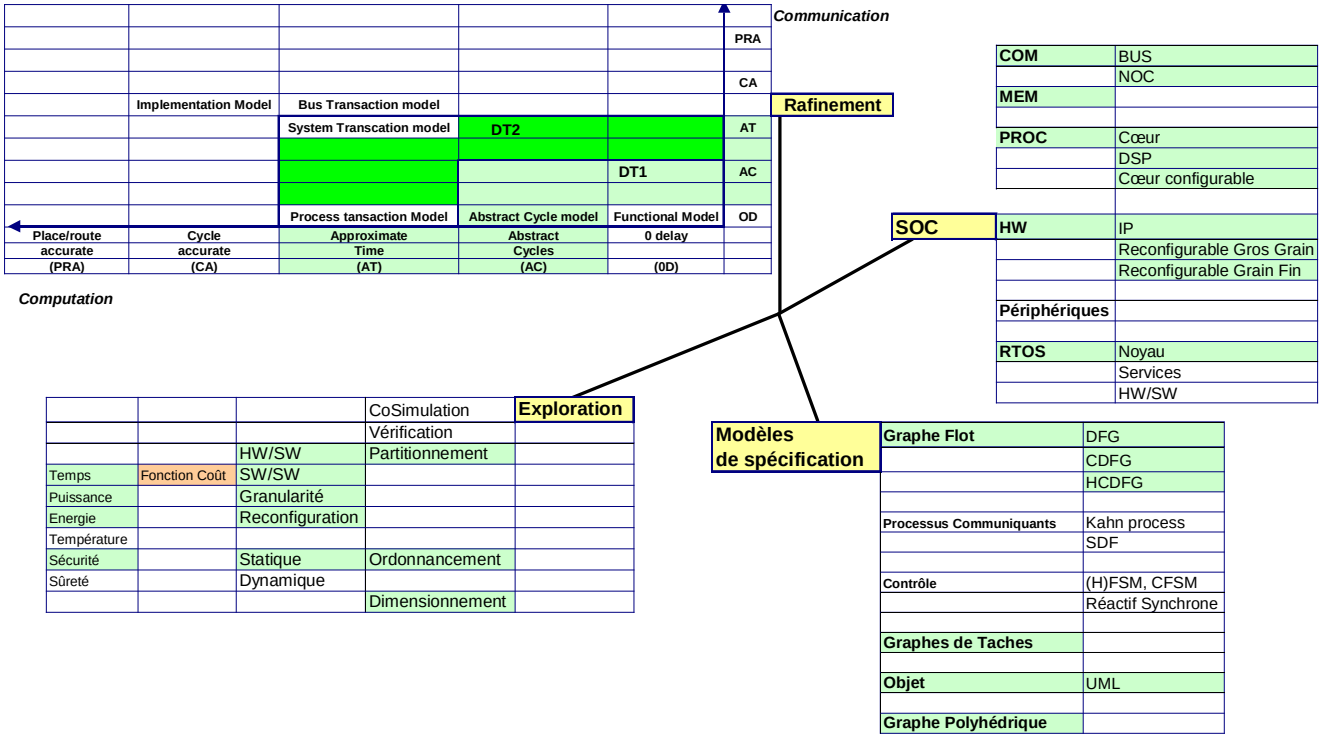


FIG. 2.1 – Parcours de l'espace de conception. En vert, les thèmes abordés.

2.2 Exploration au niveau fonction, DT

2.2.1 Introduction

J'ai initié en le projet Design Trotter (DT) peu après mon arrivée au LESTER en 1998. Les motivations à l'origine de ce travail mené avec Jean-Luc Philippe (Pr.) et Guy Gogniat (MCF) étaient et demeurent le guidage de l'utilisateur vers un choix d'architecture et un environnement adapté à une ou un ensemble d'applications dans le domaine de l'électronique embarqué.

En premier lieu vient la question de la taille de l'espace des solutions offertes au concepteur qui rend nécessaire l'insertion d'une étape d'exploration avant les outils de synthèse ou compilation. Il s'agit d'une part de guider l'utilisateur vers le choix d'architecture ou de configuration d'architecture. D'autre part la question est l'extraction des parties pertinentes de l'application ainsi que les contraintes (niveau de granularité, nombre de cycles, librairie de composants, déroulage de boucle, hiérarchie mémoire) à imposer aux outils identifiés pour la synthèse architecturale.

D'un point de vue architectural cet aspect s'est considérablement étoffé ces dernières années avec le développement des SoC (re)configurables intégrant des blocs mémoires, des blocs spécialisés (DSP, IO, ..) et des processeurs configurables tels que le NIOS d'Altera, le MicroBlaze de Xilinx ainsi que les architectures DRP de NEC, ISEF de Stretch Inc, XPP de PACT ou MRC6011 de Freescale. Les DSP configurables de MorphICs, PicoChip et Morpho Tech. dans le domaine de la radio-logicielle offrent des plate-formes possédant leur propre environnement de développement. Une autre approche telle que Cascade de CriticalBlue propose un environnement visant la synthèse de co-processeurs à partir de processeurs embarqués du commerce.

Parallèlement on observe que l'offre d'outils de synthèse architecturale (Forte/Cynthesizer, Mentor/CatapultC, Synfora/PICO, Synopsis/CCSS, DK/Celoxica) traduit la disponibilité d'outils industriels dans ce secteur malgré le marché de niche auquel semble-t-il ceux-ci s'adressent (quelques dizaines de clients). Ces outils de plus en plus efficaces sont cependant limités techniquement. De l'avis des utilisateurs (ST, Thales notamment) la génération suivante devra intégrer une étape d'exploration permettant d'identifier les parties critiques, de sélectionner les configurations les plus prometteuses et de contraindre la synthèse en fonction des performances souhaitées ou des ressources disponibles. C'est le rôle d'un outil comme DT.

Enfin les outils de co-simulation atteignent une certaine maturité (Coware, Cofluent) en grande partie grâce aux facilités de co-simulation apportées par l'approche C++/systemC. Cette orientation se traduit aussi par l'apparition d'outils de synthèse prenant en entrée une description sur la base du langage C (SystemC, Stream-C, Handel C, Impulse C, μC++). Par contre les outils consacrés à l'exploration avant synthèse comportementale, restaient [9] et demeurent peu exploités. Cet espace de conception est complexe et diversifié car aucune solution unique n'a pu être trouvée afin de répondre aux challenges de l'électronique embarquée actuelle et future. Suivant les contraintes et objectifs visés le compromis, que l'exploration doit contribuer à déterminer, favorisera plus ou moins les aspects suivants :

- le temps de conception qui constitue le premier critère d'un point de vue industriel ;
- la consommation qui sera d'autant plus réduite que l'architecture sera spécialisée et donc les transistors et le temps utilisés efficacement ;
- le respect des contraintes de temps exploitant plus ou moins le parallélisme disponible ;
- la programmabilité ou flexibilité qui rejoignent le premier point.

En second lieu, l'étape d'exploration doit aider au choix de l'outil, du type de spécification et de la méthodologie de conception qui sera adapté au type de synthèse à effectuer pour chaque partie de l'application. Cet aspect, que nous traitons notamment à l'aide de métriques, adresse

tout d'abord l'extraction des parties critiques de l'application en précisant si elles sont orientées traitement, contrôle et ou mémoire. Dans le cas d'une architecture du type processeur augmenté d'un co-processeur matériel (e.g. NIOS + accélérateur synthétisés sur FPGA Altera), le contrôle peut être pris en charge par le processeur, et donc réalisé sous une forme logicielle avec éventuellement un jeu d'instruction enrichi. Dans le cas d'une réalisation entièrement matérielle, une combinaison d'outils de synthèse d'architectures orientés traitement (graphe flot de données, e.g. GAUT, MMAlpha) et contrôle (machine d'état et gestion des boucles, e.g SPARK, UGH) peut être envisagé. Un traitement spécifique peut également être distingué lorsque les applications sont dominées par les transferts mémoire, c'est par exemple le partie pris de l'environnement DTSE (Atomium, Adopt) de l'IMEC à présent distribué par PowerEscape Inc.
Le troisième point inhérent à l'exploration est la flexibilité du modèle d'architecture cible car celui-ci peut être initialement inconnu ou partiellement figé. C'est pourquoi nous nous sommes dirigés vers une solution permettant de spécifier une architecture générique dont le parallélisme peut évoluer en fonction du compromis ressources / temps d'exécution. Ce compromis inclue l'exploration des besoins en mémoire au niveau local (e.g. registres internes) ou global (mémoire externe). Le corollaire de cette flexibilité est la nécessité de développer une méthode qui permette d'obtenir des résultats rapidement afin de procéder à des modifications aux niveaux architectural et algorithmique.

Enfin, l'exploration peut s'envisager à différents niveaux de granularité, le parallélisme exploitable peut se situer au sein d'un DFG au niveau instruction ou opération selon le contexte, entre les différents niveaux d'un cœur de boucle, entre les différentes parties plus ou moins dépendantes d'un algorithme, enfin entre les tâches constituant une application complexe. Afin d'appréhender de manière unifiée cet espace de recherche nous avons adopté une approche par graphe hiérarchique (HCDFG) constituant les nœuds d'une graphe de tâche. La section 2.2 présente l'exploration au niveau fonction, la section 2.3 détaille le niveau tâche.

2.2.2 Flot d'exploration

L'outil DT a été conçu comme un outil d'analyse rapide et interactif pouvant être utilisé avec différents niveaux de raffinement en fonction de la connaissance de l'architecture visée. Le premier se traduit par une architecture abstraite reposant sur un modèle algorithmique. Le principe de DT est de transformer une application spécifiée sous la forme d'un programme (C à ce jour), en un unique HCDFG (*hierarchical control data flow graph*) constituant un espace générique d'exploration afin de mesurer les grandeurs suivantes :

- Orientation, ou type d'opérations dominantes : test, accès aux données, traitement ;
- Parallélisme aux niveaux des accès aux données et traitements ;
- Localité des données : placement des données dans une hiérarchie de mémoires.

Cette exploration s'effectue suivant trois axes :

- La granularité : depuis les opérations élémentaires jusqu'au graphe complet représentant l'application ;
- Le nombre de cycles alloués pour l'exécution : potentiellement depuis le chemin critique jusqu'au chemin séquentiel, ce choix peut entraîner une décision de déroulage de boucle ad hoc ;
- Le modèle de l'architecture cible : opérateurs et instructions spécifiques dans l'unité de traitement.

Le flot d'exploration décrit en section 2.2 se divise en quatre étapes principales :

- la spécification et la représentation sous forme de graphes hiérarchiques ;
- la caractérisation ;
- le calcul de courbes de compromis ressources / budget de cycles en fonction des choix du concepteur ;
- la projection physique.

2.2.3 Spécification

Contrôle / flot de données

Notre approche de la spécification repose sur la séparation entre les parties contrôle dominées par les tests ne pouvant être décidés à la compilation et les parties de type flot de données. Dans le projet RNTL EPICURE, cette séparation est effectuée par le concepteur qui introduit l'appel de fonctions C depuis une spécification par machines d'états SSM à l'aide de l'outil ESTEREL studio [10]. Au sein de notre outil RTDT (section 2.3) la limite est établie via la définition d'un graphe de tâches dont le comportement est spécifié à l'aide d'un programme C (fig.2.3) traduit sous la forme d'un HCDFG. Le projet EPICURE a mis en évidence l'impact du choix de cette séparation sur l'architecture résultante [11] : une description par FSM à très bas niveau ne permet pas d'exploiter le parallélisme alors qu'une granularité trop élevée rend inefficace les choix d'implantation matérielle. Il existe deux façons de traiter la question. La première consiste, à partir d'une description orientée contrôle, à combiner les différents états du système jusqu'à l'obtention de chemins de données exploitables au sens du parallélisme. La seconde débute avec un modèle de type HCDFG et consiste ensuite à mesurer le poids du contrôle pour éventuellement affiner le découpage. Nous abordons cette question suivant cette seconde voie en deux temps. Tout d'abord le calcul de métriques permet de mesurer au sein du HCDFG, le ratio contrôle / flot de données (cf. 2.2.4) pour ajuster le modèle de spécification.

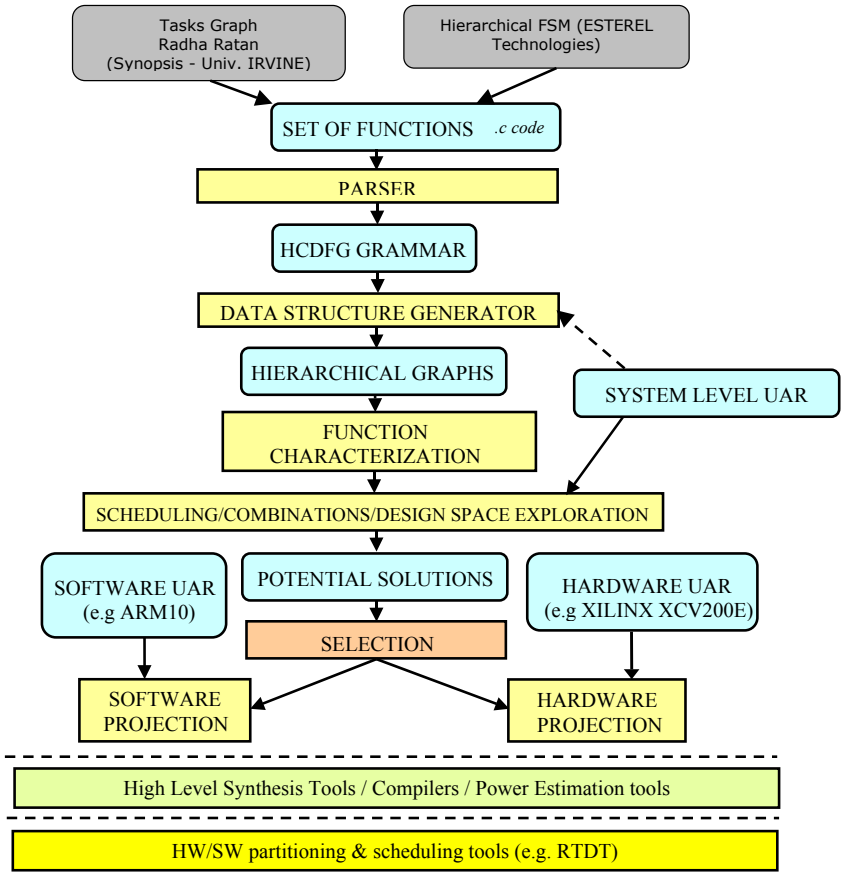


FIG. 2.2 – Flot d'exploration Design Trotter

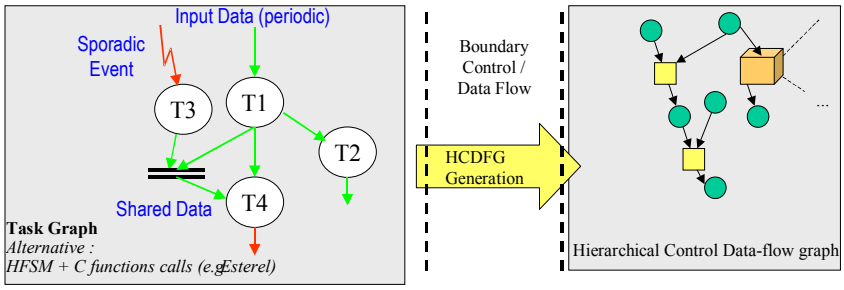


FIG. 2.3 – Séparation contrôle / flot de données

Le modèle HCDFG

La définition du HCDFG repose sur GCC et un parser qui a été défini afin de ne conserver que l'information utile du point de vue de la conception. L'approche choisie consiste à tout traiter sous la forme de graphes avec les règles suivantes :

- Les anti-dépendances sont résolues par une assignation unique avec une politique de renommage permettant de conserver le lien avec la définition de la spécification originelle, celle-ci peut-être de type scalaire ou tableau. La technique mise en œuvre consiste à numéroter les lectures consécutives à chaque écriture pour chaque donnée ;
- La hiérarchie du graphe est l'image de la structure du programme afin de préserver la localité des données ;
- L'approche retenue repose sur l'*instantiation* de composants de type graphes ;
- La représentation choisie supporte les données multi-dimensionnelles.

Le graphe est construit selon une méthode de type *depth-first search* jusqu'à atteindre les nœuds élémentaires qui représentent des traitements ou des accès aux données décrits dans l'architecture générique (UAR). Cette architecture décrite en pratique par un fichier XML a pour fonction de préciser le modèle d'architecture utilisé lors de l'exploration. Ce modèle est plus ou moins précis selon le degré de raffinement souhaité. Trois types de nœuds élémentaires peuvent y être décrits :

- les opérations simples : ce sont par exemple les traitements arithmétiques et logiques et les lectures/écritures de données ;
- les patterns représentant une suite d'opérations simples chaînées (ex. +,-,*) ;
- les fonctions intrinsèques correspondant aux opérations de granularité élevée. Il s'agit par exemple des appels de fonctions (butterfly, FIR, calcul de Syndrome, etc.).

Chaque nœud élémentaire est décrit par des attributs, tels que le nombre de cycles, plus ou moins précis suivant le type d'exploration effectué. Un DFG est construit à partir de nœuds élémentaires. Dès qu'une structure de contrôle (IF THEN ELSE, FOR(;;), etc.) est rencontrée un CDFG est construit. Finalement un HCDFG résulte de la combinaison hiérarchique de DFGs, CDFG, HCDFGs. Les nœuds de type données, potentiellement multi-dimensionnelles dans le cas de boucles, représentent les échanges entres les différents graphes. Les arcs représentent les dépendances de contrôle et de données (scalaire ou multi-dimensionnelle) entre les différents graphes et nœuds élémentaires.

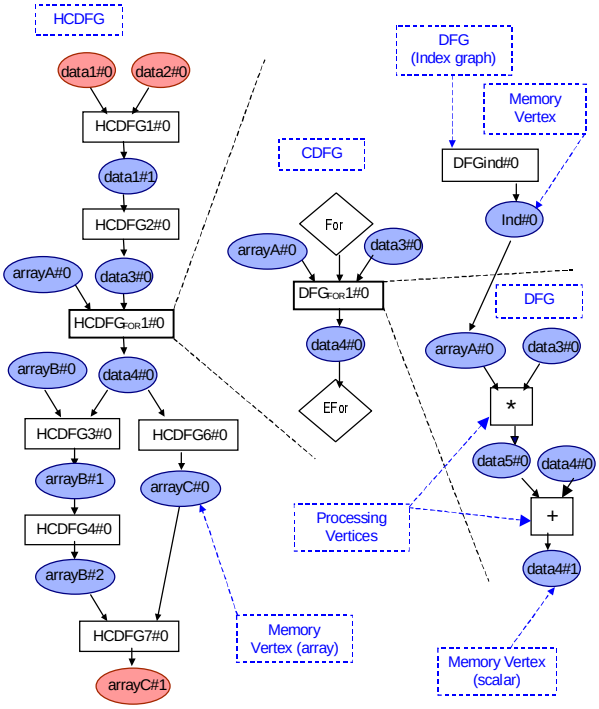
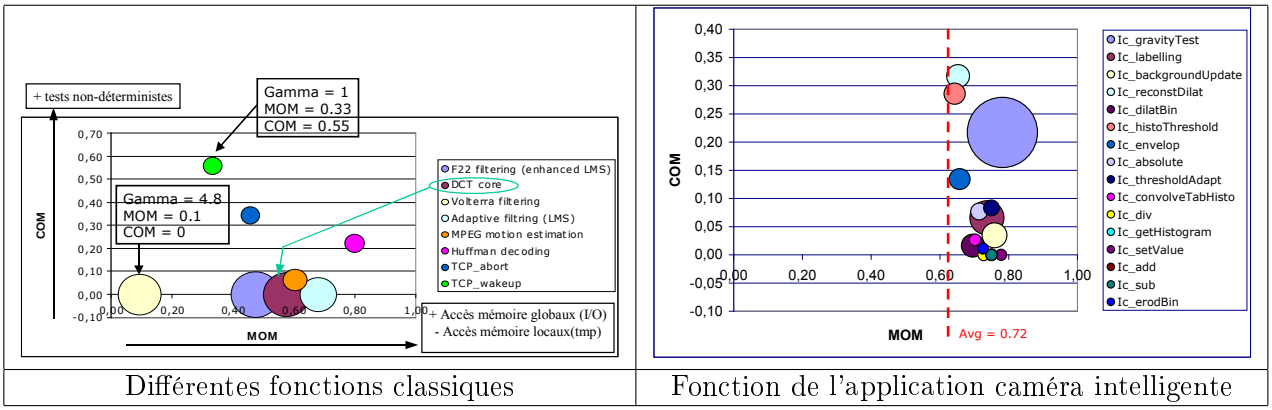


FIG. 2.4 – Exemple de HCDFG

2.2.4 Caractérisation

L'objectif de la caractérisation est d'effectuer une analyse rapide du graphe HCDFG afin d'un extraire, pour chaque niveau de hiérarchie, les informations pertinentes quant au choix de son implantation. Les métriques, détaillées dans [12, 13, 14], sont calculées pour chaque graphe i depuis chaque DFG jusqu'au HCDFG global. Ci-dessous sont données les définitions simplifiées des métriques MoM, MoC, γ et DRM *data reuse metric* avec $N_m(i)$,$N_p(i)$, $N_c(i)$ respectivement les nombres d'opérations élémentaires de type accès mémoire, traitement et test (contrôle). $CC(i)$ représente le chemin critique en nombre de cycles. En pratique les structures de contrôle entraînent des définitions spécifiques. La métrique γ exprime le parallélisme moyen sous la forme du nombre d'opérations de traitements et d'accès mémoire divisé par le chemin critique. Concrètement le calcul de ces métriques dépend des structures de contrôle rencontrées lors de la construction du graphe i considéré. Par ailleurs des règles de combinaison sont utilisées pour déduire les valeurs des métriques à un niveau de la hiérarchie en fonction des résultats obtenus dans les niveaux inférieurs. La figure 2.5 montre les distinctions qui apparaissent entre des fonctions de nature différente. On observe par exemple qu'une fonction LMS possède un MOC très faible, une orientation traitement très prononcée et un parallélisme moyen très important. Une fonction issue du protocole TCP sera, par contre, dominée par le contrôle avec un taux de parallélisme très faible. L'exemple de l'application caméra intelligente utilisée dans le projet Epicure révèle différents degrés de parallélisme pour les différentes fonctions mais une orientation générale mémoire très prononcée.

- Orientation mémoire : $MOM(i) = \frac{N_m(i)}{N_m(i)+N_p(i)}$
- Orientation contrôle : $MOC(i) = \frac{N_c(i)}{N_m(i)+N_p(i)+N_c(i)}$
- Parallélisme moyen : $Gamma(i) = \frac{N_m(i)+N_p(i)+N_c(i)}{CC(i)}$
- Taux de réutilisation des données $DRM(i) = DonneesLues(i)/DonneesUtilisees(i)$



Différentes fonctions classiques | Fonction de l'application caméra intelligente

FIG. 2.5 – Exemples de caractérisations, MoM,MoC, γ est représenté par le diamètre

2.2.5 Principe de l'Exploration par courbes de compromis hiérarchiques

Les courbes de compromis ont pour objet de fournir au concepteur le nombre de ressources à mettre en œuvre pour exécuter l'application en un nombre de cycles donné. Compte tenu de l'approche hiérarchique, ces résultats sont disponibles pour tous les graphes en faisant partie.

Si aucune restriction n'est spécifiée par l'utilisateur, l'exploration est effectuée depuis le chemin critique jusqu'au chemin ne réclamant qu'une seul occurrence de chaque type de ressource requise (chemin séquentiel). L'utilisateur peut cependant restreindre cette exploration en délimitant le niveau de parallélisme, le nombre de cycles et de déroulage de boucle afin d'extraire un nombre raisonnable de solutions vis à vis des cibles matérielles envisageables. L'exploration repose sur l'algorithme simplifié suivant :

Alg. 1 Ordonnancement HCDFG

```
G = HCDFG complet
EXPLORE(G){
IF G non exploré ou présent en librairie THEN
    IF G = DFG THEN schedule(G,selected_policy)
    ELSE SWITCH
         CASE{CDFG BOUCLE}
            traitement spécifique CDFG BOUCLE;
         CASE{CDFG CONDITIONNEL}
            traitement spécifique CDFG CONDITIONNEL;
         WHILE G non exploré
            sélectionner 2 sous graphes G1 et G2 de G;
            IF G1 non exploré THEN EXPLORER(G1);
            IF G2 non exploré THEN EXPLORER(G2);
            IF G1 et G2 indépendants THEN Combinaison parallèle(G1,G2),
            ELSE Combinaison série(G1,G2);
Produire la structure hiérarchique des résultats.
```

2.2.6 Ordonnancements des DGF

Lors de l'exploration les ordonnancements sont exécutés pour les différentes contraintes de temps afin de produire les bases des courbes de compromis ressources / nombre de cycles.

Trois algorithmes originaux ont été développés pour l'environnement DT [15, 16, 17]. Les trois reposent sur le principe du *List scheduling* sous contraintes de nombre de cycles. Cette base

algorithmique a été retenue pour des raisons de rapidité et de qualité d'optimisation suffisante pour une exploration à haut niveau d'abstraction. Les trois types d'algorithmes sont :

1. *Memory First* : ordonnancement prioritaire des accès mémoire ;
2. *Processing First* : ordonnancement prioritaire des traitements ;
3. *Mixte* : traitement simultané des deux types d'opérations.

Ils reposent sur deux techniques particulières. La première "*OARC*" est l'estimation en cours d'ordonnancement des ressources disponibles. À chaque pas d'ordonnancement le nombre minimum de ressources à considérer est réévalué afin de rendre immédiatement utilisables des ressources qui deviendront nécessaires avant la fin de l'ordonnancement. La seconde est la technique de "*la pioche*" employée pour les deux premiers algorithmes qui sont exécutés en deux temps. Après l'ordonnancement d'un premier type de ressources (accès mémoire ou traitement), la marge de cycles éventuellement disponibles après la première étape est utilisée lors de la seconde. Ces cycles sont exploités en insérant des cycles qui décalent l'ordonnancement des opérations précédemment traitées sans en modifier l'ordre. Ce décalage est effectué lorsque la seconde étape requiert ponctuellement d'augmenter la mobilité des nœuds restant à ordonnancer. L'ordonnancement des DFG prend en compte la disponibilité des données en mémoire locale. Si la taille de la mémoire locale est bornée par l'utilisateur, l'ordonnancement peut insérer des cycles de lecture / écriture entre les mémoires locales et globales si la taille fixée s'avère insuffisante. Le choix des données présente en mémoire est effectué en tentant, par une approche glouton, qui tente de minimiser l'*overhead* dû aux transferts supplémentaires.

2.2.7 Cas du déroulage de boucle

Le déroulage intervient dans le processus d'ordonnancement des cœuds de boucles lorsque la contrainte de temps imposée lors de l'exploration ne peut être respectée. Pour résoudre cette question nous avons inversé l'approche couramment employée [18] en traitement numérique du signal, celle-ci consiste à calculer le déroulage nécessaire pour atteindre le débit optimal atteignable pour un algorithme donné. Ainsi nous calculons le déroulage requis pour gagner les G cycles qui font défaut. Le principe est équivalent à ce que l'on observe dans le domaine du pipeline logiciel puisqu'il résulte de l'analyse de dépendances inter-itération et aboutit à l'exécution simultanées d'opérations (resp. d'instruction) issues d'itérations différentes. Les détails sont disponibles dans [14, 17], les équations ci-dessous ainsi qua la figure 2.6 résument le principe.

$$\alpha = \frac{1}{1 - \frac{G}{T - d_{min}}} \tag{2.1}$$

$$d_{min} = \lceil max_{cycles}(\frac{T_{cr}}{\Delta_{cr}}) \rceil \tag{2.2}$$

Où G est le nombre de cycles à gagner, Δ_{cr} est le nombre de délais (repérés par les indices de tableaux dans le HCDFG) et T_{cr} la latence cumulée du cycle critique.

Dans un deuxième temps $\alpha - 1$ clones du cœur de boucle sont fusionnés avec ce dernier pour former un graphe unique qui est ordonnancé avec l'algorithme en vigueur, chaque clone i au préalable voit ses dates ASAP décalées $i * d_{min}$ cycles.

Les dépendances de type accumulation ($y = y + x(i) * h(n - i)$) ne sont pas traitées avec l'approche précédente car elles introduisent une distance de dépendance égale à un. Celles-ci sont néanmoins fréquentes en traitement numérique du signal. Ce type de problème peut être résolu par une transformation algorithmique du type décomposition en arbre dont le degré, et donc l'accélération résultante, sont calculables en fonction du besoin. En effet si CC est le chemin critique initial de la boucle et α le facteur de déroulage alors l'accélération A est (sans les initialisations et opérations de contrôle par souci de simplification) : $A = \frac{\lceil CC/\alpha \rceil + \lceil Log_2(L) \rceil}{CC}$.

Cette transformation a été formalisée lors de la thèse de Y. Le Moullec [14], son intégration est inscrite dans la liste mises à jour en cours (fig. 2.6).

Ces transformations de boucles décidées en fonction d'un budget de cycles et de la disponibilités des ressources est un exemple de ce que l'exploration peut apporter en amont de la synthèse d'architecture.

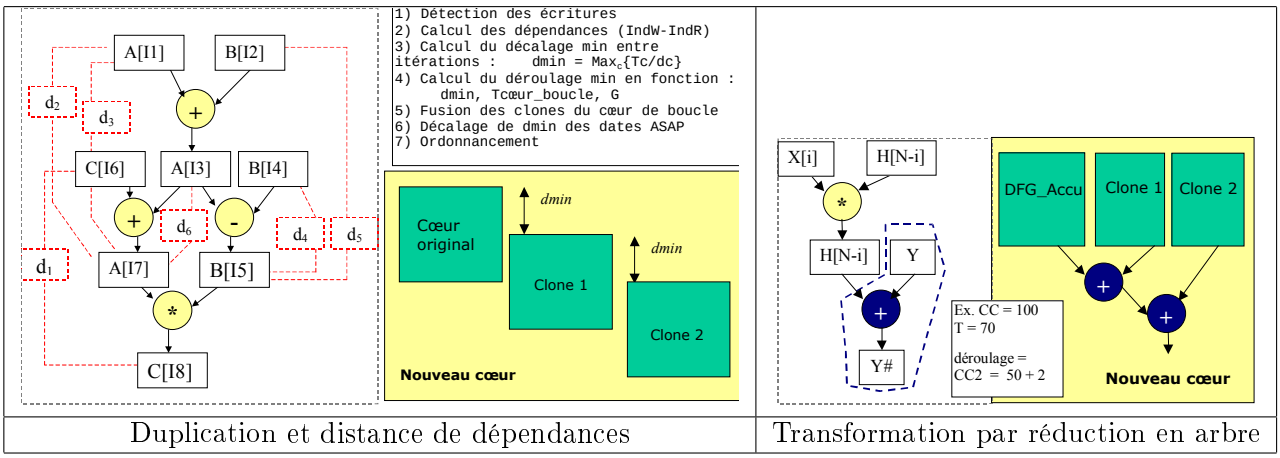


FIG. 2.6 – Calcul du déroulage de boucle minimum

2.2.8 Combinaisons

Les combinaisons ont été introduites lors de la thèse de S.Bilavarn [19] afin d'obtenir rapidement les courbes de compromis ressources/nombre de cycles. L'algorithme de combinaisons a été accéléré [20] lors de la thèse Y.Le Moullec grâce à l'usage des points particuliers qui permettent d'obtenir une solution polynômiale. Comme indiqué sur la figure 2.7, les sous-graphes sont combinés deux à deux en série ou parallèle afin de converger vers une solution unique regroupant les différentes solutions issues de l'exploration. Les branches conditionnelles sont traitées de manière spécifique, la contrainte de temps à respecter est vue globalement en supposant un nombre de passages dans chaque branche qui dépend de la probabilité associée à chaque test.

2.2.9 Estimation des mémoires

L'aspect mémoire est traité à différents niveaux pour différents types d'analyse :

1. Estimation de la mémoire globale de stockage (statique) ;
2. Estimation de la mémoire locale (e.g. file de registres) utilisée lors de l'exécution des DFGs ;
3. Histogramme fournissant le type de données (booléen, octet, long) ainsi que la nature des données (tableaux, scalaire, variable temporaire, constante) ;
4. Expérimentation de l'estimation mémoire suivant la méthode des treillis (graphes polyhédriques).

Le point 1 correspond à l'estimation statique de la mémoire principale pour le stockage des données lues (tableaux, scalaires, constantes) nécessaires à l'exécution du programme aux différents niveaux de hiérarchie. Le nombre de cycles d'accès à la mémoire globale est spécifié dans l'UAR, un *Hit Ratio* peut également être fixé afin d'introduire des délais de manière aléatoire.

L'aspect 2 est traité lors des phases d'ordonnancement puis utilisé lors des combinaisons. Si aucune valeur n'est imposée, la mémoire locale est estimée lors de l'ordonnancement en

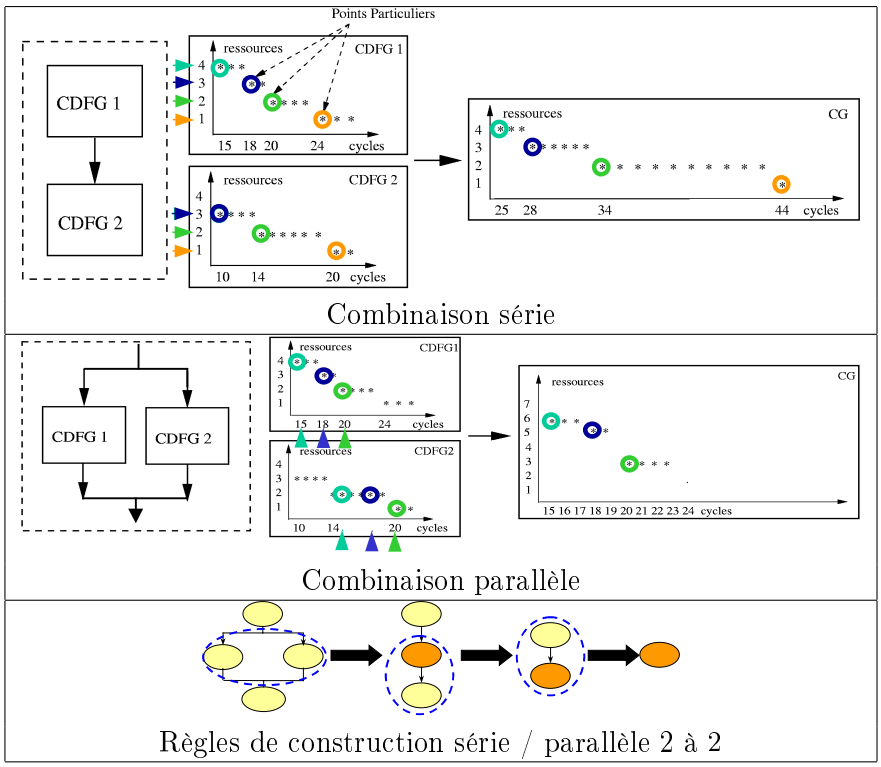


FIG. 2.7 – Combinaisons série / parallèle

considérant qu'une donnée lue une fois lors de l'exécution d'un DFG peut être réutilisée sans nécessiter de cycles de lecture. Le résultat de ce type d'approche est l'estimation de la taille mémoire locale requise. Par contre si une limite est fixée dans la définition de l'architecture alors des cycles d'écriture et de lecture sont ajoutées.

L'histogramme a pour objectif de dresser une carte hiérarchique des types de données manipulées dans les différentes parties de l'application.

Le dernier point 4 correspond à l'utilisation de la méthode développée par Balasa à l'IMEC/ Irvine [21, 22]. Celle-ci repose sur le librairie Polylib de calcul polyédrique de l'INRIA et permet de traiter de manière analytique (i.e. en évitant la simulation) l'évolution de la taille mémoire en fonction des accès en lecture et écriture aux différents tableaux de l'application. Le résultat est une borne minimum et maximum. Cette approche a été implantée et testée sur un nombre de cas typiques de nids de boucle. Cependant, il est apparu que le temps de calcul est très important et les restrictions quant à son usage dans un programme C complexe limitent en pratique sa portée. Cette piste pourrait être de nouveau activée dans le cadre d'une collaboration avec T.Risset du LIP.

Enfin, la gestion des accès à la mémoire est dépendante de caractéristiques telle que la taille du bus, l'usage de *burst* pour les accès aux tableaux ainsi que le placement des tableaux à un niveau donné de la hiérarchie mémoire. Cet aspect a également été formalisé, en modifiant le placement des données dans la hiérarchie mémoire le concepteur peut jouer sur le compromis taille mémoire / performances. Cette option doit être rendue interactive et rapide, un effort d'ingénierie est en cours pour que cela soit simplifié dans la prochaine version.

2.2.10 Projection Physique

Il s'agit du travail de thèse de S.Bilavarn [19], encadré principalement par Guy Gogniat et Jean-Luc Philippe. Après sélection de solutions prometteuses avec un modèle d'architecture

(UAR) simplement algorithmique, le concepteur peut produire une projection de ses solutions sur une cible matérielle de type FPGA par exemple à l'aide d'une UAR dédiée et précaractérisée. La projection physique, traduit les cycles abstraits en temps et les ressources en blocs logiques et cellules dédiées (ex. BRAM), et évalue le coût de la machine d'état résultante. Ce type d'estimation rapide a été utilisée dans le cadre du projet RNTL Epicure [23] afin, après une sélection de solutions réalisées au niveau algorithmique, de produire l'estimation (temps surface) des modules potentiellement instanciables sur la zone dynamiquement reconfigurable de l'architecture cible. Une première approche de l'estimation de consommation a également été introduite [24]. La question de la spécification et de l'exploration de l'architecture FPGA visée lors de la projection a été enrichie et quantifiée dans le cadre de la thèse de L.Bossuet [25] notamment au niveau des communications et de la localité des données. Nous avons également, dans le cadre d'une coopération avec le CISS de l'université d'Aalborg, commencé à explorer la projection logicielle. Il s'agissait en l'occurrence d'un processeur MIPS [26].

2.2.11 Outil, état actuel

À l'heure actuelle, DT a été utilisé pour l'analyse d'un certain nombre d'applications telles que le traitement d'image pour le projet EPICURE [17, 16], la compression vidéo avec l'EPFL dans le cadre du postdoc de S.Bilavarn [27], les télécommunications [28] avec le CISS d'Aalborg, le cryptage [25, 29]. D'autres algorithmes classiques (DCT2D, DWT, FIR, ...) de traitement du signal ont également été présentés dans les différentes thèses du projet.

L'outil DT est actuellement en phase de durcissement afin d'être mis en ligne avant la fin de l'année 2005. L'objectif est de proposer un démonstrateur avec un panel d'algorithmes représentatifs des applications visées dans le domaine de l'électronique embarquée (traitement vidéo, synthèse d'image, compression vidéo, (dé)chiffrement, OFDM, blocs de base de type DCT, DWT). La figure 2.8 présente la version de l'outil d'octobre 2004, la figure du haut illustre le lien entre le graphe sélectionné par l'utilisateur et les courbes de compromis associées. La figure du bas montre comment l'utilisateur peut, à partir de la sélection d'une solution particulière au plus haut niveau de la hiérarchie, atteindre progressivement l'ordonnancement des DFGs dont celle-ci est issue.

2.2.12 Approches concurrentes

L'approche choisie pour l'exploration dans le projet DT connaît peu d'équivalents dans le domaine du codesign sinon au niveau de la caractérisation. Dans ce domaine on trouve les travaux de l'UGS pour la sélection de processeurs prédéfinis avec des métriques de type contrôle, d'accès mémoire et de traitement de données, les fonctions visées sont du type grain épais et la recherche est statique. La solution offerte par Polimi [30] est plus précise et calcule des métriques permettant d'associer une fonction C, sans exploration de granularité, à un type particulier de processeur (DSP, RISC, ASIC). En ce qui concerne l'exploration, outre les méthodes de partitionnement logiciel / matériel qui seront mentionnées dans la section 2.3, les approches existantes reposent sur des plate-formes permettant une spécification rapide et la simulation d'architectures plus ou moins hétérogènes dans le but de réduire le temps de conception sous contrainte de consommation et de performances. Dans cette catégorie une des plus récentes et plus abouties est la plate-forme StepNP [31] qui offre un environnement de simulation complet et des perspectives pour la programmation à destination des applications orientées réseau. Une autre approche remarquable est l'environnement Metropolis développé à Berkeley [32] dont l'originalité réside dans la définition d'une sémantique précise permettant de spécifier les applications et architectures à différents niveaux de granularité à des fins de simulation principalement. Sur le même principe l'environnement SPADE [33] offre différents niveaux de raffinement à partir d'une spécification par processus de Kahn. Dans le domaine des environnements permettant de produire des courbes de compromis de type ressource / délai, la solution offerte par Platune [34]

est une exploration reposant sur le choix des paramètres pour chaque composant de l'architecture visée. L'équipe Matador de l'IMEC propose une approche intéressante pour l'exploration [35, 36] sur cible multi-processeurs. Celle-ci prend la forme de courbes de compromis correspondant à diverses solutions d'ordonnancement et de partitionnement effectuées hors ligne. La seconde phase a lieu en ligne et consiste à effectuer dynamiquement le choix de l'ordonnancement en fonction des contraintes rencontrées. Les autres solutions consistent à utiliser des outils de synthèse d'architecture pour lesquelles différentes contraintes sont à spécifier. L'approche DT se situe en amont de ces outils de conception et/ou de simulation, avant toute décision de composant ou d'outil de synthèse.

2.2.13 Conclusion

L'ambition du projet DT était de fournir un outil d'exploration offrant rapidité et flexibilité au concepteur à un haut niveau d'abstraction. Ce type d'exploration vise, avant le choix de l'architecture, la sélection de solutions d'ordonnancement et de granularité pour l'implantation d'une tâche sur cibles logicielles ou matérielles. Cette exploitation du parallélisme est générique et s'entend à la fois sur cibles multi-processeurs (e.g. architectures multi-cores), sur cibles (re)configurables ou matérielles. Il offre aussi le degré de parallélisme nécessaire à l'estimation de consommation. L'objectif est atteint du point de vue de la preuve de concept. Il demeure cependant un certain nombre de restrictions concernant le déroulage qui est limité au dernier niveau de boucle, le sous-ensemble du langage C en entrée et la hiérarchie mémoire qui devrait être affinée pour s'adapter aux niveaux de hiérarchie du graphe. La suite du projet dépend du juste équilibre à trouver entre l'effort d'ingénierie qui relève à présent à mon sens du transfert technologique et la poursuite des travaux de recherche que permet l'environnement DT notamment autour du thème de la reconfiguration dynamique. Les débouchés potentiels de ce travail se situent au niveau des *front end* des futurs outils de synthèse comportementale et d'estimation de consommation et pour l'analyse des transformations algorithmiques.

2.3 Exploration au niveau tâche, RTDT

2.3.1 Introduction

Le domaine de l'embarqué (robotique, télécoms, avionique, automobile, etc.) tire profit des progrès d'intégration au point de pouvoir concevoir à présent des SoC extrêmement hétérogènes (GPPs,DSPs, I/O processeurs). L'appétit des concepteurs croît dans des proportions au moins aussi importantes et la complexité des applications embarquées a suivi sinon précédé les besoins. Ainsi, l'intérêt de RTOS devient réel afin de gérer l'ensemble des tâches et/ou configurations à considérer. Le concepteur doit donc faire face à des exigences antagonistes relatives au domaine de l'embarqué (faible consommation et faible coût lequel inclut le temps de conception) et du temps réel (systèmes réactifs, contraintes temps réel, prévisibilité, sur-dimensionnement). L'hétérogénéité des applications ajoute encore à la complexité du problème, lorsqu'il s'agit de combiner des applications dominées par le traitement intensif (e.g. multi-média : temps réel non strict, dominé par les communications, contrôle statique) ou le contrôle (e.g. contrôle-commande : temps réel strict, contrôle dynamique, peu de communication).

Le compromis entre l'hétérogénéité des architectures potentielles et les contraintes de l'embarqué donne un sens à la spécialisation des processeurs paramétrables ou configurables au sens d'un domaine d'application donné. Ce type d'architectures s'entrevoit déjà avec la famille Virtex IV qui se décline en trois classes de FPGA qui embarquent des ressources de différents types en quantité variable selon le composant choisi. Il s'agit de blocs logiques, de blocs DSP, de mémoires dédiées, de cœurs de processeurs embarqués (PowerPC), de transceivers, de MAC Ethernet. Cette spécialisation de processeurs se retrouve également au niveau des SOC récem-

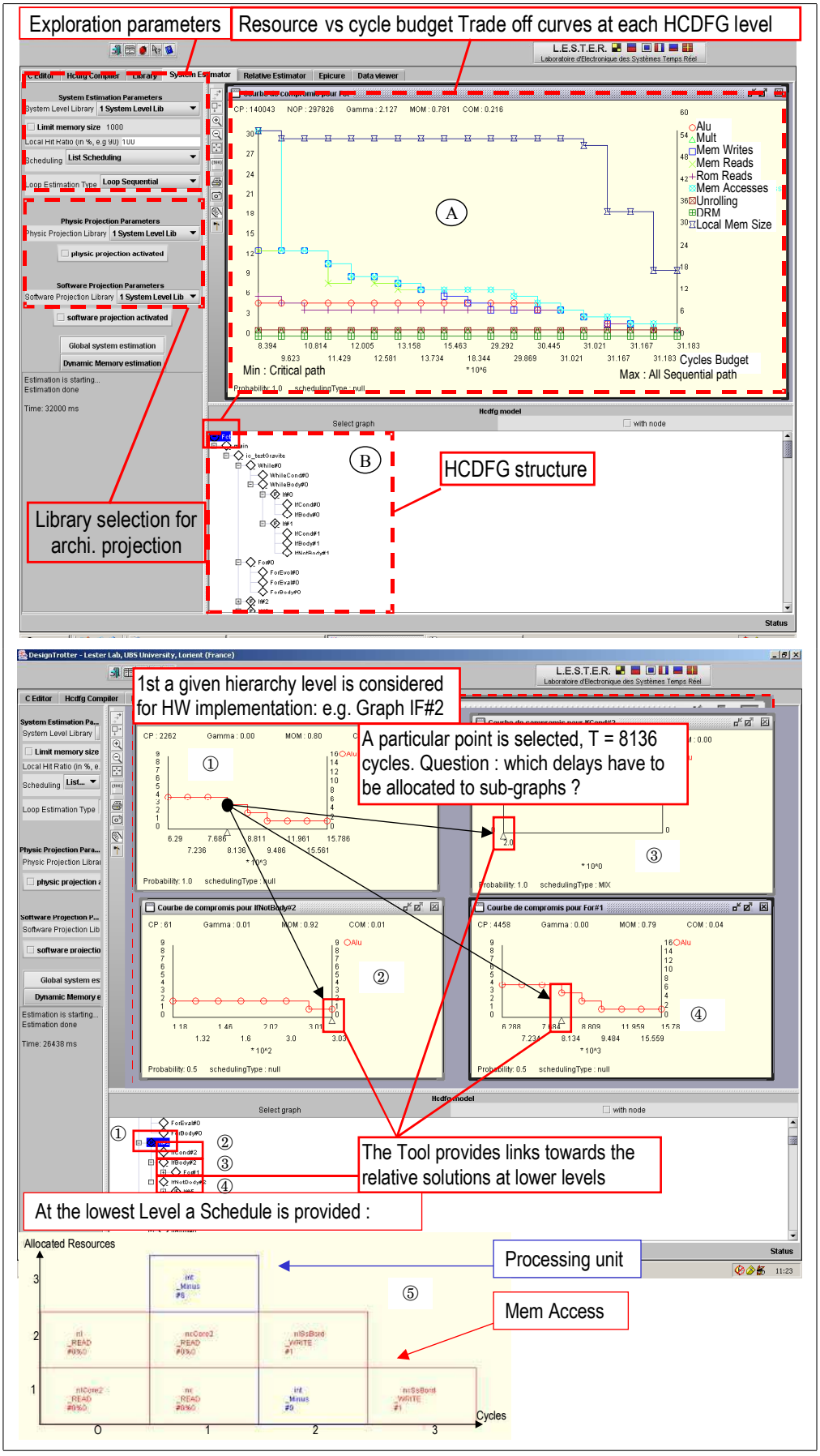


FIG. 2.8 – Outil DT, octobre 2004

ment apparus tels que ceux de la famille MPCxx de Freescale constitués de un ou deux PowerPC e600 augmentés entre autres de co-processeurs (Ativec) vectoriels. De la même manière TI propose différentes versions de DSP intégrant des accélérateurs spécialisés tels que ceux dédiés au codage canal (Turbocodes, Viterbi) dans la famille 320C64x. On citera également le STW22000 de ST qui offre l'un des plus éloquents représentant des SoC multi-formes, celui-ci intègre un processeur ARM, un DSP double MAC, un FPGA embarqué, un accélérateur dédié au décodage canal et un certain nombre de périphériques. Enfin, les architectures reconfigurables décrites dans la section 2.2.1 apportent autant de candidats à la spécialisation d'architectures.

L'environnement de conception RTDT s'inscrit dans cette logique d'adéquation application / système, mais s'appuie sur un modèle d'architecture générique (fig.2.9) qui ne fait pas le choix a priori du nombre et de la granularité de ses composants. À ce niveau l'exploration prend la forme du partitionnement de tâches matérielles ou logicielles ordonnancées suivant une approche temps réel. La première étape consiste à regrouper les tâches par *clusters* sur un modèle générique d'architectures mono-processeur. Ce travail est réalisé dans le cadre de la thèse de I.Maalej [37] par une approche génétique multi-critères. Chacune des tâches est un code C dont l'exploration architecturale est potentiellement le fruit d'une analyse effectuée à l'aide de l'environnement DT. Les différentes solutions sélectionnées lors de l'exploration au niveau HCDFG constituent un point d'entrée pour la seconde étape qui se traduit par un partitionnement logiciel / matériel. Les solutions ordonnançables sont retenues et classées suivant une fonction de coût dépendant de la consommation et de la surface. Un résumé de l'environnement d'exploration RTDT est fourni ci-après.

Outre le partitionnement proprement dit, l'objectif de l'environnement est principalement de fournir un outil d'analyse offrant au concepteur la possibilité de tester rapidement un modèle d'architecture qu'il pourra affiner afin de mesurer l'impact de ses choix sur la consommation, les performances et la surface du circuit résultant.

RTDT a été l'occasion d'expérimenter un certain nombre de modèles et techniques qu'il serait fastidieux de détailler ici. Les paragraphes suivants ont pour but d'en résumer les principales contributions.

2.3.2 Spécification de l'espace à explorer

Dérivation de contraintes

La question des contraintes est un point essentiel du problème du codesign pourtant souvent éludé. Or, les contraintes d'un système sont généralement spécifiées aux entrées et sorties indépendamment du découpage en tâches choisi par l'utilisateur, alors que le problème du partitionnement nécessite généralement d'imposer des contraintes de temps aux différentes tâches. Ensuite un système temps réel embarqué peut nécessiter des tâches de nature très hétérogène : sporadique et périodique de période variable. Ce dernier point est directement lié à la granularité retenue pour l'exécution du traitement (ex. Image, Ligne, colonnes cf fig.2.11). Nous avons appréhendé ce travail en collaborant avec A.Dasdan (UC Irvine) qui a réalisé l'outil Radha-Ratan [38] dont le rôle est de dériver des contraintes d'entrées / sorties en contraintes pour chacune des tâches du graphe. Ainsi nous disposons pour chacune d'un intervalle $[T_{min}, T_{max}]$ correspondant aux délais extrêmes entre deux activations de la tâche. En prenant T_{min} comme période, l'utilisateur choisit par exemple un respect strict des contraintes de temps et périodise de fait les tâches apériodiques. Les tâches ne nécessitant pas le respect strict du temps réel sont affectées à une tâche serveur de priorité minimale. En ce qui concerne les tâches ayant une contrainte stricte, le pire cas est généralement considéré comme la seule solution garantissant le temps-réel. Cependant la conjugaison des pires cas au niveaux des contraintes et des temps d'exécution (WCET) peut entraîner un sur-dimensionnement en contradiction avec les contraintes des sys-

tèmes embarqués. Aussi, nous avons introduit dans le flot de codesign la notion de QoS au sens du respect du temps pour modérer l'impact de ce type de choix (cf. section 2.3.5).

Spécification générique des solutions

Nous l'avons vu précédemment au niveau HCDFG, le choix de la granularité permet ou non de tirer profit du parallélisme disponible. Dans le cas d'une implantation matérielle, ce choix se traduit par un compromis à déterminer entre la taille de la mémoire locale et le gain apporté par des communications de type *burst*.

Ainsi, nous avons choisi un modèle d'entrée, (fichier .cde sur la figure 2.10) qui permet d'une part de décrire le graphe de tâches où les nœuds sont les tâches et les arcs les dépendances de données, et qui d'autre part, inclut le détail des différents types d'implantation logiciels avec ou sans co-processeur et d'implantions matérielles avec plusieurs niveaux de granularité. Ainsi, le fichier de spécification apporte trois types d'information, il précise les dépendances de données ou de contrôle entre les tâches, ainsi que le partage de ressources critiques. Ensuite il fournit les quantités de données échangées entre les tâches ainsi que les contraintes temps réel issues de la dérivation. Enfin il permet d'entrer les caractéristiques temps, surface, consommation pour chaque solution candidate retenue lors de l'exploration intra tâche. Concernant ce dernier point, les données fournies sont relatives et dépendent des grandeurs indiquées dans le fichier de spécification de l'architecture (.arch).

La spécification de l'architecture précise les grandeurs relatives au choix des ressources matérielles pour différents modes de fonctionnement. Ainsi, pour le processeur, le concepteur indique différents paramètres comme le délai de changement de contexte, les fréquences du processeurs, du bus et celle du matériel dédié, les tensions d'alimentations, la puissance statique par unité de surface, la puissance des accès aux mémoires, le coût en surface et en consommation d'un point mémoire etc. Autrement dit le fichier .arch fournit les différents paramètres nécessaires pour transformer les spécifications exprimées sous la forme de valeurs relatives en grandeurs physiques.

2.3.3 Méthode d'exploration

Algorithmes d'exploration

Deux méthodes de parcours ont été implantées, la première est exacte et du type *Branch and Bound*. La seconde est heuristique du type *Simulated annealing* et se révèle nécessaire à partir d'une vingtaine d'implantations cumulées sur l'ensemble des tâches. Le concepteur, outre les modes spécifiés dans le fichier .arch, dispose d'un certain nombre de paramètres. Il s'agit de la balance entre la surface (S) et la consommation (P) pour la fonction de coût Fc ($Fc = \alpha S + \beta P$), les paramètres du recuit simulé, la méthode de calcul d'ordonnançabilité (exacte ou simple *rate monotonic*) et la part du temps CPU attribué à la tâche serveur.

Prise en compte des communications et mémoires

Les communications sont traitées en insérant un nœud supplémentaire dans le graphe pour toute communication entre un accélérateur matériel et le processeur. Ces tâches, comme les autres, sont traitées suivant un modèle multi-cadences rendu nécessaire d'une part par la périodisation aléatoire des tâches sporadiques et d'autre part en raison de l'exploration des différents niveaux de granularité. La figure 2.12 illustre le principe de l'insertion de tâches de communication par un exemple et la figure 2.11 montre la nécessité des cadences multiples.

La taille de la mémoire est calculée en fonction de la quantité de données transmises pour le niveau de granularité traité. Le modèle permet de prendre en compte des cadences de tâches différentes par l'analyse de l'ordonnancabilité sur une hyper-période.

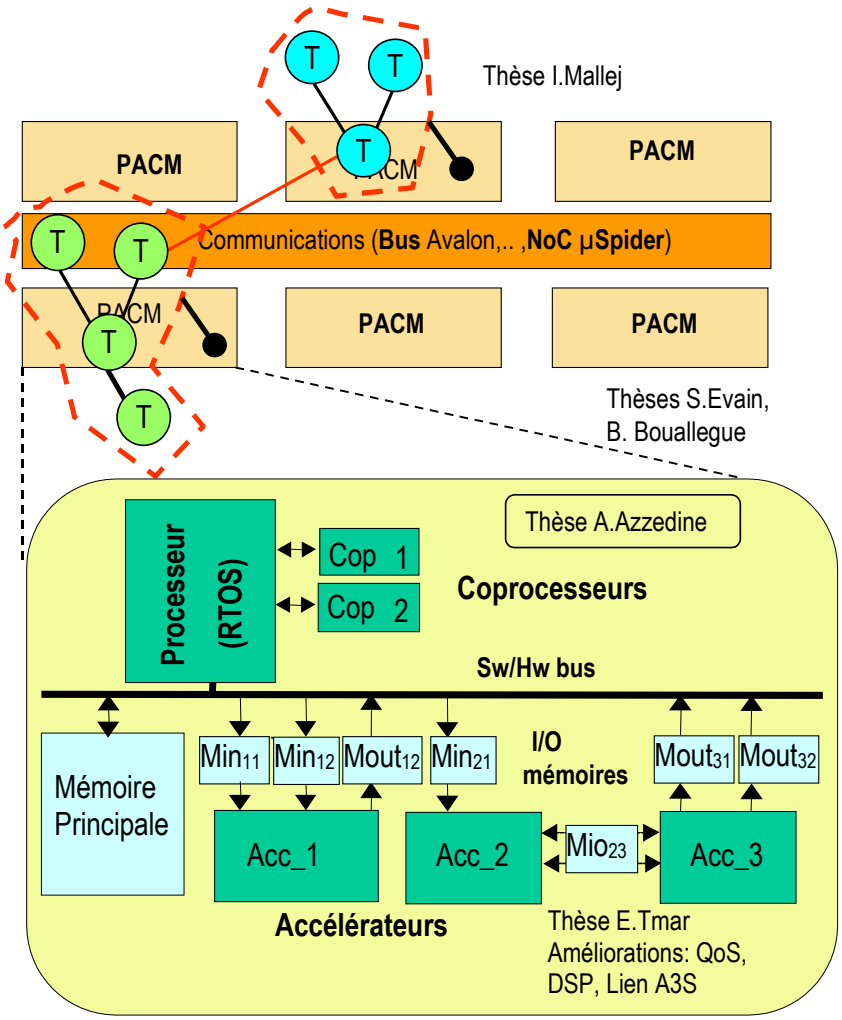


FIG. 2.9 – Architecture multi-PACM

Ordonnançabilité

Compte tenu du domaine d'application visé, à savoir les systèmes temps-réel embarqués, le choix s'est naturellement porté sur un ordonnancement statique de type HPF *Hign priority first* tel qu'on le trouve dans la plupart des RTOS du domaine (μCOSII, eCOS, RTLinux). La priorité des tâches est calculée comme l'inverse de leur période. L'ordonnançabilité est effectuée par une analyse RMA *rate monotonic analysis* suivie, en cas de succès, d'un calcul exact du temps de réponse. La première méthode est analytique et donc rapide mais entraîne une surestimation, la seconde est précise mais itérative et donc plus lente. Le concepteur peut choisir de la désactiver si une analyse grossière suffit. Les deux techniques supposent une indépendance des tâches, mais la réalité est différente puisqu'il existe des contraintes de dépendances, des ressources partagées ainsi que des délais introduits lors des changements de contexte. La question est résolue d'une part en calculant des dates de réveil et d'autre part en utilisant une hyper-période qui permet de considérer les itérations des différentes tâches comme indépendantes. Par la suite, si la condition nécessaire de l'analyse RMA est respectée, le temps de réponse est calculé comme suit en tenant compte du temps de blocage par des tâches de priorité plus élevée ou moins prioritaires mais

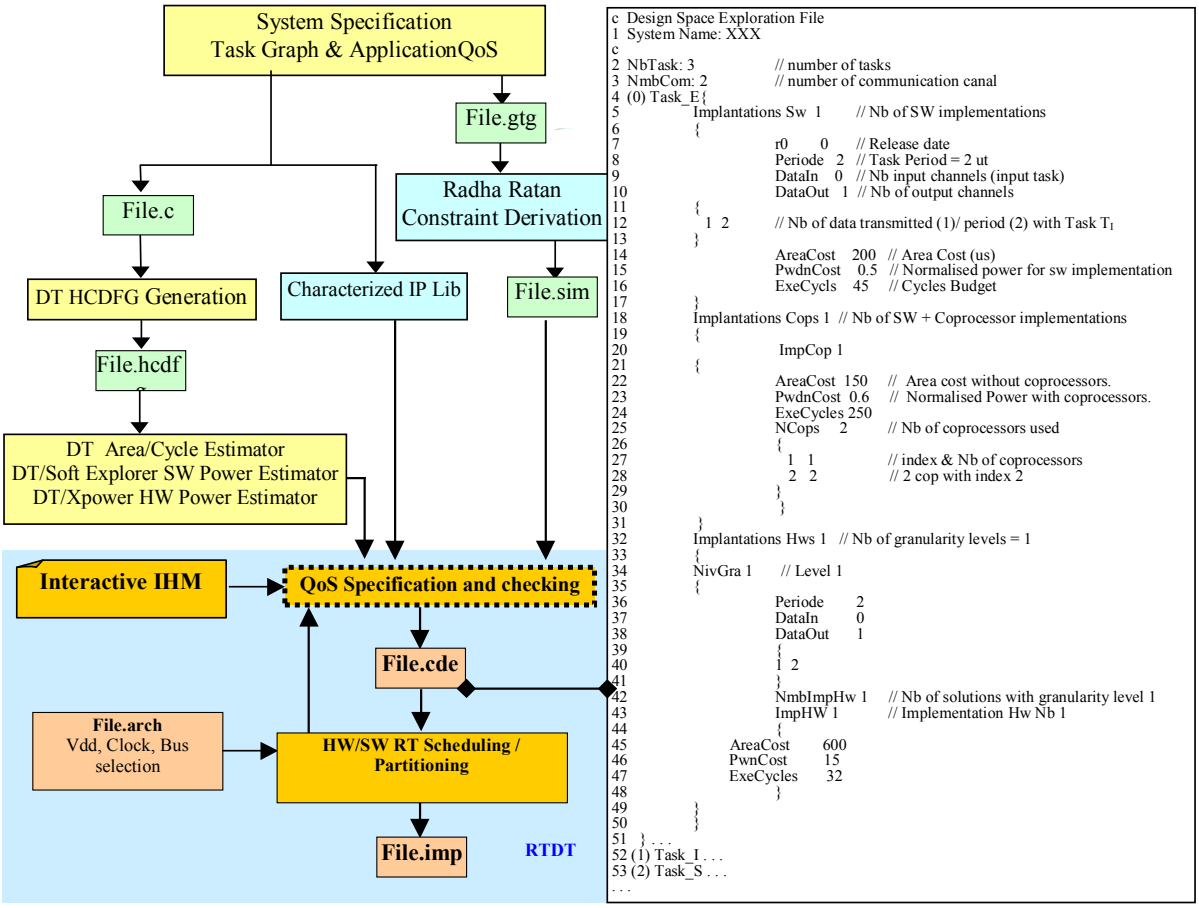


FIG. 2.10 – Flot RTDT

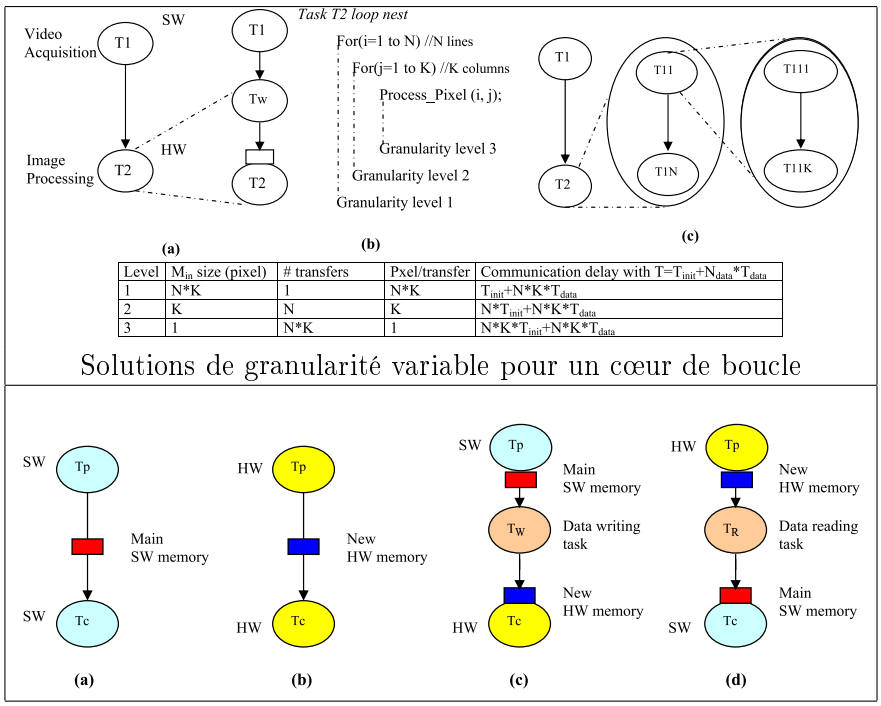
ayant pris le contrôle d'une ressource critique :

$$\forall T_j \in HP(i), \exists R_i \leq D_i \backslash R_i = (C_i + B_i) + \sum_{j \in HP(i)} \left\lceil \frac{R_i}{P_j} \right\rceil * (C_j + C_{sw}) \tag{2.3}$$

avec :

- $HP(i)$: Ensemble des tâches plus prioritaires que T_i ;
- R_i : Temps de réponse au pire cas de de T_i ;
- C_i : Temps d'exécution de T_i ;
- B_i : Retard maximum dû à une ressource partagée ;
- P_j : Période de T_i
- $C_{sw} = \delta_0 + \sum_k \delta(k)$ avec δ_0 le délai de changement de contexte sans co-processeur et δ_k le délai additionnel dû à l'ajout du co-processor k.

Fonction de coût et modèle de consommation Le modèle de coût utilisé pour classer les solutions est un compromis entre la surface et la consommation d'énergie. L'aspect surface pour les co-processeurs, accélérateurs matériels et mémoires est classique, par contre le modèle de consommation propose une approche détaillée que le concepteur peut renseigner plus ou moins finement en fonction des informations dont il dispose. Le sujet de la consommation est vaste et, à l'instar de la question de l'exploration, se décline à différents niveaux depuis le niveau transistor avec le contrôle fin des fréquences d'horloge et tensions d'alimentation (puissance



Level	M_{in} size (pixel)	# transfers	Pxel/transfer	Communication delay with $T=T_{init}+N_{data}*T_{data}$
1	N*K	1	N*K	$T_{init}+N*K*T_{data}$
2	K	N	K	$N*T_{init}+N*K*T_{data}$
3	1	N*K	1	$N*K*T_{init}+N*K*T_{data}$

FIG. 2.11 – Implantation des mémoires pour différents schémas de communication

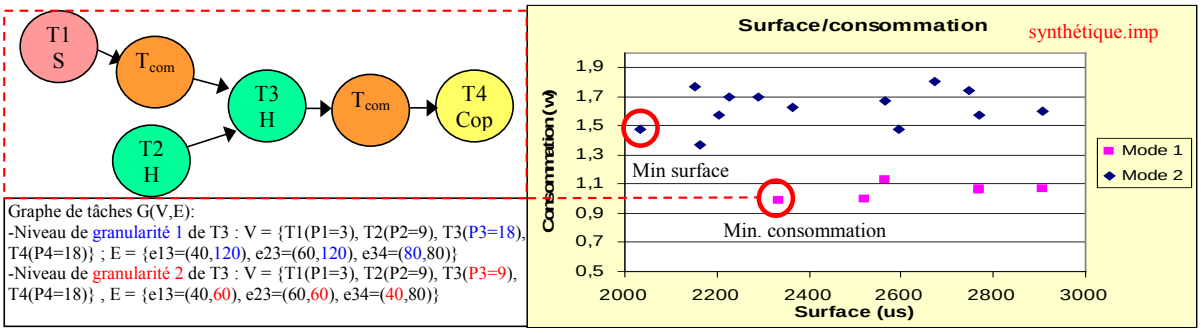


FIG. 2.12 – Exemple synthétique

dynamique) et de la polarisation du substrat (puissance statique) jusqu'au niveau de l'ordonnancement de tâches dans un système multi-processeurs. Compte tenu du niveau d'abstraction et des phénomènes aléatoires (dates d'activation des tâches, jitter, délais d'exécution et de communication), il est illusoire de penser traiter la question de manière extrêmement précise, nous avons à ce niveau besoin de comparer différentes solutions entre elles. Ce constat sur l'incertitude des décisions effectuées lors de la conception est une tendance lourde dans le domaine des systèmes embarqués, le chapitre 4 présente une réponse que nous proposons pour traiter ce problème à l'aide d'architectures dynamiquement reconfigurables. Par ailleurs, le problème général de la consommation des circuits sous-entend plusieurs aspects inter-dépendants : la puissance moyenne, la puissance crête, l'énergie, la température.

Dans notre approche les tâches sont périodiques ou périodisées, ainsi minimiser la puissance moyenne sur l'hyper-période est équivalent à traiter la question de la minimisation de l'énergie consommée. L'environnement RTDT permet à l'utilisateur de choisir les paramètres influençant la consommation (différents modes Vdd/Fréquence Horloge, puissance statique par unité de

surface, etc.), l'objectif est de mesurer l'impact des choix technologiques.

$$Pw(S) = Pw_d + Pw_s \tag{2.4}$$

avec : Pw_d la consommation dynamique moyenne pendant l'hyper-période T_G et Pw_s la consommation statique moyenne. En résumé le modèle est le suivant pour la puissance dynamique : $Pw_d = \frac{E_d(sw)+E_d(hw)}{T_G}$ avec :

- $E_d(sw) = E_d(idle) + E_d(switch) + E_d(exe)$
- $E_d(exe) = T_G \sum_{i \in SW} P_{wd}(i) \frac{C_i}{P_i}$ avec $P_{wd}(i)$ la puissance dynamique moyenne pour la Tâche i.
- $E_d(switch) = T_G\, P_{wd}(switch) \sum_{i \in SW} \frac{C_{sw}}{P_i}$ avec $P_{wd}(switch)$ la puissance moyenne du changement de contexte.
- $E_d(idle) = P_{wd}(idle) T_G \left(1 - \sum_{i \in SW} \frac{C_i + C_{sw}}{P_i}\right)$ où $P_{wd}(idle)$ est la puissance dynamique moyenne du processeur inoccupé.
- $E_d(hw) = T_G \sum_{i \in HW} P_{wd}(i) \frac{C_i}{P_i}$

et pour la puissance statique : $Pw_s = f\left(N_{tr} K_{design} I_{leakage} V_{dd}\right)$

Le modèle utilise $Pw_s(sw)$ et $Pw_s(hw)$ pour les parties logicielles et matérielles. Si une stratégie de mise en veille est utilisée alors $Pw_s = Pw_{offSW} * Area(SW) + Pw_{offHW} * \sum_{i \in HW} Area(i) \frac{C_i}{P_i}$. On notera que la prise en compte de la puissance statique implique que les grandeurs consommation et puissance ne sont indépendantes.

2.3.4 Mise en œuvre

Le logiciel RTDT a été développé dans le cadre de la thèse d'A.Azzedine [39]. Les applications test sont issues de domaine de l'automobile [40], de la robotique [41] et des télécoms (projet A3S). La principale difficulté rencontrée à été la mise en œuvre concrète d'applications. En effet, ce travail s'insère dans un cadre général de conception de SoC, il nécessite également un certain nombre de données concernant la consommation statique et l'*overhead* du système d'exploitation. Nous sommes partis du principe que ces données étaient disponibles et issues des différents outils développés par exemple au LESTER : de l'exploration au niveau fonction (DT), de la synthèse d'architecture (Gaut), de l'estimation de consommation (soft explorer), de la modélisation UML de systèmes (projet A3S). Or, en pratique chacun sait l'effort de développement que requière la mise en place d'un tel environnement cohérent qui plus est au sein d'une structure académique. Nous avons avancé en supposant disponibles les passerelles encore manquantes au sein de l'espace de conception. La figure 2.13 présente, pour l'application robotique, l'évolution des solutions en fonction de la fonction de coût. L'utilisation de modules matériels dédiés (plus efficace d'un point de vue énergétique) à des niveaux de granularité différents, permet d'améliorer la localité des données et de réduire le taux d'utilisation du processeur. L'effet observé est une réduction de la consommation qui s'accompagne d'une augmentation de la surface. L'effet de la consommation statique atténue le gain en consommation. L'impact grandissant de la consommation statique est à l'origine d'importants efforts de recherche aboutissant au contrôle dynamique de la polarisation du substrat [42].

2.3.5 Travaux en cours et perspectives

Approche probabiliste de la QoS : Analyse statique en cours

Le domaine des systèmes temps réel embarqués impose des contraintes antagonistes entre le temps de conception réduit et l'optimisation de la consommation d'énergie. Par ailleurs seul un ordonnancement statique permet de garantir le temps réel de manière simple mais en considérant le WCET comme temps d'exécution pour chaque tâche. Or, ce pire cas peut s'avérer en pratique

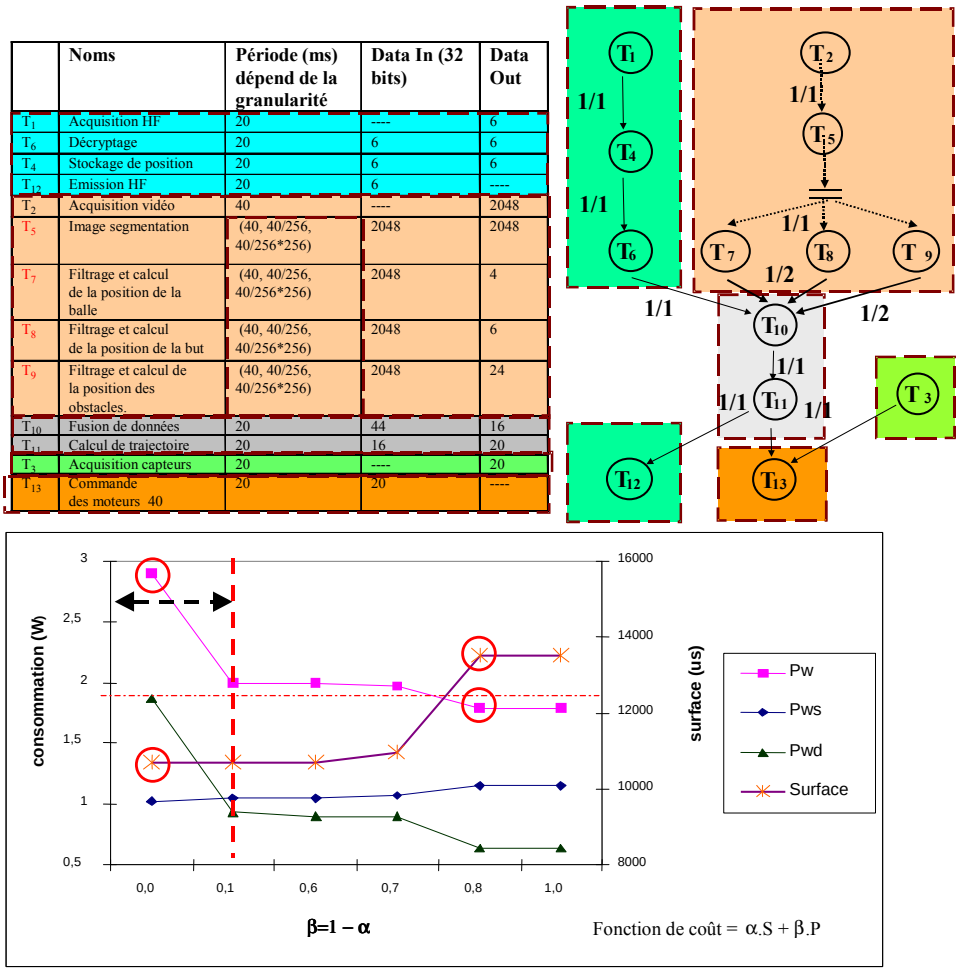


FIG. 2.13 – Évolution du compromis Surface Énergie pour Application Robot "footballeur"

très fluctuant et se traduire par un sur-dimensionnement du système. Cette variabilité des temps d'exécution (et de la puissance consommée) s'accentue avec la complexité des architectures (caches, gestion dynamique des alimentations, NoC, etc ..) et l'hétérogénéité des applications qui pour des raisons d'optimisation intègrent des techniques d'adaptation du traitement aux données (ex. estimation de mouvement en spirale) et à l'environnement (niveau de bruit, luminosité, vitesse, etc.). Cette dépendance du temps de calcul en fonction des données et de l'environnement tend à rendre prohibitif l'approche au pire cas dans le contexte des systèmes embarqués et conduit à transformer la question du temps réel en gestion de la qualité de service. Un tour d'horizon complet de cette problématique au niveau des RTOS est fourni par le projet ARTIST [43].

Au vue de ce constat, nous avons orienté le projet RTDT vers une approche probabiliste dans le cadre de la thèse de H.Tmar (co-tutelle SFAX). L'approche analytique qui consiste à établir une loi exprimant le temps d'exécution en fonction de paramètres applicatifs (ex. contenu d'une image) a été abandonnée pour les raisons qui suivent. Tout d'abord elle nécessiterait une étude systématique et laborieuse des paramètres pertinents, ensuite les temps d'exécution des différentes tâches sont généralement interdépendants. Enfin elle ne permettrait pas d'aboutir, dans un cadre de SoC complexe, à une précision suffisante en raison des fluctuations inévitables liées à l'architecture (ex. contenu du cache, délai de traversée d'un NoC etc.). Ainsi le principe adopté est le suivant, chaque tâche est caractérisée non plus par une valeur WCET mais par

une loi exprimant la probabilité d'obtenir un temps d'exécution resp. une puissance consommée inférieure à une valeur donnée [1]. Un module intermédiaire (*Static QoS mamager*) est inséré entre l'étape de spécification et le partitionnement logiciel/materiel spécification, celui-ci permet à l'utilisateur de choisir le niveau de QoS souhaité pour chaque dimension du tableau associé à la tâche i : $TQoS(i)[x_1, .., x_n]$. Deux dimensions sont pour l'heure prises en compte dans la cadre du traitement d'applications vidéo, il s'agit de la cadence ($x_1 = AQoS$) c'est à dire la contrainte temps réel imposée et de $x_2 = RTQoS$ représentant la probabilité de respect de la contrainte. L'outil effectue deux types de traitement, le premier est une vérification de cohérence des QoS demandés en fonction des dépendances de données, des ressources partagées et de la priorité des tâches. Le second est une analyse originale après partitionnement/ordonnancement de la fiabilité de l'architecture (*Scheduling Safety Rate*) vis à vis du temps réel. L'outil retourne au concepteur la probabilité que le dépassement des contraintes de temps réel soit supérieur au temps accordé à la tâche serveur de plus faible priorité, autrement dit la probabilité que la tâche serveur ne puisse absorber le sur-coût de traitement. En fonction des résultats l'utilisateur décide du compromis qu'il juge acceptable. La figure 2.14 illustre des résultats préliminaires obtenues avec l'application "robot footballeur" 2.14. On observe que relâcher quelque peu les contraintes au niveau du temps réel induit des gains considérables en termes de surface et de consommation (75% de probabilité de respect de la contrainte du temps réel permet de réduire la consommation de moitié). Ce type d'analyse offre de manière rapide les différentes options possibles, libre ensuite au concepteur de choisir le compromis le plus adapter à la situation en toute connaissance de cause. Ce type d'information, dont le concepteur est actuellement privé, constituerait un indéniable un gain de temps et une meilleure justification des choix finalement entérinés.

Approche probabiliste de la QoS : Traitement dynamique en perspectives

Dans un second temps l'objectif est d'effectuer dynamiquement le choix du partitionnement logiciel / matériel en fonction de l'état du système. Cet état est mis à jour par une tâche d'observation qui récupère les grandeurs nécessaires telles que le temps d'exécution courant des tâches, le niveau de la batterie plus un certain nombre d'informations spécifiques aux applications traitées. Nous avons dissocié les décisions correspondant au niveau local de celles relevant des décisions globales. Cet aspect est au cœur du projet ACI RaaR consacré aux architectures intelligentes et détaillé dans le chapitre 4.

Nous allons ajouter de plus un paramètre déterminant dans le cadre des architectures embarquées, celui de la consommation induite. En effet dans une application vidéo sans fil par exemple, le processeur en tant que tel influence relativement peu sur la consommation en regard de la caméra et du module Wifi, par contre les tâches exécutées par le processeur induisent un usage des I/OS vidéo et HF. L'objectif est de lier la consommation du système à la qualité de service de l'application dans le cas où celle-ci n'est pas contrôlée mais flexible.

Lien avec les spécifications UML du profile A3S-Software Radio

Projet A3S Le projet RNRT A3S est consacré à la modélisation et au prototypage d'applications de radio-logicielle selon une approche MDA (*Model Driven Architecture*). Dans ce projet, nous avons enrichi le méta-modèle UML pour la radio-logicielle[2], celui-ci est intégré dans l'outil Objecteering de Softeam partenaire du projet. Les extensions ont eu comme objet la définition d'un profil permettant de prendre en compte les contraintes liées au ressources matérielles. La figure 2.15 présente le flot. La thèse de S.Rouxel, encadré par G.Gogniat, J-L.Philippe et moi-même suivant les aspects abordés, est en partie consacrée à la mise en œuvre

[1] Cette loi est obtenue consécutivement à une campagne de mesures ou de traces sur la base de benchmarks
[2] UML profile for Software Radio

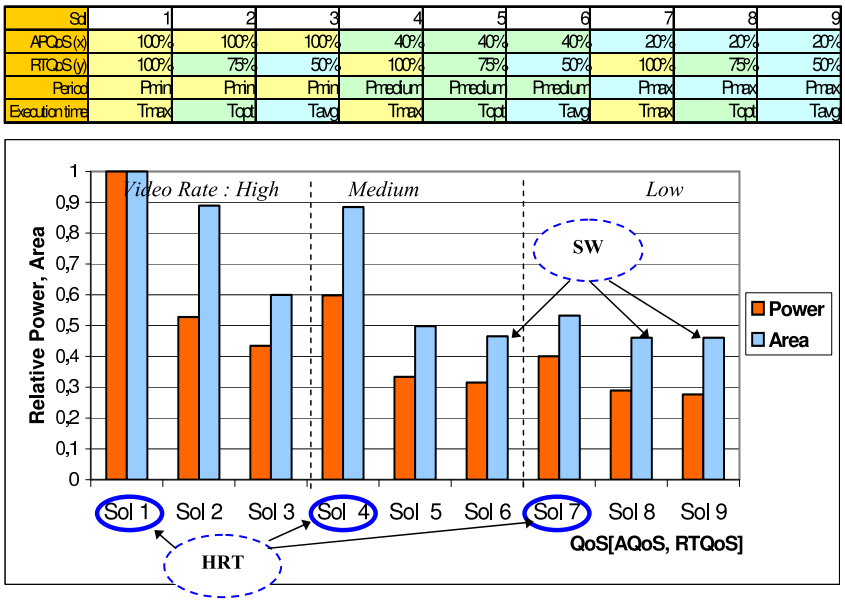


FIG. 2.14 – Compromis QoS, Consommation, Surface

de ce flot. La première étape est une représentation sous forme de diagramme d'activités indépendant de l'architecture. La seconde étape consiste à spécifier l'architecture sous la forme de *hardware components*. La troisième phase consiste à déployer des *software components* sur la plate-forme précédemment décrite. Les étapes 2 et 3 offrent au concepteur une aide sous la forme de vérifications de contraintes (format, connexion, espace mémoire, type de données etc.) Enfin la quatrième étape se traduit par une nouvelle série de vérifications et par la génération des fichiers d'entrée .cde et .arch de l'outil RTDT permettant de procéder automatiquement une analyse d'ordonnançabilité et de consommation. Il est à noter que la méthode de dérivation de contraintes à l'origine de l'outil Radha-Ratan a été adaptée et intégrée dans l'environnement A3S. Les plate-formes cibles sont des cartes Pentek à base de DSPs et de FPGA et l'application fournie par Mitsubishi est une chaîne de traitement UMTS.

Adaptation de RTDT Cette interface avec RTDT a occasionné un certain nombre de modification dans l'outil RTDT. Nous avons du transformer la notion d'accélérateur afin qu'elle puisse être multi-tâches et intégrer les composants de type DSP. Nous pouvons à présent viser des architectures telle que l'OMAP ou le STW22000. Nous avons également enrichi les modes de communication par des FIFO, enfin l'analyse des échanges de données à été améliorée afin notamment de prendre en compte les données rémanentes entre deux exécutions d'une même tâche (cas typique d'un filtre numérique avec fenêtre glissante). Depuis le début du projet nous présentons régulièrement l'avancement de l'outil dans le cadre du SDR Forum où notre approche reçoit un accueil favorable et attentif. À titre d'exemple les deux objectifs principaux mis en exergue par les groupes de travail en conclusion du forum étaient : "modélisation unifiée" et "formalisation du lien vers les plate-formes matérielles". Cette année l'ensemble du flot sera présenté [44]. Les extensions à venir traiteront de la gestion dynamique des configurations et des interactions avec le RTOS.

2.3.6 Positionnement dans le domaine du codesign

À l'origine du projet RTDT en 2000, l'objectif était de proposer un outil de partitionnement logiciel / matériel, dans un contexte de systèmes temps-réel, adapté aux cœurs configurables (ac-

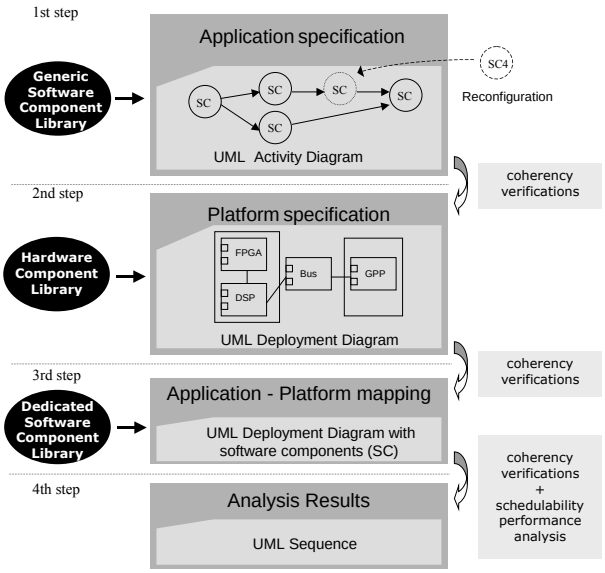


FIG. 2.15 – Flot de conception A3S

célérateurs, co-processeurs) émergeants dotés d'un RTOS. Outre l'aspect QOS, l'originalité des travaux repose sur un traitement simultané des points suivants que l'on ne trouve généralement pas réunis au sein d'un unique environnement.

- Exploration de différents degrés de granularité (co-processeurs et accélérateurs à différents niveaux des nids de boucles) ;
- Traitement de tâches apériodiques ;
- Analyse d'ordonnancabilité ;
- Pise en compte de l'*overhead* du RTOS ;
- Modélisation des tâches de communications ;
- Analyse de dépendance multi-cadence ;
- Modèle d'architecture générique pour l'estimation (délai, consommation et surface) ;
- (Suite au projet A3S, l'interface avec les UML a été intégrée).

Des méthodes incluant co-processeurs et accélérateurs ont été proposées dans [45] avec un modèle de spécification de type CDFG. L'allocation de tâches multi-cadences est traitée dans [46] sur une cible mono-processeur avec co-processeurs et dans [47, 48, 49] les techniques proposées adressent l'ordonnancement de tâches multi-cadences sur cible multi-processeurs. Les deux cas de figure considèrent des clusters indépendants de tâches cadencées avec une période unique. Le traitement de la granularité variable (hormis la distinction accélérateur / co-processeur) a été exploité dans [50, 51] sans cependant d'analyse approfondie des conséquences sur l'ordonnancement des tâches de communication. Enfin, l'aspect consommation (puissance, énergie, température) a pris peu à peu l'ascendant sur les optimisations en surface et cela s'est traduit par son intégration dans les modèles de coût [47]. L'ordonnancement tire à présent profit de la gestion statique et dynamique du couple (Fréquence d'horloge, Vdd) [52, 53]. Concernant la consommation l'incertitude des valeurs pose un problème sérieux comparable à celui du WCET rencontré pour l'ordonnancement. En effet l'incertitude sur les dates de démarrage des tâches rend potentiellement caduque un ordonnancement a priori. Il en est de même pour les mesures de consommation rendues périlleuses en raison des techniques employées sur les processeurs récents pour réduire la consommation (ex. Pentium M : gestion dynamique des alimentations du cache et du bus et sélection automatique du couple Freq/Vdd). Un ordonnancement dynamique semble

être la solution mais il doit dès lors disposer d'une vue globale de l'évolution du système sans induire un overhead au niveau performance et consommation. Notre approche initiale a été de fournir un cadre permettant de modéliser cette consommation en fournissant les paramètres nécessaires, ensuite nous avons évolué vers le traitement probabiliste du partitionnement. L'étape en cours nous mène aux architectures intelligentes capables de décider de leur configuration à faible coût.

Chapitre 3

Réseaux et NoC

3.1 Introduction

Les entrées sorties, au sens général du terme, ont une influence déterminante sur les performances d'un système embarqué, j'ai abordé cette question suivant un certain nombre de travaux complémentaires. La première approche visait à concevoir un ASIP de type cœur de processeur dédié au protocole IP et au cryptage. Ce travail a débuté en 1999 dans le cadre la thèse de S.Montfort. J'ai, par ailleurs, participé à l'encadrement de la thèse de B.Boualllegue (00/05), étudiant inscrit en co-tutelle UBS/Faculté des Sciences de Monastir (Tunisie). L'objectif de la thèse était une approche unifiée de la synthèse d'interface entre un réseau (ATM ou *Network On Chip*) et une application multi-média (MPEG2). Les critères d'optimisation privilégiés ont été le respect des contraintes temps réel et la qualité de service au sens de l'impact des pertes de paquets. Enfin, depuis 2003 j'encadre la thèse de S.Evain portant sur la réalisation d'un outil de CAO pour la synthèse et le dimensionnement de NoC reconfigurables.

3.2 IP Processeur réseau embarqué

3.2.1 Principe

La mise en œuvre des protocoles de communication de type TCP/IP se traduit par des traitements, des tests et accès mémoire effectués au niveau bit, ils sont de plus caractérisés par le maintien et l'évolution de machines d'états. Les architectures de processeurs classiques sont peu adaptées à ce type de traitement qui impliquent de fréquents accès à la mémoire ne serait-ce que pour la mise à jour de simples *flags*. D'importants gains en termes de performances et d'énergie sont envisageables avec un effort d'optimisation de l'adéquation application / architecture. En partant de ce constat, l'objectif de la thèse était en premier lieu d'extraire les instructions spécifiques au protocole TCP/IP et ensuite de proposer une architecture flexible permettant de traiter les configurations courantes du protocole afin de proposer un ASIP capable de décharger le processeur embarqué de la gestion des protocoles réseaux (couches logicielles 3 et 4 dans un système UNIX). Il est à noter que Intel sortait en décembre 1999 le premier de la famille IXP de *network processor* pour des applications de type serveur, notre approche ne suivait pas la même optique puisque nous visions spécifiquement le domaine de l'embarqué qui n'est évidemment pas sujet au même type de contraintes.

3.2.2 Réalisation

La première étape a abouti à l'extraction des instructions pertinentes [54] ainsi qu'à un modèle d'architecture originale disposant d'unités dédiées adaptées au type de traitements rencontré. Cette extraction, sur le modèle de DT, a été précédée d'une étape de caractérisation

permettant de mettre en lumière le poids relatif du contrôle, du traitement, des accès mémoires (données et variables de contrôle). La figure 3.1 illustre l'analyse des principales fonctions du protocole TCP et propose la matérialisation d'un composant spécifique dédié aux différentes modes de la fonction *Established*. Des opérateurs dédiés au cryptage (blowfish et DES) ont également été intégrés au modèle. Ce travail a hélas été interrompu suite à l'obtention par le candidat d'un poste d'ingénieur de recherche en région parisienne. Ma charge de travail en tant que maître de conférence à l'époque ne m'a pas permis d'en poursuivre le développement.

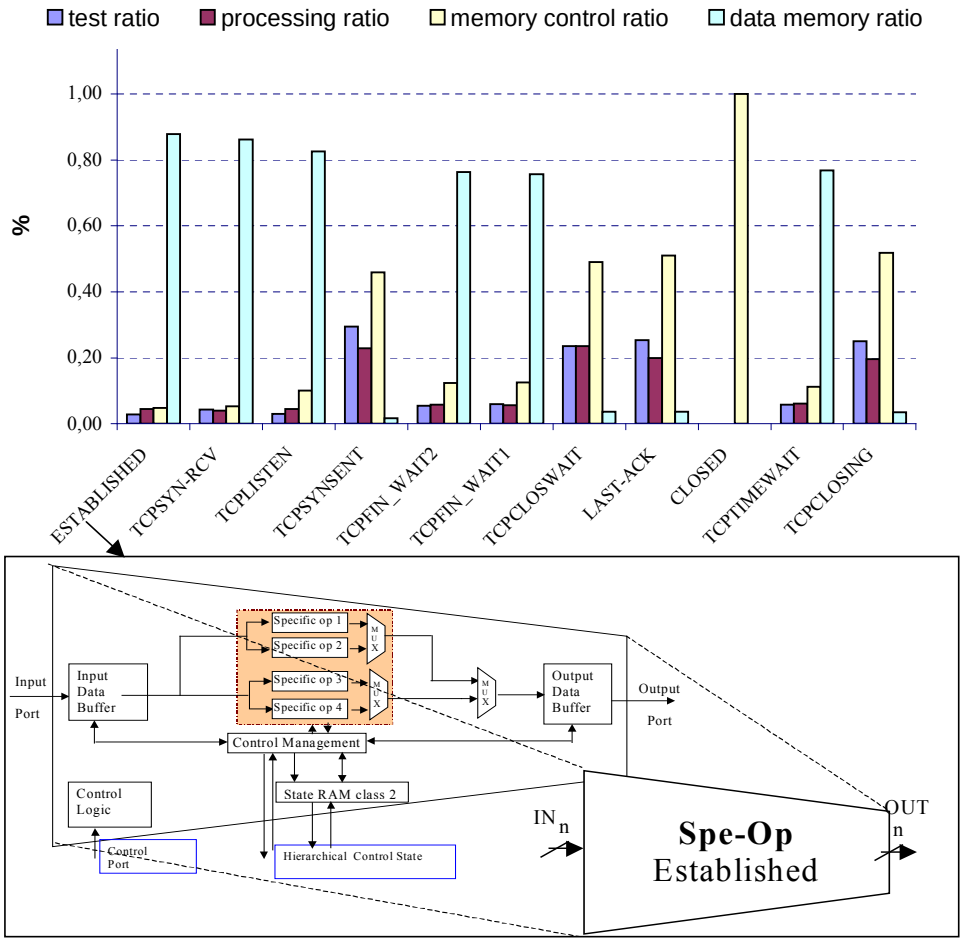


FIG. 3.1 – Caractérisation du protocole TCP

3.3 Architecture Interface Réseau / MPEG

3.3.1 Objectif

Le domaine visé est celui des protocoles de communications offrant une bande-passante négociée pour le transfert de flux multi-média. Deux cas de figure ont été considérés. Le premier et le plus abouti est le protocole ATM. Le second vise un *Network On Chip* ad hoc. Différents types d'erreurs de transmission peuvent intervenir dans un tel contexte et se traduire par une dégradation de la qualité du flux multimédia. Deux sources d'erreurs ont été traitées, la première est la perte de paquets qui peut frapper les données utiles ou les en-tête de paquets avec une probabilité beaucoup plus faible. La seconde est due aux fluctuations de la bande passante pouvant entraîner un engorgement au niveau de l'interface et une perte de données en raison

des capacités mémoire limitées.

3.3.2 Réalisations

Démonstrateur MPEG / ATM / MPEG Les travaux effectués se situent à trois niveaux différents :

1. Stategie d'encapsulation du flux MPEG ;
2. Contrôle de flux pour le traitement de l'engorgement ;
3. Gestion dynamique de la mémoire.

Ces trois points ont été publiés notamment dans [55, 56], le journal électronique [57] fournit une synthèse accessible en ligne.

La première tâche à consister à proposer une stratégie d'encapsulation des trames MPEG au sein des cellules ATM. De base la couche d'adaptation AAL5 ne prévoit pas de gestion de la qualité de service en cas de variation brutale du débit. Quelques techniques ont été proposées comme notamment l'affectation de priorités aux différentes trames après analyse [58]. L'approche proposée AAL5+ est adaptée à des applications de type VOD *video on demand* qui ne supportent pas la réémission de paquets en cas d'erreur et imposent l'emploi d'une méthode de correction. La stratégie implantée est la suivante. Contrairement à la méthode, habituelle l'encapsulation n'est pas effectuée au niveau macro-bloc MPEG mais consiste à composer en premier lieu un paquet élémentaire (PES) pour chaque image compressée. Celui-ci est ensuite segmenté en blocs (TS) de 188 octets organisés par groupes de trois au sein d'un paquet SSCS. Un en-tête de quatre bits est associé à chaque groupe, celui-ci correspond à un numéro de séquence codé et dispose d'un bit de parité. En fin de paquet un code reed-salomon est ajouté. Cette technique décrite sur la figure 3.2 est mise en œuvre conjointement avec le contrôle de trafic.

Le second point est l'usage de l'entrelacement pour lisser l'effet de la perte de cellules ATM, due à l'engorgement ou à une erreur de transmission, sur la qualité globale des images reçues. Cet entrelacement est effectué au niveau de la source MPEG avant la constitution de cellules ATM de 48 octets. Les paquets SSSC sont organisés en un tableau de N lignes par M colonnes qui est écrit suivant les lignes et lu suivant les colonnes. Différentes configurations ont été testées, la solution N=M=48 a finalement été retenue. Celle-ci offrait un bon compromis entre complexité (notamment taille mémoire) et efficacité. À la réception, les paquets sont écrits suivant les colonnes et lues suivant les lignes.

Le troisième aspect est le contrôle de trafic associé à la gestion dynamique de la mémoire. L'objectif est d'une part d'adapter la taille de la mémoire allouée aux différentes connexions et d'autre part de traiter le phénomène de rafale, c'est à dire l'arrivée de paquets avec un débit supérieur à celui négocié initialement. Au débit négocié est associé le délai théorique minimum T_{th} entre deux paquets successifs. L'idée mise en œuvre consiste à accepter une augmentation relative du débit d'un facteur α pouvant être fixé dynamiquement. Une valeur de α élevée favorise la qualité vidéo mais implique un surcoût mémoire important, en définitive un bon compromis a été atteint en fixant $\alpha = 15\%$ pour les flux testés.

La figure 3.3 illustre la méthode. Si le délai est inférieur au seuil alors la cellule est stockée en mémoire suivant la méthode décrite ci-après, sinon la cellule est perdue. La mémoire est organisée à l'aide d'une table de connexion (CT) contenant les adresses des listes de cellules (LL) associées. Ces listes de cellules partagent le buffer commun dont la taille fixe la capacité maximale du système. Cette méthode évite la copie onéreuse de cellules et accélère les accès mémoire qui sont le point critique de l'architecture notamment en raison de l'entrelacement. La figure 3.4 illustre les gains obtenus grâce à l'entrelacement. La mesure utilisée pour quantifier l'erreur est le rapport signal sur bruit, on observe ainsi la résistance du système aux pertes de paquets. En pratique, l'amélioration se juge de manière visuelle. Les exemples d'images permettent de noter que la qualité visuelle est nettement améliorée lorsque la perte des paquets ne se traduit pas par une disparition localisée de pixels.

Le démonstrateur a été mis en œuvre, l'architecture à été réalisée en VHDL et testée avec des flux MPEG lues sous la forme de fichiers ainsi qu'avec un émulateur de réseau ATM.

Travaux en cours : application de la méthode au NoC L'évolution de la taille et du degré d'intégration des SoC va amplifier de manière aiguë les problèmes d'incertitudes et d'immunité aux fautes aux niveaux des traitements et surtout des interconnexions. En effet la largeur du canal des transistors diminuant la fréquence des erreurs de type SEU (*Single Event Upset*) ne pourra plus être négligée. Par ailleurs la réduction des tensions d'alimentation entraîne une diminution de la marge de bruit. L'immunité des circuits aux sources de bruit telles que les phénomènes de *crosstalk*, les radiations liées à l'environnement et aux alimentations ne sera plus suffisante et des techniques de correction d'erreur devront être mises en œuvre de manière systématique. De plus les interconnexions, sur des puces de grande taille, nécessitent un nombre de cycles variable entre la source et la destination. Le NoC, en plus de son intérêt pour simplifier les communications entre IP, est une solution pour la mise en œuvre du contrôle de trafic et des mécanismes de correction d'erreur. Ainsi l'objectif de la dernière année de thèse est de transposer l'expérience réalisée autour du réseau ATM sur un NoC utilisé pour la communication de processeurs NIOS et de deux mémoires sur une cible Altera Stratix II. Le NoC a été conçu à des fins de démonstration, il ne s'agit pas de μSpider présenté ci-après. Les résultats seront disponibles pour la fin de la thèse prévue avant la fin de l'année 2005.

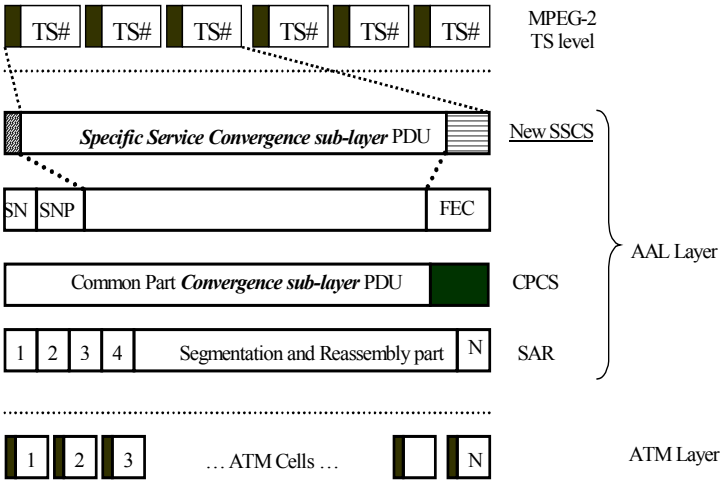


FIG. 3.2 – Nouveau mapping de MPEG-2 sur la couche AAL5+

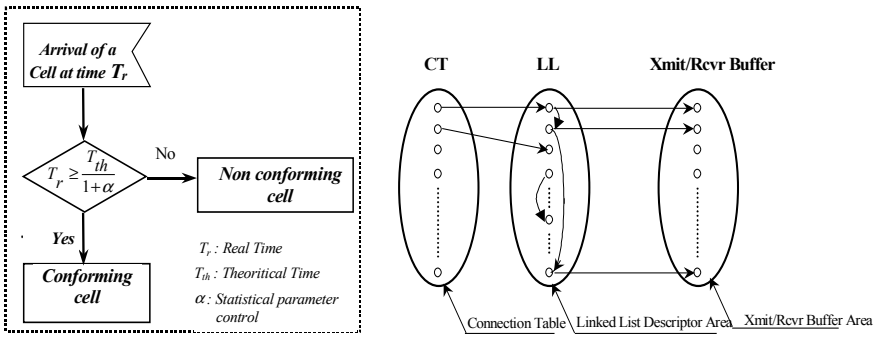
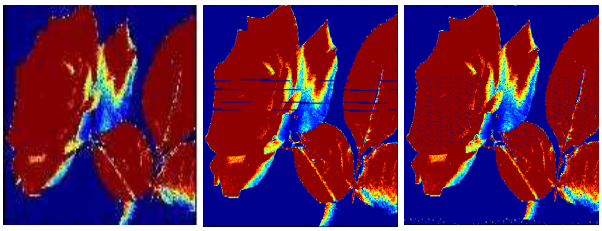


FIG. 3.3 – Contrôle de traffic et gestion mémoire

Test Encoded video sequence	*Loss Probability*	*PSNR(dB) without Interleaving*	*PSNR (db) with Interleaving*
Engine benchmark	10^{-4}	28,2	19,4
	10^{-3}	19,8	16,8
	10^{-2}	10	9,5
Vanseg benchmark	10^{-4}	30	27,4
	10^{-3}	20	19,7
	10^{-2}	12,1	10

PSNR de séquences de test MPEG ;



Original, sans et avec entrelacement

FIG. 3.4 – Mise en œuvre de l'entrelacement

3.4 Conception de NoC ad hoc : μSpider

3.4.1 Introduction

Le thème des NoC a progressé par transitions progressives depuis les structures multi-bus utilisées pour interconnecter les systèmes multi-processeurs. Initialement est apparu le concept de routeur, en tant que circuit, utilisé pour interconnecter un grand nombre de processeurs communiquants à l'aide d'une technique de routage de type *Wormhole* [59]. Ensuite l'apparition des *Systems On Chip* a ouvert la possibilité d'implanter plusieurs processeurs sur un circuit unique et la notion de NoC est naturellement apparue [60] pour assurer des transferts au niveau paquet. Les techniques employées reposent sur des travaux précédents dans le domaine des réseaux de télécommunications, ou d'ordinateurs ou des architectures multi-processeurs. Ces techniques ont été redimensionnées pour être adaptées à l'échelle et aux nombres de nœuds potentiellement connectés au sein d'un SOC. Un concept important est par exemple celui des canaux virtuels [61] nécessaires à la mise en œuvre de la garantie de service au sein d'un NoC notamment lorsque des contraintes temps réel sont à respecter. Historiquement le NoC est issu des réseaux de processeurs (ordinateurs) où la notion de temps réel était supplantée par celle de *best effort*. Cela explique que peu d'approches ont abordé la question de trafic garanti, l'autre raison est le fait que peu de mises en œuvre concrètes de systèmes ont été effectuées notamment dans les domaines du multi-média et des télécoms où la notion de QoS a un sens associé au respect des contraintes de temps. C'est ainsi que l'on ne trouve guère que Ætheral [62] chez Philips et μSpider [63] qui ne traitent réellement de la question. Les NoC deviennent intéressants par rapport au multi-bus lorsque le nombre d'IP (processeurs, mémoire, accélérateurs dédies) devient important [64] donc lorsque la taille des SOC est conséquente. Or, le problème du timing devient extrêmement complexe lorsque la taille des circuits croît et la technologie diminue puisque les phénomènes de *skew* et de *crosstalk* rendent l'estimation exacte des délais dans les interconnexions quasi impossible. Le succès de la start-up française ARTERIS [65] s'explique justement par le fait qu'ils sont les seuls a priori à offrir une solution concrète de NoC suffisamment robuste pour résoudre partiellement cette question. Leur technologie repose sur le concept des GALS (Globalement Asynchrone Localement Synchrone). Cela se traduit par des routeurs synchrones auxquels sont connectés des IP fonctionnant sur la même horloge alors que les communications entre routeurs sont réalisées à l'aide FIFOs asynchrones.

3.4.2 Philosophie du projet μSpider

Le projet est né au sein de l'équipe projet CNRS Radio-Logicielle (EPML CNRS 16) entre l'IETR (Rennes) et le LESTER, initiée en 2001 et débutée en 2002. L'EPML a été structurée autour de deux projets, le premier est consacré à la consommation des FPGA (Directrice de thèse N.Julien au LESTER) et le second aux NoC (Directeur de thèse D.Houzet à l'IETR), en inversant la localisation des directeurs de thèse et étudiants afin de favoriser les liens inter-

laboratoires. En pratique j'encadre au quotidien S.Évain sur le sujet au LESTER.

L'objectif de ce projet se mesure en deux temps. La première étape a consisté à créer un environnement de CAO permettant de personnaliser un NoC et d'en effectuer la synthèse de manière automatique sur cible FPGA. La seconde phase, qui est en cours, s'appuie sur l'environnement flexible disponible. Le but est de proposer et d'implanter de nouvelles techniques dans le domaine des *Network On Chip*. Les thèmes traités, en cours où à venir sont dans l'ordre :

1. Le dimensionnement (topologie, mapping, chemins, ordonnancement) ;
2. La mise en œuvre du trafic garanti dans un contexte de communications asynchrones ;
3. La reconfiguration dynamique des chemins et ordonnancements en fonction des modes de fonctionnement ;
4. La sécurité et la tolérance aux fautes ;
5. L'estimation de la consommation et de la surface à haut niveau d'abstraction.

Beaucoup d'approches dans le domaine des NoC se sont focalisées sur des simulations (ex. simulateur NS2 [66]) probabilistes ou statistiques des communications [67]. Nous n'avons pas souhaité aller dans cette direction et avons privilégié l'aspect CAO, le dimensionnement et les futures directions citées précédemment qui nous semblent pertinentes du point de vue des perspectives de recherche. De plus, l'IETR va se charger d'ajouter à notre environnement une sortie au format SystemC qui permettra d'accélérer les temps de simulation.

3.4.3 Réalisation

L'outil de CAO μSpider permet de spécifier un grand nombre de paramètres pour la production d'un NoC répondant au contraintes imposées par le concepteur dans un contexte de SOC. La stratégie suivie est celle d'un NoC permettant :

- d'assurer le transfert de données au niveau paquet (couche OSI 3) ;
- à l'aide de techniques du type canaux virtuels, de garantir la latence et la bande passante des trafics garantis (GT) lorsque ceux-ci sont requis et d'offrir des solutions efficaces de type *Best Effort* (BE) avec différents niveaux de priorité dans le cas contraire ;
- de proposer différentes solutions de routage statiques et d'arbitrage qui soient adaptées au niveau de QoS souhaité et qui garantissent l'absence de *deadlocks*
- de fournir les composants peut être les plus importants à savoir les interfaces réseaux (NI) permettant de simplifier les échanges entre IP au travers d'une interface unique disposant de *wrapper* pour l'adaptation de protocole (OCP, OPB, Avalon, etc.). L'interface prend en charge l'ordonnancement par une approche de type TDMA (*Time Division Multiple Access*) des accès au réseau pour chacun des trafics GT et globalement pour l'ensemble des trafics BE. L'idée est ici, qu'un trafic BE puisse accéder à tout moment au réseau au travers de l'interface si aucun trafic GT n'est actif.

Les caractéristiques de l'outil de synthèse sont présentées dans [68, 63]. Il serait fastidieux de détailler ici tous les paramètres à disposition du concepteur ou de l'outil de dimensionnement. La figure 3.5 résume l'espace de conception en faisant un parallèle avec les couches OSI et la figure 3.6 présente l'architecture générique du NoC (dans cet exemple le routeur a une arité de 3 ports et chaque port n'a qu'un seul canal virtuel par souci de synthèse). La figure 3.8 compare les caractéristiques de μSpider aux principales autres solutions publiées.

Notre environnement est relativement proche de la solution Ætheral notamment depuis la publication [62] qui présente un flot de CAO. Ils proposent également dans cet article une méthode de dimensionnement qui nous semble être relativement grossière et adapté uniquement aux trafics homogènes, cette stratégie peut se justifier d'un point de vue industriel. Les différences notables entre les outils se situent au niveau du degré de généricité, du nombre de canaux virtuels limité à deux pour Ætheral (1 BE, 1 GT), par ailleurs leur politique d'attribution des chemins

associés aux ordonnancements est fixe. Enfin, l'outil constitue un moyen pour nos recherches futurs et non un but en soi.

Modèle OSI	Equivalent NOC	Fonctions et paramètres génériques µSpider
7 Application 6 Présentation		
5 session	**Communication entre IP Maître et Escaves**	Contraintes Applicatives par tâche de communication **Latence, Taille Burst, Bande Passante**
4 Transport	**NI (Interface Network)**	**Table TDMA Wrapper protocole Mapping mémoire (Globale => Port NI) Ordonnancement TDMA Paramètres (Taille FIFO, Nb Canaux, Nb Ports)**
3 Réseau	**Paquet**	**Chemin (source routing) Routage (XY, Street Sign, ...) Adresse memoire locale Arbitrage (Sans, Fixe, Round Robin) Sécurité(SPA*, ...) Paramètres Topologie (Nb routeurs, Arité, Topologie)**
2 Liaison	**Contrôle de flux entre routeurs FLIT (Flot control digIT)**	**Wormhole Flot de contrôle (Hand shake, cédit d'émission) Sécurité (Bit parité, ..) Nb et type de Canaux Virtuels (GT, BE1, ..) Paramètres (Taille Champs, FIFO, ..)**
1 Physique	**PHIT (PHysical digIT)**	**Nombre de bits**

FIG. 3.5 – Espace de configuration μSpider

3.4.4 Travaux en cours

La figure 3.7 présente le flot global de conception qui est construit autour de l'outil de synthèse dont le point d'entrée est un fichier .xml permettant de spécifier les options choisies ou calculées.

Dimensionnement Le dimensionnement du NoC adresse un grand nombre de paramètres, tels que la topologie du routeur (MESH 2D, Arbre, Anneau etc.), le nombre de routeurs, le nombre de bits des liens, la taille des différents champs de contrôle, la taille des FIFOS, le nombre de slots de temps et leur affectation au sein des fenêtres TDMA, etc. Le problème est évidemment trop complexe pour être résolu de manière exhaustive, d'autant plus que nous souhaitons l'utiliser en ligne dans le contexte d'architectures dynamiquement reconfigurables. L'approche développée est présentée sur la figure 3.7. Elle consiste à ordonner les différentes décisions en minimisant la remise en cause des décisions précédentes. La première difficulté, et non la moindre, consiste par exemple à formaliser mathématiquement les contraintes exprimées par l'utilisateur (bande passante, latence, taille de *bursts*) en incluant les échanges des différents paquets de contrôle et les dépendances entre les trafics. Le modèle analytique permet notamment de borner l'espace de recherche relatif aux nombres de routeurs, de liens, de ports d'interfaces etc. À titre d'exemple représentatif, la formulation suivante exprime le problème de type ILP à résoudre pour déterminer la taille minimum unique S (en nombre de slots) des tables TDMA utilisées au sein des NI (*Network Interface*) pour l'ordonnancement des accès au réseau.

$$Min(S) \mid \sum_i \lceil k_i S \rceil \leq S \tag{3.1}$$

où $0 < k_i < 1$ représente le pourcentage de la bande passante affectée à chaque trafic garanti i. Compte tenu des critères de convergence rapide souhaités, la question est résolue à l'aide d'une approche itérative qui tire profit du fait qu'en pratique le taux de trafic de type (BE) nécessite une bande-passante non négligeable qui permet de relâcher les contraintes du problème.

Les résolutions des différentes étapes ont été formalisées, une première version de l'outil qui reste à durcir est opérationnelle.

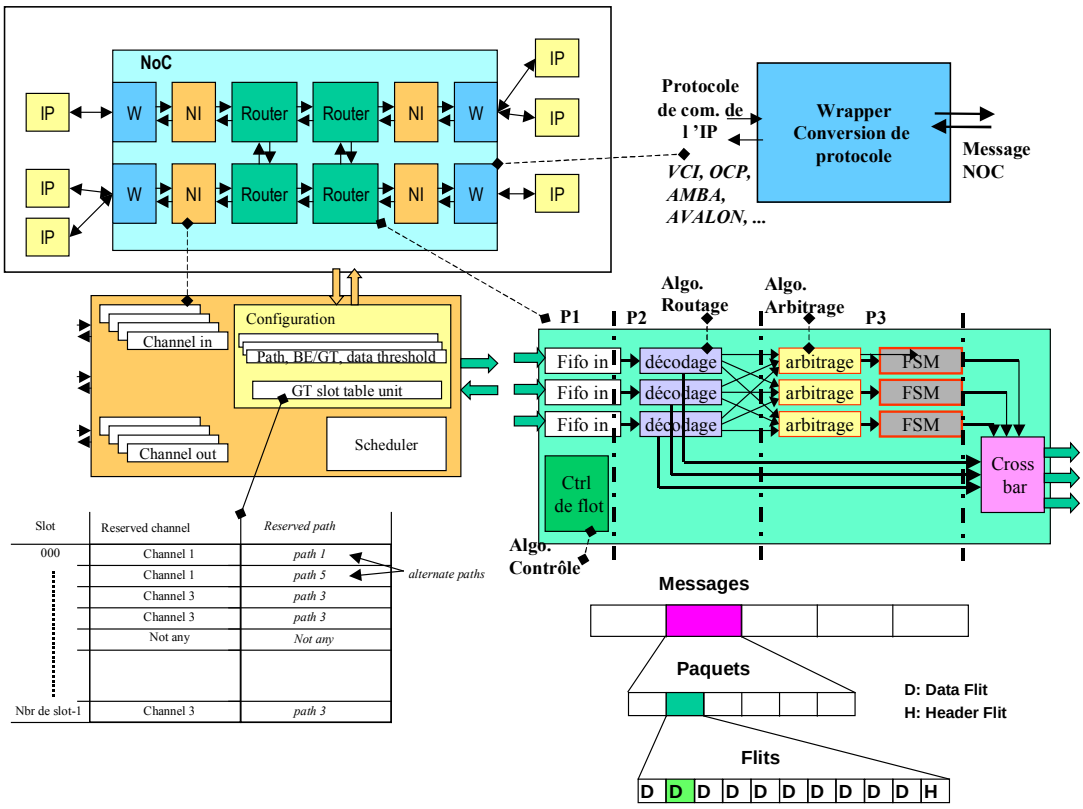


FIG. 3.6 – Architecture générique du NOC

Communications asynchrones Une approche du problème consiste à rendre déterministe les délais sur les différentes interconnexions par l'insertion de répéteurs et de registres. L'inconvénient de la méthode est la complexité de l'analyse qui en découle et qui, de notre point de vue, va à l'encontre de l'objectif de flexibilité propre au NoC.

Notre approche du problème est originale et tire profit du fait que μSpider est conçu de manière à pouvoir produire des NoC hiérarchiques c'est à dire un NoC de NoCs[1]. Celle-ci revient à considérer des NoC synchrones (par ex. 1 routeur + 2 NI) connectés entre eux de manière asynrchone au travers d'une NI dépourvue de *wrapper* et dont le rôle sera de répartir l'accès aux NI situées sur le NoC destinataire. Ainsi, les bandes passantes peuvent être garanties, la latence peut être bornée en fonction du délai maximal introduit lors de la communication asynchrone entre les NI des NoC situés dans des domaines d'horloge différents. Le prix à payer pour la résolution de la question du timing est le surcoût dû à l'introduction d'interfaces aux frontières des domaines synchrones.

Reconfiguration dynamique À ce niveau, le premier objectif est de permettre à un manager de configuration de reprogrammer dynamiquement les chemins et les ordonnancements TDMA au sein de chaque interface lorsque le mode de fonctionnement du système est modifié (nouvelles contraintes de communication). L'architecture de μSpider a été pensée afin qu'un maître doté de droits particuliers puisse jouer le rôle de configurateur en accédant aux tables (chemins, TDMA) des interfaces. Par ailleurs, les algorithmes de dimensionnement de complexité polynômiale sont conçus de manière à pouvoir être exécutés en ligne en un temps raisonnable compte tenu du nombre limité en pratique (quelques dizaines) de nœuds (maîtres et esclaves) du NoC.

[1] "on the heaven's door" suggéra le mélomane...

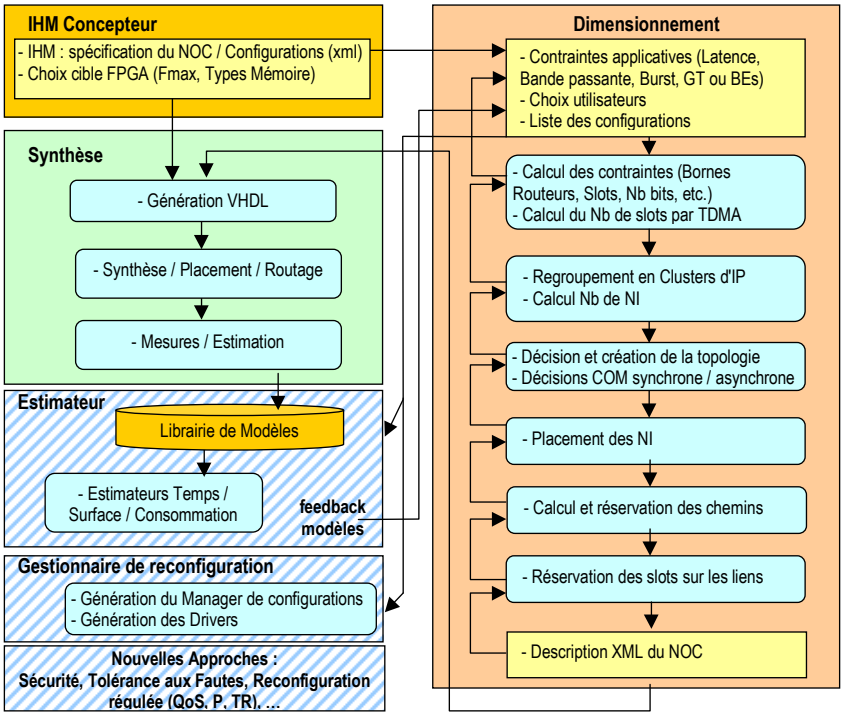


FIG. 3.7 – Flot de conception μSpider

Sécurité et sûreté de fonctionnement Nous avons effectué une étude approfondie de la question de sécurité dans le NoC en abordant les différents cas de figures : déni de service, injection de faute, extraction d'information, etc. À notre connaissance, aucune analyse de ce type n'a encore été publiée. Nous avons également proposé un nouveau type de codage SCP (*Self Complemented Path coding*) de chemin qui permet d'une part de réaliser de manière simple et fiable l'authentification de l'interlocuteur et d'autre part de gérer facilement la reconfiguration dynamique du placement des IP autour du NoC. Ces travaux ont été soumis à SIPS'05 [69].

Estimation de consommation et de surface Ces travaux sont des perspectives de recherche que nous mèneront en commun avec l'équipe consommation du LESTER. L'idée est d'effectuer des mesures sur des circuits générés par μSpider et de déterminer des corrélations entre celles-ci et les paramètres architecturaux d'une part (nombre de routeurs, nature des interface, type de canaux virtuels, topologie, ...) et les caractéristiques des contraintes de communications imposées par les applications (débit, latence, *bursts*) d'autre part.

3.4.5 Applications

Les contraintes imposées pour le test de l'outil de dimensionnement sont issues de deux types d'applications aux caractéristiques très différentes. La première, fournie pas l'IETR, est issue du domaine des télécoms, il s'agit de l'application MC-CDMA du projet IST 4more. Celle-ci se traduit par des communications principalement de type point à point entre des opérateurs de grain relativement fin. La seconde se situe dans le domaine du traitement vidéo, il s'agit du *tracking* d'objets dans un contexte d'architectures adaptatives. Suivant la configuration logicielle/materielle choisie la granularité des transferts de données varie de quelques octets à une image entière. Nous envisageons d'associer aux traitements précédents d'autres applications que nous développons dans le cadre des architectures adaptatives telles que la cryptographie

(AES) et le codage canal (Turbodécodeur). L'objectif est de tester la solution NoC dans un contexte hétérogène de contraintes.

3.4.6 Conclusion

L'intérêt des NoC a donné lieu à un certain nombre de discussions, en définitive son principal atout est, à mon sens, la simplification de la synthèse des communications lorsque le nombre d'IP communiquant est important voire variable dans le cas d'architectures dynamiquement reconfigurables. Dans ce cas le sur-coût matériel est acceptable et justifié par les gains obtenus en termes de flexibilité, de temps de conception et de fiabilité. Par ailleurs un environnement de conception de NoC offre un cadre formel et précis permettant d'appréhender de manière globale des questions telles que la reconfiguration dynamique, la correction des erreurs et la gestion du *timing*. À présent, la maturité du projet et de nos réflexions nous amène à un phase intensive de publications.

NoC name	**SPIN**	**RASoC**	**Æthereal**	**µSpider**
Topology	multi-level Fat-tree	2-D grid or torus	generic	n dimension mesh topology or any
Router ports (I/O) number	8	5	1 to 6	Generic
Data width (in bits)	32	Generic	32	Generic
A/Synchronous System Clock	Synchronous	Synchronous	Unknown	Both available
Link flow control protocol	Credit based	Handshake	Credit based for BE, no back control for GT	Credit based , Handshake or no back control
Buffering	Input buffer + 2 shared output buffers	Input buffer	Input buffer	Input buffer
Routing algorithm	Adaptive upward routing. Deterministic downward routing.	Deterministic source-based routing algorithm (XY).	Source routing	2 deterministic source-based routing algorithms: dimension ordered or street sign
Output Arbitration	Round robin	Round robin	Round robin for BE. No arbitration for GT	Round robin, fixed channel priority or no arbitration
Traffic classes supported	BE	BE	BE + GT	BE + GT + intermediate QoS classes
Virtual channels number	1	1	Can be related to 2 VC	Generic
Specification Input	No	VHDL code Level	Yes (since 2005)	Explicit CAD tool

FIG. 3.8 – Comparatif des solutions NoC

Chapitre 4

Reconfiguration Algorithmique Architecturale Régulée (RAAR)

4.1 Motivation

Ce projet est né du bilan des recherches précédentes et du constat sur l'évolution des systèmes embarqués de type multimédia communiquants. En effet, il apparaît une double contradiction formant le nouveau paradigme à résoudre :

1. Pour atteindre les objectifs de performance et de consommation, une architecture optimisée donc dédiée serait en théorie nécessaire, cependant l'accroissement exponentiel du coût de fabrication des circuits impose d'exploiter au maximum la réutilisation voire la reconfiguration de composants matériels et logiciels. La contrainte économique incite à la flexibilité alors que les exigences d'efficacité s'y opposent.
2. La variabilité des caractéristiques des systèmes va croissant en raison des techniques d'optimisation qui dépendent des données et sont implantées aux niveaux architectural et algorithmique. De plus la taille des applications considérées se traduit par un poids dominant des aspects contrôle au niveau système. Dans ces conditions comment évaluer a priori les performances et la consommation d'un système dont le comportement varie en fonction des données et de l'environnement ?

J'ai obtenu en octobre 2004 le financement d'un programme de trois ans (ACI JC9169 "RaaR") dont l'objet est de proposer une réponse au dilemme précédent. L'approche repose sur l'idée que le système doit être conscient de son état pour s'adapter lui même à son environnement et aux tâches qu'il doit exécuter. Le modèle proposé repose sur de nouveaux service de RTOS. Il doit permettre d'une part de prédire en temps réel le comportement d'un système fluctuant et d'autre part d'exploiter ce modèle pour effectuer dynamiquement le paramétrage des applications afin d'optimiser le compromis consommation / qualité de service (QoS) / performances. Ainsi notre objectif est de considérer le problème sous un angle à notre connaissance encore inexploré. Il s'agit de l'asservissement de processus discrets tels que traité en automatique mais appliqué à un système constitué d'une architecture reconfigurable à base de processeur(s) et d'application(s) dotées de différents modes de fonctionnement. Dans ce contexte, j'encadre la thèse de Yvan Eustache depuis octobre 2004.

4.2 Positionnement

Ce thème de recherche approche quatre types de domaine de recherche sans se fondre dans un seul. Le premier relève du domaine des architectures adaptatives (niveau VDD et horloge [70], pipeline [71], gestion du cache [72], unités fonctionnelles [73], bande passante de NoC [74] ou de

la sélection au niveau matériel du choix de l'algorithme [75]. Les approches existantes traitent le problème localement alors que notre intention est de traiter le problème de manière globale afin de fournir une solution transposable. Le second domaine est celui des architectures dynamiquement reconfigurables à grain élevé telles DART à l'IRISA ou Systolic-Ring au LIRMM où la question de la décision de reconfiguration n'est pas vraiment abordée. Le troisième domaine est celui des RTOS étendus à la reconfiguration matérielle qui a récemment été introduit [76, 77]. Ces approches montrent que le lien RTOS /reconfiguration matériel est pertinent en tant que perspective de recherche. Si l'on suppose que les architectures réellement dynamiquement reconfigurables seront bientôt disponibles alors la question de la stratégie de reconfiguration se pose. Enfin, le dernier domaine est purement logiciel et concerne les politique d'OS pour le contrôle du QOS. Cet aspect a été exploré en détail notamment pour les applications web [78]. L'aspect automatique qui nous intéresse a été récemment introduit dans ce domaine, l'asservissement étant proposé en tant que solution potentielle pour traiter la question de l'incertitude relative au WCET. Des solutions pour l'ordonnancement de tâches vu comme une optimisation de QoS plutôt que purement temps réel sont détaillées dans [79] et [80].

4.3 Approche

4.3.1 Principe

Le thème de la reconfiguration embrasse plusieurs aspects à considérer avant de choisir l'implantation de la méthode de décision. Le premier point est la localité, un compromis doit être trouvé entre les décisions à prendre localement et celles qui doivent être résolues globalement. Les premières sont locales en raison de la spécificité de l'information, du temps de réaction et de l'impact. Les secondes sont à prendre globalement lorsqu'une vue d'ensemble est requise et que l'abstraction des éléments de décision est généralisable à l'ensemble des tâches. Le second critère à prendre en compte est la complexité de l'algorithme de décision, là encore un compromis est à déterminer entre la recherche de l'optimum et l'énergie à dépenser pour trouver la solution. Dans un contexte embarqué, le coût de la décision doit être négligeable par rapport aux gains escomptés.

La méthode en cours de développement repose sur trois principes :

- **Le système apprend**. Une table RST (*reconfiguration space table*) permet de conserver, de mettre à jour et de trier rapidement les caractéristiques (Qualité de Service normalisée, Puisance, Temps d'exécution) des configurations sélectionnées lors de la conception. La sélection de la configuration adaptée découle d'une simple opération de tris successifs (ex. Durée de vie, puis QoS, puis le temps d'exécution)
- **Le système s'adapte**. Un système est modélisé par un asservissement en boucle fermée (cf. Fig. 4.2) disposant d'un observateur et d'un régulateur. Dans un souci de compromis complexité / efficacité, le système observe et construit son propre modèle dynamiquement, lorsque c'est nécessaire il observe son état à l'aide des capteurs disponibles (mesures temps d'exécution, niveau batterie, qualité du résultat), sinon fait appel au modèle pour estimer son état. La fréquence de l'adaptation ainsi que celle des mesures est adaptée en fonction des erreurs constatées par rapport aux consignes (Durée de vie, QoS, ou Temps réel) et par rapport au modèle respectivement.
- **L'OS dispose des services pour le contrôle**. Il offre au moins deux nouveaux services : l'accès aux "capteurs" et le contrôle de la reconfiguration.

Ce travail est actuellement en cours, un premier algorithme pour la mise en œuvre est décrit dans [81], celui-ci précise les conditions de stabilité de l'asservissement ainsi que la méthode d'identification du modèle par une méthode issue du monde des télécoms et offrant un bon compromis complexité / efficacité, à savoir l'algorithme du gradient stochastique (LMS). La figure 4.1 présente l'organisation du système régulé. Ce système se compose de deux types de

tâches, celles qui sont régulées et celles qui ne le sont pas (sporadique). Trois principales tâches "système" sont incluses dans la spécification de l'application, il s'agit de l'observation (récupérations des données), de l'estimation (fonction de transfert du système) et de la décision (choix des configurations et fréquences de mesure, d'estimation et de reconfiguration). De nouveaux services sont ajoutées à l'OS afin de généraliser l'accès aux données de contrôle et de configuration. Le système est conçu de manière à se stabiliser dans la configuration qui sera la plus favorable au respect des contraintes imposées par l'utilisateur. Lorsqu'un état stable est atteint, la fréquence de l'adaptation et des mesures décroît régulièrement jusqu'à atteindre un limite correspondant au temps de réaction maximum spécifié. Au contraire dans un contexte très perturbé, le système se stabilisera dans un état où des gains peuvent être atteints (i.e. le système ne consommera pas inutilement son CPU et sa batterie en tentant de faire évoluer le système vers une illusoire meilleure configuration).

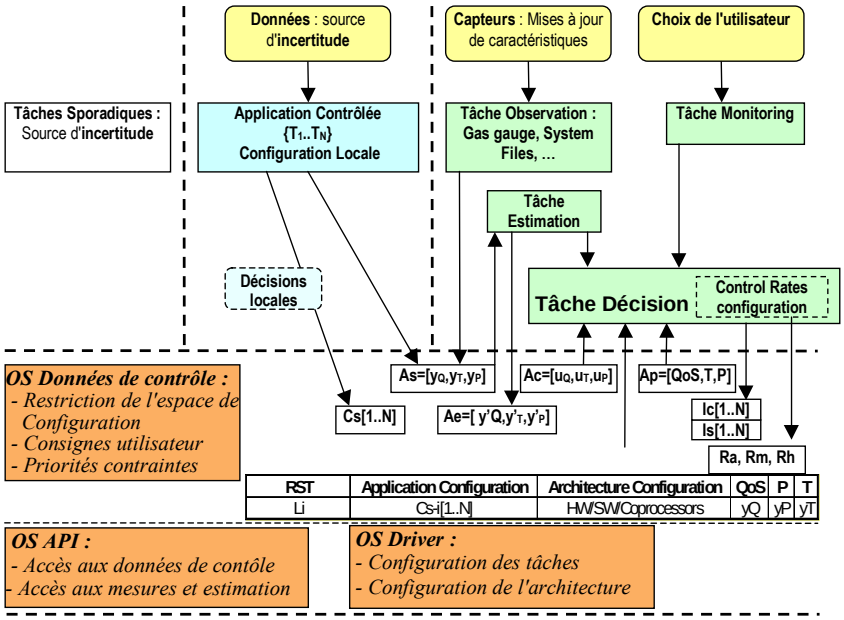


FIG. 4.1 – Organisation du système régulé

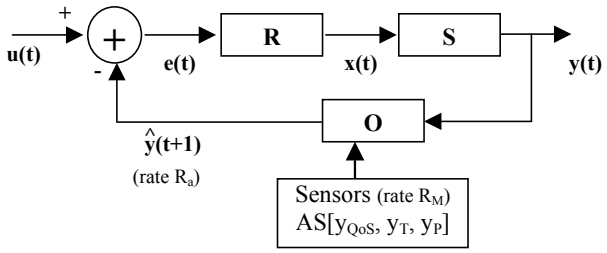


FIG. 4.2 – Asservissement de la configuration du système embarqué. R : régulateur ($ke(t)$), S : Adaptateur de configuration (tri des solutions par rapport aux priorités des consignes et erreurs), O : Observateur du système (mesures + estimateur du modèle de la fonction de transfert)

4.3.2 Expériences en cours

Nous avons choisi une approche par l'expérimentation indispensable dans le domaine visé, les trois types d'applications visées sont les suivantes dans l'ordre de priorité.

Application vidéo À partir de l'application de caméra intelligente fournie par le CEA/LIST dans le cadre du projet RNTL Epicure, nous avons imaginé une application de *tracking* d'objets consistant à suivre à l'aide d'un filtre de Kalman les objets apparaissant dans le champs de la caméra. L'intérêt de l'application réside dans le grand nombre de paramètres de configuration situés à la fois aux niveaux local et global. Il s'agit notamment du compromis en l'estimation et l'extraction d'objets nécessitant l'acquisition d'une nouvelle images. Le dispositif en cours de développement est dans un premier temps construit sur une carte Altera stratix II. Celle-ci se compose d'un processeur NIOS disposant du RTOS μCOS II ainsi que d'une caméra, d'une sortie VGA, d'une batterie et de sa jauge. Plusieurs configurations logicielles et matérielles sont en cours d'implantation. La seconde étape consistera à migrer sur Xilinx Virtex II afin de tirer profit de la reconfiguration dynamique.

Application cryptage L'application cryptage est réalisée en collaboration avec Guy Gogniat qui effectue une année 2004/05 sabbatique à l'université UMASS (USA). Le travail effectué avec l'aide de stagiaires ingénieurs a abouti à une série d'architectures sur un principe incrémental permettant d'augmenter progressivement le pipeline afin d'offrir différents compromis entre les ressources et le débit. Cette évolution se traduit par différents rendements (nJ/bit) pour le cryptage et/ou le décryptage. L'algorithme de contrôle est un régulateur permettant d'adapter la configuration au débit dans le cadre d'un profil supposé négocié au préalable. Le projet en est au stade de l'intégration sur Virtex II pro.

Applications communication numériques Le troisième domaine est celui des communications numériques, il s'agit d'une collaboration avec C.Jego et E.Priou de l'ENST Bretagne autour d'une architecture flexible dédiée au turbo-décodage en bloc. Dans le cadre du stage de DEA de S.Huang nous rendons adaptative l'architecture statique initialement développée à Brest en SystemC. L'idée est ici de piloter dynamiquement, entre autres, le nombre d'itérations et les critères de convergence de l'algorithme en fonction du rapport signal sur bruit.

Une seconde étude est consacrée à la réalisation d'une architecture flexible de récepteur MIMO OFDM. Ce travail se déroule dans le cadre d'une collaboration construite autour du stage de DEA (UBS) que M.Bey effectue au CISS avec Y.Le Moullec à présent Assistant Professor à l'Université d'Aalborg au Danemark. L'objectif est ici d'adapter dynamiquement le rendement du codage en fonction du rapport signal sur bruit, l'architecture cible est construite autour d'un Microblaze sur Xilinx Virtex II pro.

Ces deux expériences constituent un travail exploratoire et préparatoire à une thèse en cotutelle LESTER(UBS)/TAMCIC (ENST Bretagne) qui doit débuter avant fin 2005.

4.3.3 Projet simulateur

Afin de disposer d'un environnement de test pour évaluer rapidement différentes options ou réglages du régulateur nous avons développé un simulateur. Celui-ci a été conçu de manière à être indépendant de l'application cible, la première mise en œuvre est une application de synthèse 3D implantée simultanément sur Stratix par N.Ben Amor dans le cadre de sa thèse (transition de DT vers RaaR). Les paramètres de l'application sont le nombre de polygones et le choix entre deux algorithmes d'ombrage Gouraud et Flat pour lesquels deux architectures à base de NIOS avec et sans co-processeur sont disponibles. Les sources d'aléas sont dues aux formes des objets, au nombre d'objets et de polygones utilisés, à leur vitesse et à la charge du

CPU pouvant varier en fonction du démarrage de tâches sporadiques. La première version du simulateur (fig. 4.3) en date de juin 2005. Celle-ci permet de modifier les paramètres suivants :

- Niveau de charge initiale d'une batterie Li-Ion, le modèle utilisé est celui décrit dans [82] (plus précis que celui de Peukert) ;
- Loi de probabilités des aléas (Délai Puissance) dues aux données et tâches sporadiques ;
- Le coefficent K du LMS ;
- Le gain du régulateur k_p ;
- Le coût (délai, puissance) des mesures, estimations et reconfigurations ;
- Les périodes minimales des tâches d'observation et d'estimation ;
- $Th_{1,2,3,4}$ Thresholds for dynamic rate management ;
- Les choix de priorité et de consigne pour QoS, Délai and Puissance (durée de vie).

et retourne l'évolution des valeurs de QoS, de durée de vie et de temps d'exécution. L'outil n'est pas encore stable à ce jour, l'objectif est d'obtenir une version fiable avant la fin de l'année 2005.

4.3.4 Liens

Deux liens forts existent avec deux autres projets de l'équipe. Il s'agit de l'utilisation des capacités de reconfiguration fournies par un NoC produit par μSpider. Le second est l'aspect dynamique du partitionnement logiciel/matériel sur contrainte de QoS.

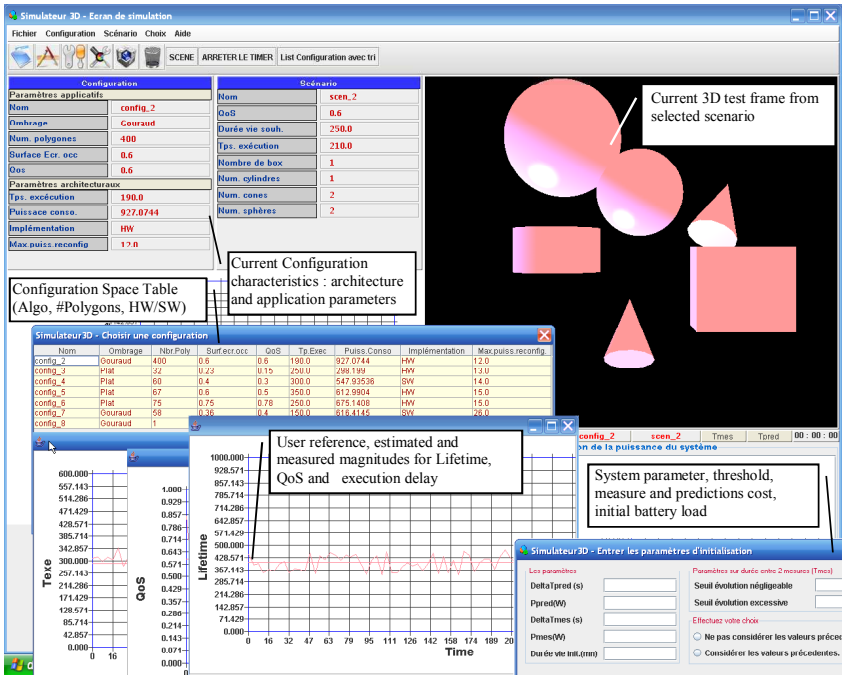


FIG. 4.3 – Version initiale de l'environnement de simulation, application synthèse 3D

4.3.5 Conclusion

Le projet RaaR est dans sa première année, les bases ont été posées et différentes mises en œuvre sont en cours. Un important effort de développement est cependant encore nécessaire pour prouver la pertinence de l'approche. Les aspects les plus critiques se situent au niveau de la mise en œuvre de la reconfiguration dynamique et de la modification du RTOS afin d'intégrer les nouveaux services requis. Ce projet fait l'objet de collaborations avec l'équipe d'Y. Trinquet à

l'IRCCyN pour les aspects RTOS et l'ENST Bretagne pour les applications de communications numériques.

Chapitre 5

Responsabilités diverses

La figure 5.1 résume l'ensemble de mes activités depuis la thèse.

5.1 Enseignement

- 93/96 : Boursier MESR et vacataire à l'ENSSAT/Univ. Rennes I. Enseignements (TP, TD) : VHDL, FPGAs, Synthèse Logique, Traitement numérique du signal.
- 97/02 : **Maître de conférence** en licence/maîtrise EEA puis en IUP GEII.
- 00 / 02 : **Directeur d'études de la licence de l'IUP GEII.**
  Cours (CM, TD, TD) :
  - Traitement du signal : traitement numérique du signal (Maîtrise), signaux et systèmes (licence)
  - Automatique : Asservissements, Modèle d'états (Maîtrise)
  - Circuits et systèmes : conception VLSI, Synthèse logique, Consommation (Maîtrise), familles logiques (DEUG)
  - Informatique : algorithmique, langage C, Mapple (DEUG),
- 00/02 : **Intervenant DEA :** Modules "Algorithmes d'optimisation exactes et heuristiques pour la conception VLSI" à l'ENSTB et à Rennes, Module "filtrage adaptatif" à Lorient, DEA Électronique co-habilité UBS, ENSTB, INSA, SUPELEC.
- 02/03 : Mise en disponibilité de l'enseignement pour activités de recherche et transfert technologique.
- 04/05 : **Intervenant M2 électronique** (UBS / ENSTB) et DESS MSO (UBS) "Algorithmes d'optimisation exactes et heuristiques pour la conception VLSI"

5.2 Co-Encadrements de thèse

Les thèses que j'ai encadrées l'ont été en co-direction avec J-L.Philippe, Professeur à l'UBS hormis celle de Samuel Évain dont le directeur de thèse est D.Houzet (MCF HDR, IETR, Rennes) :

- Yannick Le Moullec, soutenue le 10 avril 03, "Aide à la conception de systèmes sur puce hétérogènes par l'exploration paramétrable des solutions au niveau système" ; situation actuelle : Assistant Professor à l'Université d'Aalborg, Danemak, laboratoire CISS ;
- Abdenour Azzedine, soutenue le 2/06/04. "Méthode de conception matériel/logiciel pour les systèmes mono-processeur temps réel embarqués", situation actuelle : postdoc dans l'équipe codesign de G.Bois à l'École polytechnique de Montreal ;
- Nader Ben Amor, co-tutelle ENIS(SFAX)/LESTER, "Approche de conception de processeur de vision embarqué", soutenance avant fin 05, encadrement ENIS : M.Abid ;

- Belgacem Bouallegue, co-tutelle Fac. des Sciences Monastir/LESTER, "Contribution à l'interfaçage d'une application multi-média à un réseau à haut débit", soutenance avant fin 05, encadrement Monastir : Ridha Djemal ;
- Thèse Samuel Evain, co-encadrement depuis janvier 04, "Synthèse et personnalisation de NoC", soutenance prévue en décembre 05 ;
- Thèse Co-tutelle Hedi Tmar, débutée en décembre 03, "Codesign orientée QoS sur cible multi-processeur", encadrement ENIS : M.Abid ;
- Thèse Yvan Eustache, débutée en octobre 04, "Contrôle sous contraintes multiples de la reconfiguration algorithmique et architecturale d'un système temps réel embarqué" ;
- Thèses interompues : Hervé Thomas (97/99, raisons familiales), "Exploration algorithmique de l'espace de conception" et Stéphane Montfort, "IP Processeur réseau embarqué", (99/00, réussite au concours d'ingénieur de recherche, mutation à Paris)
- PostDoc : Montage du projet et collaboration autour du Post-Doc de Sébastien Bilavarn (thèse au sein du projet Design Trotter, 2002) à l'EPFL, Grant Intel 11409, "Processor architecture enhancement for media delivery".

5.3 Projets et contrats

Au cours de ces dernières années, j'ai participé au montage et à l'animation scientifique des projets suivants :

5.3.1 Projets en cours ou passés

- Projet régional MARSM (UBO, LESTER, 98-99). Montage et Co-reponsabilité avec E.Martin, "MPEG4 : analyse de complexité et propositions d'architecture sur FPGA".
- Projet CNRS (R2D2/LANNION, CNRS I3S/NICE, 00-02). Participation, "Méthode d'Aide à la Conception Globale des Terminaux de Télécommunications".
- Projet RNTL EPICURE (CEA/LIST, Thales, Esterel Tech., I3S, 00-02). Montage et responsabilité scientifique pour le LESTER, "Environnement de Partitionnement et de Co-développement adapté aux processeurs à architectures Reconfigurables".
- Projet RNRT A3S (Thales Com., Mitsubichi ITE, Softeam, 03-05). Montage en 2002 puis responsabilité scientifique en 04/05 en l'absence de G.Gogniat, "Adéquation Architecture - Application Système : approche MDA et modélisation UML appliquées à la radio-logicielle".
- Projet RAAR : ACI JC 9169, Fonds National de Science, (oct. 04/07). Montage, responsable administratif et scientifique, "Modélisation et régulation algorithmique de systèmes temps réels embarqués pour la maîtrise du compromis performance / consommation / QoS"

5.3.2 Montage de projets en cours d'examen :

- Projet Européen FET, AETHER, 2006-2010, 19 partenaires, recherche à long terme (horizon 2020), informatique diffuse et architectures spontanément adaptatives. Soumis en mars 05.
- Projet Européen, ITEA SPICES piloté par Thales, projet commun avec l'équipe consommation du LESTER (N.Julien), modélisation de composants et estimation de consommation, migration de l'approche composant vers les plate-forme matérielles de type FPGA. Application : avionique, soumis en mai 05. Le projet a été validé au niveau européen, la phase de demande de financement au niveau national est en cours.

	96-97	97-98	98-99	99-00	00-01	01-02	02-03	03-04	04-05	05-06
Thèses		H.Thomas (Interruption raison familiale)								
				Y.Le Moullec			10/04/2003			
				S.Montfort (Interruption Mutation)						
					A.Azzedine			02/06/2004		
					B.Bouallegue (co-tutelle UBS/ENIS)				(soutenance)	
						T.Gourdeaux (alternance)		Ingé. Recherche		
						N.Ben Amor (co-tutelle)			(soutenance)	
								H.Tmar (co-tutelle)		
									Y.Eustache	
							S.Evain		(soutenance)	
DEA		S.Montfort		P.Chedmail		A.Menhoudj			S.Wuang	
									B.Maiz (CISS)	
									M.Elkhodary	
Ingé/DESS/IUT		0/0/1	0/0/1	0/0/1	0/0/1	0/0/1	0/1/0	0/2/4	1/0/1	
Projets			Création et responsabilité du groupe ***Design trotter***							
								Projet ***µSpider***		
									Projet ***RaaR***	
Contrats		Projet Regional **MARSM** (UBO)			Projet CNRS **MACGTT** (I3S, LASTI)					
					Projet RNTL **EPICURE** (I3S, Thalès, Esterel Tech., CEA-List)					
						Projet RNRT **A3S** (Thales, Mitsubichi, Softeam)				
									ACI **RaaR**	
Collaborations		**IMEC** (fin postdoc)			Univ. **Aalborg / CISS** (DK)					
(visite, échanges						**EPFL** (CH) / **INTEL**(USA)				
publis, postdocs)						**UMASS** (USA)				
							PROSYR projet CMCU (SFAX, Tunise)			
Projets Déposés									IP FET AETHER	
									ITEA SPICES avec N.Julien	
								Structure Fédérative **IRCASE** avec B.Pottier		
Transfert Tech.			**US Patent** (IMEC, Memory Hierarchy)				Bretagne Innovation, Anvar			
					Brevet FR Écran micro-électro-mécanique			**Brevet FR**/PCT Dixip		
								Création PME dixip		
Situation	**Post-Doc IMEC**	**MCF Université Bretagne Sud**					**Disponibilité** sep.02/janv.04		**CR1 CNRS**	

FIG. 5.1 – Panorama des activités

5.3.3 Projet de structure Fédérative IRCASE

Convaincu de l'existence d'un pôle de compétence en Bretagne dans le domaine des outils et méthodes pour la conception de systèmes sur silicium et de la nécessité d'organiser les recherches afin d'atteindre une masse critique suffisante et une visibilité internationale, j'ai repris avec Bernard Pottier (LESTER/UBO) le projet de structure fédérative IRCASE[1]. Celui-ci a été pensé dans le sens d'une alliance de compétences à différents niveaux du flot de conception, elle tire profit de la complémentarité des équipes de recherche de la région et s'inscrit dans la logique de structures ayant montré leur efficacité telles que l'IMEC en Belgique.

Le montage du dossier, déposé en décembre 2004, s'est traduit par un recensement exhaustif et une classification des thèmes de recherches et collaborations. Les équipes participantes sont citées ci-après, cette liste pourra s'étendre dans le futur à toute équipe désireuse de participer au projet.

- LESTER (UBS et UBO à Lorient et Brest), FRE CNRS ;
- l'IRISA, équipes R2D2, Symbiose (UR1 et ENSSAT à Rennes et Lannion), UMR CNRS et INRIA ;
- l'ENST-Bretagne, laboratoires CNRS LIT et TAMCIC équipe MACS ;
- l'IETR, équipe SPS, UMR CNRS. (INSA, Supelec, UR1) ;
- l'ENSIETA, équipe DTN (à Brest).

Formellement le dossier est présenté au MENESR par trois universités : UBS, UBO, UR1, et appuyé par trois écoles (INSA, ENSIETA, ENST-Bretagne). Ensemble elles représentent une des toutes premières forces de recherche dans ce domaine au niveau national avec près de 120 chercheurs dont 60 doctorants, avec une quinzaine de projets communs financés, un partenariat industriel d'une vingtaine de sociétés dont plusieurs grands groupes, et une visibilité internationale certaine (recensement fait sur les 4 dernières années). Les domaines techniques

[1] *Institut de Recherche en Conception d'Architectures et Systèmes Embarqués*

concernés se situent à l'interface physique des STIC :

- Modélisation de systèmes et d'architectures ;
- Synthèse d'architecture, synthèse de circuits reconfigurables ;
- Compilation, transformations de modèle ;
- Communications numériques, radio-logicielle, traitement du signal et de l'image dans les aspects mise en œuvre ;
- Architectures de traitement, parallélisme ;
- Optimisation des traitements, exploration architecturale ;
- Sécurité, cryptographie.

Le rôle de l'IRCASE :

- Guichet unique du pôle de compétence breton dans le domaine de la conception d'architecture ;
- Coordination des recherches ;
- Plate-forme de test et bases de données applicatives ;
- Diffusion de la connaissance ;
- Veille et représentation.

En tant que fédération scientifique, la structure est organisée autour d'un comité de direction et d'un conseil scientifique. Elle labellise et coordonne des projets tels que ceux qui sont actuellement menés de manière conjointe par les équipes. Le comité de direction joue aussi un rôle de conseil et d'orientation stratégique et propose un portail public commun vis à vis des collectivités et des industriels. Il s'appuie sur une cellule technique qui gère la présentation commune des équipes (serveurs, partages de logiciels), qui assure l'intégration technique des outils logiciels, et l'écoute des industriels notamment via un club de partenaires.

5.3.4 Collaborations internationales

PROSYR Projet CMCU - PROtotypage de SYstèmes mixtes flot de données et/ou Réactifs, Application à la conception des systèmes sur puce, 2003 - 2005, LESTER, ENIS (Tunisie). Thèses en co-tutelles, missions, accueil de stagiaires.

CISS/CTIF-Univ.Aalborg, DK. Les compétences du CISS se situent principalement au niveau applicatif dans le domaine des Télécoms, l'aspect architectural est secondaire alors que la situation est inverse au sein de notre équipe. Outre l'étude commune d'applications télécoms, j'ai mis en place un projet de collaboration plus formelle sur les architectures reconfigurables pour la radio-logicielle lors de ma visite en Août 2005. Elle se traduit pas un stage de master recherche de l'UBS au Danemark et l'utilisation de Design Trotter lors de projets d'étudiants Danois.

EPFL Je collabore également depuis 2001 avec l'équipe de P.Vandergheynst au LTS à l'EPFL autour de l'utilisation de Design Trotter pour l'implantation matérielle et reconfigurable d'une nouvelle technique de compression vidéo appelée Matching Pursuit [27]. Nous avons bâti cette coopération en définissant le projet de post-doc de S.Bilavarn financé par Intel via l'équipe de E.Debes. Le projet a été étendu par la suite à l'étude de normes plus industrielles telles que H264. Cette bourse a été obtenue après une sévère et concurrentielle procédure de sélection. L'outil a été présenté à deux reprises lors de colloques internes chez Intel et le projet a reçu un échos très favorable ce qui a permis à S.Bilavarn de bénéficier exceptionnellement d'une reconduction de contrat. Par ce projet, nous avons contribué aux efforts des partisans (encore marginaux) des technologies reconfigurables chez Intel.

UMASS/USA W.Bulerson associate professeur, a passé quelques mois au LESTER. Après avoir collaboré de manière informelle sur les architectures adaptatives, nous préparons un projet

autour de l'utilisation de μSpider pour générer des NoC comme solution pour l'asynchronisme et la tolérance au fautes. Ce sujet s'intègre dans les préoccupations actuelles de l'UMASS, il est également supporté par Guy Gogniat qui y effectue actuellement une année sabbatique sur le thème de la sécurité appliquée aux architectures reconfigurables.

5.4 Divers

- Membre élu du conseil scientifique du LESTER ;
- Animation avec E.Martin de l'AS CNRS 202 intergiciel pour la radio-logicielle ;
- Membre de la commission de spécialiste 61e section en tant que MCF UBS (2002/2003) et membre extérieur CNRS depuis janvier 2005 ;
- Relecteur pour : IEEE Transactions on VLSI, ACM Trans. on Embedded Computing Systems, IEE Proceedings - Circuits, Devices and Systems, conférences : DAC, ISSS.

Chapitre 6

Conclusion

6.1 Réflexions et critiques

6.1.1 La nécessité du transfert technologique

Le premier point concerne la proximité entre le domaine recherche et les attentes de l'industrie. Les outils et méthodes de conception sont au cœur d'un enjeu économique considérable que l'on mesure à l'aune des contraintes antagonistes qui entrent en conflit. Il s'agit d'une part de la réduction des délais, et donc des coûts de conception, imposée par le marché de masse où se situent les systèmes embarqués et d'autre part de l'accroissement de la complexité des systèmes et du coût d'intégration résultant de l'adoption de nouvelles technologies. De fait, la valeur ajoutée induite par les outils de CAO de manière générale sont une nécessité industrielle pour l'Europe dans un contexte de réduction des coûts de production. Le secteur des systèmes embarqués suit à présent le modèle économique de l'automobile [83].

Cette situation implique que les solutions proposées soient rapidement opérationnelles et surtout qu'elles s'intègrent dans les standards existants. L'étude récente effectuée à l'occasion de DAC 2005 a montré qu'il s'agissait d'une critique récurrente chez les concepteurs : "trop d'outils et de méthodes incompatibles". La conséquence sur les travaux de recherche dans notre domaine est que la pertinence des approches proposées est souvent mesurée d'abord sous l'angle des applications tests et de l'usage de standards avant de l'être sur les contributions scientifiques. Ainsi, la démonstration d'un environnement de codesign impose après la phase de conceptualisation, une lourde charge de développement de l'outil et de spécification d'applications. Ce type d'effort relève pour beaucoup de l'ingénierie et non de la recherche proprement dite. Ce constat souligne le dilemme des laboratoires de recherche. En effet, ceux-ci doivent pour exister dans ce domaine, procéder très rapidement à un rapprochement avec une structure disposant des ressources pour effectuer le transfert technologique où alors accepter des délais de développement qui mettent en péril l'originalité initiale de leurs solutions et donc leur reconnaissance par les publications. Design Trotter, par exemple, n'échappe pas à cette règle et après un effort de développement conséquent l'outil sera amené à un état de démonstrateur raisonnable en septembre 2005. Les développements futurs concernant les interfaces par exemple avec d'autres outils sont à prévoir dans un contexte de collaborations industrielles.

Mon expérience dans la création d'entreprise m'a permis de faire ce type d'analyse depuis l'autre côté du miroir. En effet on observe que le sort de nombre de jeunes entreprises innovantes est une transformation graduelle en société de service classique. Cette évolution est due à une double difficulté. La première est le financement de la recherche et du développement pour offrir un produit fiable à un coût raisonnable. La seconde est le financement de la phase de commercialisation qui s'avère généralement longue et difficile compte tenu du caractère nouveau des solutions proposées. Le recours aux activités d'ingénierie traditionnelle pour financer la recherche est une stratégie nécessaire qui doit être manié avec précaution.

La complémentarité est donc évidente entre les besoins et aspirations des laboratoires de recherches et ceux des entreprises visant l'innovation. Malgré les progrès effectués dans cette relation, notamment au niveau des structures de transfert et de valorisation, on observe qu'en pratique il reste encore du chemin à faire pour que ces échanges vertueux s'activent plus spontanément.

6.1.2 L'écueil du spectre trop large

Outre mon passage dans les projets Gaut (LASTI/LESTER) et Atomium (IMEC puis PowerEscape), j'ai piloté le développement de trois logiciels dans le domaine de la conception de systèmes électroniques : DT, RTDT et μSpider et contribué activement à celui résultant du projet A3S (prototypage reposant sur UML). Dans ce type de projet, la tentation est grande de vouloir sans cesse ajouter de nouvelles contraintes ou paramètres à l'environnement de conception afin de traiter tel ou tel aspect. Je dois reconnaître avec le recul que je procéderais de manière légèrement différente si je devais débuter de tels projets aujourd'hui. En l'occurrence, j'adopterais une approche plus graduelle en évitant de mettre trop de fers au feu simultanément. Dans le cadre du projet RaaR nous étudions différentes applications mais la démarche est relativement focalisée sur des aspects théoriques qui seront peu à peu alimentés et corrigés en fonction des expériences.

6.1.3 Organisation et coopérations

Un autre point me semble important à souligner, il concerne la nécessité de définir des interfaces ou des surfaces d'échanges entre équipes de recherche au sein d'un laboratoire ou d'une structure distribuée. Contrairement à ce que l'on imagine habituellement, la force des habitudes fait que cet aspect ne va pas de soi. L'objectif n'est pas de cantonner les chercheurs dans un domaine, au contraire, mais plutôt de régulièrement re-tracer les contours du périmètre de recherche principal des différents projets. Il ne s'agit pas simplement de présenter son travail et ses résultats mais de faire l'effort de préciser régulièrement ce que sont les hypothèses de travail, les limites et l'horizon des recherches à court et long terme. Ce type de démarche doit s'inscrire dans une politique de gestion de laboratoire au sens large afin que de manière naturelle soient provoquées les coopérations. Le projet IRCASE qui doit être de mon point de vue une priorité politique, s'inscrit dans cette perspective avec le but initial d'éviter la redondance des développements, la mise en commun des expériences et de prouver la pertinence d'un point précis en tirant profit des avancés d'un ensemble complet et cohérent. L'évolution du LESTER et de ses rapprochements régionaux à venir est à penser également dans ce sens.

6.2 Perspectives

6.2.1 Le devenir des acteurs

Revenons sur nos trois principaux personnages : le concepteur, l'usager et le circuit (SOC) et tentons de placer le triptyque en perspective.

Le concepteur. Selon lui les priorités actuelles dans le domaine des systèmes sur silicium convergent vers les thèmes suivants :

- Concevoir encore plus vite et à plus faible coût :
  Réutiliser, Raffiner et Standardiser ;
- Rationaliser la consommation d'énergie en contrôlant les points chauds :
  Maximiser les grandeurs #MIPS/nJ et #bits/nJ ;
- Considérer l'incertitude et le taux d'erreur non négligeable :
  Contrôler ;
- Intégrer l'hétérogénéité des SOC (HF, CAN, Multi-processeurs, NoC, ...) :
  Abstraire.

Si l'on se fie aux prévisions de l'ITRS repris sur la figure 6.1, le concepteur devra élaborer avant 2009 les méthodes et outils qui lui permettront de suivre le rythme de productivité minimum requis, sachant que d'un point de vue économique toute amélioration des prévisions sera saluée. L'exploration, en raccourcissant la phase de sélection des architectures, constitue une des voies qui lui permettra d'atteindre sinon de dépasser ces objectifs. En conférant au SOC des facultés d'auto-configuration il pourra également gagner du temps lors de la conception.

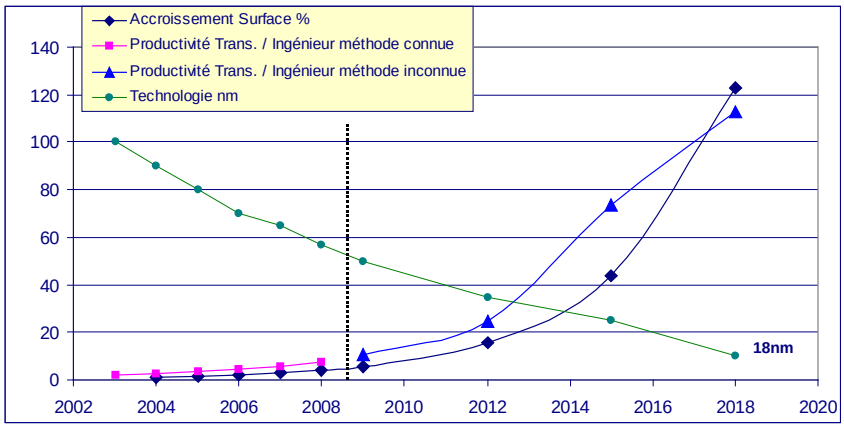


FIG. 6.1 – Prévisions de l'ITRS

L'usager. De son côté, il n'a cure de la manière dont est réalisé le circuit (système enfoui) à sa disposition mais souhaite par contre qu'il rende en toutes circonstances et tous lieux de réels services c'est à dire qu'il lui simplifie la vie. De la même manière que l'on ne s'émerveille plus devant une ampoule qui s'allume sur commande, l'usager considérera comme naturel le fait de recevoir et d'échanger en permanence des informations avec ses semblables ou son environnement. Le spectre des applications n'est limité que par l'imagination et la réponse à un besoin qu'il s'agisse de communication, d'accès à l'information pertinente géo-localisée, de loisirs multimédia, de réalité enrichie, de surveillance et de contrôle médical, d'accessibilité sécurisée etc. La simplification impose que le système soit fiable et puisse s'adapter aux exigences de l'usager. L'omniprésence et la généralisation des services sous-entendent de fait que le coût du système

soit négligeable. Il signifie également qu'un système électronique sera dans le futur rarement isolé, éloigné de ses semblables.

Ainsi, le concepteur n'aura que très peu de temps pour réaliser un système qui n'utilise que le minimum d'énergie nécessaire pour rendre à ses congénères humains ou artificiels, les services attendus. Dans ces circonstances trois contraintes apparaissent clairement au centre de la mire :

- Le système doit s'adapter ;
- Le système doit coopérer ;
- Le système doit donc être conscient de son environnement et de son état.

Le SOC À court terme son évolution est tracée pour les 15 ans à venir. Si l'on se réfère aux perspectives de l'ITRS (6.1), il continuera de voir diminuer la taille de ses transistors. Au delà les candidats à la succession affichent déjà leurs ambitions, il s'agit notamment des SET (*Single Electrons Transistor*), des nano-structures de carbone, des interrupteurs moléculaires, du calcul quantique sur un support organique. On peut supposer que l'espèce la mieux adaptée à la situation sera opérationnelle pour prendre le relais au cours de la décennie 2020/30.

Pour autant si comme l'écrit Hubert Reeves "devenir Adulte c'est apprendre à vivre dans le doute et l'incertitude" alors le SOC sortira définitivement du monde de l'enfance et de l'insouciance avant même de quitter celui du CMOS traditionnel. En effet les sources d'incertitudes et d'erreurs seront nombreuses et le SOC devra en être conscient pour s'auto-contrôler. Elles ne se situent plus simplement au niveau système au travers du WCET des tâches variant en fonction des données traitées ou au niveau de l'architecture par exemple dans la gestion des caches mais se niche au niveau physique. Il s'agit par exemple des délais des interconnexions[1] variant de manière significative en fonction de la température, du phénomène de *crosstalk* et de la localisation dans le circuit. De même la marge de bruit réduite en raison de celle des tensions d'alimentations ne permettra pas de garantir complètement le bon fonctionnement du système y compris dans un environnement faiblement bruité. Par ailleurs l'accroissement exponentielle du taux de rebut en sortie de production imposera de faire appel à l'auto-configuration pour réaliser une fonction avec un taux non nul de transistors défectueux. Afin de limiter la consommation d'énergie et la température, l'architecture du SOC continuera d'évoluer vers le multi-processeurs et l'exploitation du parallélisme aux niveaux *thread* et tâches. L'efficacité énergétique devrait conduire à une forte spécialisation des différents cœurs de processeurs embarqués à faible coût. Une partie d'entre eux seront dédiés aux standards en vigueur en matière de gestion d'entrées / sorties (protocoles IP, communication sans fil), de traitements graphiques, de compression vidéo, de sécurité. Les autres devront se configurer en fonction des besoins. Au moins, deux types de solutions peuvent cohabiter à ce niveau. La première est la reconfiguration du jeu d'instructions et des ressources des processeurs élémentaires en fonction des besoins. La seconde est la coopération entre processeurs élémentaires dans le but d'exploiter au mieux le parallélisme disponible. Le principe de la coopération ne s'arrête aux frontières du SOC, en effet dans un contexte d'électronique ambiant les SOC pourront coopérer pour améliorer le service qu'ils pourront collectivement remplir.

[1] plus de 100ps pour franchir 1mm de cuivre au delà de 90nm, source S.Borkar Intel

6.2.2 Directions de recherches personnelles

Les convergences entre les besoins exprimés par les trois acteurs précédents sont séduisantes sur le papier : lorsqu'un ou plusieurs systèmes rencontrent un service c'est à dire une série d'applications à exécuter, ceux-ci explorent leur espace de configuration propre et collectif et décident de qui va faire quoi et le font. En pratique évidemment les verrous scientifiques surgissent à tous les niveaux que ce soient à propos des règles et langages de programmation, de la mise en œuvre même du concept de coopération, des choix architecturaux, des règles de communications, des techniques de configuration, des systèmes d'exploitations, des méthodes d'exploration, des méthodes d'estimations, des règles de décisions etc.
Comme je le précisais plus haut, lors de la définition d'un projet de recherche, le premier travail consiste à définir la surface d'échanges entre les équipes, les laboratoires et les industriels concernés. L'objectif est d'ouvrir un espace de recherche qui sera plus ou moins vaste et plus moins précis en fonction des aspirations de chacun. Une voie que je privilégie dans l'immédiat repose sur le contrôle dynamique de l'adéquation application / système en supposant vérifiée l'hypothèse de travail suivante : "les architectures auto-reconfigurables offrant un rendement énergétique satisfaisant existent". J'ajoute qu'elle disposent d'outils de synthèse et de compilation opérationnels. On peut dans un premier temps laisser en suspens la nature de ces systèmes en utilisant comme pis-aller à des fins expérimentales ce qui dans l'existant se rapproche au mieux de l'hypothèse. La recherche de l'adéquation dans ce contexte suppose principalement mais non exclusivement :

- De modéliser l'espace de conception ou de configuration ;
- De définir les paramètres pertinents prévalant lors de la prise de décision (disponibilité des ressources, durée de vie, énergie, qualité de service, sécurité, fiabilité, temps-réel etc.) ;
- De préciser le moyen d'obtenir ou d'estimer les données (ex. évolution batterie) nécessaires à la décision ;
- De procéder à une séparation des niveaux de décisions (local ... global) en fonction des dépendances entre les tâches, des ressources qu'elles partages et de leur degré de spécialisation.
- ...

Cette rapide carte des recherches se situe dans la continuité du projet RaaR et reste confinée au sein du SOC multi-processeurs à un niveau proche du matériel. Un autre aspect est à prendre en considération, celui du lien avec le niveau *middleware* dans le but de mettre en correspondance l'appel d'une fonction ou d'une méthode avec les ressources matérielles nécessaires. Il est clair que l'objectif n'est pas de refaire ce qui existe dans le domaine des composants en informatique, nous sommes là face à un cas concret de surface d'échanges intéressante. Celle-ci offre l'opportunité de créer un pont vers le monde du logiciel où l'on œuvre à la mise au point d'approches par assemblage de composants (programmation *in the large*) éventuellement adaptables (entre autres le Valoria au sein de l'UBS). Ces méthodes font généralement abstraction du matériel sous-jacent et donc des possibilités d'optimisation de l'adéquation algorithme architecture. Ce type de problématique est particulièrement sensible dans le domaine de la radio-logicielle par exemple, où l'objectif serait de considérer des composants qui puissent s'implanter sur GPP, DSP ou FPGA à l'aide d'un modèle de spécification unifié. Si la migration de l'approche composant du GPP vers le DSP est en cours, le chemin vers le FPGA reste à faire. C'est un des objectifs du projet ITEA SPICES auquel je collaborerai s'il est validé.
Dans un deuxième temps l'objectif est de considérer un ensemble de systèmes embarqués connectés entre eux. Le type de connexion peut prendre plusieurs formes et se matérialiser par des communications sans fil entre appareils mobiles (WIFI, WIMAX, UWB, BLUETOOTH, ZIGBEE, WUSB, etc.), par des communications sur courant porteurs dans un environnement professionnel ou domestique, ou encore par des connexions IP à travers Internet. Dans ce cadre la notion de ressources doit s'élargir et se hiérarchiser. Au sein de DT par exemple l'UAR spécifie les

ressources sous la forme d'opérateurs de granularité variable allant des opérations arithmétiques et logiques jusqu'à l'accélérateur dédié à une fonction particulière (DCT, AES). En augmentant la granularité, la définition des ressources évoluera et devra préciser notamment la bande passante requise, les tailles des mémoires de données et de programme, les méthodes ou fonctions disponibles, les services de l'OS etc. Un langage comme RSL *resource specification language* [84] utilisé dans le domaine du *grid computing* permet de spécifier les contraintes de temps et de mémoire, il pourrait être enrichi ou localement complété pour être utilisé à des fins de reconfigurations logicielles et matérielles. Ainsi, nous pouvons imaginer que le téléchargement d'une tâche sur un SOC provoque automatiquement l'auto-reconfiguration du SOC récepteur à l'issue des deux étapes suivantes. La première est l'analyse de l'adéquation entre l'application et les architectures candidates, la seconde est une l'étape de décision . Ce type de mise en œuvre est envisageable à moyen terme.
À un niveau d'abstraction supérieur se pose la question de la répartition des applications au sein du système distribué. La réponse relève de stratégies de coopération déjà étudiées par ailleurs. Il sera nécessaire de considérer de nouvelles surfaces d'échanges et de faire preuve de curiosité puis d'intérêt à l'égard de domaines connexes issus de disciplines a priori éloignées. Il s'agit notamment de l'informatique traitant des systèmes coopératifs à base d'agents par exemple, de celui des réseaux spontanés ad hoc, des principes d'auto-adaptation connus en biologie moléculaire et du domaine des sciences cognitives.
Le SOC reconfigurable peut être vu comme le maillon élémentaire d'un système distribué au sein duquel s'enchaînent des séries de décisions de configuration depuis les plus globales aux jusqu'aux plus locales. Cette vision des choses provoque un étonnant parallèle avec la biologie moléculaire potentiellement candidate à la réalisation même des SOC. En effet, un ensemble de processeurs inter-connectés et reconfigurables au point de pouvoir se spécialiser pour effectuer efficacement une fonction particulière s'apparente aux réseaux de cellules souches tels que ceux qui furent à l'origine du concepteur et de l'usager.

Chapitre 7

Publications

Revues internationales avec comité de lecture

- Y.Le Moullec, J-Ph.Diguet, N.Ben Amor, T. Gourdeaux, J-L.Philippe, *Algorithmic-level Specification and Characterization of Embedded Multimedia Applications with Design Trotter*, Jour. of VLSI Signal Processing, Springer (formely Kluwer Academic Publishers), to appear, Accepted in Feb. 2005.
- N.Ben Amor, Y.Le Moullec, J-Ph.Diguet, J-L.Philippe and M Abid, *Design of multimedia processor based on metric computation*, Jour. Advances in Engineering Software, Elsevier Science, vol. 36, no.7, july, 2005.
- S.Bilavarn, E.Debes, P.Vandergheynst and J-Ph.Diguet, *An optimization Study for Media Delivery : Processor Issues*, Jour. of VLSI Signal Processing, Springer (formely Kluwer Academic Publishers), vol.41, no.2, sep. 2005
- Y.Le Moullec, J-Ph.Diguet, T.Gourdeaux, J-L.Philippe, *Design Trotter : System-Level Dynamic Estimation Task a 1st step towards platform architecture selection*, 2nd review step, minor revisions, in (JEC) Journal of embedded computing, Cambridge Int. Science Pub, Oct. 2004.
- B.Bouallegue, R.Djemal, G.Hattab, J-Ph.Diguet, J-L.Philippe and R.Tourki, Video Coding Protocol Architecture over High Speed Network, IEE Proceedings-communications, feb., 2004.
- H.Guesmi, R.Djemal, B.Bouallegue, J-Ph.Diguet and R.Tourki, *High performance architecture of integrated protocols for encoded video application*, Jour. of Computer standards & interfaces, Elsevier Science, dec, 2003.
- J-Ph.Diguet, D.Chillet, O.Sentieys, *A Framework for High Level Estimations of Signal Processing VLSI Implementations*, Jour. of VLSI signal processing, Kluwer Academic Publishers, Vol . 25, pp 261-284, Jul. 2000.
- S.Wuytack, J.Ph.Diguet, F.Catthoor, H.De Man, *Formalized methodology for data reuse exploration for low-power hierarchical memory mappings*, IEEE Trans. on VLSI Systems, Vol . 6, No.4, pp 529-537,Dec. 1998.
- P.Amayenc, J-Ph.Diguet, M.Marzoug, T.Tani ; *A class of single and dual-frequency algorithms for rain-rate profiling from a spaceborne radar. Part II : Tests from airborne radar data* ; Jour. of Atmospheric and Oceanic Technology ; Feb. 96, Vol. 13, pp 142-164.
- J-L.Philippe, O.Sentieys, J-Ph.Diguet, E.Martin ; *Novel approaches in logic and architecture synthesis, chapter Synthesis : From digital signal processing specifications to layout*, ed. Chapman & Hall, pp 307-313, 1995.

Revues nationales avec comité de lecture

- Y.Le Moullec, J-Ph.Diguet et J-L.Philippe, *Estimation du parallélisme au niveau système*

pour l'exploration de l'espace de conception de systèmes enfouis, Technique et Science Informatiques (RSTI-TSI), Vol. 22, n°3/2003, pp. 315-349, Hermes-Science.
- J-Ph.Diguet, J-L.Philippe, O.Sentieys, E.Martin, Revue TS, Ed. Spéciale sur l'Adéquation Algorithmes Architectures, *Mesures probabilistes de l'Adéquation Algortihmes Architectures*, vol.14, No. 6, mars 1997.
- D.Chillet, J-Ph.Diguet, J-L.Philippe, O.Sentieys, *Méthodologie de conception des unités de mémorisation appliquée au traitement du signal temps réel*, Revue TSI, Vol.16, No. 4,1997.

Conférence internationales avec comité de lecture

- S.Evain, J-Ph.Diguet, D.Houzet, *μSpider : A CAD Tool for Efficient NoC Design*, IEEE NORCHIP 2004, Oslo, NORWAY, November 8-9, 2004.
- S.Evain, J-Ph.Diguet, D.Houzet, *A Generic CAD Tool for Efficient NoC Design*, IEEE Int. Symp. on Intelligent Signal Processing and Communication Systems (ISPACS), Seoul, Korea, November 18-19, 2004.
- Y.Le Moullec, N.Ben Amor, J-Ph.Diguet and P.Koch, *Follow-up Modelling for Wireless Personal Communication Systems*, 7th Int. Symp. on Wireless Personal Multimedia Com., sep. Italy, 2004.
- A.Delautre, J-E.Goubard, G.Gogniat, S.Rouxel, J-Ph. Diguet, C.Moy and N.Bulteau, *UML profiles towards waveform performances verification*, Wireless World Research Forum (WWRF), Oslo, Jun, 2004.
- N.Ben Amor, Y.Le Moullec, J-Ph.Diguet, J-L.Philippe, M.Abid, *Design of an adaptive multimedia system*, 16th Int. Conf on Microelectronics, Tunis, Tunisia, Dec.,
- 2004 H.Tmar, A.Azzedine, J-Ph.Diguet, J-L.Philippe, M.Abid, *RTDT a static QoS Manager, RT scheduling, HW/SW partitioning CAD Tool*, 16th Int. Conf on Microelectronics, Tunis, Tunisia, Dec., 2004
- Y.Le Moullec, N.Ben Amor, J-Ph.Diguet, J-L.Philippe, M.Abid, *Multi-granularity Metrics For The Era Of Strongly Personalized SOCs*, Design Automation & Test in Europe (DATE), March, Munich, 2003.
- S.Bilavarn, E.Debes, P.Vandergheynst and J-Ph.Diguet, *Reconfigurable Coprocessor for Media Streaming*, IEEE Int. Conf. On Multimedia and Expo (ICME), June, Taiwan, 2004.
- C.Moy, M.Raulet, S.Rouxel, J-Ph.Diguet, G.Gogniat, P.Desfray, N.Bulteau, J-E.Goubard, Y.Denef, *UML Profiles for Waveform Signal Processing Systems Abstraction*, SDR Forum Technical Conference, Phoenix, USA, novembre 2004
- M.Auguin, K.Ben Chehida, J-Ph.Diguet, X.Fornari, A-M.Fouilliart, C.Gamrat, G.Gogniat, P.Kajfasz, Y.Le Moullec, *Partitioning and CoDesign tools & methodology for Reconfigurable Computing : the EPICURE philosophy*, 3rd Int. Work. on Systems, Architectures, Modeling Simulation (SAMOS03), Greece, Jul., 2003.
- T.Gourdeaux, J-Ph.Diguet and J-L.Philippe, *Design Trotter : Inter-function Cycle Distribution Step*, 11th Int. Conf. RTS embedded Systems, Paris, April 2003.
- Y.Le Moullec, P.Koch, J-Ph.Diguet, J-L.Philippe, *Design Trotter : Building and Selecting Architectures for Embedded Multimedia Applications*, , IEEE Int. Symp. on Consumer Electronics (ISCE03), Dec. 3-5, 2003, Sydney, Australia
- Y.Le Moullec, J-Ph.Diguet and J-L.Philippe, *Design-Trotter : a Multimedia Embedded Systems Design Space Exploration Tool*, IEEE Int. Work. on Multimedia Signal Processing, St. Thomas, US, Dec.02.
- A.Azzedine, J-Ph.Diguet and J-L.Philippe, *Large Exploration for HW/SW Partitioning of Multirate and Aperiodic Real-Time Systems*, 10th IEEE/ACM Int. Symp. on Hardware/Software Co-Design, Estes Park USA, May 6-8 2002.

- Y.Le Moullec,J-Ph.Diguet, D.Heller and J-L.Philippe, *Fast and Adaptive Data-flow and Data-transfer Scheduling for Large Design Space Exploration*, ACM GLSVLSI 2002, April 18-19,New-York, USA.
- Y.Le Moullec, P.Koch and J-Ph.Diguet, *A Power Aware System-Level Design Space Exploration Framework*, IEEE DDECS 2002, April 17-19, Brno, Czech Republic.
- L.Bossuet, G.Gogniat, J-Ph.Diguet, J-L.Philippe, *A Modeling Method for Reconfigurable Architectures*, Int. Workshop on System-on-Chip for Real-Time Applications, July 6-7, 2002, Banff, Canada.
- R. Djemal, B.Bouallegue, J-Ph.Diguet and R.Tourki, *A Flow Control approach for encoded video applications over ATM Network*, 6th IEEE Symp.On Computers & Communications, ISCC, 2001.
- Y.Le Moullec, J-Ph.Diguet, J-L.Philippe, *A Scheduling Framework for System-Level Estimation*, 7th IEEE Int. Conf. on Electronics, Circuits & Systems, Dec., 2000, Lebanon.
- S.Montfort, J-Ph.Diguet, J-L.Philippe, *ASIP Design Methodology for Telecommunications*, Int. Conf. on Information Society (IS2000), Nov.2000, Aizu, Japan.
- J-Ph.Diguet, G.Gogniat, P.Danielo, M.Auguin, J-L.Philippe, *The SPF model*, Forum on Design Language, FDL, Tübingen, Germany,sep.2000.
- H.Thomas, J-Ph.Diguet, J-L.Philippe, *A methodology for an application profiling at system level*, IEEE Work. on Signal Processing Systems (SiPS), Taiwan, 20-22 Oct. 1999.
- J-Ph.Diguet, S.Wuytack, F.Catthoor, H.De Man, *Formalized methodology for data reuse exploration in hierachical memory mappings*, ACM/IEEE Int. Symp. on Low Power Elec. & Design (ISPLED), Monterey, USA aug, 1997.
- J-Ph.Diguet, O.Sentieys, D.Chillet, J-L.Philippe, IEEE ICASSP, *VLSI High Level Synthesis of Fast Exact Least Mean Square Algorithms*, Munich, 1997.
- O.Sentieys, J-Ph.Diguet, J-L.Philippe, E.Martin ; *Hardware Module Selection for Real Time Pipeline Architectures using Probabilistic Cost Estimation* ; ASIC 96 ; Rochester, NY, 23-27 sep.96.
- S.Gailhard, N. Julien, J-Ph. Diguet, E. Martin, *Methods to transform easily classical architectural synthesis tools to low power ones*, in 8th Great Lakes Symp.on VLSI, Louisiana, USA, Feb. 1998.
- O.Sentieys, D.Chillet, J-Ph.Diguet, J-L.Philippe ; *Memory Module Selection for High Level Synthesis* ; in IEEE Workshop on VLSI S.P. ; San-Francisco, USA 1996.
- J-Ph.Diguet, O.Sentieys, J-L.Philippe, E.Martin ; *Probabilistic Resource Estimation for pipeline architecture* ; IEEE Workshop on VLSI S.P. ; Sakai, Japan Oct. 95 ; pp 217-226.
- J-Ph.Diguet, O.Sentieys, J-L.Philippe, E.Martin ; *How Specify an Algorithm in VLSI architectural Synthesis, a Vocal Coding Application* ; IEEE Workshop on VLSI S.P. ; San Diego, Oct.94 ; pp346-355.
- J-L.Philippe, D.Chillet, O.Sentieys, J-Ph.Diguet, Trieste, Italy, EUSIPCO, *Memory Aspect in Signal Processing and HLS Tool : some Results*, sep. 1996.
- J-L.Philippe, O.Sentieys, J-Ph.Diguet and E.Martin ; *Adequacy Architecture Algorithm, an experiment in signal processing using FPGAs* ; VHDL-Forum for CAD, Spring'94 ; pp 47-56.

Brevets

- J-Ph.Diguet, P.Danielo, *Procédé et système d'échange d'informations point à point par l'intermédiaire d'un réseau de diffusion*, FR 03 11556, 2 oct. 2003, dépôt PCT en cours.
- J-Ph.Diguet, P.Danielo, *Écran micro-électro-mécanique*, FR 0100237, 9/01/01.
- S.Wuytack, J-Ph.Diguet, F.Catthoor, H.De Man, *Method for determining an optimized memory organization of a digital device* ; US Patent No. 09/360,042, July 99.

Divers

- J-Ph.Diguet, Y.Eustache, N.Ben Amor and S.Hammami, *Feedback control modelling for learning recongurable embedded systems*, Invited paper, RESOSOC, june 2005, Montpellier, France.
- J-Ph.Diguet, *System On Chip Design Space Exploration*, Internal CISS meeting, Univ.Aalborg, Aug. 2004.
- J-Ph.Diguet, *Besoins identifiés pour les OS dans les systèmes reconfigurables*, Journée thèmes émergents du chapitre français de l'ACM-SIGOPS Systèmes temps réel embarqués, 18 novembre 2004, IRCCYN, Nantes.

récentes soumissions en revue :

- H.Tmar, J-Ph.Diguet, A.Azzedine, M.Abid and J-L.Philippe, *RTDT : a Static QoS Manager, RT Scheduling, HW/SW Partitioning CAD Tool*, The Microelectronics Journal, Elsevier, submited April 2005.
- J-Ph. Diguet, Y.Le Moullec, S.Bilavarn, K.Ben Chehida, M.Auguin, C.Gamrat, G.Gogniat, J-L.Philippe, X.Fornari, A-M. Fouillart, P.Kajfasz, *EPICURE : A Partitioning and Co-Design Framework For Reconfigurable Computing*, ACM Transactions on Embedded Computing Systems, Special Issue, Submitted in Feb 2005.

Bibliographie

[1] "Analyse the futur," http ://www.idc.com/.

[2] "Le pc à 100 dollars est pour bientôt," http ://www.lexpansion.fr/art/126.0.131221.0.html.

[3] P.Amayenc, J-Ph.Diguet, M.Marzoug, and T.Tani, "A class of single- and dual-frequency algorithms for rain-rate profiling from a spaceborne radar. part ii : Tests from airborne radar measurements," *Jour. of Atmospheric and Oceanic Technology*, vol. 13, no. 1, pp. 142–164, Feb. 1996.

[4] J-Ph.Diguet, *Estimation de complexité et transformations d'algorithmes de traitement du signal pour la conception de circuits VLSI*, Ph.D. thesis, Univ.Rennes I, LASTI/ENSSAT, Lannion, oct 1996.

[5] J-Ph.Diguet, D.Chillet, and O.Sentieys, "A framework for high level estimations of signal processing implementations," *Jour. of* VLSI *Signal Processing Systems*, vol. 25, no. 25, pp. 261–284, July 2000.

[6] S.Wuytack, J-Ph.Diguet, F.Catthoor, and H.De man, "Formalized methodology for data reuse exploration for low-power hierarchical memory mappings," IEEE *Trans. on* VLSI *Systems*, vol. 6, no. 4, pp. 529–537, Dec. 1998.

[7] L.Cai and D.Gajski, "Transaction level modeling : An overview," in *CODES+ISSS*, Newport Beach, USA, oct 2003.

[8] A. K.Deb, A.Jantsch, and J.Oberg, "System design for dsp applications in transaction level modeling paradigm," in *DAC*, San Diego, USA, June 2004.

[9] M. Santarini, "Codesign tools aren't ready for soc era," EE Times, September 2001.

[10] "Esterel studio web page," http ://www.esterel-technologies.com/.

[11] K.Ben Chehida, *Méthodologie de partitionnement logiciel matériel pour plateformes reconfigurables dynamiquement*, Ph.D. thesis, Univ. de Nice, I3S, Sophia Antipolis, France, 2004.

[12] Y.Le Moullec, N.Ben Amor, J-Ph.Diguet, J-L.Philippe, and M.Abid, "Multi-granularity Metrics For The Era Of Strongly Personalized SOCs," in *Design Automation and Test in Europe (DATE)*, Munich, Germany, Mar. 2003.

[13] Y.Le Moullec, N.Ben Amor, J-Ph.Diguet, T.Gourdeau, and J-L.Philippe, "Algorithmic-level specification and characterization of embedded multimedia applications with design trotter," *to appear in Jour. of VLSI Signal Processing*, vol. -, no. -, pp. –, - 2005.

[14] Y.Le Moullec, *Aide à la conception de systèmes sur puce hétérogènes par l'exploration paramétrable des solutions au niveau système*, Ph.D. thesis, Univ. de Bretagne Sud, LESTER, Lorient, France, Apr. 2003.

[15] Y.Le Moullec, J-Ph.Diguet, D.Heller, and J-L.Philippe, "Fast and Adaptive Data-flow and Data-transfer Scheduling for Large Design Space Exploration," in *ACM/SIGDA Great Lake Symp. on VLSI* , NY, USA, Apr. 2002.

[16] Y.Le Moullec, D.Heller, J-Ph.Diguet, and J-L.Philippe, "Estimation du parallélisme au niveau système pour l'exploration de l'espace de conception de systèmes enfouis," *Technique et Science Informatiques (RSTI-TSI)*, vol. 22, no. 3, pp. 315–349, 2003.

[17] Y.Le Moullec, J-Ph.Diguet, T.Gourdeaux, and J-L.Philippe, "Design trotter : System-level dynamic estimation task a 1st step towards platform architecture selection," *submitted to Jour. of embedded computing (JEC), Cambridge Int. Sc. Pub*, vol. 2nd review step, 2005.

[18] D-J.Wang and Y.H.Hu, "Rate optimal scheduling of recursive DSP algorithms by unfolding," IEEE *Trans. on Circuits and Systems*, vol. 41, pp. 672–675, october 1994.

[19] S. Bilavarn, *Exploration Architecturale au Niveau Comportemental - Application aux FPGAs*, Ph.D. thesis, Univ. de Bretagne Sud, LESTER, Lorient, 2002.

[20] Y.Le Moullec, J-Ph.Diguet, and J-L.Philippe, "Design-Trotter : a Multimedia Embedded Systems Design Space Exploration Tools," in *IEEE Work. on Multimedia Signal Processing (MMSP02)*, Dec. 2002.

[21] F.Balasa, F.Catthoor, and H.De Man, "Background memory area estimation for multidimensional signal processing systems," IEEE *Trans. on* VLSI *Systems*, vol. 3, no. 2, June 1995.

[22] N.Dutt G.Grun and F. Balasa, "System level memory size estimation," Tech. Rep., University of California, Irvine, 1997.

[23] M.Auguin, K.Ben Chehida, J-Ph.Diguet, X.Fornari, A-M.Fouilliart, C.Gamrat, G.Gogniat, P.Kajfasz, and Y.Le Moullec, "Partitioning and CoDesign tools & methodology for Reconfigurable Computing : the EPICURE philosophy," in 3^{rd} *Int. Work. on Systems, Architectures, Modeling Simulation (SAMOS03)*, Greece, July 2003.

[24] S.Bilavarn, G.Gogniat, and J-L.Philippe, "FPGA area time power estimation for DSP applications," in *Int. Conf. on Signal Proc. Appli. & Techno.* (ICSPAT), Dallas, USA, Oct. 2000.

[25] L. Bossuet, *Exploration de l'espace de conception des architectures reconfigurables, 10 Septembre 2001. pdf*, Ph.D. thesis, Univ. de Bretagne Sud, LESTER, Lorient, 2004.

[26] Y.Le Moullec, J-Ph.Diguet, and P.Koch, "A Power Aware System-Level Design Space Exploration Framework," in *IEEE DDECS*, Brno, Czech Republic, Apr. 2002.

[27] S.Bilavarn, E.Debes, P.Vandergheynst, and J-Ph.Diguet, "An optimization study for media delivery :processor issues," *Journal of VLSI Signal Processing*, accepted in 2004.

[28] Y.Le Moullec, S.Christensen, W.Chenpeng, P.Koch, and S.Bilavarn, "Fast system-level design of wireless applications," in *WPMC*, Aalborg, Denmark, 2005.

[29] L.Bossuet, G.Gogniat, and J-L.Philippe, "Generic design space exploration for reconfigurables architectures," in 12^{th} IEEE *Reconfigurable Architectures Workshop, (RAW)*, Denver, USA, april 2005.

[30] D.Sciuto, F.Salice, L.Pomante, and W.Fornaciari, "Metrics for design space exploration of heterogeneous multiprocessor embedded systems," in 10^{th} *Int. Symp. on* H/S *Codesign*, Estes Park, USA, May 2002.

[31] P.G.Paulin, C.Pilkington, and E.Bensoudane, "Network processing challenges and an experimental npu platform," in *Design, Automation and Test in Europe conf. (DATE)*, 2003.

[32] G.Yang, Y.Watanabe, F.Balarin, and A.Sangiovanni-Vincentelli, "Separation of concerns : Overhead in modeling and efficient simulation techniques," in 4^{th} *ACM Int. Conference on Embedded Software (EMSOFT'04)*, September 2004.

[33] T.Stefanov, C.Zissulescu, A.Turjan, B.Kienhuis, and E.Deprettere, "System design using kahn process networks : The compaan/laura approach," in *Design, Automation and Test in Europe conf. (DATE)*, 2004.

[34] T.Givargis, F.Vahid, and J.Henkel, "System-level exploration for pareto-optimal configurations in parameterized systems-on-a-chip," in *ICCAD*, Nov. 2001.

[35] P.Yang, C.Wong, P.Marchal, F.Catthoor, D.Desmet, D.Verkest, and R.Lauwereins, "Energy-aware runtime scheduling for embedded-multiprocessor socs," *IEEE Design & Test*, vol. 18, no. 5, pp. 46–58, 2001.

[36] C.Wong, P.Marchal, and P.Yang, "Task concurrency management methodology to schedule the mpeg4 im1 player on a highly parallel processor platform," in 9^{th} *Int. Symp. on* H/S *Codesign*, New York, NY, USA, 2001, pp. 170–177, ACM Press.

[37] M.Abid J-L.Philippe I.Maalej, G.Gogniat, "High level analysis of multiprocessor system on chip," in *Embedded RT Systems Implementation workshop in* IEEE *Int. Real-Time Systems Symposium (RTSS)*, Lisbon, Portugal, Dec. 2004.

[38] A.Dasdan, D.Ramanathan, and R.K.Gupta, "Rate derivation and its application to reactive real-time embedded systems," in *35th* ACM/IEEE *Design Automation Conf.*, San Francisco, USA, 1998.

[39] A.Azzedine, *Outil d'analyse et de partitionnement/ordonnancement pour les systèmes temps réel embarqués*, Ph.D. thesis, Univ. Bretagne Sud, LESTER, Lorient, june 2004.

[40] A.Azzedine, J-Ph.Diguet, and J-L.Philippe, "Large exploration for hw/sw partitioning of multirate and aperiodic real-time systems," in *10th Int. Symp. on* H/S *Codesign*, Estes Park, USA, May 2002.

[41] H.Tmar, J-Ph.Diguet, A.Azzedine, M.Abid, and J-L.Philippe, "Rtdt : a static qos manager, rt scheduling, hw/sw partitioning cad tool," in 16^{th} *Int. Conf. on Microelectronics (ICM)*, Tunis, Tunisia, Dec. 2004.

[42] J.W.Tschanz, S.G.Narendra, Y.Ye, B.A.Bloechel, S.Borkar, and V.De, "Dynamic sleep transistor and body bias for active leakage power control of microprocessors," IEEE *Jour. of Solid-State Circuits*, vol. 38, no. 11, pp. 1838–1845, 2003.

[43] http ://www.artist embedded.org/Overview/, "IST ARTIST," 2002.

[44] S.Rouxel, J-Ph.Diguet, N.Bulteau, J.Carre-Gourdin, J-E.Goubard, and C.Moy, "UML framework for PIM and PSM verification of SDR systems," in *SDR Forum, technical conf.*, Orange, CA, USA, Nov. 2005.

[45] M.Auguin, L.Capella, F.Cuesta, and E.Gresset, "CODEF : a system level design space exploration tool," in *ICASSP*, Salt Lake City, USA, May 2001, pp. 1031–1034.

[46] R.Kamden, A.Fonkua, and A.Zenatti, "Hardware/software partitioning of multirate system using static scheduling theory," in IEEE *Int. Conf. on Computer Design*, Texas, Sept. 1999.

[47] B.P.Dave, G.Lakshminarayana, and N.K.Jha, "COSYN : Hardware-software co-synthesis of heterogeneous distributed embedded systems," IEEE *Trans. on Software Engineering*, vol. 7, no. 1, Mar. 1999.

[48] H.Oh and S.Sha, "A hw/sw co-synthesis technique based on heterogeneous multiprocessor scheduling," in *7th Int. Work. on textsch/s Codesign*, Roma, May 1999.

[49] T.Y.Yen and W.Wolf, "Sensitivity-driven cosynthesis of distributed embedded systems," in 8^{th} IEEE/ACM *Int. Symp. on System Synthesis*, Cannes, France, Sept. 1995.

[50] T.Y.Lee, P.A.Hsiung, and S.J.Chen, "Hardware-software multi-level partitioning for distributed embedded multiprocessor systems," *IEICE Trans. Fundamentals of Electronics, Communications, and Computer Sciences*, vol. E84-A, no. 2, pp. 614–626, Feb. 2001.

[51] P.Eles, Z.Peng, K.Kuchcinski, and A.Doboli, "System level hardware/software partitioning based on simulated annealing and tabu search," *Kluwer Journal on Design Automation for Embedded Systems*, vol. 2, no. 1, pp. 5–32, Jan. 1997.

[52] P.Guitton-Ouhamou, H.Ben Fradj, C.Belleudy, and S.Nikolaidis, "Low power co-design tool and power optimization of schedules and memory system," in *PATMOS*, Santorini, Greece, 2004, pp. 603–612.

[53] J.Luo and N.K.Jha, "Static and dynamic variable voltage scheduling algorithms for real-time heterogeneous distributed embedded systems," in *ASP-DAC/VLSI*, 2002, p. 719.

[54] S.Montfort, J-Ph.Diguet, and J-L.Philippe, "Asip design methodology for telecommunications," in *Int. Conf. on Information Society (IS2000)*, Aizu, Japan, Nov. 2000.

[55] H.Guesmi, R.Djemal, B.Bouallegue, J-Ph.Diguet, and R.Tourki, "High performance architecture of integrated protocols for encoded video application," *Computer standards and interfaces, Elsevier Science*, Dec. 2003.

[56] B.Bouallegue, R.Djemal, G.Hattab, J-Ph.Diguet, J-L.Philippe, and R.Tourki, "Video coding protocol architecture over high speed network," *IEE Proceeding communications*, Feb. 2004.

[57] B.Bouallegue, R.Djemal, H.Guesmi, R.Tourki, and J-Ph.Diguet, "A flow control approach and interleaving method for real-time application in high-speed network," *Dedicated Systems e-Magazine*, vol. Q1, 2003,
http ://www.dedicated-systems.com/Magazine/emagazine/fulltext/2003q1_4.pdf.

[58] A.Mehaoua and R.Boutaba, "Performance analysis of cell discarding technique for best effort video communication over atm networks," *Computer Networks and ISDN Systems*, vol. 29, no. I7-I8, pp. 2021–2037, Feb. 1998.

[59] Y-W.Lu, G.K.Yeh, and J.B.Burr, "A 15mw 1.6gb wormhole data router for 2d mesches," in *Int. Symp. on VLSI Circuits*, 1996.

[60] P.Guerrier and A.Greiner, "A generic architecture for on-chip packet-switched interconnections," in *Design, Automation and Test in Europe conf. (DATE)*, 2000, pp. 250–256.

[61] W. J. Dally, "Virtual-channel flow control," I*EEE Trans. on Parallel. & Distributed Systems*, vol. 3, no. 2, pp. 60–68, Mar. 1992.

[62] Andrei Rădulescu, John Dielissen, Santiago González Pestana, Om Prakash Gangwal, Edwin Rijpkema, Paul Wielage, and Kees Goossens, "An efficient on-chip network interface offering guaranteed services, shared-memory abstraction, and flexible network programming," I*eee Transactions on CAD of Integrated Circuits and Systems*, vol. 24, no. 1, pp. 4–17, Jan. 2005.

[63] S.Evain, J.Ph.Diguet, and D.Houzet, "A generic cad tool for efficient noc design," in I*eee Int. Symp. on Intelligent Signal Processing and Communication Systems (ISPACS)*, Seoul, Korea, Nov. 2004.

[64] L.Benini and G.De Micheli, "Networks on chip : A new paradigm for systems on chip design," in *Design, Automation and Test in Europe conf. (DATE)*, 2002, p. 418.

[65] "Arteris, the noc company," http ://www.arteris.net, 2005.

[66] "Ns2 network simulator," http ://www.isi.edu/nsnam/ns/.

[67] E.Bolotin, I.Cidon, R.Ginosar, and A.Kolodny, "Qnoc : Qos architecture and design process for network on chip," *Journal of Systems Architecture, Elsevier*, 2003.

[68] S.Evain, J-Ph.Diguet, and D.Houzet, "μSpider : a cad tool for efficient noc design," in *NORCHIP*, Oslo, NORWAY, Nov. 2004.

[69] S.Evain and J.Ph.Diguet, "From noc security analysis to design solutions," in *Submitted to* Ieee *Work. on Signal Processing Systems (SIPS)*, Athènes, Greece, Oct. 2005.

[70] J.L.Wong, G.Qu, and M.Potkonjak, "An on-line approach for power minimization in qos sensitive systems," in *ASP-DAC*, 2003.

[71] S.Manne and A.Klauser D.Grunwald, "Pipeline gating : speculation control for energy reduction," in 25^{th} *Int. Symp. on Computer Architecture*, Spain, 1998, pp. 132–141.

[72] D.H.Albonesi, "Selective cache ways : On-demand cache resource allocation," in 32^{nd} *Annual International Symposium on Microarchitecture*, 1999.

[73] R.Maro, Y.Bai, and R.I.Bahar, "Dynamically reconfiguring processor resources to reduce power consumption in high-performance processors," in *Work. on Power-Aware Computer Systems*, 2000.

[74] V.Nollet, T.Marescaux, D.Verkest, J-Y.Mignolet, and S.Vernalde, "Operating-system controlled network on chip," in 41^{th} ACM/IEEE *Design Automation Conf.*, San Diego, USA, 2004.

[75] J.Liang, A.Laffely, S.Srinivasan, and R.Tessier, "An architecture and compiler for scalable on-chip communication," IEEE *Trans. on* VLSI *Systems*, vol. 12, no. 7, pp. 711–726, July 2004.

[76] J-Y.Mignolet, V.Nollet, P.Coene, D.Verkest, S.Vernalde, and R.Lauwereins, "Infrastructure for design and management of relocatable tasks in a heterogeneous reconfigurable system-on-chip," in *Design, Automation and Test in Europe conf. (DATE)*, Munich, Germany, Mar. 2003.

[77] H.Walder and M.Platzner, "Reconfigurable hardware operating systems : From design concepts to realizations," in *Int. Conf. on Engineering of Reconfigurable Systems and Algorithms, ERSA'03*, Las Vegas, USA, June 2003.

[78] B.Noble, M.Satyanarayanan, D.Narayanan, J.E.Tilton, J.Flinn, and K.R.Walker, "Agile application-aware adaptation for mobility," in 16^{th} *ACM Symp.on Operating Systems Principles*, 1997.

[79] C.Lu, J.Stankovic, G.Tao, and S.Son, "Feedback control real-time scheduling : Framework, modeling and algorithm," *special issue of RT Systems Journal on Control-Theoretic Approaches to Real-Time Computing*, vol. 23, no. 1/2, pp. 85–126, july/september 2002.

[80] B.Li and K.Nahrstedt, "A control-based middleware framework for quality of service adaptation," *IEEE Journal on Selected Areas in Communication*, Sept. 1999.

[81] J-Ph.Diguet, Y.Eustache, N.Ben Amor, and S.Hammami, "Feedback control modelling for learning recongurable embedded systems," in *RESOSOC*, Montpellier, France, June 2005.

[82] D.Rakhmatov, S.Vrudhula, and D.A.Wallach, "A model for battery lifetime analysis for organizing applications on a pocket computer," IEEE *Trans. on* VLSI *Systems*, vol. 11, no. 6, pp. 1019–1030, Dec. 2003.

[83] M.J.Bass and M.Christensen, "Customization and speed-to-market will drive the industry from the bottom up," *IEEE Spectrum*, vol. 39, no. 4, Apr. 2002.

[84] www.globus.org/, "The Globus Alliance," 2004.

Annexe A

Articles joints

Deux articles encadrant mes travaux depuis la thèse.

- **Formalized methodology for data reuse exploration for low-power hierarchical memory mappings, Paru dans IEEE Trans. on VLSI Systems, Vol.6, n.4, p.529-537, Dec. 98.**
- **Algorithmic-level Specification and Characterization of Ebedded Multimedia Applications with Design Trotter, À paraître dans Journal of VLSI Signal Processing, Springer.**

Formalized Methodology for Data Reuse Exploration for Low-Power Hierarchical Memory Mappings

Sven Wuytack, Jean-Philippe Diguet, Francky V. M. Catthoor, *Member, IEEE,* and Hugo J. De Man, *Fellow, IEEE*

***Abstract*—Efficient use of an optimized custom memory hierarchy to exploit temporal locality in the data accesses can have a very large impact on the power consumption in data dominated applications. In the past, experiments have demonstrated that this task is crucial in a complete low-power memory management methodology. But effective formalized techniques to deal with this specific task have not been addressed yet. In this paper, the surprisingly large design freedom available for the basic problem is explored in-depth and the outline of a systematic solution methodology is proposed. The efficiency of the methodology is illustrated on a real-life motion estimation application. The results obtained for this application show power reductions of about 85% for the memory subsystem compared to the case without a custom memory hierarchy. These large gains justify that data reuse and memory hierarchy decisions should be taken early in the design flow.**

***Index Terms*—Low-power design, memory, system level, tradeoffs, video processing.**

I. Introduction

A large part of the power dissipation in data dominated applications is due to data transfers and data storage. This power component can often be reduced by introducing an optimized custom memory hierarchy that exploits the temporal locality in the data accesses. The impact of this can be very large, as has been demonstrated by us on an H.263 video decoder [1] and a motion estimation application [2].

The idea of using a custom memory hierarchy to minimize the power consumption is based on the fact that memory power consumption depends primarily on the access frequency and the size of the memory. For on-chip memories, which are not very much partitioned, memory power increases with the memory size. In practice, the relation is between linear and logarithmic depending on the memory library. For off-chip memories, the power is much less dependent on the size because they are internally heavily partitioned. Still they consume more energy per access than the smaller on-chip memories. Hence, power savings can be obtained by accessing heavily used data from smaller memories instead of from large background memories. Such an optimization requires architectural transformations that consist of adding layers of smaller and smaller memories to which frequently used data can be copied [3]. Memory hierarchy optimization introduces copies of data from larger to smaller memories in the data flow graph. This means that there is a tradeoff involved here: on the one hand, power consumption is decreased because data is now read mostly from smaller memories, while on the other hand, power consumption is increased because extra memory transfers are introduced. The memory hierarchy design task has to find the best solution for this tradeoff.

Memory hierarchy design for power optimization is basically different from caching for performance optimization [4]. The latter determines how to fill the cache such that data has been loaded from main memory before it is needed. Instead of minimizing the number of transfers, the number of transfers is often increased to maximize the chance of a cache hit, leading to wasted power by prefetching data that may never be needed.

Some custom memory hierarchy experiments on real-life applications can be found in literature. What is not solved, however, is how to decide on the optimal memory hierarchy. In this paper, we present a formalized methodology for this decision and give an indication of how large the search space really is. The latter is much larger than conventionally exploited in state-of-the-art designs. For more information on the context of this work we refer to [5].

The rest of the paper is organized as follows. Section II discusses the related work. Section III defines the global memory hierarchy problem. Section IV defines the data reuse decision problem and points out important issues toward a methodology. Section V presents our methodology for solving the data reuse exploration and decision problem. Section VI discusses the results of the data reuse exploration experiment for a motion estimation application. Section VII concludes the paper.

II. Related Work

The main work related to data reuse exploration lies in the parallel compiler area, especially related to the cache hierarchy. Here, several papers have analyzed memory organization issues in processors [6]. This, however, has not resulted yet in any formalizable method to guide the memory organization issues. In most work on parallel multiple instruction multiple data (MIMD) processors, the emphasis in terms of storage hierarchy has been on hardware mechanisms based on cache coherence protocols [7]. Partitioning or blocking strategies for loops to optimize the use of caches have been studied in several contexts [4]. The main focus is on performance improvement though and not on memory cost. Recently, also in a system synthesis context, applying transformations to improve the cache usage has been addressed [8], [9]. None

Manuscript received December 15, 1997; revised May 15, 1998.

The authors are with Inter-University Micro-Electronics Center (IMEC), Leuven B-3001 Belgium.

Publisher Item Identifier S 1063-8210(98)08514-X.

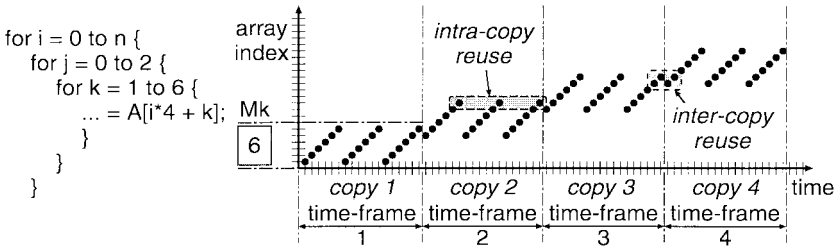


Fig. 1. Exploiting data reuse local in time to save power.

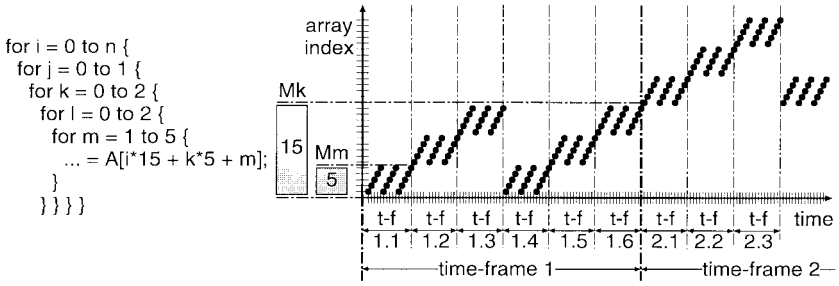


Fig. 2. Possibility for multilevel hierarchy.

of these approaches determine the best memory hierarchy organization for a given (set of) applications and only few address the power cost. Only an analysis of memory hierarchy choices based on statistical information to reach a given throughput expectation has been discussed recently [10]. In the hardware realization context, much less work has been performed, mainly oriented to memory allocation [11]–[13]. This paper discusses how to decide on the optimal use of the memory in a systematic way.

III. Memory Hierarchy Design

This section defines the memory hierarchy design task. First, it shows that memory hierarchy design is about exploiting temporal locality. Then, it shows that memory hierarchy design consists of two steps: *data reuse decision*, which is the topic of this paper, and *memory layer assignment*, which is left for a future paper.

A. Exploiting Temporal Locality

Memory hierarchy design exploits data reuse local in time to save power by copying data that is reused often in a short time period to a smaller memory, from which the data can then be accessed. Fig. 1 illustrates this for all read operations to a given array.

The horizontal axis is the time axis. It shows how the data accesses are ordered relatively to each other in time. The vertical axis shows the index of the array elements. Every dot represents a memory read operation, scheduled at a certain time and accessing a given array element. In this example, most values are read multiple times. Assuming that the data is still needed later on, all of the array elements have to be stored in a large background memory. However, when we look at smaller time frames (indicated by the vertical dashed lines), we see that only part of the data is needed in each time frame, so this part of the data would fit in a smaller, less power consuming memory. If there is sufficient reuse of the data in that time frame, it can be advantageous to copy the data that is used frequently in this time frame to a smaller memory. Consequently, the second time an array element has to be read, it can be read from the smaller memory instead of the larger memory. This leads to the following definitions.

Definition 1—Time Frame: The execution time of an application can be subdivided into a number of nonoverlapping time intervals, called level 1 time frames. Each level i time frame can be subdivided further into nonoverlapping level $i+1$ *time frames*.

Definition 2—Copy Candidate: A copy candidate corresponding to an array A and time frame TF_i is a set of array elements of A that are read in TF_i and are considered for copying to a lower hierarchy level.

During the data reuse task it will be decided which copy candidates are really worth to be copied in order to save power. Once this decision is made, the transfers for making the copies will be added to the application code and the copy candidates become real (partial) copies of their corresponding arrays.

Data reuse is the result of **intra***copy reuse* and **inter***copy reuse* (cfr. Fig. 1). *Intracopy* reuse means that each array element is read several times from its memory during one time frame. *Intercopy* reuse means that advantage is taken from the fact that part of the data needed in the next time frame could already be available in the memory from the previous time frame, and therefore does not have to be copied again.

Taking full advantage of temporal locality for power, usually requires architectural transformations that consist of adding several layers of memories (each corresponding to its own time frame level) between the large background memories and the small foreground memories (registers in the data path). Every layer in the memory hierarchy contains smaller memories than the ones used in the layer above it. An example of this is shown in Fig. 2. It shows time frames that are subdivided into smaller

time frames. Each level of time frames potentially corresponds to a memory layer in the memory hierarchy.

B. Steps in Memory Hierarchy Design

Two steps can be identified in memory hierarchy design:

1) the *data reuse exploration and decision* step decides *which* intermediate copies have to made for accessing the data in a power efficient way;
2) the *memory layer assignment* step decides for each array and copy of an array on *which layer* in the common custom memory hierarchy it will be stored.

After memory layer assignment, an optimal memory architecture has to be derived for each layer. This is done by the subsequent memory allocation and array-to-memory assignment tasks in our data transfer and storage exploration (DTSE) methodology [5]. This paper focuses on the data reuse decision step only.

IV. Data Reuse Decision

This section discusses the data reuse exploration and decision step of memory hierarchy design. It points out important elements for a systematic methodology described in the next section.

A. Search Space

The following assumptions allow to focus the data reuse task, without really restricting the search space in practice:

- only read operations have to be considered.

The reason for this is that repeated reading of the same data *value* makes sense (i.e., repeatedly reading the same memory location without intermediate writes to that memory location), whereas writing *the same* data *value* usually does not make sense. So there is no need for creating a memory hierarchy for repeatedly written data. The only thing that has to be decided for write operations is in which layer a certain (temporary) array will be written. This is decided in the memory layer assignment step *after* data reuse decisions are made:

- only one array has to be considered at a time.

Data reuse *exploration* can be tackled for each array separately. The main reason for this is that each copy in the memory hierarchy has its own root array, i.e., the copies form a tree where the *root* is the original array that is being accessed. A copy cannot be obtained as the mix of different arrays. We will also assume that the data reuse *decision* can be taken for each array separately. As explained later on, this limits the search space to some extent.

B. Data Reuse Factor

The usefulness of memory hierarchy for saving power is strongly related to the array's data reusability, because this is what determines the ratio between the number of read operations from a copy of an array in a smaller memory, and the number of read operations from the array in the larger memory on the next hierarchical level.

The reuse factor of a group of data D stored in a memory on layer i relative to layer $i-1$ where layer 1 is the furthest from the data paths, is defined as

$$F_R(i, D) = \frac{N_R(M_i, D)}{N_R(M_{i-1}, D)} \tag{1}$$

where $N_R(M_i, D)$ is the total number of times an element from D is read from a memory at layer i. A reuse factor larger than 1 is the result of *intra*copy and *inter*copy reuse (cf., Fig. 1).

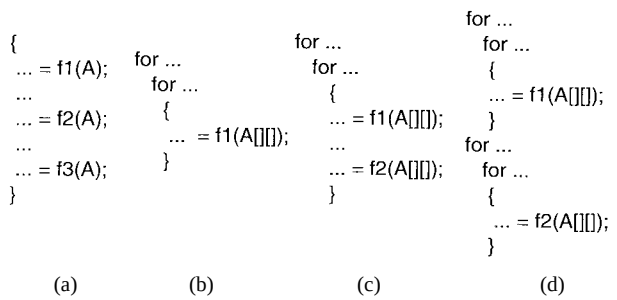


Fig. 3. Classification of data reuse opportunities.

C. Classification of Data Reuse

Fig. 3 presents a classification of four cases in which data reuse can be exploited by means of memory hierarchy. These four cases are:

1) No Loops: In case there are no loops, there is no structured use of arrays. In fact, each array element is treated independently from the others similar to scalars. This case is *not* considered in our methodology. It is left for scalar methodologies which are more suited for this.

2) One Read Instruction in a (Nested) Loop: This is the basic case on which our methodology is based. Both intracopy and intercopy reuse are possible here, when the loop nesting (i.e., the order of the different nested loops, and the direction in which the loops are traversed) is fixed. Indeed, in this case, the ordering of the different copies is known and every two consecutive copies can be examined for overlap (i.e., intercopy reuse).

3) Multiple Read Instructions in a (Nested) Loop: Here it is assumed that each read instruction has a different index expression, because otherwise they can be reduced to the previous case by reading once from background memory and storing the result in a foreground register. When the read instructions are accessing different parts of the array, a different memory hierarchy can be constructed for each of them. In practice, these memory hierarchies can contain partly the same data, and are then best combined. Determining which part of the memory hierarchy can be shared, can be done with a geometrical data flow analysis.

4) Read Instructions in Different Loop Nests: In this case, a memory hierarchy can be derived for each of the loop nests separately. Because they are in different loop nests, these memory hierarchies can be very different from each other. Depending on the temporal locality (which is only known when the ordering of all loop nests is already fixed), it may be useful to copy the data that is common to these loops to an extra memory layer. Again a polyhedral analysis can be used to determine which part of the array is used in common.

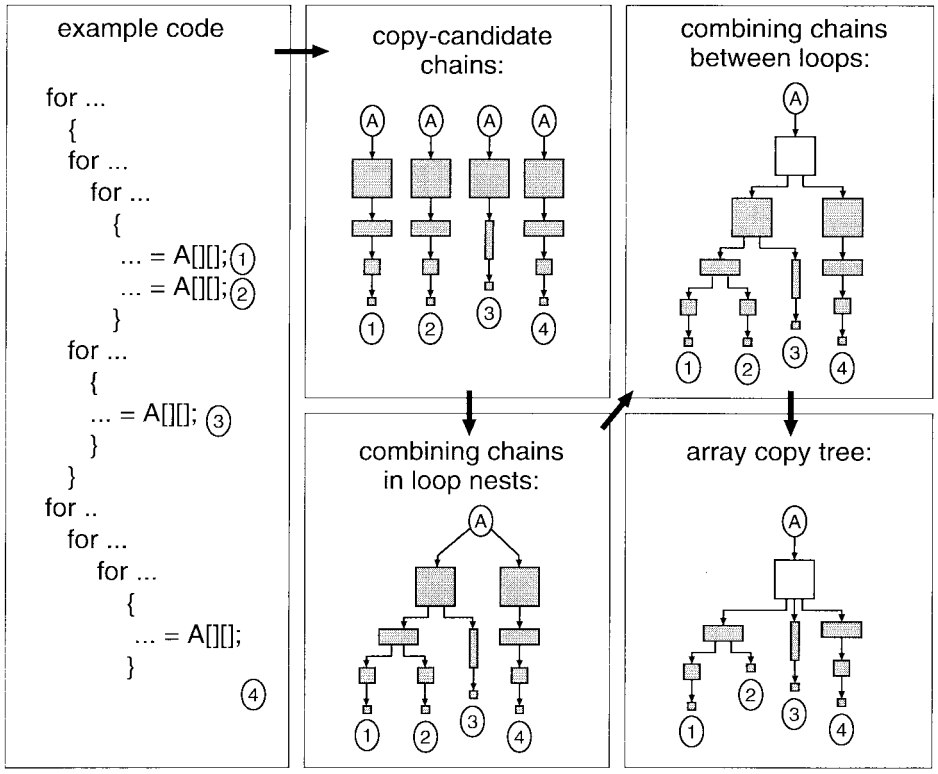


Fig. 4. Data reuse exploration methodology example.

Remark that when more than two loops are involved, part of the array can be common to only a few of the loops involved, making things much more complex.

V. Proposed Methodology

In this section, we propose a methodology for data reuse exploration and decision based on a number of assumptions to make the solution feasible for real-life applications.

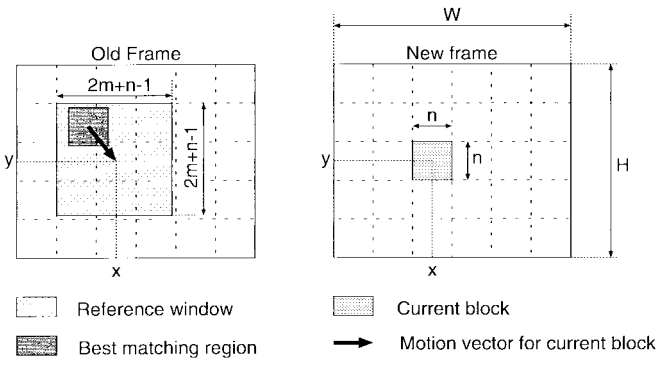
A. Assumptions

The following is assumed in our methodology:

- *Nesting order and direction of nested loops is fixed.*
  The fixed nesting order is required to determine the time frames and copy candidates corresponding to each read instruction in the loop nest. The fixed iteration direction is required to determine the overlap for estimating the *intercopy* reuse.
- *Time frames are determined by loop boundaries.*
  Finding an optimal time frame hierarchy is a very complex problem. However, we believe that the optimal time frame boundaries are likely to coincide with the loop boundaries of loop nests. Therefore, we use as a heuristic that the loop boundaries correspond to time frame boundaries, instead of trying to find globally optimal time frame boundaries.
- *A copy candidate contains all data read in its time frame.*
  It is assumed that *all* data being accessed by a certain read instruction in a certain time frame will be copied to a copy candidate, such that all data required by the read instruction can be found inside the copy candidate. Ideally, only part of the data that is accessed more than once should be copied. The loss due to this restriction is small in practice.
- *Copy candidates of a time frame level are stored in-place.*
  It is assumed that at the end of a time frame, the data copied into the intermediate memory is not needed anymore, and will be overwritten by the data needed in the next time frame. Therefore, the size of the copy candidate corresponding to a certain time frame level is determined by the time frame leading to the largest copy candidate.
- *Copy candidates are stored in perfectly fitting memories.*
  It is assumed that a copy candidate will be stored in a memory with word depth and bit width equal to those of the copy candidate. In general, this assumption leads to an underestimation of the power cost, as usually several copy candidates will have to share a memory. This real memory size is only known until after array-to-memory assignment and can therefore not be taken into account.
- *Intercopy reuse is fully exploited.*
  It is assumed that intercopy reuse will be fully exploited. This means that data that is already present in the smaller memory from the previous copy will not be copied again. Only data that is not yet present has to be copied over the data of the previous copy that is not needed anymore. This affects the number of accesses to each of the copy candidates. The size of the copy candidates is unaffected.

B. Data Reuse Exploration

Based on the assumptions listed in the previous subsection, we propose a systematic data reuse exploration methodology. Fig. 4 illustrates the different steps for a small example.




```
for (g=0; g<H/n; g++)          /* vertical CB counter */
 for (h=0; h<W/n; h++) {       /* horizontal CB counter */
  Δopt[g][h] = +∞;
  for (i=-m; i<m; i++)         /* vertical searching of RW */
   for (j=-m; j<m; j++) {      /* horizontal searching of RW */
    Δ = 0;
    for (k=0; k<n; k++)        /* vertical traversal of CB */
     for (l=0; l<n; l++) {     /* horizontal traversal of CB */
      Δ += abs(New[g.n + k][h.n + l]
                -Old[g.n + i + k][h.n + j + l];)
     }
    Δopt[g][h] = min(Δ, Δopt[g][h]);
   }
 }
```

Fig. 5. The motion estimation algorithm and its parameters.

1) Copy Candidate Chains: For each read instruction inside a loop nest, a *copy candidate chain* can be determined in the following way. The first copy candidate in the chain contains *all* array elements accessed by the given read instruction during the execution of the loop nest. The second copy candidate in the chain is associated with the iterations of the outer loop. Each iteration has a corresponding time frame. All array elements that are accessed by the read instruction during a given time frame are stored in the corresponding copy candidate. The storage space of the copy candidate is shared among the different iterations of the loop. In case the number of elements accessed by the read instruction varies between iterations, the size of the copy candidate is determined by the iteration that accesses the most array elements. This can be repeated for each of the remaining loop nest levels. Together these copy candidates form the *copy candidate chain* for the considered read instruction. Such a chain represents the maximal exploitation of data reuse of type (b) in Fig. 3.

2) Copy Candidate Trees: Copy candidate chains of different read instructions accessing the same array can be combined into a *copy candidate tree* for that array.

1) *Read instructions belonging to the same loop nest.*
   Often, the copy candidates of read instructions belonging to the same loop nest contain the same data. If this is the case for all iterations, the copy candidate can be shared between the read instructions. Moreover, if a copy candidate can be shared, also the copy candidates before it in the chain can be shared. Because the first copy candidate, which is the array itself, can always be shared, the copy candidate chains can always be combined into a *copy candidate tree*. Such a tree represents the maximal exploitation of data reuse of type (b) and (c) in Fig. 3.
2) *Read instructions belonging to different loop nests.*
   Copy candidates that cannot be combined in the previous way because they belong to different loop nests [cf., Fig. 3(d)], can still share the same data. In this case, an extra intermediate copy candidate can be inserted in the copy candidate tree to exploit this data reuse opportunity. When more than two copy candidates share data in this way, the search space grows quickly: for every possible subset of them an extra copy candidate could be introduced. This freedom will be explored in future research and will not be considered further in this paper. So we will treat the read instructions as operating on independent arrays. Two simple rules can be applied to prune copy candidates from copy candidate trees because they will never occur in an optimal memory hierarchy: the size of the copy candidates must decrease from one layer to the next, and the reuse ratio (1) of each layer must be larger than 1.

3) Copy Candidate Graphs: From the previous step we can conclude that for every array, a copy candidate tree can be determined. Each node in the tree represents a copy candidate. A copy candidate can be characterized by its required memory size, number of write operations to copy data into it, and the total number of read operations to copies on lower layers. Also its data reuse-factor can be calculated. From this tree other valid copy trees can be derived because it is allowed to copy data from any ancestor node in the tree, not only the parent node. Therefore, we extend the copy candidate tree to a *copy candidate graph* in the following way: for every node in the tree, we add edges starting from all its ancestor nodes toward the node itself. All possible trees that can be derived by selecting a single path from the root node to every leaf node, represents a valid copy tree.

4) Array Copy Trees: The *array copy tree* is the lowest cost tree obtained from the copy candidate graphs in the way described above. Selecting the array copy tree from all possible copy trees is called data reuse *decision* (cf., Section V-D). The cost function for selecting the array copy tree is defined next.

C. Cost Function

The cost function for selecting the optimal copy tree is a weighted sum of a power and area estimate for the copy tree CT. The cost function is given by

$$\begin{aligned}\text{cost}(CT) = \alpha \cdot \sum_{c \in CT} & [P_r(N_{\text{bits}}(c), N_{\text{words}}(c), f_{\text{read}}(c)) \\ & + P_w(N_{\text{bits}}(c), N_{\text{words}}(c), f_{\text{write}}(c))] \\ & + \beta \cdot \sum_{c \in CT} A(N_{\text{bits}}(c), N_{\text{word}}(c)) \end{aligned} \quad (2)$$

where

- c is a copy candidate of the considered copy tree CT;
- $P_{r/w}(N_{\text{bits}}, N_{\text{words}}, f_{\text{access}})$ is the power estimate for read/write operations of a memory with bit width N_{bits},

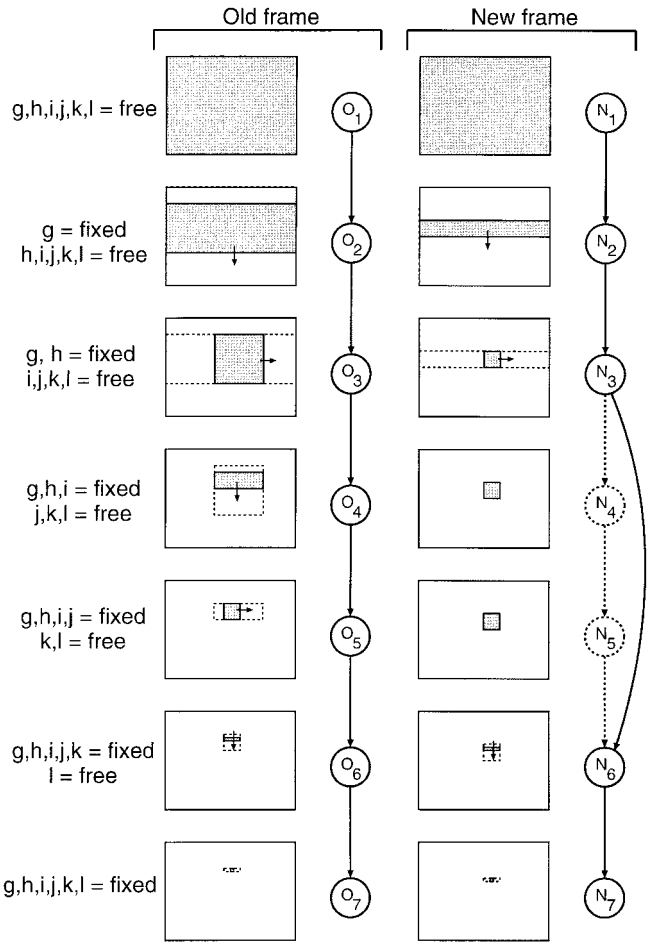


Fig. 6. Copy candidate chains for ME application.

word depth N_{words}, and that is accessed with a *real* access frequency f_{access};
- $A(N_{\text{bits}}, N_{\text{words}})$ is the area estimate for a memory with specified parameters;
- α and β are weighting factors for area/power tradeoffs.

The *real* access frequency f_{access} of a memory is obtained by multiplying the number of memory accesses per frame with the frame rate (**not** the clock frequency).

D. Data Reuse Decision

The end result of the data reuse decision task is an optimal *array copy tree* for each array in the application. Here, it is assumed that the optimal copy tree can be derived for each array independently from the other arrays. This is not completely true because the optimal copy trees depend partly on how well the different copies can share memory space (interarray in-place [15]). However, because the data reuse decision is best taken early in the design flow, not enough data about interarray in-place is available to include it in the decision process.

VI. Test Vehicle: Motion Estimation Algorithm

The motion estimation (ME) algorithm is used as a test-vehicle to illustrate the proposed methodology.

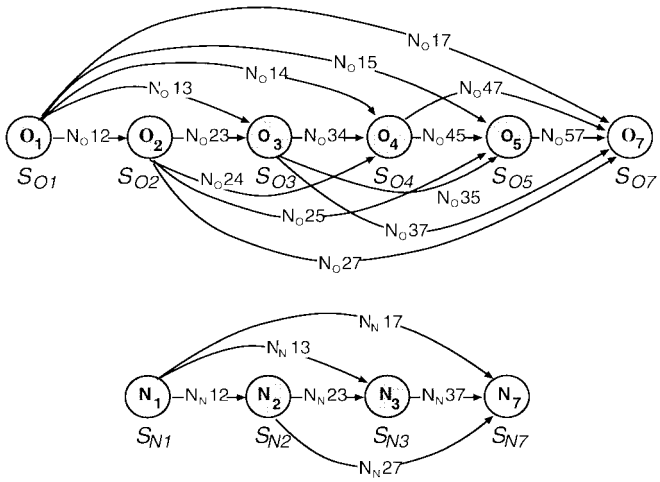


Fig. 7. Copy candidate graphs for ME application.

A. Algorithm and Cost Functions

The motion estimation algorithm is used in moving image compression algorithms. It allows to estimate the motion vector of small blocks of successive image frames. The version we consider here is the kernel of what is commonly referred to as the "full-search full-pixel" implementation [16]. The algorithm and its parameters are shown in Fig. 5.

For the experiments we have used the parameters of the QCIF format $(W = 176, H = 144, m = n = 8)$ with a frame rate of 30 frames/s. We are using an accurate but proprietary model for estimating the power and area of the memory modules from a specific library for which we are not allowed to publish absolute values on area and power. Therefore only relative values are provided.

B. Data Reuse Exploration for ME Application

1) Copy Candidate Chains: For the motion estimation application only the frame arrays *Old* (O) and *New* (N) will be considered here, because the other arrays can easily be stored in a foreground register. Fig. 6 shows the copy candidate chains for the two read instructions accessing array O and N, respectively. Copy candidates N_4 and N_5 can be pruned because they are not smaller than N_3.

2) Copy Candidate Trees: Since there is only one read instruction from the *Old* frame and one from the *New* frame, there is nothing to combine in this example: the copy candidate chains for the frame arrays *are* the copy candidate trees. The nodes O_6 and N_6 can now be pruned, though, because they have a reuse ratio of one (i.e., no reuse).

3) Copy Candidate Graphs: The copy candidate graphs for the two frame arrays are shown in Fig. 7.

4) Array Copy Trees: In this case the copy candidate trees are in fact chains.[1] Because of the chains, the search process for finding the array copy tree can be represented as a simple search tree. Fig. 8 shows these search trees for our motion estimation example. The (optimal) array copy trees are indicated in gray. The dashed lines divide the search trees in a number of layers. Each layer corresponds to a copy candidate

[1]In general, this is not true as demonstrated by us before for the H.263 video conferencing decoder test vehicle in [1].

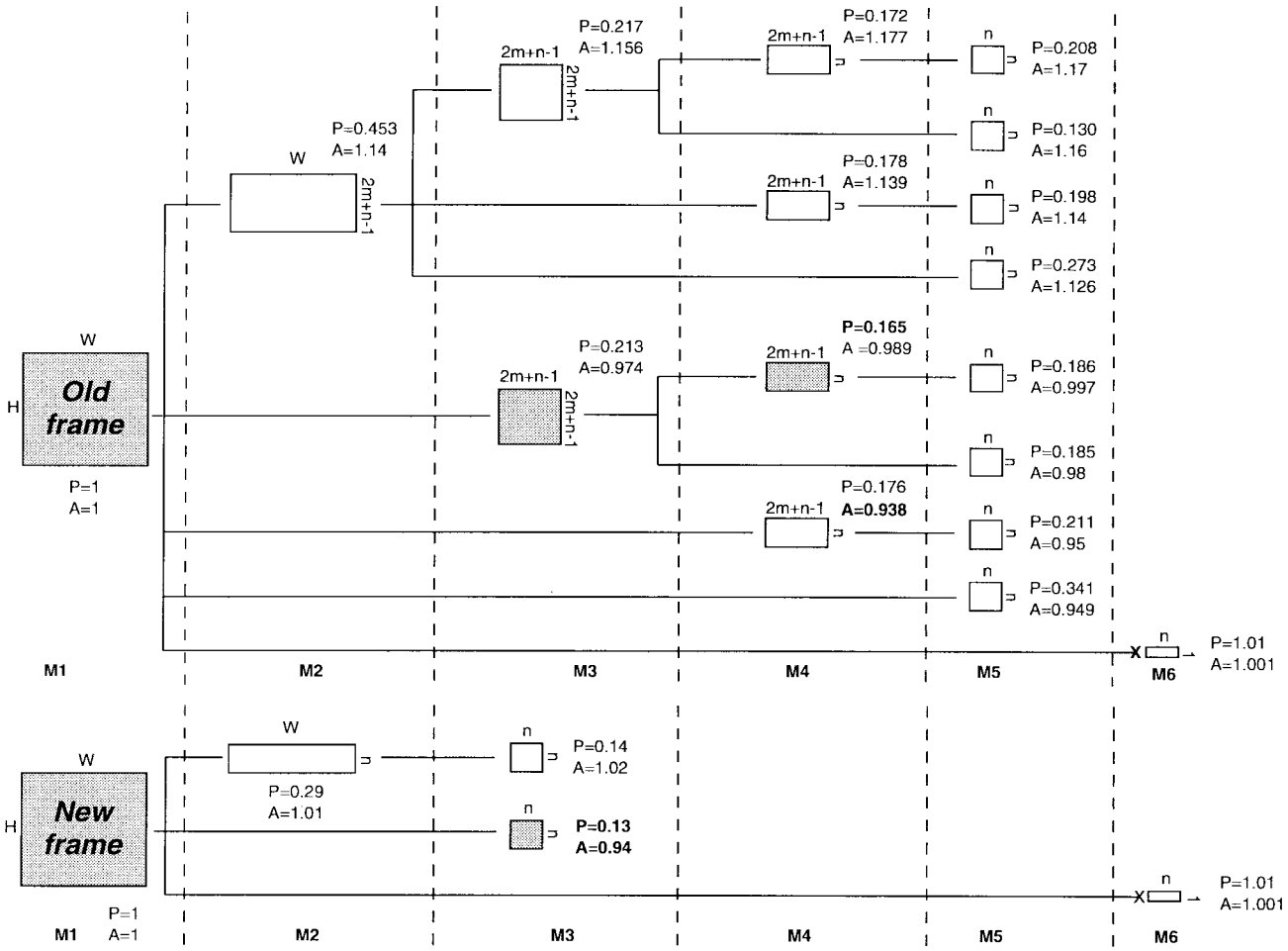


Fig. 8. Search trees for data reuse decision.

(or time frame level). The solutions can either include this copy candidate (copy is shown in search tree) or skip it (copy is not shown). Every node in the tree represents a solution: the copy candidate on that node is the lowest layer in the hierarchy, and all copy candidates on the path between the root node and the node itself are intermediate layers in the hierarchy. For each solution, the area (A) and power (P) of the complete array copy tree relative to the solution without hierarchy are indicated.

C. Discussion

A surprising result is that the total memory area can *decrease* by adding extra memory layers (cf., Fig. 8). The reason for this is that the maximum access frequency of the memories is taken into account in our estimations. If a certain memory would be accessed above its maximum access frequency, this memory will be split into two memories of half the size to increase the memory bandwidth. This splitting introduces overhead. By adding extra memory layers with small memories, the bandwidth requirements of the large background memories can be reduced, and therefore splitting can be avoided for the large memories. This area gain can be larger than the area lost by adding a few small memories.

The optimal memory hierarchy for power is

- for the *Old* frame, a three-level hierarchy that leads to a power saving of 83% compared to the solution without memory hierarchy;
- for the *New* frame, a two-level hierarchy that leads to a power saving of 87% compared to the solution without memory hierarchy.

These figures do not include the power dissipation in the interconnect. Taking this into account will result in even larger gains, as off-chip communication dissipates much more power than on-chip communication. Without memory hierarchy all data transfers are off-chip. With memory hierarchy, most of these are replaced by less power consuming on-chip transfers.

If we compare the result with the one we proposed in an *ad hoc* way in an earlier paper [2], we note that a different memory hierarchy with only two levels was selected which results in a higher power consumption. This clearly shows that by using a more systematic design space exploration methodology which exploits the full search space available, as proposed in this paper, better results can be obtained.

VII. Conclusion

Exploiting temporal locality in the memory accesses by means of an optimized memory hierarchy can effectively

reduce the power dissipation of data-dominated applications. The memory hierarchy design task can be split into two steps: data reuse decision and memory layer assignment. The first step is the topic of this paper, the second is left for a future paper.

A systematic methodology for the data reuse decision step has been proposed based on a number of realistic assumptions. The feasibility and the large impact of the proposed techniques have been shown on a real-life video application. The results obtained for the motion estimation application show power reductions of about 85% for the memory subsystem compared to the case without memory hierarchy. Similar results have been obtained for other applications not presented in this paper. These figures do not include the power dissipation in the interconnect. Taking this into account will result in even larger gains. These large power gains justify that the memory hierarchy should be decided early in the design script.

Currently we are extending the methodology to work also for instruction set processors with a (partially) fixed memory hierarchy.

REFERENCES

[1] L. Nachtergaele, F. Catthoor, B. Kapoor, D. Moolenaar, and S. Janssen, "Low power storage exploration for H.263 video decoder," in *Proc. IEEE Workshop VLSI Signal Processing*, Monterey CA, Oct. 1996; also in *VLSI Signal Processing IX*, W. Burleson, K. Konstantinides, and T. Meng, Eds. Piscataway, NJ: IEEE Press, 1996, pp. 116–125.

[2] S. Wuytack, F. Catthoor, L. Nachtergaele, and H. De Man, "Power exploration for data dominated video applications," in *Proc. IEEE Int. Symp. Low Power Design*, Monterey, CA, Aug. 1996, pp. 359–364.

[3] S. Wuytack, F. Catthoor, F. Franssen, L. Nachtergaele, and H. De Man, "Global communication and memory optimizing transformations for low power systems," in *Proc. IEEE Int. Workshop Low Power Design*, Napa, CA, Apr. 1994, pp. 203–208.

[4] D. Kulkarni and M. Stumm, "Linear loop transformations in optimizing compilers for parallel machines," Comput. Syst. Res. Inst., Univ. Toronto, Toronto, Ont., Canada, Tech. Rep., Oct. 1994.

[5] F. Catthoor, S. Wuytack, E. De Greef, F. Franssen, L. Nachtergaele, and H. De Man, "System-level transformations for low power data transfer and storage," in *Low Power Design*, B. Brodersen and A. Chandrakasan, Eds. Piscataway, NJ: IEEE Press, 1998.

[6] A. Faruque and D. Fong, "Performance analysis through memory of a proposed parallel architecture for the efficient use of memory in image processing applications," in *Proc. SPIE'91, Visual Commun. Image Processing*, Boston, MA, Oct. 1991, pp. 865–877.

[7] L. Liu, "Issues in multi-level cache design," in *Proc. IEEE Int. Conf. Comput. Design*, Cambridge, MA, Oct. 1994, pp. 46–52.

[8] D. Kolson, A. Nicolau, and N. Dutt, "Elimination of redundant memory traffic in high-level synthesis," *IEEE Trans. Computer-Aided Design*, vol. 15, pp. 1354–1363, Nov. 1996.

[9] P. Panda, N. Dutt, and A. Nicolau, "Memory organization for improved data cache performance in embedded processors," in *Proc. 1996 Int. Symp. Syst. Synthesis*, La Jolla, CA, Nov. 1996, pp. 90–95.

[10] B. Jacob, P. Chen, S. Silverman, and T. Mudge, "An analytical model for designing memory hierarchies," *IEEE Trans. Comput.*, vol. C-45, pp. 1180–1193, Oct. 1996.

[11] P. Lippens, J. van Meerbergen, W. Verhaegh, and A. van der Werf, "Allocation of multiport memories for hierarchical data streams," in *Proc. IEEE Int. Conf. Computer-Aided Design*, Santa Clara, CA, Nov. 1993, pp. 728–735.

[12] L. Ramachandran, D. Gajski, and V. Chaiyakul, "An algorithm for array variable clustering," in *Proc. European Design and Test Conf.*, Paris, France, Mar. 1994, pp. 262–266.

[13] F. Balasa, F. Catthoor, and H. De Man, "Dataflow-driven memory allocation for multi-dimensional processing systems," in *Proc. IEEE Int. Conf. Computer-Aided Design*, San Jose, CA, Nov. 1994.

[14] L. Stok, "Data path synthesis," *Integration, the VLSI Journal*, vol. 18, pp. 1–71, June 1994.

[15] E. De Greef, F. Catthoor, and H. De Man, "Array placement for storage size reduction in embedded multimedia systems," in *Proc. 11th Int. Conf. Application-Specific Systems, Architectures and Processors*, Zurich, Switzerland, July 1997, pp. 66–75.

[16] T. Komarek and P. Pirsch, "Array architectures for block matching algorithms," *IEEE Trans. Circuits Syst.*, vol 36, Oct. 1989.

Sven Wuytack was born in Leuven, Belgium, in 1970. He received the degree in electrical engineering from the Katholieke Universiteit, Leuven, Belgium, in 1993. He is currently working towards the Ph.D. degree at the Inter-University Micro-Electronics Center (IMEC), Heverlee, Belgium.

His research interests include system and architecture-level power optimization, mainly oriented toward memory organizations, and memory management in general. The major target application domains where this research is relevant are data structure dominated modules in telecom networks and real-time signal and data processing algorithms in image, video, and end-user telecom applications.

Jean-Philippe Diguet was born in France in 1969. He received the engineering degree in electronics from ESEO, France, in 1992 and the Ph.D. degree in signal processing from University of Rennes, France, in 1996. He is currently working on his thesis with the Lasti Laboratory, Lannion, France.

He then joined the Inter-University Micro-Electronics Center (IMEC), Leuven, Belgium, in 1997, as a Postdoctoral candidate supported by the French government fellowship Lavoisier. He is now an Assistant Professor of Electrical Engineering at UBS University, Lorient, France, and a member of the Lester Laboratory. His current research interests are in high-level estimations for system design methodologies applied to signal-processing and telecommunication domains.

Francky V. M. Catthoor (S'86–M'87) received the engineering and the Ph.D. degrees in electrical engineering from the Katholieke Universiteit, Leuven, Belgium, in 1982 and 1987, respectively.

From September 1983 to June 1987, he has been a Researcher in the area of VLSI design methodologies for Digital Signal Processing, with Prof. H. De Man and Prof. J. Vandewalle as Ph.D. dissertation advisors. Since 1987, he has headed several research domains in the area of high-level and system synthesis techniques and architectural methodologies, all within the VLSI System Design Methodology (VSDM) division at the Inter-University Micro-Electronics Center (IMEC), Heverlee, Belgium. He is an Assistant Professor at the Electrical Engineering Department of the Katholieke Universiteit Leuven since 1989. His current research activities belong to the field of architecture design methods and system-level exploration for area and power, mainly oriented toward memory management and global data transfer optimization. The major target application domains are real-time signal and data processing algorithms in image, video and end-user telecom applications, and data structure dominated modules in telecom networks

Dr. Catthoor received the Young Scientist Award from the Marconi International Fellowship in 1986. In 1995, he became an Associate Editor for the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATED (VLSI) SYSTEMS.

Hugo J. De Man (M'81–SM'81–F'86) was born in Boom, Belgium, on September 19, 1940. He received the electrical engineering degree and the Ph.D. degree in applied sciences from the Katholieke Universiteit Leuven, Heverlee, Belgium, in 1964 and 1968, respectively.

In 1968, he became a Member of the Staff of the Laboratory for Physics and Electronics of Semiconductors at the University of Leuven, Leuven, Belgium, working on device physics and integrated circuit technology. From 1969 to 1971, he was at the Electronic Research Laboratory, University of California, Berkeley, as an ESRO-NASA Postdoctoral Research Fellow, working on Computer-Aided Device and Circuit Design. In 1971 he returned to the University of Leuven as a Research Associate of the NFWO (Belgian National Science Foundation). In 1974, he became a Professor at the University of Leuven. During the winter quarter from 1974 to 1975, he was a Visiting Associate Professor at the University of California, Berkeley. From 1984 to 1995, he was Vice-President of the VLSI systems design group of Inter-University Micro-Electronics Center (IMEC), Leuven, Belgium, where the actual field of this research division is design methodologies for integrated systems for telecommunication, examples of spread spectrum, and ATM components design. This group's research has been at the basis of the EDC-Mentor Graphics DSP-Station. Since 1995, he has been a Senior Research Fellow of IMEC responsible for research in system design technologies.

Dr. DeMan was an Associate Editor for the IEEE JOURNAL OF SOLID-STATE CIRCUITS from 1975 to 1980 and was European Associate Editor for the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS from 1982 to 1985. He received a Best Paper Award from the ISSCC of 1973 on Bipolar Device Simulation and from the 1981 ESSCIRC conference for work on an integrated CAD system. In 1986, he received (together with L. Claesen) the Best paper Award in CAD from the ICCD-86 Conference and in 1987 for Best Publication in 1987 in the *International Journal of Circuit Theory and Applications.* In 1989 he received the Best Paper Award at the Design Automation Conference. He is a corresponding member of the Royal Academy of Sciences, Belgium, and a member of the Royal Flemish Engineering Society (KVIV).

Algorithmic-level Specification and Characterization of Embedded Multimedia Applications with Design Trotter

Yannick Le Moullec, Jean-Philippe Diguet, Nader Ben Amor,
Thierry Gourdeaux, Jean-Luc Philippe
LESTER, Université de Bretagne Sud, 56321 Lorient Cedex, France
Jean-Philippe.Diguet@univ-ubs.fr

Abstract

Designing embedded system is a non-trivial task during which wrong choices can lead to extremely costly re-design loops. The extra cost is even higher when wrong choices are made during the algorithm specification and mapping over the selected architecture. In this paper we propose a high-level approach for design space exploration, based on a usual standard language. More precisely we present the characterization step which is located at the very beginning of our hardware / software codesign framework. Once transformed into our internal representation, the specification is rapidly and automatically characterized and explored at the algorithmic level. Metrics are provided to the designer in order to evaluate very early in the design process, the impact of algorithmic choices on resources requirements by means of processing, control, memory bandwidth capabilities and potential parallelism at different levels of granularity. The aim is to improve the algorithm / architecture matching that sorely influences the implementation efficiency in terms of silicon area, performances and energy consumption. We give examples which illustrate how designers can refer to the outcomes of our methodology in order to select or build suitable architectures for specific applications.

Keywords: specification, characterization, algorithm / architecture matching, design-space exploration, SoCs.

1 Introduction

The context of our work is the hardware/software co-design in the domain of embedded multimedia applications. In this area, algorithmic and architectural choices have a strong impact on the power vs. performance trade-off which is a key issue regarding the evolution of mobile electronic devices. Thus, designers have to face a number of challenges. Three main situations can be considered: i) a chosen target architecture must be used, in that case optimizations have to be carried out on the specification and its implementation; ii) the specification cannot be changed, in that case optimizations have to be performed on the architecture which has to be selected

and/or tailored to match the specification and iii) neither the specification nor the architecture are fixed, in that case optimizations have to be performed using feedbacks between them. In all cases the designer needs relevant knowledge about the specification. Such knowledge includes operation granularity, potential parallelism, orientation (processing, control, memory) and data locality (memory hierarchy).

Consequently, a fast automated exploration process is required to alleviate the designer with the tedious task consisting in evaluating a large number of potential solutions based on the previously mentioned algorithmic options.

We tackle this problem by considering a high-level algorithmic approach using a standard procedural language for specifying the application. This specification is then automatically transformed into a fully graph-based representation which enables a fast and automatic characterization and exploration of the application in terms of algorithmic options.

Namely we consider a system as event-based at the highest levels of hierarchy - Hierarchical Finite State Machines (HFSMs) or task-graphs (TG)- encapsulating function calls. These functions are described with Hierarchical and Control Data-flow Graphs (HCDFGs), presented in this paper. It is the responsibility of the designer to choose the granularity of the specification, and therefore his responsibility to choose what should be described by means of HFSMs/task-graphs and HCDFGs. However, since this task is not trivial the designer can use the characterization step (presented in section 4) to get metrics about the control or data-flow orientation of the functions and iterate towards the most appropriate separation. It is worth noting that tools are available for simulation, formal proof and code generation at the event-based level. The goal of our work is to performed automatically and rapidly the tedious algorithmic exploration for the functions called from the event-based level.

This work is part of a complete design framework called Design Trotter, presented in [1]. This is a set of cooperative tools which aim at guiding embedded system designers early in the flow by means of design space exploration, as summarized in Fig. 1. It operates at a high-level of abstraction (algorithmic-level). Firstly the different functions of the applications are handled separately. For each fonction, the first step presented in this paper, includes the graph building (HCDFG specification) and the characterization metric computation. Then a scheduling step [1] is performed for each data-flow graph and loop nest within the function, the design space exploration is obtained by means of a large set of time constraints (e.g. from the critical path to a complete sequential execution). Finally, the results are combined to produce trade-off curves (#ressource vs. #cycles). The scheduling process offers different options including the balance between data-transfers and data-processings and the use of loop unrolling to meet time constraints. After the

intra-function scheduling an inter-function scheduling step [2], based on the previous trade-off curves, can be performed if functions can be executed concurrently. Then a projection step enables the exploration of the design space targeting reconfigurable architectures (FPGAs) [3], [4] and processors (HW and SW projections in Fig. 1 respectively). Finally results can be used within our HW/SW partitioning and real-time scheduling tool [5].

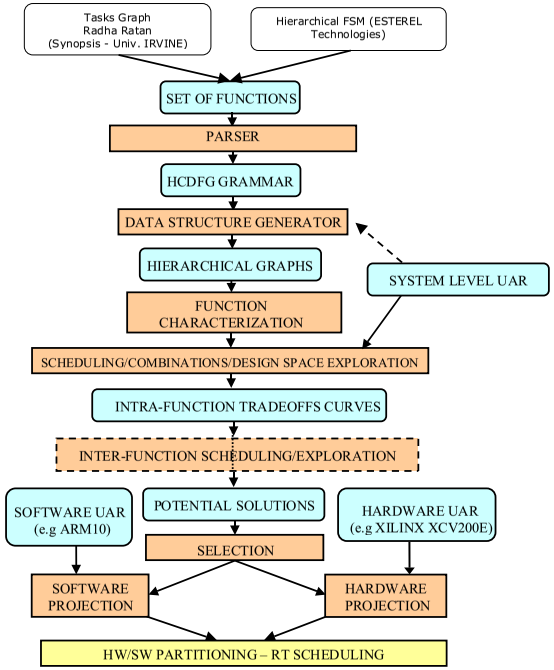


Figure 1: Design Trotter design flow.

The rest of the article is organized as follows: in section 2 we give an overview of existing specification models and characterization tools. Then, we expose the contributions of our paper for both specification and characterization aspects. Sections 3 and 4 detail these two points respectively. In section 5 we present some results of the characterization step showing the interest of the proposed method. Finally in section 6 we conclude about our work and present some perspectives.

2 Related work

2.1 Part 1: specification

One of the first issue when designing a system is its specification. Specifying a system is a complex task which can have a major influence on the subsequent steps of the design flow. Choosing a specification model can, for example, stress more or less hardware vs. software orientations, event-based vs. data-flow based approaches, data-processings vs. data-transfers. Granularity is another important feature which greatly influences the possibilities of the exploration process. For example a fine grain granularity specification focusing on implementation details usually enables very accurate results at the cost of long exploration processes. It is crucial to control these aspects and not to be restricted to one level of granularity (for subsequent steps such as characterization, cf. section 4).

2.1.1 Existing work for specification

Generally, existing design approaches consider only partial design flow. The decomposition of the design flow results from the architectural target and/or from the type of information handled by the application. We can mention control oriented models [6], [7],[8],[9], data-flow models [9],[10], Khan process networks [11], Ptolemy so-called domains (e.g. DE, DF).

To alleviate this problem, some attempts to define an unified modeling and design framework have been proposed. They are generally based on co-simulation (e.g., CoWare, VCC from Cadence). However, these approaches do not really support a complete and seamless design flow, from high-level specification to the generation of VHDL code for synthesis and assembly code for processors.

SystemC [12] is a design and verification language enabling the description of systems from high-levels (algorithmic-level) down to implementation in hardware and software. The modeling features of SystemC are provided as a C++ class library. It is a possible candidate for becoming a standard for the design of embedded systems. Reducing the productivity gap in system design can be achieved by raising the level of abstraction. However, as mentioned in [13], it is important to well-define the abstraction levels and models. The authors propose system-level semantics that cover the system design process and define properties and features for each model. By formalizing the flow, design automation for synthesis and verification can be performed and productivity gains can be achieved. Moreover, customizing the semantics enables the creation of specialized design methodologies.

2.1.2 Contribution

Initiatives like SystemC and SpecC [14] aim to provide the designer with a common language for progressively modeling an application by refining steps, from the system un-timed level to the implementation cycle accurate level. Such framework are necessary, however the designer remains responsible for choosing the appropriate computation model. One of the main influent specification decision is the separation between an event-based FSM model and a data-flow model, the boundary is not necessarily clear and can vary depending on the designer background.

The requirements of the tools included in Design Trotter have guided our choice towards a suitable internal representation model. This model should have the following features:

- Hierarchy: enables structured exploration of the specification such as bottom-top and top-bottom approaches but also partial exploration;
- Multi-granularity: enables coarse and/or fine grain specification depending on the utilized tool: fast characterization, system-level estimation (scheduling/combinations/exploration in Fig. 1), architectural estimations (HW and SW projections in Fig. 1);
- Attributes: used to save critical information. We consider two types of attributes: those created during the parsing step (e.g. data-type) and those created during the exploration/characterization steps (e.g. metrics);
- Parallelism specification: parallelism is a key feature in design space exploration. The model must enable the clear specification of processing and data-transfers parallelism in order to detect and exploit them;
- Openness: the model must be open to new languages and can be easily changed/extented.

We found out that none of the existing (and easily available) models had all the required features. Therefore we have defined our own model: HCDFG. The definition of its formalism (and underlying grammar rules) has been guided by a pragmatic approach considering that the most important point is to stay close to the original algorithmic specification of the application. This internal representation model, based on graphs, offers the flexibility required for the development of the tools implemented in Design Trotter.

We preferentially use the Esterel framework [15] to perform the first specification step since it offers various interesting features such as concurrent and hierarchical finite state machines (HFSM called Safe State Machine in Esterel framework) and tools for formal proof computation, simulation and code generation. Our approach enables a clear control of the model separation in a top down

approach: when the designer estimates that a data-flow specification can be efficiently used at a given state level of the HFSM he can insert calls to C functions. The HCDFG description starts at this level. Then depending on metric computation results (cf. 4), the designer can decide to modify his specification choices in terms of HFSM/C separation. We use the C language for three main reasons, firstly a lot of standards (e.g. ITU) and applications are written with this language, secondly it can be naturally inserted in various framework like SystemC and Esterel, and finally the GCC compiler provides a good starting point for building a graph representation. Actually, we use GCC to perform lexical and grammatical checkings and then introduce our own syntactic tree in order to extract the information we consider relevant (see 3.3).

In our approach the next step after the specification is the characterization of the application. In what follows we present existing work on this aspect.

2.2 Part 2: characterization

Several design-flows include a step used to characterize the specification of an application. The main objective of characterization is to extract relevant information from the specification to guide the designer and/or synthesis steps towards an efficient application-architecture matching. For this purpose, metrics can be efficiently used to rapidly stress the proper architecture style for the application or for part of it (sub-tasks, functions). Some of the relevant features include the wider/deeper trade-off, namely the ratio of explicit parallelism versus the pipeline depth, the necessity of complex control structures, the requirements in terms of local memories and specific bandwidth, and the need for processing resources for specific computations or address generation.

Contrary to partial available approaches, we consider that an efficient characterization step should include all the following requirements:

- Independence & flexibility: during the characterization step the designer should have the option to specify or not an architectural target since the objective of this step is to guide either the choice of an existing architecture or the construction of a new one. Moreover, the implementation of new metric computations must be easy;
- Hierarchy & multi-granularity: enables the characterization of the different ganularity levels: e.g i) loop body, ii) loop, iii) sequence of loops, it also offers the possibility to reuse characterizations for different "mappings" of a given graph (e.g. various calls of the same sub-function);
- Data and control dependency analysis: the information about critical paths at different levels of granularity is required for in-depth parallelism characterization.

2.2.1 Review of existing metrics

Recently works dealing with metrics in the domain of high-level synthesis [16], [17] and hardware software co-design [18], [19], [20] have been proposed. In [16] the metrics provide algorithm properties regarding a hardware implementation; the quantified metrics address the concurrency of arithmetic operations based on uniformed scheduling probabilities and the regularity that measures the repetition rate of a given pattern. In [17], some probability based metrics are proposed to quantify the communication link between arithmetic operators (through memory or registers). These metrics focus on a fine grain analysis and are mainly used to guide the design of data-paths, especially to optimize local connections and resource reuse. The metrics from [18] are computed at the functional level to highlight resource, data and communication channel sharing capabilities in order to perform a pre-partitioning resulting in function clustering to guide the next design step (hardware/software partitioning). The main issue is the placement of close functions on the same component in order to optimize communications and resource sharing. An interesting method for processor selection is presented in [19]. Three metrics representing the orientation of functions in terms of control, data transformation and data accesses are computed by counting specific instructions from a processor independent code. Then a distance is calculated, with specific characteristics of processors regarding their control, bandwidth and processing capabilities. In this framework a coarse and fixed granularity level is considered and the target is limited to predefined processors. Moreover the technique does not take into account instruction dependencies and there is no detail about the different types of memory accesses regarding the abstract processor model used. However we can reuse the concept of distance during the design steps located at lower levels. Finally, in [20] finer metrics are defined to characterize the affinity between functions and three kinds of targets: GPP, DSP and ASIC. The metrics are the result of the analysis and counting of C code instructions in order to highlight instruction sequences which can be DSP-oriented (buffer circularity, MAC operations inside loops,...), ASIC-oriented (bit level instructions) or GPP-oriented (conditional or I/O instructions ratio). Then a HW/SW partitioning tool is driven by the affinity metrics. Like in [19] these metrics are dedicated to HW/SW partitioning, they do not exploit instruction dependencies and address a fixed (C procedures) granularity. Moreover, the locality of data bandwidth is not clearly taken into account.

2.2.2 Contributions

Although a number of existing works dealing with metrics can be found in the literature, some important features are not yet covered. In our work we propose to fulfill this requirements by

means of new metrics which are detailed in section 4.

Firstly, the analysis performed in other works is generally dedicated to a class of applications and architectures; our framework provides the designer with a generic library UAR (User Abstract Rules, described in 3.3.4) that can be more or less specialized to offer either independency or to match a given architecture. Secondly, regarding the heterogeneity of applications, different pieces of an application can present various features, so metrics at different levels of granularity can help to localize parts with specific features (e.g. parallelism). As explained later, our metrics are computed at each level of granularity and thus are naturally available at all levels from the leaf DFGs (Data-Flow Graph) to the complete HCDFG representing the whole C code. Finally, the system metrics from [20, 19] do not address data dependencies so can not include the critical paths during the parallelism analysis; our metric computation includes this feature.

The characterization step implemented in Design Trotter analyzes the functions of an application in order to determine the *orientation* and the *criticity* of a function. The orientation indicates if a function is processing, control or memory oriented. The criticity indicates the average parallelism available in a function. For that purpose a set of metrics has been defined, it is presented in section 4.

3 HCDFG Specification

The aim of the HCDFG model is to represent a function in a way that facilitates its estimation and exploration. The notion of function differs according to the designer point of view. In our work two specification models are considered. The first one, HFSM-HCDFG is based on Hierarchical Finite State Machines, the second one, TG-HCDFG is based on task-graphs (TG).

3.1 HFSM-HCDFG

This type of representation is presented in Fig. 2. The HFSM model has been used in the EPICURE project [21]. The hierarchical decomposition at the system-level is made with Esterel Studio [15]. The specification is based on a two level decomposition approach: event-based with Esterel and calls to C functions. During the specification step, the designer inserts calls to C functions (in the states of the system) when he considers that the HFSM model is no longer suitable. Then, regarding the model presented in Fig. 2, the specification corresponds to the "Process" view. In this view the representation is based on a graph of functions enabling sequentiality, mutual exclusion and parallelism. The designer is responsible for organizing the function setup and for defining their granularity. Finally, the "Function" view corresponds to a specific function, described with the

HCDFG model.

3.2 TG-HCDFG

The other type of specification is related to a classic real-time specification model, i.e., a task-graph. This type of specification is used to performed HW/SW partitioning and real-time scheduling considering cyclic and a-cyclic tasks [5]. This model is described in Fig. 3. It is composed of four levels: the system-level (describing the inputs/outputs constraints), the task-graph, the function graph and the HCDFG level (used to describe a function). We use the Radha Ratan model [22] to specify the task-graph.

3.3 The HCDFG model

3.3.1 HCDFG basics

As previously mentioned (see 2.1.2), the HCDFG model is used to describe the application when the designer evaluates that a control-data flow is well suited at a given hierarchy level during the specification procedure. The HCDFG generation is based on GCC from which a parser has been devised in order to extract and structure only relevant information for design exploration. Namely we build a HCDFG using the following principles :

- Single assignment with comprehensive renaming rules. Firstly we eliminate false dependencies by introducing dependence edges. Then the name of scalar or array variables is built by combining the original name with integer values that are incremented after each read and write accesses (cf. 3.4.2);
- The initial hierarchical structure is used in order to preserve data locality;
- The code hierarchy is also exploited to perform a component based-approach through graph patterns usage. A given C function/block is seen as a single graph-component (HCDFG) that can then be instantiated several times distinguished with specific numbers;
- An efficient multi-dimensional data representation;
- Data-transfer and data-processing nodes are mapped onto a generic library which can be personalized depending on the target architecture (e.g., a graph-component can be associated to a processing unit);
- To summarize, all the elements of the model are represented by graphs. Thus, during the

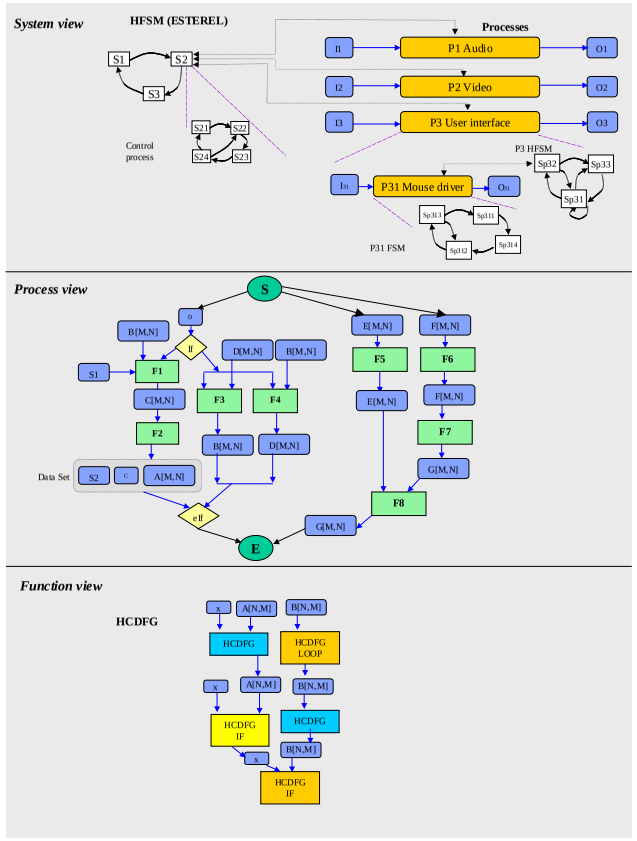


Figure 2: *Specification example with the HFSM-HCDFG model.*

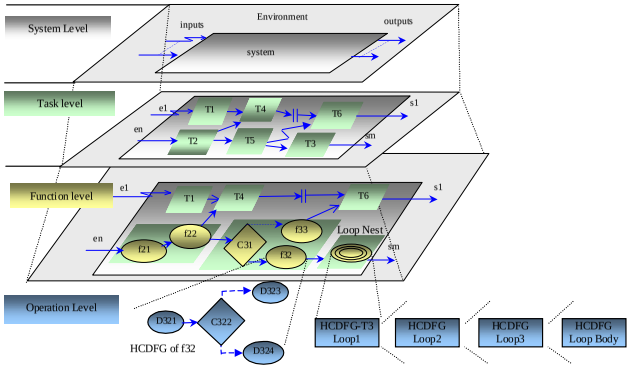


Figure 3: *Specification example with the TG-HCDFG model.*

characterization and estimation steps, important elements and structures can be easily identified by means of our uniform model.

3.3.2 Definitions

The HCDFG model enables the representation of a function in the form of a hierarchical graph containing control structures and processing operations manipulating scalar and array data. This graph has been defined in order to make the characterization and estimation steps efficient. The required information and the results are stored as attributes in the graph. Attribute examples are the ASAP and ALAP dates of nodes, hierarchy levels for memory nodes, metric results and so on. Each function described in the C language is parsed to a HCDFG, an example is given in Fig. 4. A HCDFG is composed of elementary nodes (processing, memory, control), dependence edges (control, data) and graphs that can be hierarchical. The different key features are detailed hereafter.

Elementary nodes A *processing node* (processing vertex) can represent several types of operations: fine grain arithmetic /logic operations but also coarse grain computations (MAC, Butterfly, FIR, DCT, Pixel Shader,...). As the association between operations and resources is defined in the UAR file (described in 3.3.4), it is possible to reference the name of a sub-graph (instance of a graph pattern) in the UAR, indicating that a resource is dedicated to this computation. In that case the graph is seen as a black box, which may already have been estimated (in particular the cycle budget). Processing nodes can be seen on the right part of Fig. 4.

A *memory node* (memory vertex) represents a data-transfer. The main node parameters are the transfer direction (read / write), the data format and the hierarchy level which can be fixed by the designer. Data are explicitly represented by nodes in the graph, and are not, like in many other models, associated to edges. The advantage of such a model is to not duplicate information concerning data and to not overload the specification. For multi-dimensional data (arrays, vectors), addressing mechanisms are explicitly represented in the graph through index DFGs. Memory vertices (scalar and array) are exposed on the right and left part of Fig. 4 respectively.

A *conditional node* represents a test operation (if, case, loops, etc.).

Dependence edges Three types of oriented edges are used to indicate scheduling constraints.

A *Control dependency* indicates an order between operations without data-transfer, for instance a test operation must be scheduled before mutual exclusive branches.

A *Scalar data dependency* between two nodes A and B indicates that node A uses a scalar data produced by node B.

A *Multidimensional data dependency* is a data-dependency where the data produced is not a scalar but an array. For instance such an edge is created between a *loop* CDFG (Control-Data Flow Graph) reading an array transformed by another *loop* CDFG.

Graphs There are six kinds of graphs.

A *DFG* is a graph which contains only elementary memory and processing nodes. Namely it represents a sequence of non-conditional instructions of the 'C' code. The graph on the right part of Fig. 4 is a DFG.

A *CDFG* is a graph which represents a test or a loop pattern with associated DFGs. A CDFG is made of: two control nodes (begin and end) which indicate the type of the structure (*if, switch-case, while, do-while and for*), an evaluation graph, plus an evolution graph in the case of a FOR structure, and finally one or more H/CDFGs which represent the processing conditioned by the control node. Moreover, an attribute enables the saving of the execution probability of each branch (for control structures). The probabilities can be obtained by profiling or can be specified directly by the designer through an interactive and user-friendly interface. The graph in the center of Fig. 4 is a CDFG.

An *Evaluation graph* produces a boolean data, it corresponds to the computation of a condition. The boolean data node is connected to the control node by an order edge.

An *Evolution graph* is found in FOR structures. It represents the increment mechanism of loop indexes (only affine increment mechanism are allowed). An evolution graph corresponds to

the computation of the index increment. The data node representing the index of a FOR loop is connected to the control node by an order edge.

An *Index graph* represents index computations (e.g., $i + (j * 2)$ in array[i+(j*2)] = 0). This is a key feature for detecting the need for address generation units (AGUs).

A *HCDFG* is a graph which contains elementary conditional nodes, HCDFGs and CDFGs. It represents the application hierarchy, i.e., the nesting of control structures and graphs executed in sequential or parallel patterns. The graph on the left part of Fig. 4 is a HCDFG.

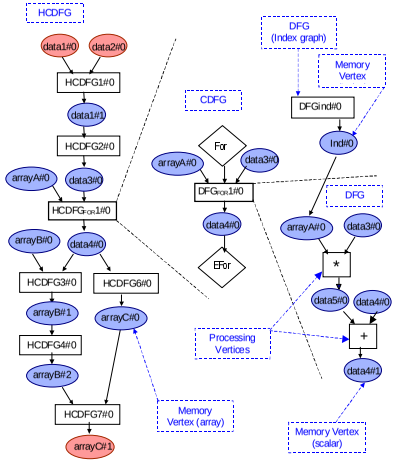


Figure 4: *Elements of a HCDFG.*

3.3.3 Graph creation rules

The composition principle is quite natural. The graph is traveled with a depth-first search algorithm. When no more conditional nodes are found, a DFG is built. Then a H/CDFG is created each time a conditional node is found in the upper hierarchy level. Another stop condition is encountered when the name a graph pattern can be associated to a function already referenced in the architectural model (UAR). In order to facilitate the estimation process, classic CDFG patterns have been defined to identify rapidly the usual nodes like *loop, if, etc.*.

3.3.4 Architecture definition

The designer defines a set of rules, named "UAR" (User Abstract Rules) which aims at describing an architectural model for both characterization (optional) and exploration (at system-level or for

HW and SW projections). When no architecture has been selected the designer can describe any kind of abstract architecture to start the exploration process. Then the UAR can be refined using some feedback from the exploration process. As the UAR model is flexible, the designer can also describe an existing architecture.

The processing part of the architecture is characterized by the type of available resources : ALU, MAC, etc. and the operations they can perform; a number of cycles is associated to every type of operator. The granularity of the operators is not fixed, and coarse grain operators (MAC, FFT or more complex functionality) can be described and associated to operations/functions of the HCDFG description of the application. Regarding the memory part, the user defines the number of levels (L_i) of the hierarchy and the number of cycles (l_i) associated for each type of access.

Fig. 5 shows an example of two User Abstract Rules files. The first one (left) is the initial file where all resources have a latency equal to one cycle. When the designer starts to refine the architectural model (using results given by the system-level estimation), he can add new types of resources or specify resource latencies like in the second file (right). Thus the designer may improve his analysis by means of system-level estimation.

```
<LIBRARY> system
  <OPERATOR> Alu
      <OPERATIONS>
          "+","-","*","/",
          "<=","=","!=",">="
      <ENDOPERATIONS>
      <ATTRIBUTES>
          latency:cycle:=1
          datawidth:INT:=8
      <ENDATTRIBUTES>
  <ENDOPERATOR>
  <OPERATOR> Mac
      <OPERATIONS> "*+"
      <ENDOPERATIONS>
      <ATTRIBUTES>
          latency:cycle:=1
          datawidth:INT:=8
      <ENDATTRIBUTES>
  <ENDOPERATOR>

/* MEMORY */
  <MEMORY> RAM_DP
      <ATTRIBUTES>
          access_mode:=rw
          latency_read:cycle:=1
          latency_write:cycle:=1
      <ENDATTRIBUTES>
  <ENDMEMORY>
<ENDLIBRARY>
```

```
<LIBRARY> system
  <OPERATOR> Alu
      <OPERATIONS>
          "+","-","*","/",
          "<=","=","!=",">="
      <ENDOPERATIONS>
      <ATTRIBUTES>
          latency:cycle:=2
          datawidth:INT:=32
      <ENDATTRIBUTES>
  <ENDOPERATOR>
  <OPERATOR> Mac
      <OPERATIONS> "*+"
      <ENDOPERATIONS>
      <ATTRIBUTES>
          latency:cycle:=3
          datawidth:INT:=32
      <ENDATTRIBUTES>
  <ENDOPERATOR>

/* MEMORY */
  <MEMORY> RAM_DP_LEVEL_1
      <ATTRIBUTES>
          access_mode:=rw
          latency_read:cycle:=1
          latency_write:cycle:=2
      <ENDATTRIBUTES>
  <ENDMEMORY>
  <MEMORY> RAM_DP_LEVEL_2
      <ATTRIBUTES>
          access_mode:=rw
          latency_read:cycle:=2
          latency_write:cycle:=3
      <ENDATTRIBUTES>
  <ENDMEMORY>
<ENDLIBRARY>
```

Figure 5: UAR file examples (left: first approach; right: refinement)

3.4 Java data structure

3.4.1 Two step construction

This section briefly presents how the graph-component approach is implemented in Java. Basically, the internal model is built in two main steps as illustrated in Figs. 6, 7 and 8. The first step consists in translating the C code into the HCDFG grammar and the second one converts the HCDFG description into a Java data-structure. The main advantage of this construction is the independence during the first step from the architectural model defined in the UAR file. The first step extracts the relevant information and the second one exploits it regarding the target architecture.

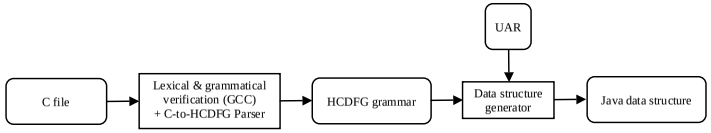


Figure 6: *From C to data structure*

3.4.2 Data and graph representation

The main issues of a graph representation for system design is the size of the resulting structure and its capabilities for information retrieving and synthesis. To cope with this trade-off, we first implement a graph-component approach where a single graph definition can be mapped several times. The second point is the data representation: in order to limit the memory requirements for representing all data and accesses to these data, we use a single assignment rule. It implies that a data is represented only once and that each data access (R/W) can be clearly identified through false data-dependencies elimination. This is done by means of a specific formalism: data names are kept but are extended with two suffixes. For instance, if we consider accesses to data A, the node A♯k%n is interpreted as follows : the "♯ k" symbol indicates the k^{th} write access to data A and the "% n" symbol indicates the n^{th} read access to data A following the k^{th} write acces.

For example, the following C code:

```
y = a*b;
b = y+2;
c = y;
b = c + 1;
```


will lead to the following *accessEdge* elements:

```
a#0%0 , b#0%0 , y#1 ; y#1%0 , b#1 ; y#1%1 , c#1 ; b#2 , c#1%0
```

To avoid size explosion, the array accesses are treated in the same way, independently from the index values. Firstly the index computations are considered during the characterization step as processing operations since they also can impact the critical path computation. Secondly, they are taken into account during the scheduling/combinations/design space exploration step (Fig. 1) since they introduce data dependencies. Finally, the index evolution is considered during the memory size fine-estimation based on polyhedral computation [23]. This is not the topic of this paper but this point also explains our choices. Briefly, the default memory size estimation considers array definitions, the dynamic memory size estimation has been integrated in our tool as an option that takes benefit from our graph structure and scheduling algorithms to implement a polyhedral-based memory size estimation method.

Examples of transformations from a C file to a HCDFG and from a HCDFG to the data structure are given in Fig. 7 and Fig. 8 respectively.

3.4.3 Interactive facilities

The aim of our work is to provide the designer with a fast and easy to use analysis tool. Thus the designer can modifiy, once HCDFGs have been built, the following parameters :

- *Constant values* such as loop bounds, array size, etc. The designer can also choose a set of values in order to observe the evolution of the metrics regarding a given constant;
- *Test probabilities* such as IF or CASE conditions which are equiprobable by default. When such a modification occurs the occurrence probabilities of all graphs are updated and stored.

The result data are stored in an XML file to facilitate exchanges with other tools, to keep the graph hierachy and to get a generic graphic interface.

4 Characterization

The functions composing an application can have very different features in terms of orientation (processing, control, memory) and potential parallelism. Characterizing functions has two objectives: i) guide the designer in his architectural choices and ii) guide the function estimation step in order to use the most adequate scheduling algorithms [1].

We have defined the following metrics:

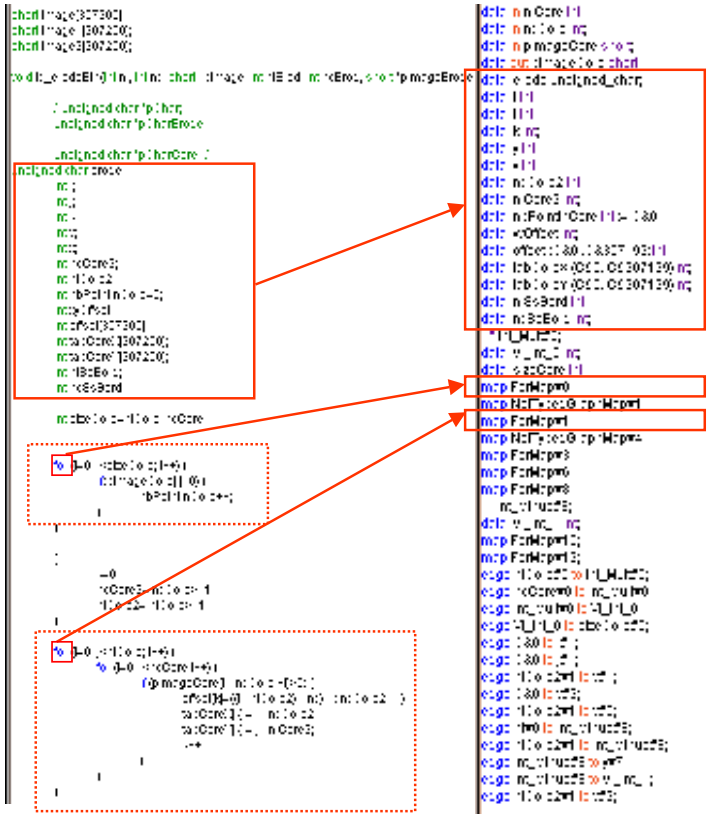


Figure 7: *From C to HCDFG.*

Figure 8: *From HCDFG to data structure.*

- orientation metrics: *Memory Orientation Metric (MOM)*, *Control Orientation Metric (COM)*.
- criticity metric: γ.
- data reuse metric (DRM) and Hierarchical Data Reuse Metric (HDRM).

4.1 Definitions and memory model

Before to give the metric formula details, we introduce some terms which will be used in what follows.

- **Processing**: includes computation operations (ALU, MAC...), addresses computation (which can be performed on ALUs or specific units such as AGUs *Address Generator Unit*) [24], deterministic control (e.g. constant loop bounds that can be eliminated by unfolding);
- **Control**: includes tests, namely control operations that are not computable by a compiler (data-dependant control);
- **Memory**: includes read/write data-transfers. We distinguish the size, read/write access timing and the simultaneous number of accesses. Moreover the memory can be hierarchical. In this work, we have considered two levels from an architectural point of view: i) the local memory which includes cache units and internal registers and ii) the global memory which include RAM, ROM, hard-drives, cf. section 4.1.1.

4.1.1 Memory model

A key point in the following sections is the notion of local and global accesses from a graph point of view. This notion is used during the characterization and the estimation steps. We have distinguished several types of memory nodes (the type is saved as an attribute for each node in a graph):

- N1: inputs/outputs: data identified as inputs/outputs of a graph;
- N2: temporary data: produced by internal processings;
- N3: reused data: input data (N1 subset) reused in a graph (detected with suffixes values);
- N4: accumulation data: annotated with pragmas during the specification.

At the system-level the local memory size associated to the processing unit is not yet known. It is therefore necessary to define a general notion of locality related to the application and not to the architecture. A global access is a data transfer to/from a data which is defined outside the

considered graph. A local access is a data transfer to/from a data defined in the considered graph (register-register transfer from an architectural point of view). N1 data are always global, N4 data always local, N2 and N3 data are initially local but can generate local/global swapping during the scheduling steps if the local memory size is limited [1].

4.2 Computation of the metrics for a function (i.e., for a HCDFG)

The computation of the metrics is performed hierarchically, with an ascending approach. First, the metrics are computed for the leaf graphs (DFGs), then for control graph (CDFG) and hierarchical graphs (HCDFGs) using mutual-exclusive rules (CDFGs) and parallel and sequential rules (CDFGs, HCDFGs). The metrics at a level i of the hierarchy are not simply obtained by a combination of metrics at level i-1, instead they are computed using the features stored a level i-1 such as the number of processing operations, memory accesses, control nodes and the critical paths as explained in sections 4.3.1, 4.3.2, 4.4.3 and 4.4.3.

Note also that on the one hand the orientation metrics (section 4.3) are computed without knowledge of the node ASAP/ALAP dates, only by counting the relevant elements. On the other hand the criticity (section 4.4) and (H)DRM (section 4.5) metric computations use ASAP/ALAP dates computed with UAR features.

Abbreviation	Meaning
Np	number of processing operations
Nc	number of tests (control operations which can not be eliminated at compile time)
Nm	number of global memory accesses operations with Nb_p = number of operations of type ALU, Macs,...+ Index computation + tests
tr/fa	graphs of mutually exclusive branches in "IF-THEN-ELSE" structures
p_{tr}, p_{fa}	execution probabilities of true and false branches respectively (obtained by profiling or specified by the designer, equals 0.5 by default)
for	graph of the core of a "FOR" structure
eval	graph of the evaluation part of a "FOR" structure
evol	graph of the evolution part of a "FOR" structure
N_{ite}	number of iterations in a loop structure

Table 1: *Abbreviation used*

4.3 Orientation metrics

For the three orientation metrics we firstly give a general formula. Then we detail the computations for DFGs, CDFGs (IF-THEN-ELSE and FOR) and HCDFGs. The computation of SWITCH, WHILE and DO-WHILE structures are not presented since they are generalization of IF-THEN-ELSE and FOR computations. The abbreviations used are presented in table 1.

4.3.1 Memory Orientation Metric (MOM)

MOM indicates the frequency of memory accesses in a graph. The general formula for MOM is the ratio between the number of global memory accesses and the number of global memory accesses plus the number of processing operations. Its value is bounded in the interval [0;1]. High MOM values indicates that processing operations are applied to new data (i.e., data entering the graph, as opposed to data computed previously which might reside in the local memory). The more MOM $\rightarrow$ 1, the more the function is data transfer oriented (if MOM = $\frac{3}{4}$ then a processing operation requires, in average, three memory accesses). In the case of hard time constraints, high performance memories are required (large bandwidth, dual-port memory,...) as well as an efficient use of memory hierarchy and data locality [25].

- **DFG MOM**

  For a DFG MOM is computed as follows:

$$MOM = \frac{Nm}{Nm + Np}$$

- **IF-THEN-ELSE MOM (and SWITCH by extension)**

  For this structure MOM is given as the ratio between the number of memory operations in the branches multiplied by their respective execution probabilities and the sum of all the operations in the branches multiplied by their respective execution probabilities.

$$MOM_{IF} = \frac{Nm_{tr} * p_{tr} + Nm_{fa} * p_{fa} + Nm_{eval}}{\sum_{x=p,c,m}(Nx_{tr} * p_{tr} + Nx_{fa} * p_{fa} + Nx_{eval})}$$

- **FOR MOM (and WHILE DO-WHILE by extension)**

  For this structure MOM is given as the ratio between the number of memory operations in each part of the loop (evaluation, core and evolution) and the sum of all the operations in each part.

$$MOM_{FOR} = \frac{Nm_{eval} + Nmfor + Nm_{evol}}{\sum_{x=p,c,m}(Nx_{eval} + Nx_{for} + Nx_{evol})}$$

- **HCDFG MOM**

  For a HCDFG MOM is given as the ratio between the sum of all memory operations in the sub-graphs of the HCDFG and the sum of all the operations in the sub-graphs.

$$MOM_{HCDFG} = \frac{\sum_{\text{sub-graphs j}} Nm_j}{\sum_{\text{sub-graphs j, x=p,c,m}} Nx_j}$$

4.3.2 Control Orientation Metric (COM)

COM indicates the appearance frequency of control operations (i.e., tests that cannot be eliminated during compilation) in CDFGs and HCDFGs (since there is no test within a DFG).

The general formula of COM is the ratio between the number of tests and the total number of operations including processing operations, tests and accesses to global memories. COM values are bounded in the interval [0;1]. The closer to 1 COM is, the more the function is control dominated, so needs complex control structures (if COM = $\frac{3}{4}$ then 1 operation out of 4 is a non-deterministic test). It also indicates that the use of the pipeline technique is not efficient for such functions.

- **DFG COM**

  For a DFG, COM equals 0 since there is no control in a DFG.

- For a CDFG the generic formula for COM is computed as follows:

$$COM = \frac{Nc}{Np + Nc + Nm}$$

- **IF-THEN-ELSE COM (and SWITCH by extension)**

  For this structure COM is given as the ratio between the number of control operations in the branches multiplied by their respective execution probabilities and the sum of all the operations in the branches multiplied by their respective execution probabilities.

$$COM_{IF} = \frac{Nc_{tr}p_{tr} + Nc_{fa}p_{fa} + Nc_{eval}}{\sum_{x=p,c,m}(Nx_{tr} * p_{tr} + Nx_{fa} * p_{fa} + Nx_{eval})}$$

- **FOR COM (and WHILE DO-WHILE by extension)**

  For this structure COM is given as the ratio between the number of control operations in each part of the loop (evaluation, core and evolution) and the sum of all the operations in each part.

$$COM_{FOR} = \frac{Nc_{eval} + Ncfor + Nc_{evol}}{\sum_{x=p,c,m}(Nx_{eval} + Nx_{for} + Nx_{evol})}$$

- **HCDFG COM**

  For a HCDFG COM is given as the ratio between the sum of all control operations in the sub-graphs of the HCDFG and the sum of all the operations in the sub-graphs.

$$COM_{HCDFG} = \frac{\sum_{\text{sub-graphs j}} Nc_j}{\sum_{\text{sub-graphs j, x=p,c,m}} Nx_j}$$

4.4 Criticity metric

The approach used in our methodology consists in estimating the most critical functions first (the less critical ones may reuse the resources allocated to the most critical ones and are therefore estimated after). Criticity is defined by the γ metric such as:

$$\gamma = \frac{\text{Nb processing and memory accesses operations}}{\text{Critical Path}}$$

The critical path of a DFG is defined as the longest chain of sequential operations (expressed in cycle number). The critical path for a function is computed hierarchically by combining the critical path of its HCDFGs.

γ indicates the average parallelism available at a specific hierarchy level: lets consider a HCDFG composed of 5 identical parallel DFGs. If each DFG is internally executed in a sequential manner then γ for each DFGs equals 1 but γ for the whole HCDFG equals 5.

A function with a high γ value can benefit from an architecture offering high parallelism capabilities. On the other hand, a function with a low γ value has a rather sequential execution. In that case the acceleration of this function can be made via temporal parallelism (e.g., long pipeline), depending on COM value. From a consumption point of view a function with a high parallelism offers the opportunity to reduce the clock frequency by exploiting the spacial parallelism.

4.4.1 IF-THEN-ELSE γ (and SWITCH by extension)

The criticity metric for this structure is the ratio between the number of operations in the sub-parts of the control structure multiplied by their respective probabilities and the sum of the critical paths of the sub-parts.

$$\gamma_{IF} = \frac{\sum_{x=p,c,m}(Nx_{tr} * p_{tr} + Nx_{fa} * p_{fa} + Nx_{eval})}{p_{tr} * CP_{tr} + p_{fa} * CP_{fa} + CP_{eval}}$$

4.4.2 FOR γ (and WHILE DO-WHILE by extension)

The criticity metric for this structure is the ratio between the sum of the operations in each part of the loop and the sum of the critical paths of the sub-parts.

$$\gamma_{FOR} = \frac{\sum_{x=p,c,m}(Nx_{evol} + Nx_{for} + Nx_{eval})}{CP_{evol} + CP_{for} + CP_{eval}}$$

4.4.3 HCDFG γ :

For a HCDFG representing a serial execution of sub-graphs the criticity metric is given as the ratio between the sum of all control operations in the sub-graphs of the HCDFG and the sum of all the critical paths.

$$\gamma_{\text{serial}} = \frac{\sum_{\text{sub-graphs j, x=p,c,m}} Nx_j}{\sum_{\text{sub-graphs j}} CP_j}$$

For a HCDFG representing a serial execution of sub-graphs the criticity metric is given as the ratio between the sum of all control operations in the sub-graphs of the HCDFG and the longest of all the critical paths.

$$\gamma_{\text{parallel}} = \frac{\sum_{\text{sub-graphs j, x=p,c,m}} Nx_j}{MAX_{\text{sub-graphs j}}(CP_j)}$$

4.5 DRM and HDRM metrics

4.5.1 DRM metric

Balancing processing and memory access operations is a critical point in system design to face the memory bandwidth bottleneck (as compared to the processors performances). The balance can be obtained through the use of scheduling algorithms adapted to the function orientation. In order to guide function analysis and DFG scheduling we have defined a metric called *DRM: Data Reuse Metric*. At system-level the only memory hierarchy information available is naturally extracted from the algorithmic memory hierarchy (i.e., from the high-level language specification). This metric takes into account the local memory size, which has to be fixed by the designer or estimated. The estimation is performed as explained in the next paragraph. Moreover, as we want to obtain estimates rapidly, methods such as clique coloring have been discarded. The DRM metric gives the global/local accesses ratio. A local access which produces a memory conflict (local memory full) involves a global read and write, thus the number of extra global accesses is $\beta \times EGT$ with β the average number of cycles required by an access to the global memory (main memory

with or without cache capabilities.) In the case of a single main memory which requires only one cycle per access $\beta = 2$ (read plus write).

We use average data lifetime to estimate the quantity of data alive in each cycle, from which the minimum memory size can be derived. Minimum and maximum data-life of data d are defined as follows:

$$MinDL(d) = ASAP(d^n) - ALAP(d^1) + 1$$

$$MaxDL(d) = ALAP(d^n) - ASAP(d^1) + 1$$

where $ASAP$ and $ALAP$ are the earliest and latest scheduling dates respectively, d^1 and d^n the earliest and the latest read access to data d for a given time constraint respectively. The average data-life of data d is then given by:

$$AvDL(d) = \frac{1}{2}\left(MinDL(d) + MaxDL(d)\right)$$

Finally, the number of data alive per cycle is given by:

$$AAD = \frac{1}{T}\sum_{d} AvDL(d)$$

where T is the number of cycles allocated to the estimated function (the estimation process is based on a time constraint scheduler, using several time constraints [1]). The number of local transfers turning into global transfers because of a too small local memory is given by:

$$EGT = \begin{cases} (AAD - UM)T \ if \ AAD > UM \\ 0 \ otherwise \end{cases}$$

where UM is the local memory sized. UM can be defined by the user, its default value is AAD.

If we consider a memory hierarchy with the following characteristics, Level 1 (L1): $latency1(l1) = 1cycle$, L2: $l2 = 2cycles$, L3: $l3 = 3cycles$, and if MR_i is the miss ratio of the cache level i, then:

$$\beta = 1.(1 - MR_1) + 2.MR_1.(1 - MR_2) + 3.MR_1.MR_2$$

If we generalize to K levels of hierarchy (including hard disk and network for instance) we obtain:

$$\beta = \sum_{k=1}^{K}\left(l_k.(1 - MR_k).\prod_{j=1}^{K-1} MR_j\right)$$

Finally the DRM metric is given by:

$$DRM = \frac{N1 + \beta.EGT}{N1 + N2 + N3 + N4}$$

The DRM metric is computed for DFGs and can be used for selecting the most appropriate scheduling algorithm during the estimation step [26].

4.5.2 HDRM metric

The hierarchical DRM (HDRM) extends the DRM metric to inter-HCDFG data reuse (remember that a HCDFG is a graph which contains elementary conditional nodes and parallel and serial HCDFGs and CDFGs). Its computation is general and used for all hierarchical estimations, it is based on two by two HCDFGs clustering. The principle is given in Fig. 9: A1 and A2 are the amount of exclusive input data read by the HCDFG1 and HCDFG2 respectively. A12 quantifies the input data which are common to both graphs and A3 is the amount of result data from the first graph transmitted to the second one. Thus the HDRM computed by the formula 8 provides the ratio of reused data between two HCDFG which can be parallel (A3 = 0), sequential (A12=0) or a combination.

The HDRM metric is computed as follows:

$$HDRM = \frac{A_3 + A_{12}}{A_1 + A_2 + A_3 + A_{12}}$$

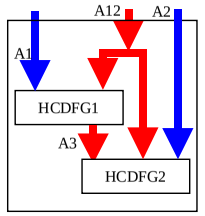


Figure 9: *Illustration of the HDRM metric.*

HDRM = 1 means that the reuse ratio is maximal: all data read by the HCDFG2 are shared or produced by HCDFG 1, it also denotes that a local memory could be efficiently implemented. On the other hand DRM12 = 0 means that no data-reuse is available for memory optimization. In multimedia applications, data reusing has a important impact especially because the optimization opportunities are mainly due to memory management of loop nests. According to that point, we

observe that the HRDM computation can be computed while considering HCDFG1 and 2 as two successive loop iterations.

Illustrative example

We now present how the HDRM computation can be applied to the six nested loops of a classical motion estimation algorithm. Our aim here is to use the HDRM in order to highlight data reuse between subsequent iterations at a given level of hierarchy. If we refer to the illustration in Fig. 9, it means that we virtually consider a loop unrolling where each iteration is embodied by a HCDFG. The HDRM has been computed for the different loop levels: column and row of a $n*n$ (with n=8), column and row of the $2m+n-1*2m+n-1$ reference window (with m=16) and column and row of a WxH frame (QCIF format: W=174, H=144). Table 2 presents the results regarding the Ai and HDRM values for each hierarchical level. (OF and NF mean Old and New frame respectively). The last column shows the size of the memory candidate to store the old frame data at each level of hierarchy. We observe that the best data reuse opportunities are available when the HCDFG-core of the following loop indices are considered: column of the reference window (HDRM = 88%), row of the reference window (HDRM = 81%) and row of the frame (HDRM=87%). It means that including an ad hoc memory hierarchy to locally store these highly reused data can provide high performance and power optimizations.

Level	A1-OF	A1-NF	A2-OF	A2-NF	A12-OF	A12-NF	A3	HDRM	OF Memory size
block column	1	1	1	1	0	0	1	0,20	1
block row	8	8	8	8	0	0	1	0.03	n
window column	8	0	8	0	56	64	0	0,88	$n*n$
window row	39	0	39	0	273	64	0	0,81	$(2m+n-1)*n$
frame column	312	312	312	312	1209	1209	0	0,66	$(2m+n-1)^2$
frame row	1408	1408	1408	1408	5456	25336	0	0,87	$W*(2m+n-1)$

Table 2: *HDRM metric Motion estimation theoretical results. OF: Old Frame, NF: New Frame*

Final remark on the metrics

Compiler optimizations and transformations can have strong impacts on the final code and it would be therefore desirable to evaluate these impacts as regard to the metrics. This point is out of scope of this paper, however it should be possible to evaluate these impacts by accessing post target-independent optimized code and to characterize this code. Finally by comparing the two characterizations, the compiler impact could be evaluated.

5 Results

We have applied the previously defined metrics to functions widely used in embedded systems. Hereafter we present results computed with the aim to be as much as possible independent from any architecture, namely we consider an algorithmic characterization based on a unspecialized UAR. The following examples are detailed: a wavelet transform (DWT) and a 2D-DCT transform, a G722 audio decoder, a TCP protocol and two video applications matching pursuit (MP): Object Motion Detection (OMD). Regarding the OMD application we also provide estimations results on a FPGA target.

5.1 DWT

The DWT algorithm [27] has been implemented using the lifting scheme. It is composed of two sequential blocks (C sub-functions translated into HCDFGs, level N-1) which operate sequentially in the horizontal and vertical dimensions respectively. Each block is made of 6 sub-blocks (C sub-functions translated into HCDFGs, level N-2). The C code is made of one function (translated into a HCDFG, level N) englobing the code for the sub-blocks. The lower level of granularity depend on the structure of each sub-blocks whose metrics are automatically computed like for upper levels. The delay for the whole characterization step equals 500ms on a PIII@700Mhz with Java implementation. Table 3 provides the results for the six functional sub-blocks and for the whole function (top graph). The first observation is that the COM metric equals zero for all graphs, since this application is composed of deterministic loops and does not contain any test. Secondly we observe that MOM values for the wavelet functional blocks are higher than 0,7; this means that more than 7/10 of operations are data accesses, so the application is clearly, at all levels, memory oriented. Finally, γ values are around 1,5 for all the functional blocks. The horizontal and vertical blocks have gamma values equal to 2,704. As the two blocks are executed sequentially, the gamma value for the whole function (top graph) also equals 2,704. This indicates that spatial parallelism is rather weak considering the fine grain sub-blocks and a that a coarse grain parallelism is available, i.e., parallelism between the horizontal and vertical blocks. By analyzing the metrics values, the designer can notice that there i) is no need for complex control structures, ii) are important needs for high data-access requirements and iii) is a coarse grain parallelism. This means that optimizations can be obtained with a pipelined architecture with possible coarse grain dedicated hardware modules providing a large bandwidth. So if high performances are required, a (programmable) dedicated hardware can be introduced within the SOC.

Sub-blocks N-2	**MOM N-2**	**COM N-2**	γ **N-2**
HFirstLiftingStepFOR11	0,721	0	1,576
HFirstDualLiftingStepFOR21	0,721	0	1,576
HSecondLiftingStepFOR31	0,721	0	1,576
HSecondDualLiftingStepFOR41	0,722	0	1,579
HScalingFOR51	0,802	0	2,136
HRearrangeFOR61	0,904	0	1,843
VFirstLiftingStepFOR71	0,721	0	1,576
VFirstDualLiftingStepFOR81	0,721	0	1,576
VSecondLiftingStepFOR91	0,721	0	1,576
VSecondDualLiftingStepFOR101	0,722	0	1,579
VScalingFOR111	0,802	0	2,136
VRearrangeFOR121	0,904	0	1,843
Blocks N-1	**MOM N-1**	**COM N-1**	γ **N-1**
Hlines	?	?	2,704
Vlines	?	?	2,704
Top-graph N	**MOM N**	**COM N**	γ **N**
DWT (top graph)	0,765	0	2,704

Table 3: *DWT characterization*

5.2 G722 Decoder

The UIT-T G722 recommendation is one of the audio part of the H320 standard for video-conference. We have studied the coder part of the application, and more specifically the adaptive predictor block (predicSup). This block is made of 8 sub-blocks (filters) which execute concurrently. The characterization results are found in table 4. We can notice that the results are quite similar to those of the previous example. First of all the COM values are very small, which indicates that there is almost no tests. Next we observe high MOM values which reflect a large number of global memory accesses. Finally the parallelism is weak at fine grain levels (between 1.33 and 2.33 for the eight sub-blocks) and increases at the highest levels of the hierarchy, since the sub-blocks execute concurrently (3.60 and 3.62 for predic and predicSup respectively). The parallelism evolution is quite similar to the DWT example since the parallelism is increasing from level N-2 to N-1 and remains stable at level N, however larger gains are obtained. By considering a cross analysis of MOM and γ we observe that in order to exploit the available parallelism ($\gamma = 3.62$) the architecture should provide enough simultaneous memory accesses since more than 70% of operations are data-transfers. By referring to the metrics the designer should select an architecture with good I/O capabilities and enough computational power to execute the sub-blocks concurrently. For example a large DSP such as the Texas Instrument TMS320C6201 coupled with a I/O co-processor featuring large FIFOs could be used.

Sub-blocks N-2	**MOM N-2**	**COM N-2**	γ **N-2**
parrecEecons	0,714	0	2,333
upzero	0,758	0,039	1,686
uppol2	0,674	0,087	2,045
uppol1	0,743	0,086	2,188
recons	0,603	0,081	2,128
filtez	0,688	0	1,375
filtep	0,5	0	2,000
predic	0,75	0	1,333
Sub-blocks N-1	**MOM N-1**	**COM N-1**	γ **N-1**
predicSup	0,738	0,037	3,602
Sub-blocks N	**MOM N**	**COM N**	γ **N**
predictorSup (top graph)	0,739	0,037	3,621

Table 4: *G722 characterization*

5.3 2D DCT

This application is a well known 2D-DCT for 8x8 image blocks, this example is interesting to illustrate metric interpretations. From a structural point of view, it is composed of two identical and sequential 1D-DCT sub-blocks (operating on lines and columns), so the corresponding graphs have the same metric values as seen in table 5. We can notice that the γ metric extracts the high degree of parallelism (5,71) provided at the lowest level of granularity (N-1). The parallelism does not increase at the second level of granularity (N) because of strict data-dependencies between sub-functions. We also observe that MOM metric is near to 0,5 compared to the DWT example where MOM is greater than 0,7. It means that the reuse of temporary local data is here much more important, it is also related to the larger degree of available parallelism.

Sub-blocks N-1	**MOM N-1**	**COM N-1**	γ **N-1**
DCT8L	0,575	0	5,714
DCT8C	0,575	0	5,714
Sub-blocks N	**MOM N**	**COM N**	γ **N**
DCT8x8 (top graph)	0,575	0	5,714

Table 5: *2D DCT characterization*

5.4 TCP

We have computed the TCP protocol metrics in order to test another kind of classical application. Each function represents a TCP state within a FSM specification. Table 6 shows the analysis result for some representative functions of TCP. We can notice that the functions have relatively high COM values denoting heavily conditioned data-flows. The MOM metric values (greater than 1/3) also indicate an important data accesses frequency. It means that these functions are control-

oriented and require high memory bandwidth. So, a suitable target architecture is a GPP powered by efficient I/O devices. There is no need for a DSP and for a complex data path structure, since the parallelism cannot be exploited since the functional blocks are clearly control oriented. Note that another very efficient architecture could be implemented using a dedicated FSM associated with fast FIFOs.

Functional blocks	**MOM**	**COM**
TCPTIMEWAIT	0,482	0,06
TCPFINWAIT2	0,534	0,055
TCPABORT	0,457	0,343
TCPwakeup	0,333	0,556
Tfinsert	0,5	0,01
TCPdodat	0,375	0,06
TCPSENT	0,508	0,320
TCPRESET	0,667	0,148

Table 6: *TCP characterization*

5.5 Matching pursuit

We have been provided with the Matching pursuit application in the context of a collaboration project with EPFL [28]. The matching pursuit application is a new compression mode which does not operate on pixels but on "atoms" representing basic patterns in a picture. This example is interesting because it is still under development: the specification is still evolving, it is therefore interesting for the designer to be able to evaluate rapidly any modification of the specification. In this example modifications include classical algorithmic transforms such as loop unrolling and so on, but also structural transforms and organization of the code, which can have considerable effects on the performances. This shows that performing fast algorithmic characterization as proposed in our method is justified. Fig. 10 shows the elements of the processing setup. The encoder is based on a genetic algorithm and implemented on a server, we have not focused on this part. The decoding part can be implemented on several systems, including embedded systems. Fig. 10 shows the 4 main blocks of the decoding part.

Fig. 11 shows the results for the 4 main functions of the decoder. Clearly, "DecodeVideo" is the only function which includes some tests, limited however to 3%. We also notice that global memory accesses are frequent, this is due to the reads of the data from the video stream. "DecodeVideo" and "SetPixelValue" have the highest γ values, therefore they are to be examined first for optimization. These metrics have been computed with the first initial specification, these results have been used to refine the specification of the MP application [28], especially to increase the intrinsic parallelism of the "ComputeNorm" function.

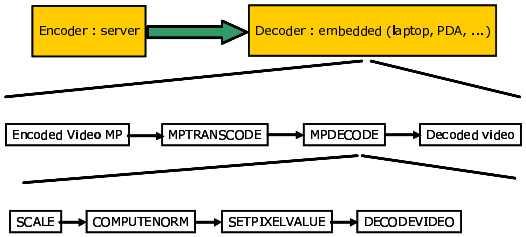


Figure 10: *Matching Pursuit setup (courtesy S. Bilavarn)*

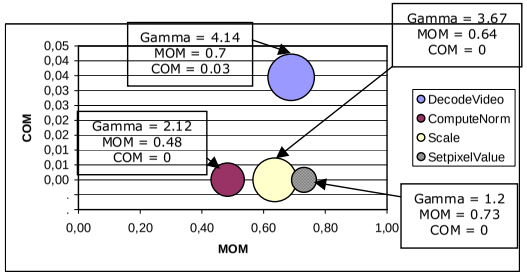


Figure 11: *Matching Pursuit characterization. γ is proportional to the size of the circle.*

5.6 Object Motion Detection (OMD)

This motion detection application have been developed by the LIST laboratory of the CEA research center [29] for the EPICURE project [21]. The typical target architecture is presented in Fig. 12. This is an intelligent video camera made of a CMOS sensor and a processor along with some reconfigurable logic. This application is typically embedded in video cameras and used for parking lot monitoring (detection of car and person moves), person counting in places such as subways and so on. We have used a large set of representative input data (from a parking lot monitoring appliance) to produce a profiling of the application functions used to fill the probability attributes of the graphs (cf.3.3). Fig. 13 illustrates how the OMD application works.

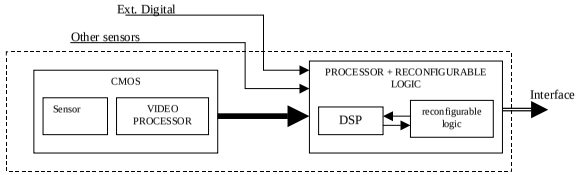


Figure 12: *Motion detection architecture. (Copyrights CEA).*

Figure 13: *OMD motion detection example. Video / background detection / moving object detection*

The OMD application is made of a hundred of functions. The main processing part is made of 31 functions, representing 1740 lines of C code. We have characterized these functions and found out that 16 of them are the most critical ones (i.e., those with the highest criticity metric (γ) values). These functions are those which should be optimized. The functions are quite complex, for example the function "Ic_gravityTest" is composed of 378 C code lines, translated into 2408 lines of HCDFG, the corresponding graph is made of 200 sub-graphs. The function "Ic_labelling" is made of about 200 lines of C code and 1200 lines in the HCDFG description. The results of the OMD characterization are presented in Fig. 14 and table 7. The possibility of characterizing an application rapidly is a important and very useful feature which enables the designer to sketch a

new architecture or to tune an existing one. In this case, all OMD functions have been characterized in a few seconds.

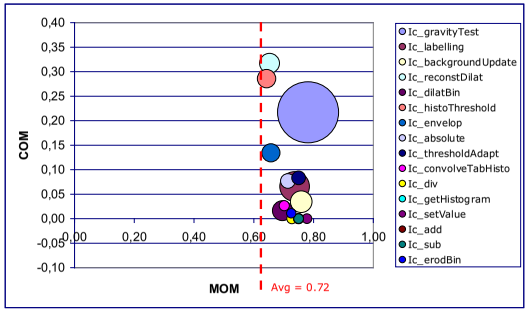


Figure 14: *OMD characterization. γ is proportional to the size of the circle.*

Solution number	**Function name**	**Critical Path (Nb cycles)**	**Gamma**	**MOM [0,1]**	**COM [0,1]**
1	Ic_gravityTest	2102	43,88	0,78	0,22
2	Ic_labelling	395269	10,31	0,74	0,07
3	Ic_BackgroungUpdate	73	5,62	0,76	0,03
4	Ic_reconstDilat	848144	4,75	0,65	0,32
5	Ic_dilatBin	49	4,69	0,70	0,02
6	Ic_histoThreshold	3	4,00	0,64	0,29
7	Ic_envelop	6098184	3,91	0,66	0,13
8	Ic_absolute	327683	2,60	0,71	0,08
9	Ic_thresholdAdapt	327683	2,20	0,75	0,08
10	Ic_convolveTabHisto	15879	1,27	0,70	0,03
11	Ic_div	524291	1,25	0,73	0,00
12	Ic_getHistogram	591622	1,22	0,75	0,00
13	Ic_setValue	1795	1,14	0,78	0,00
14	Ic_add	294919	1,11	0,75	0,00
15	Ic_sub	294919	1,11	0,75	0,00
16	Ic_erodBin	4776219	1,10	0,73	0,01

Table 7: *OMD characterization.*

The first observation which can be made is that all the functions have high MOM values, (0,72 on the average, which indicates that more than 2 operations out of 3 are memory accesses). This is due to the fact that there are numerous reads of data from the video stream and that the application is highly hierarchical (nested control structures for example), it also indicates that the inner DFGs are rather small so few local temporary data can be reused. This implies that the OMD application requires either a large local memory to store reused image data or high end input/output mechanisms (parallel data reading/writing).

Next we observe that γ variations are very significant since it evolves from 1,27 for "Ic_convolveTabHisto" up to 43.8 for "Ic_gravityTest". Using these values it is possible to sort the functions and to find out in what order they should be considered regarding the design of a specific architecture. Focusing on the most critical ones first enables to sketch an appropriate architecture and also to take reusing into account (the functions less critical can be implemented on existing resources allocated to the most critical ones). Finally COM values are comprised between 0 and 0,3 which denotes that that tests are not dominant (most of the control in the application is deterministic).

In the Epicure project we have performed the next steps after characterization, i.e., the scheduling/combinations/exploration step (Fig. 1) and a HW architectural estimation (HW projection in Fig. 1). In the overall design flow the characterization has been used to i) select the 16 most critical functions, ii) explore and detect the functions which have been implemented on FPGA (Xilinx V400EPQ2) and iii) choose a processor for the other functions (ARM922). The functions chosen for hardware implementation (Ic_gravityTest, Ic_labelling, Ic_BackgroundUpdate, Ic_dilatBin, Ic_envelop and Ic_absolute) are those which present high parallelism opportunities (high γ), high MOM and low MOC. In order to illustrate the usefulness of the metrics we present some results of the next steps of Design Trotter, namely the system-level estimation presented in table 8 computed with generic UAR and its projection on a Xilinx V400EPQ2 architecture given in table 9. The functions selected for hardware implementation by means of the metrics have been scheduled/combined/explored with Design Trotter. The results found during that step corroborate the indication given by γ and MOC: in terms of speed-up is has be found for example that Ic_gravityTest can be accelerated with factors up to 2614 as shown in table 8. Finally table 9 gives the results for the hardware projection of Ic_gravityTest on the Xilinx V400EPQ2 FPGA. The choice of the V400EPQ2 is based on the analysis of the metrics and the result of the scheduling/combination/exploration step, since its features suit well the need highlighted by the metrics and the system-level estimation.

6 Conclusion

In this paper we have proposed a high-level methodology and presented an interactive CAD tool which aims at guiding designers of embedded systems. More specifically, the framework enables the rapid characterization and exploration of applications specified using a usual standard language. The outcome is a set of metrics characterizing the application at all levels of hierarchy in terms of processing, control and memory orientation as well as in terms of potential parallelism. This information can be used in three ways:

Solution number	Nb cycles	Speed-up	ALU	MULT	Nb memory accesses	local memory size
1	10940	2614,85	2775	1	14020	25236
2	12634	2264,24	2755	1	13764	24788
3	13265	2156,53	2755	1	13764	24788
4	13461	2125,13	2755	1	13763	24788
5	22267	1284,70	2751	1	13743	24752
6	37979	753,22	791	1	3943	7112
7	41181	694,65	789	1	3933	7094
8	43925	651,36	789	1	3933	7094
9	49413	578,92	789	1	3933	7094
10	52157	548,47	789	1	3931	7094
11	54901	521,05	789	1	3929	7094
12	57645	496,25	789	1	3929	7094
13	60389	473,70	789	1	3928	7094
14	63133	453,11	789	1	3927	7094
15	71365	400,85	789	1	3927	7094
16	75828	377,25	397	1	1967	3566
17	85085	336,21	397	1	1967	3566
18	137221	208,47	397	1	1967	3566
19	139965	204,38	397	1	1966	3566
20	142709	200,45	396	1	1965	3566
21	145467	196,65	396	1	1964	3566
22	145582	196,50	395	1	1963	3564
23	264758	108,05	115	1	563	1044
24	529260	54,05	59	1	283	540
25	1055520	27,10	31	1	143	288
26	2414976	11,85	11	1	43	108
27	4302848	6,65	7	1	23	72
28	8009992	3,57	5	1	13	54
29	8547816	3,35	5	1	13	54
30	9623464	2,97	5	1	13	54
31	10161288	2,82	5	1	11	54
32	10699112	2,67	5	1	9	54
33	11236936	2,55	5	1	9	54
34	11774760	2,43	5	1	8	54
35	12312584	2,32	5	1	7	54
36	13926056	2,05	5	1	7	54
37	16615176	1,72	5	1	7	54
38	26833832	1,07	5	1	7	54
39	27371656	1,05	5	1	6	54
40	27909480	1,02	4	1	5	54
41	28447500	1,01	4	1	4	54
42	28447503	1,01	3	1	3	52
43	28592958	1,00	2	1	3	34
44	28592960	1,00	2	1	2	34
45	28606419	1,00	1	1	2	18
46	28606421		1	1	1	18

Table 8: *System-level scheduling/exploration of IC_gravityTest*

Solution number	Time (ns)	Nb states	Nb Logic Cells	Nb Dedicated Cells
25	20191042,08	4384	868	357
26	46196075,90	4415	428	191
27	82309179,39	857	340	56
28	153223136,97	614	296	36
31	194375278,15	618	296	32
36	266391525,22	625	296	28

Table 9: *IC_gravityTest hardware projection on a Xilinx V400EPQ2 FPGA*

i) when using a fixed architecture the specification characterization guides his algorithmic choices (e.g., parallel vs. sequential execution, loop unrolling, dedicated coprocessors, etc.)

ii) for a fixed specification the characterization guides his implementation choices (e.g., DSP vs. GPP vs FPGA).

iii) when neither the specification nor the architecture are definitely set, the designer can refine conjointly both aspects.

Therefore, by using our methodology the designer is guided, very early in the design process, for evaluating rapidly the impact of his algorithmic choices and choosing or building the most appropriate architecture for his application. This step is part of a high-level co-design environment called Design Trotter of which the goal is to assist designers of embedded systems such as SOCs.

We have presented two key points of this work: the first one is the HCDFG internal representation. This model is fully based on graphs and has been designed to fulfill our own requirements for the characterization and exploration of applications originally specified with the 'C' language. The second one is the characterization step itself. Two types of information are provided for each granularity level of the application functions. Firstly two orientation metrics are given, the MOM metric indicates how influent are data-transfers compared to data-processings, this point is usually related to the data-flow graph depth. The COM metric exhibits the weight of undesirable tests within the application. The MOM metric can be interpreted as a balance between the bandwidth and processing parallelism requirements, the MOC metric predicts the efficiency of spatial and temporal parallelisms. The second kind of metric produces the average level of parallelism comparatively to the critical path. Besides the information given about the available parallelism, this metric also shows up how it is ditributed over the different levels of granularity. Thus it indicates where large gain can be obtained with spatial parallelism but also where pipeline architecture is required.

We have illustrated these concepts with experiments conducted using the Design Trotter framework, which implements the C to HCDFG parser, the computation of the metrics and user interface for results analysis. By referring to the characterization results, designers of embedded systems can get rapidly feedback on their algorithmic choices and be guided in their architectural choices during the selection or the building of an appropriate architecture for the application.

The Design Trotter tool set has been designed as an open and flexible framework for implementing and testing new methods in the area of hardware / sofware codesign of embedded system, thus it constitutes an open space for future work.

References

[1] Y. Le Moullec, J-Ph. Diguet, and J-L. Philippe, "Design-trotter: a multimedia embedded systems design space exploration tool," in IEEE *Workshop on Multimedia Signal Processing (MMSP 2002)*, St. Thomas, US Virgin Islands, December 2002.

[2] Th. Gourdeaux, J-Ph. Diguet, and J-L. Philippe, "Design trotter: Interfunction cycle distribution step," in *11th Int. Conf. RTS Embedded Systems*, Paris, France, April 2003.

[3] S. Bilavarn, G. Gogniat, J.L. Philippe, and L. Bossuet, "Fast prototyping of reconfigurable architectures >from a c program," in *ISCAS 2003*, Bangkok, Thailand, May 2003.

[4] L. Bossuet, W. Burleson, G. Gogniat, V. Anand, A. Laffely, and J.L. Philippe, "Targeting tiled architectures in design exploration," in *10th Reconfigurable Architectures Workshop (RAW)*, Nice, France, April 2003.

[5] A. Azzedine, J-Ph. Diguet, and J-L. Philippe, "Large exploration for hw/sw partioning of multirate and aperiodic real-time systems," in *International Symposium on Hardware/Software Codesign (CODES)*, Estate Park, USA, May 2002.

[6] Gérard Berry, "The esterel primer," http://www-sop.inria.fr/meije/esterel/esterel-eng.html.

[7] N. Halbwachs, "Synchronous programming of rective systems," *Kluwer Academic Publisher*, 1993.

[8] F. Balarin, M. Chiodo, P. Giusto, H. Hsieh, A. Jurecska, L. Lavagno, C. Passerone, A. Sangiovanni-Vincentelli, E. Sentovich, K. Suzuki, and B. Tabbara, *Hardware-Software Co-Design of Embedded Systems: The Polis Approach*, Kluwer Academic Publisher, June 1997.

[9] S. Edwards, L. Lavagno, E. A. Lee, and A. Sangiovanni-Vincentelli, "Design of embedded systems: formal models, validation and synthesis," *Proceedings of the IEEE*, March 1997.

[10] J. Buck and E.E. Lee, ", the token flow model," in *Data Flow Workshop*, Hamilton Island, Australia, May 1992.

[11] A. D. Pimentel, L. O. Hertzberger, P. Lieverse, P. van der Wolf, and Ed F. Deprettere, "Exploring embedded-systems architectures with artemis," IEEE *Computer*, vol. 34, no. 11, pp. 57–63, November 2001.

[12] "Systemc web page," http://www.systemc.org/.

[13] A. Gerstlauer and D. Gajski, "System level abstraction semantics," in *IEEE International Symposium on System Synthesis (ISSS)*, Kyoto, Japan, 2002.

[14] L.Cai, S.Verma, and D.D.Gajski, "Comparison of specc and systemc languages for system design," Tech. Rep. CECS-03-11, Univ. of California, Irvine, Irvine, USA, 2003.

[15] "Esterel studio web page," http://www.esterel-technologies.com/.

[16] L. Guerra, M. Potkonjak, and J. Rabaey, "System-level design guidance using algorithm properties," in IEEE *Workshop on* VLSI *Signal Processing*, San Diego, USA, October 1994.

[17] J-Ph. Diguet, O. Sentieys, J-L. Philippe, and E. Martin, "Probabilistic resource estimation for pipeline architecture," in IEEE *Workshop on* VLSI *Signal Processing*, Sakai, Japan, October 1995.

[18] F. Vahid and D. D. Gajski, "Closeness metrics for system-level functional partitioning," in *EDAC*, Brighton, U.K., September 1995, pp. 328–333.

[19] L. Carro, M. Kreutz, F. Wagner, and M. Oyamada, "System synthesis for multiprocessor embedded applications," in *Design Automation and Test in Europe Conference (DATE)*, Paris, France, March 2000.

[20] D.Sciuto, F.Salice, L.Pomante, and W.Fornaciari, "Metrics for design space exploration of heterogeneous multiprocessor embedded systems," in *International Symposium on Hardware/Software Codesign (CODES)*, Estes Park, USA, May 2002.

[21] M.Auguin, K.Ben Chehida, J-Ph.Diguet, X.Fornari, A-M.Fouilliart, C.Gamrat, G.Gogniat, P.Kajfasz, and Y.Le Moullec, "Partitioning and CoDesign tools & methodology for Reconfigurable Computing: the EPICURE philosophy," in 3^{rd} *Int. Work. on Systems, Architectures, Modeling Simulation (SAMOS03)*, Greece, July 2003.

[22] A.Dasdan, D.Ramanathan, and R.K.Gupta, "Rate derivation and its application to reactive real-time embedded systems," in 35^{th} ACM/IEEE *Design Automation Conf.*, San Francisco, USA, 1998.

[23] P.Grun, F.Balasa, and N.Dutt, "Memory size estimation for multimedia applications," in 6^{th} *Int. Work. on HW/SW Co-Design*, Seattle,USA, Mar. 1998.

[24] M. Miranda, M. Janssen, F. Catthoor, and H. De Man, "ADOPT: Efficient Hardware Address Generation in Distributed Memory Architectures," in 9^{th} IEEE/ACM *Int. Symp. on System Synthesis*, La Jolla, USA, November 1996.

[25] S. Wuytack, J-Ph. Diguet, F. Catthoor, and H. De man, "Formalized methodology for data reuse exploration for low-power hierarchical memory mappings," IEEE *Transaction on* VLSI *Systems*, vol. 6, no. 4, pp. 529–537, December 1998.

[26] Y. Le Moullec, J-Ph. Diguet, D. Heller, and J-L. Philippe, "Fast and adaptive data-flow and data-transfer scheduling for large design space exploration," in *Great Lakes Symposium on* VLSI*(GLSVLSI)*, New-York, USA, April 2002.

[27] M.Antonini, M.Barlaud, P.Mathieu, and I.Daubechies, "Image coding using wavelet transform," *IEEE Transaction on Image Processing*, vol. 1, no. 2, pp. 205–206, April 1992.

[28] S.Bilavarn, P.Vandergheynst, and E.Debes, "Methodology and tools to define special purpose processor architecture," http://ltswww.epfl.ch/ bilavarn, 2003, Intel grant 11409.

[29] L. Letellier and E. Duchesne, "Motion estimation algorithms," Tech. Rep., L.C.E.I, C.E.A, Saclay, France, 2001.