De la spécification formelle au partitionnement matériel logiciel

From Formal Specification to Hardware/ Software Partitionning

par Michel ISRAEL, Denis DUPONT

Laboratoire de Mathématiques et d'Informatique
(EA 1802) Université d'Evry Bd des Coquibus 91025 Evry Cedex

résumé et mots clés

La recherche en CAO de systèmes électroniques ainsi que l'industrie de la CAO ont eu beaucoup de succès et se sont développées, conjointement avec les progrès technologiques. Aujourd'hui, la complexité des systèmes à concevoir est devenue telle que les problèmes deviennent impossibles à gérer humainement aux niveaux bas. C'est pourquoi l'industrie et le monde académique se sont attachés à développer des outils aux niveaux les plus élevés de la conception. Ainsi, la synthèse de haut niveau devient une étape obligatoire dans la méthodologie de conception des systèmes électroniques et la co-spécification de systèmes matériel/logiciels devient un domaine de recherche ouvert.

Systèmes embarqués, co-spécification, codesign, partitionnement logiciel/matériel

abstract and key words

Research on CAD for electronic systems and the CAD industry have seen significant developments, whit technological advances. This paper adresses hardware/software codesign from formal specification to hardware/software partitionning and introduces the reader to various aspects of codesign to help the reader develop a perpective on modern digital system design.

Embedded system, Hardware/Software codesign, Formal specification

1. introduction

Dans la méthodologie traditionnelle de conception de systèmes, les conceptions des parties logicielles et matérielles sont développées de manière indépendante et ce dès les premières étapes. Ces étapes demeurent indépendantes et ont peu d'interaction. Cette approche restreint les possibilités d'exploration de différentes solutions où certaines fonctionnalités pourraient migrer du logiciel vers le matériel ou vice-versa.

Or les dernières avancées technologiques ont permis des développements de composants spécifiques (ASICs) pour un coût et dans un temps raisonnables. Ceci suggère une méthodologie de conception plus souple, où le logiciel et le matériel peuvent être conçus en interaction. Une telle approche doit permettre la réalisation de systèmes dédiés à des applications spécifiques – traitement du signal, télécommunications, ...– où le choix d'implantation matériel/logiciel peut évoluer en fonction des besoins – certaines fonctions réalisées en logiciel au départ pouvant être intégrées au matériel par la suite – ce qui rend possible l'évolution des produits en fonction de la technologie (matérielle/logicielle) et par là même une arrivée plus rapide sur le marché.

Deux problèmes clés sont à aborder dans la co-spécification : d'une part la spécification du système d'autre part le partage matériel/logiciel.

Si l'on veut pouvoir spécifier sans *a priori* matériel ou logiciel, en prenant en compte le système dans son ensemble l'utilisation de spécifications formelles s'impose. Il y a peu d'approches qui se situent au niveau de la spécification de systèmes matériel logiciel. On peut citer le travail du groupe de l'IEEE DASS sur « Z », quelques essais de spécification avec « Z » ont été réalisés notamment celle de la norme Flottante IEEE [1], des approches orientées objet ont été présentées mais elles ne se situent pas au

niveau formel. Ce niveau nécessite l'utilisation de formalismes mathématiques. Cette approche est totalement nouvelle pour les systèmes matériels mais se développe pour les spécifications en logiciel (spécifications en « B » pour la RATP, la SNCF, ...). Ce niveau est assez peu abordé dans la littérature il est cependant décisif si l'on veut assurer la qualité des spécifications et effectuer un découpage cohérent.

Les différentes équipes travaillant sur ce domaine ne l'abordent pas sous l'angle de spécifications formelles mais plus à partir de langages encore assez liés aux cibles (matérielles ou logicielles) tels que C++, Hardware C [2]. Ces langages ne permettant pas de vérifier facilement la cohérence du système, d'en effectuer la preuve, etc.

L'étape de répartition logiciel/matériel consiste à déterminer les parties du système qui seront réalisées en matériel et celles qui seront réalisées avec du logiciel. A partir des spécifications, formelles ou informelles, ou d'un algorithme décrivant le comportement du système, il faut trouver la meilleure réalisation possible en terme de surface, de coût, et de respect des contraintes temporelles qui satisfont les spécifications initiales. Cette répartition est en général faite par le concepteur en fonction de certains critères et de certaines contraintes (vitesse, complexité, ...).

Cet exposé abordera les problèmes de la conception à partir du niveau d'abstraction le plus élevé : la spécification formelle, et présentera le partitionnement matériel/logiciel à travers les développements réalisés au laboratoire LaMI de l'université d'Evry.

1.1. position du problème

Le développement des terminaux et équipements de communications constitue un des défis majeur pour le 21ème siècle. En effet, les équipements pour des applications spécifiques telles que le multimédia, la communication sans fil et les télécommunications constituent un marché important pour les industriels (un numéro spécial des Proceedings de l'IEEE vient d'y être consacré [3]). Les terminaux et autres équipements de communication évoluent rapidement vers l'intégration de systèmes complets sur un seul circuit. Comme les cycles de conception de produits sont très courts, de grandes portions de ces systèmes sont réalisées en logiciel s'exécutant sur des cœurs de processeurs intégrés. Ces cœurs de processeurs sont de plus en plus souvent de type ASIP (: avec un jeu d'instruction spécifique à l'application) intégrés en conjonction avec un cœur RISC standard, ou un microcontrôleur, et de la logique câblée.

Aujourd'hui, les processeurs de traitement du signal (DSP) pour les combinés téléphoniques et stations de base GSM sont en général des processeurs conçus spécifiquement pour ces fonctions (DSP ASIP) plutôt que des processeurs DSP du commerce, dont la consommation serait beaucoup plus élevée pour la même performance. En effet, comme les instructions d'un ASIP sont optimisées pour les nids de boucles critiques pour l'application considérée, pour qu'un DSP standard soit aussi rapide sur cette application il lui faut fonctionner à une fréquence d'horloge beaucoup plus élevée.

L'adéquation des ASIP aux tâches à accomplir permet de réduire le coût de production, d'où une dominance par rapport à d'autres architectures dans ce type de marché, à grands volumes de production et à prix peu élevés.

Les décodeurs vidéo et audio MPEG2 utilisent aussi en général des cœurs ASIP. Ceux-ci sont bien mieux adaptés aux tâches de compression et de sortie vidéo qu'un cœur de type microprocesseur 32-bit à usage général. Comme l'ASIP n'exécute qu'un ensemble de tâches soigneusement optimisé, le problème de compatibilité de logiciel ne se pose pas [4].

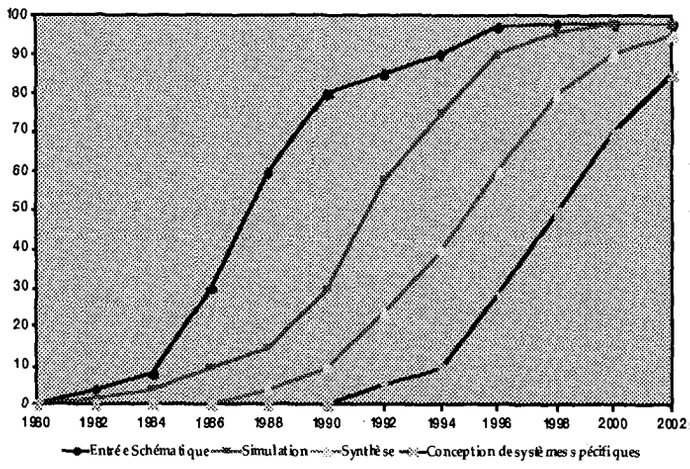
Pour réaliser une même fonction, de nombreuses combinaisons de blocs ASIP, RISC ou co-processeurs câblés sont possibles, et un outil de co-conception et de partitionnement logiciel/matériel permet d'explorer l'espace de conception (aspect spatial). La stratégie habituelle est de faire appel à un cœur de type DSP standard pour la première génération du produit puis de concevoir un ASIP dédié pour les générations suivantes, de façon à diminuer le coût. Un objectif de la co-conception des parties logicielles et matérielles à partir de spécifications de haut niveau est de faciliter cette transition (aspect temporel). Ce thème est donc important et des recherches pour définir des méthodes de conception plus fiables et plus rapides sont en cours dans le monde entier.

1.2. les Tendances

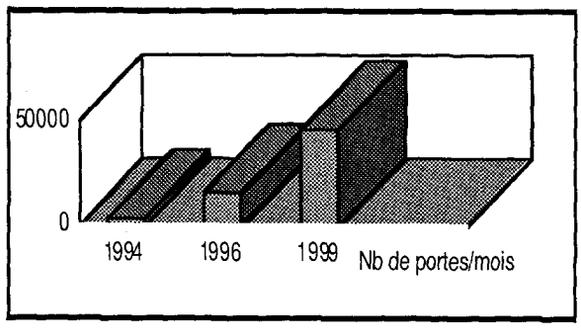
Du fait de la grande compétition existant sur le marché, les outils de co-conception vont jouer un rôle stratégique. Les prévisions des revenus liés aux ventes de circuits, et plus particulièrement ceux utilisés pour les applications spécifiques (dédiées), explique la demande croissante d'outils de conception au niveau système et systèmes spécifiques. Les ventes de ce type de circuit ont une prévision d'augmentation estimée à 34% par Dataquest pour les années 93-98.

La Figure 1(a) suivante [5] présente la pénétration des outils de conception sur le marché. Nous voyons que les outils de conception au niveau système-spécificques sont arrivés depuis 1994 et que leur courbe de croissance est identique à celles des outils de conception correspondants aux niveaux d'abstraction inférieurs (système, simulation, schématique). De la même façon, l'évolution de la technologie des circuits intégrés (0,5 μ en production, 0,3-0,1 en laboratoires) génère des densités d'intégrations de plus en plus importantes (Figure 1(b)) et par conséquent des coûts de fabrication plus élevés ce qui implique un amortissement sur des volumes de production importants.

Cette évolution amène à penser que le logiciel va jouer un rôle-clé pour différencier des systèmes implantés sur des plates-formes matérielles identiques. C'est ainsi que l'on intègre de plus en plus de logiciel dans des circuits que l'on appelle « systems on silicon ». C'est pourquoi le matériel et le logiciel sont les éléments fondamentaux des systèmes complexes.



a)



b)

Figure 1. – (a) Marché des outils de CAO, (b) Densité d'intégration en nombre de portes.

En conclusion de cette introduction, nous pouvons dire que la co-spécification matériel/logiciel est une discipline complexe basée sur différents domaines tels que la compilation, l'architecture et la conception de VLSI. C'est un problème important qui est abordé aujourd'hui de manières différentes suivant le type d'application concernée. Mais cela peut être aussi considéré comme une méthodologie de conception de systèmes hétérogènes complexes.

2. de la spécification formelle au circuit

La gestion de la complexité et de l'hétérogénéité constitue le problème central de la co-spécification. Nous pensons que l'utilisation de spécifications formelles et de la synthèse de haut niveau doit permettre la réalisation de systèmes corrects. Ces spécifications ont comme objectifs :

- la description uniforme d'un système sans *a priori* matériel/logiciel,
- la cohérence des descriptions à tous les niveaux d'abstraction,
- la preuve d'une spécification de bas niveau, par rapport à la spécification initiale,
- la réutilisabilité.

Les différentes étapes menant de la spécification formelle à la réalisation permettent de descendre dans les niveaux d'abstraction (Figure 2).

Deux grandes parties sont abordées : la spécification formelle et le partage matériel/logiciel. Dans la première partie, il s'agit de faire le partitionnement fonctionnel qui permet au concepteur de choisir le niveau de raffinement sur lequel le partitionnement matériel/logiciel doit être appliqué. Ceci peut s'effectuer à partir d'Unités Fonctionnelles Abstraites (UFA) telles que des DSP, FFT, ...; jusqu'à des unités fonctionnelles de bas niveau telles que des additionneurs, des registres,... dans la deuxième partie, une fois ce partitionnement effectué chacune des UFA pourra être affectée à du matériel ou du logiciel en fonction de contraintes telles que performances, coûts, etc., que le produit final doit respecter.

Le but d'un environnement de co-spécification est de fournir des outils permettant le développement d'un système (relatif au traitement du signal ou aux télécommunications) des spécifications formelles à son implantation.

Les environnements de développement pour la conception sont nombreux [21]. Cependant, étant donné la disparité des architectures cibles, du type d'application et des objectifs, ces approches sont difficilement comparables. L'environnement CoWare [22] propose la spécification d'un système avec plusieurs formalismes, ce qui permet de spécifier les systèmes orientés contrôle ou

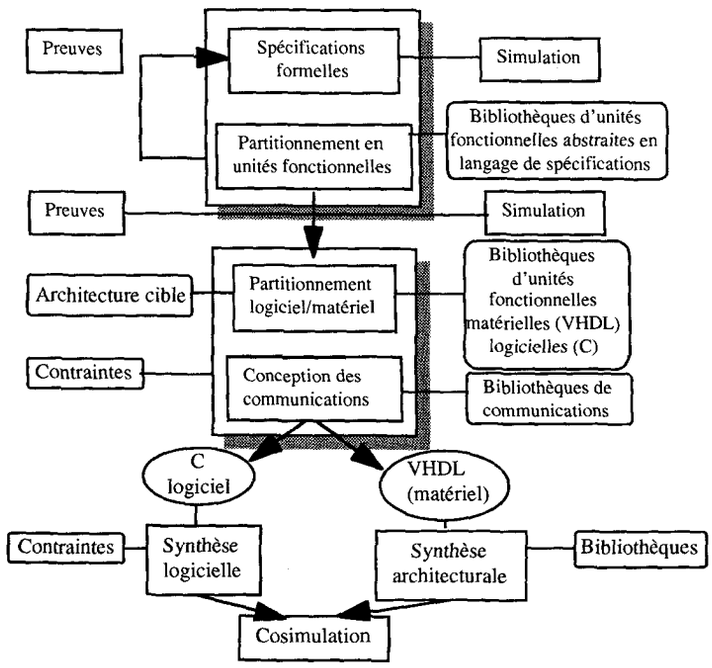


Figure 2. – Des spécifications formelles à l'implantation.

orientés flot de données. PTOLEMY [23] est également un environnement de développement qui a pour but l'encapsulation d'outils mais il permet en plus, l'encapsulation de méthodologies. La description de la méthodologie passe par la définition de domaines de simulation, de génération de code et par une définition imagée sous forme d'étoiles, de galaxies et d'univers.

3. spécification formelle

Dans ce type de spécification, le processus de conception est basé sur une représentation ayant une sémantique précise de façon à ce que la vérification et la correspondance des différents niveaux puissent être assurés. Une telle approche est classique dans certaines communautés pour lesquelles des langages possédant des propriétés formelles puissantes sont utilisés [6]. Les langages synchrones (Lustre, Signal, Esterel) [7] les langages flot de données (Lucid [10], Haskell [11]) en sont des exemples.

Un cadre de spécification est formel si il contient :

• une « syntaxe » rigoureuse définissant ce que le spécifieur est autorisé à écrire.

• une « sémantique » mathématiquement définie décrivant le(s) modèle(s) associé(s) à une spécification donnée.

• une « syntaxe » rigoureuse pour définir les propriétés « bien formées » et (pas nécessairement complète), et un ensemble de règles permettant de prouver si certaines propriétés sont satisfaites par le(s) modèle(s) d'une spécification.

Selon la définition de « spécification formelle » donnée ci-dessus, nous pouvons distinguer deux classes de cadre de spécifications formelles. Le premier cadre de spécification peut être appelé « orienté modèle » est plus largement utilisé que le second cadre de spécifications souvent appelé « orienté propriété » ou encore « déclarative ».

Dans l'approche orientée modèle (VDM, Z, B...) le spécifieur construit un unique modèle à partir de structures de données « built-in » et de constructions primitives que le langage de spécification utilisé offre. En conséquence, un programme est correcte par rapport à la spécification si il possède le « même comportement » que le modèle spécifié.

Dans l'approche orientée propriété, le spécifieur donne une liste de « noms de fonctionnalités » puis déclare un ensemble de propriétés (appelés aussi « axiomes »). Parmi tous les modèles construit sur l'ensemble des noms de fonctionnalités données précédemment, seule une partie satisfont les propriétés requises. En conséquence, un programme est correct par rapport à une spécification si il est muni de tous les noms de fonctionnalités déclarés et si la façon dont il utilise ses structures de données pour exécuter ses fonctionnalités définit un modèle qui satisfait la spécification. Une spécification est alors cohérente si il existe au moins un modèle qui satisfait cette dernière.

Les approches orientées propriétés sont souvent fondées sur les types de données abstraits et les approches les plus populaires sont définies sur des sémantiques algébriques. D'un point de vue académique, le grand avantage des approches algébriques est leur grand pouvoir d'abstraction. En effet, elles peuvent définir le comportement d'un système en s'intéressant uniquement à ce que le système est supposé faire mais non comment il est supposé le faire. De plus, à la différence des formalismes orientés modèles, elles permettent d'écrire des spécifications incomplètes et non exécutables mais possédant tout de même une sémantique (c'est-à-dire un ensemble de modèles satisfaisant la spécification). Enfin, les formalismes orientées propriétés ont un ensemble de primitives et de construction built-in beaucoup plus faible que les approches orientées modèles. En conséquence, l'ensemble des règles d'inférence n'est pas souvent très grand, les preuves sont alors plus courtes et donc beaucoup plus faciles à gérer.

Dans le cadre du projet ECOS [12], nous proposons l'utilisation d'un formalisme algébrique orientées objets (c'est-à-dire orienté propriété) de spécifications : les ETOILE-spécifications [13]. Les méthodologies orientées objets ont déjà montré leur utilité dans la conception de système matériel par leur grand pouvoir abstraction des composants. C'est la raison pour laquelle nous utilisons un langage de spécifications algébrique orienté objets qui inclut à la fois les caractéristiques d'être formel, orienté propriétés et orienté objets. La principale différence par rapport au formalismes algébriques classiques est l'introduction d'une dynamique par transformation d'états des objets.

Cette approche permet de mener dans un cadre parfaitement établi :

• des preuves avant et après raffinement,

• des vérifications de cohérence et de complétude des spécifications,

• le développement incrémental des spécifications,

• la recherche de raffinements successifs à partir de bibliothèque de modèles abstraits logiciel et/ou matériel.

Plus en détail :

Un apport décisif pour les spécifications formelles est de permettre le développement de logiciels par étapes successives de raffinements de spécifications jusqu'à l'obtention d'un programme ou d'une spécification exécutable. Le principal avantage d'une telle approche est que les formalismes utilisés ont tous les niveaux d'implantation sont les mêmes. Ainsi, ils partagent tous les mêmes techniques de preuves. Dans le cadre des ETOILE-spécifications, la correction consiste à traduire algébriquement le fait que l'implantation « simule » complètement le comportement du type d'objets implanté. Ceci revient à montrer que tous les axiomes de la spécification de haut niveau sont des théorèmes de la spécification plus concrète. Toutes ces manipulations peuvent être assistées par ordinateur et un environnement graphique pour les ETOILE-spécifications est en cours de développement. De plus les extensions envisagées sont les suivantes :

• obtention assistée de code « correct *a priori* » (C ou VHDL),

• preuve de correction *a posteriori* d'un programme C ou VHDL,

• génération automatique de test de type boîte noire.

Après que les raffinements successifs de spécifications ont été menés à leur terme, on peut déduire un ensemble de fonctionnalités qui forment la base sur laquelle le partitionnement travaille.

4. partitionnement Logiciel/Matériel

4.1. introduction

Dans une stratégie de conception conjointe, la co-spécification a pour but de permettre au concepteur d'effectuer la spécification d'un système sans se soucier du découpage sur des architectures hétérogènes logicielles et matérielles. Cette spécification abstraite (sans choix d'implantation) est ensuite raffinée (choix d'algorithmes particuliers, choix des représentations des structures de données, ...) pour aboutir à une spécification dans laquelle il est possible d'identifier les fonctionnalités à réaliser. Ces fonctionnalités de base coïncident le plus souvent avec les objets de plus bas niveau spécifiés dans la spécification la plus raffinée.

L'étape suivante consiste à rechercher pour chaque fonctionnalité une réalisation soit logicielle soit matérielle : c'est l'étape de partitionnement logiciel/matériel.

L'objectif du partitionnement est ici de trouver le meilleur compromis entre les parties logicielles et matérielles en fonction de leurs interactions et des contraintes dictées notamment par des critères de performances et de réutilisation.

Nous nous limitons dans la suite de cet article aux méthodes de partitionnement logiciel matériel pour des systèmes embarqués. Avant d'aborder la problématique liée au problème du partitionnement et différentes méthodes existantes, nous résumerons les caractéristiques des composants logiciels et matériels.

4.2. caractéristiques du matériel et du logiciel

Le tableau suivant résume ce qui caractérise les composants logiciels et matériels d'un point de vue performance, flexibilité et temps de conception.

Aux deux extrêmes, on trouve d'un côté le processeur standard offrant la souplesse de la programmation et de l'autre l'ASIC permettant des temps de traitement optimaux. Entre les deux, apparaissent des solutions intermédiaires telles qu'un processeur « brut » (sans interface d'entrées/sorties ou de mémoire associées) et un processeur dédié (DSP par exemple) offrant des compromis souplesse-performance satisfaisants pour une large classe d'applications.

Tableau. 1. – Compromis suivant différentes approches de conception.

	Processeur standard	*Cœur de processeur*	*ASIP*	*ASIC*
Performance	moyenne	moyenne	élevée	la plus élevée
puissance consommée	élevée	moyenne	faible	la plus faible
flexibilité	moyenne	élevée	élevée	faible
temps de conception	faible	moyen	le plus élevée	élevée

Lors de la conception de systèmes hétérogènes, le matériel est plutôt dédié aux parties nécessitant des performances élevées. Par contre, le logiciel est préféré pour des raisons de coût, de facilité d'évolution, de temps de développement plus court et aussi parce qu'il permet des modifications tardives dans la spécification.

Lorsque l'on regarde l'évolution d'un produit industriel (par opposition aux considérations techniques ci-dessus), nous devons préférer une conception logicielle d'un produit dans le début de son cycle de vie pour répondre plus facilement aux évolutions et améliorations de ce produit. Finalement, une réalisation plutôt matérielle à sa maturité permettra une meilleure fiabilité, de meilleures performances et un coût de production plus faible.

4.3. problématique du partitionnement

Le problème du partitionnement se pose dès que nous disposons de plus d'un composant pour réaliser une tâche. Le problème est de déterminer « qui » réalise chaque tâche (ou fonctionnalité). C'est un des problèmes classique de l'allocation des ressources. Or, les contraintes liées au partage des ressources dépendent de l'ordonnancement des tâches. En effet, le parallélisme d'action dépend du nombre de ressources disponibles sur l'architecture. Autrement dit, pour répondre correctement au « qui » il faut répondre au « quand » et réciproquement. Il est donc nécessaire d'aborder la problématique dans son ensemble.

Précisément, pour des systèmes embarqués orientés traitement de données, la problématique est pour chaque tâche :

• de déterminer sa date d'exécution (ordonnancement),

• de choisir une réalisation logicielle ou matérielle (partitionnement),

• de définir le nombre et le type de ressources pour implanter le système (allocation).

L'ordonnancement, le partitionnement et l'allocation sont trois problèmes interdépendants. L'idéal serait de les réaliser simultanément. Cependant, pour échapper à la trop grande complexité du problème général, il faut réaliser des approximations guidées par des choix locaux. Ainsi dans la majorité des cas, l'allocation, le partitionnement et l'ordonnancement sont réalisés séparément. Pour renseigner le concepteur sur la qualité des solutions trouvées, on utilise des estimateurs (estimations des performances, des coûts, ...) afin de prédire les résultats de la conception sans aller jusqu'à la réalisation du système.

Toutes ces considérations amènent une nouvelle façon d'envisager la conception de systèmes où le partitionnement logiciel/matériel devient de plus en plus stratégique.

4.4. état de l'art

Le partitionnement fait l'objet de nombreux travaux [3], qui peuvent être classés en fonction des critères suivants :

– l'interaction entre les étapes de partitionnement, d'ordonnancement et d'allocation,

– le niveau de granularité de la spécification,

– les objectifs,

– le type d'applications traitées,

– la prise en compte des communications,

– le choix d'une architecture cible.

Plus en détail :

• Les étapes de partitionnement, d'ordonnancement et d'allocation sont des étapes interdépendantes. L'approche idéal consiste, comme nous l'avons dit précédemment à réaliser ces trois étapes simultanément. Cependant la taille des graphes est alors limitée par la complexité du partitionnement. Pour les graphes de grande taille (de l'ordre d'une centaine de nœuds), la méthodologie généralement utilisée se caractérise par l'utilisation de méthodes d'optimisation ainsi que par l'emploi, dans chacune des étapes, de termes prédicteurs sur les résultats des autres étapes. Ces termes permettent de traiter séparément chacune de ces étapes interdépendantes.

• Les niveaux de granularité (qui correspond à la finesse de description) varient du niveau instruction ou groupe d'instructions, au niveau bloc système. La problématique est ici de trouver un compromis entre la taille de la granularité et le nombre de grains à traiter pour éviter une explosion combinatoire. Le niveau de granularité est en général fixe et dans le cas d'utilisation de bibliothèques de composants préexistants, le niveau de granularité correspond au niveau de granularité de la bibliothèque. Une meilleure approche serait évidemment une granularité variable, hiérarchique qui permettrait de parcourir entièrement l'espace de conception. Cependant une telle approche entraînerait une complexité impossible à traiter. L'expérience du concepteur est donc pour l'instant grandement sollicitée, afin qu'il modélise le système au niveau de granularité qui lui semble le plus approprié en fonction des bibliothèques qu'il utilise.

• Les principaux objectifs du partitionnement sont la minimisation du coût (surface, consommation,...) et de la durée d'exécution du système. Un compromis entre ces deux minimisations est généralement recherché.

• Le type d'application (orienté contrôle ou données) est également une caractéristique importante, car les applications flot de données permettent un ordonnancement statique alors que des applications orientées contrôle nécessitent un ordonnancement dynamique (la date de début d'exécution de certaines tâches est en effet indéterminée).

• Les types de communication utilisés et le degré de prise en compte des contraintes des communications lors du partitionnement sont également des critères de différentiation des approches existantes. La conception des communications est composée de deux étapes. La première étape consiste à déterminer l'architecture de communication la plus appropriée pour le transfert des données. Pour définir cette architecture, il faut déterminer le canal de transmission, le protocole et l'unité réalisant le transfert. Le problème des communications se pose donc en termes de choix de ressources logicielles ou matériels. La seconde étape consiste à implanter l'architecture qui a été définie lors de la première étape. Cette étape est généralement appelée synthèse de l'interface.

• Dans le cadre de la conception conjointe, l'architecture cible définit le support du système. Elle peut être déduite après le partitionnement par l'analyse des besoins. Pour des applications spécifiques, notamment en traitement du signal, le choix du processeur fait partie intégrante du partitionnement et des choix du matériel à effectuer. L'architecture cible peut être imposée avant le partitionnement si le concepteur possède un modèle d'architecture qu'il pense être efficace. C'est une étape de prépartitionnement qui restreint les choix de réalisations possibles.

Le tableau 2 permet une comparaison d'un certain nombre de méthodes en fonction des critères utilisés. Il est intéressant de noter l'hétérogénéité de ces différentes approches, qui empêche une réelle comparaison qualitative.

Finalement, il ne semble n'y avoir aucune méthode de partitionnement qui l'emporte parmi toutes les techniques existantes. Ceci est due principalement à la complexité intrinsèque du problème, qui exclut une formulation exacte d'une fonction de coût réaliste pour le cas général.

4.5. méthodologie de partitionnement dans le système ECOS

Maintenant que le domaine de la conception conjointe a été présenté dans sa généralité, ce paragraphe va se concentrer sur la méthodologie de partitionnement développée au sein du projet ECOS. Afin de positionner clairement le projet ECOS par rapport aux autres projets de conception conjointe une description des caractéristiques principales du partitionnement est donnée ci-dessous.

4.5.1. le partitionnement

L'algorithme développé au sein du projet ECOS détermine simultanément :

• le choix de l'implantation : assignation matérielle ou logicielle,

• l'ordonnancement : calcul de la date d'exécution,

• le choix de l'entité réalisant la tâche.

Tableau. 2. – Comparaison des méthodes de partitionnement.

Auteur	*Modèle*	*Fonction de coût*	*Algorithme*	*Communication*	*Architecture cible*
Gupta [18]	CDFG	temps	heuristique	bus, mémoire mémoire	processeur+ ASIC(s)
Kumar[8]	set-based	profiling	programmation linéaire	?	?
Chou [14]	diagramme de temps	temps, surface	min-cut	synthèse des protocoles	microcontrôleur
Kalavade [16]	acyclique DFG	temps	heuristique par liste	bus	DSP+ ASIC(s)
Henkel [19]	CDFG	profiling, synthèse	simulation annealing	mémoire partagée	processeur + coprocesseur
Vahid [9]	acyclique DFG	profiling, communication	ILP	?	processeur(s)+ ASIC(s)
Barros [15]	HDL(unity)	affinité	clustering	mémoire partagée partagée	processeur+ ASIC(s)
Ben Ismail [17]	processus communicant	résultat de simulation	hand	bus, FIFO	processeur, ASIC, FPGA, ASIP
Freund [20]	acyclique DFG	surface, communication	heuristique, exhaustif	bus, FIFO, DMA	processeur+ ASI

L'exécution simultanée de ces trois tâches permet d'obtenir de meilleurs résultats qu'une exécution séparée. En contrepartie, cette approche ne peut traiter que des graphes de petite taille (<50 nœuds).

L'objectif du partitionnement est la minimisation de la surface du circuit final sous une contrainte de temps, en proposant un ordonnancement des tâches qui minimise le nombre de ressources. L'algorithme de partitionnement privilégie la réutilisation des ressources aux dates où celles-ci sont disponibles, en tenant compte des contraintes de séquentialité des tâches du composant logiciel. La valeur minimum des coûts associés aux tâches fixe la date d'exécution et le choix de la réalisation logicielle ou matérielle.

Le coût associé aux tâches à une date est la somme des coûts des ressources utilisées par un nœud. Si ces ressources sont déjà disponibles, et donc réutilisable, le coût de cette ressource est nul.

4.5.2. spécification du partitionnement

La description initiale du système est un graphe direct sans cycle (DAG). Les nœuds du graphe représentent les calculs et les communications potentielles; les arcs du graphe représentent les dépendances de données. Une modélisation plus fine est ensuite utilisée, dans laquelle chaque nœud du graphe est décomposé en une liste de ressources. Ces ressources peuvent être des fonctions, des opérateurs arithmétiques, des supports de communications ou bien des mémoires.

Le graphe est en principe extrait de la spécification la plus raffinée. Cette extraction est actuellement manuelle. Des recherches sont en cours pour l'assister et l'automatiser. Le graphe ne contenant pas de contrôle, l'algorithme de partitionnement réalise donc des ordonnancements statiques et ne sera pas remis en cause pendant l'exécution.

4.5.3. domaine d'application

Le domaine d'application étudié par le projet ECOS est le traitement du signal. Les applications de traitement du signal visées sont des applications orientées flot de données. Les signaux d'entrée traversent donc périodiquement le système où ils subissent des traitements pour ensuite ressortir après un temps de latence fixe. L'objectif de la conception conjointe est alors de fournir une implantation qui garantisse que cette contrainte stricte de temps soit respectée.

Une autre spécificité des applications de traitement du signal réside dans le volume des données traitées. En effet, les données sont souvent des tableaux ou des matrices de grandes dimensions. De plus, les données issues d'un calcul doivent fréquemment être stockées pour être réutilisées par l'exécution suivante. Les communications et la mémorisation des données apparaissent donc comme des problèmes-clés des applications de traitement du signal.

4.5.4. les communications

Dans le projet ECOS, la conception des communications était initialement réalisée manuellement après partitionnement. Cependant au vu des résultats obtenus sur plusieurs exemples industriels, une conception automatique avec la prise en compte des temps

de communication et du coût des ressources associées dès l'étape du partitionnement, s'est avérée nécessaire.

L'approche développée dans le projet ECOS consiste donc à prendre en compte l'ensemble des contraintes des communications lors du partitionnement. En contrepartie, le nombre d'informations à traiter lors du partitionnement étant augmenté, la taille des graphes pouvant être traitée avec cette approche s'en trouve diminuée.

La méthode proposée pour optimiser les communications consiste à considérer les communications comme un nœud du graphe, dont les caractéristiques d'implantation (temps de transfert, coûts des ressources) sont rangées dans des bibliothèques. L'optimisation des communications est alors ramenée à un problème classique de partitionnement. Un des intérêts de cette approche est d'utiliser l'algorithme de partitionnement existant pour réaliser l'ordonnancement des communications et de la sélection du modèle de communication pour chaque transfert.

4.5.5. *architecture cible*

L'architecture cible est constituée d'un ou plusieurs ASICs et d'un processeur. Dans le domaine du traitement du signal, et donc dans les télécommunications, les DSPs fournissent un compromis performance/coût/consommation très intéressant. D'ailleurs, leur présence dans l'implantation des systèmes augmente de 38de processeurs est donc utilisé pour implanter la partie logicielle. Quant à la partie matérielle, elle doit exécuter des tâches le plus rapidement possible, ce qui oriente notre choix vers les ASICs.

Dans le projet ECOS, l'algorithme de partitionnement qui minimise le coût du système se limite à cette architecture simple; cependant, l'extension de l'architecture cible à plusieurs processeurs est en cours de développement.

4.5.6. *caractéristiques logicielles et matérielles*

Afin de réaliser le partitionnement, il est nécessaire de disposer des caractéristiques d'implantation des nœuds (temps d'exécution, coût...)... Or, pour ce type de processeurs, les compilateurs ainsi que les estimateurs fournissent de mauvais résultats, car ils sont obtenus par estimation à partir d'une description de haut niveau (description en C par exemple). Dans le cas de l'estimation logicielle, le taux d'erreur obtenu pour les caractéristiques est de l'ordre de 10%.

Ainsi, Le projet ECOS utilise des éléments déjà conçus et dont les caractéristiques exactes obtenues par simulation sont stockées dans des bibliothèques. Notre approche des communications est également fondée sur l'utilisation de bibliothèques contenant les temps de transfert et le coût des ressources de communication (bus, mémoire).

Cette approche a pu être possible grâce aux bibliothèques matérielles fournies par le CNET, et aux bibliothèques logicielles fournies par le CNET et l'I3S.

4.5.7. *outil graphique d'aide à la conception*

Afin d'enrichir l'environnement de développement ECOS, un outil graphique a été développé. Celui-ci intègre les algorithmes de partitionnement, notre approche des communications et la visualisation du graphe avant et après partitionnement. De plus, il permet le partitionnement automatique et manuel de la spécification en intégrant l'optimisation des communications.

La capture d'écran de cet outil montre dans la partie gauche de la fenêtre le graphe de dépendance avant partitionnement. Le système partitionné est visible dans la partie droite de la fenêtre, et les caractéristiques des nœuds et des bibliothèques dans la partie basse de la fenêtre (Figure 3).

D'un point de vue pratique nous avons proposés plusieurs architectures logiciel/matériel d'un système d'annulation d'écho acoustique [20]. La méthodologie mise en oeuvre nous permet, à partir de spécifications de haut niveau (modèle VHDL comportemental) de produire un prototype du système, composé d'un processeur de traitement du signal et d'une partie matérielle correspondant à la description structurelle d'un ASIC.

Cet environnement de développement est une première brique indispensable pour la conception d'un système. Cet environnement reste cependant incomplet. Nos efforts actuels portent sur la liaison entre les spécifications formelles ETOILE-spécifications et l'étape de partitionnement. De même, il est envisageable d'insérer des outils de preuve. Cet environnement sera ainsi peu à peu agrandi afin de superviser tout le processus de conception d'un système.

5. conclusion

Nous avons montré dans cet article l'importance des spécifications formelles et du partage matériel/logiciel pour la conception de circuits dédiés.

Il ne faut pas perdre de vue que la spécification à haut niveau n'est qu'une description des fonctionnalités d'un système mais pas de la façon dont il doit être réalisé.

Si l'on considère que le processus de conception est itératif et nous fait passer d'un niveau de raffinement à un autre en prouvant la validité de la transformation, il n'en est pas moins vrai que ce processus de conception pour la co-spécification n'en est qu'à ses balbutiements. Nous bénéficions de l'expérience acquise dans le domaine du logiciel, notamment pour la spécification de systèmes hétérogènes, appliquée aux systèmes matériel/logiciel.

Il existe cependant un espace à remplir dans le cycle de conception qui doit permettre le passage de spécifications non exécutables à des spécifications exécutables. Ce domaine est neuf et ouvert, il constitue un axe de recherche important à la fois pour le logiciel et le matériel.

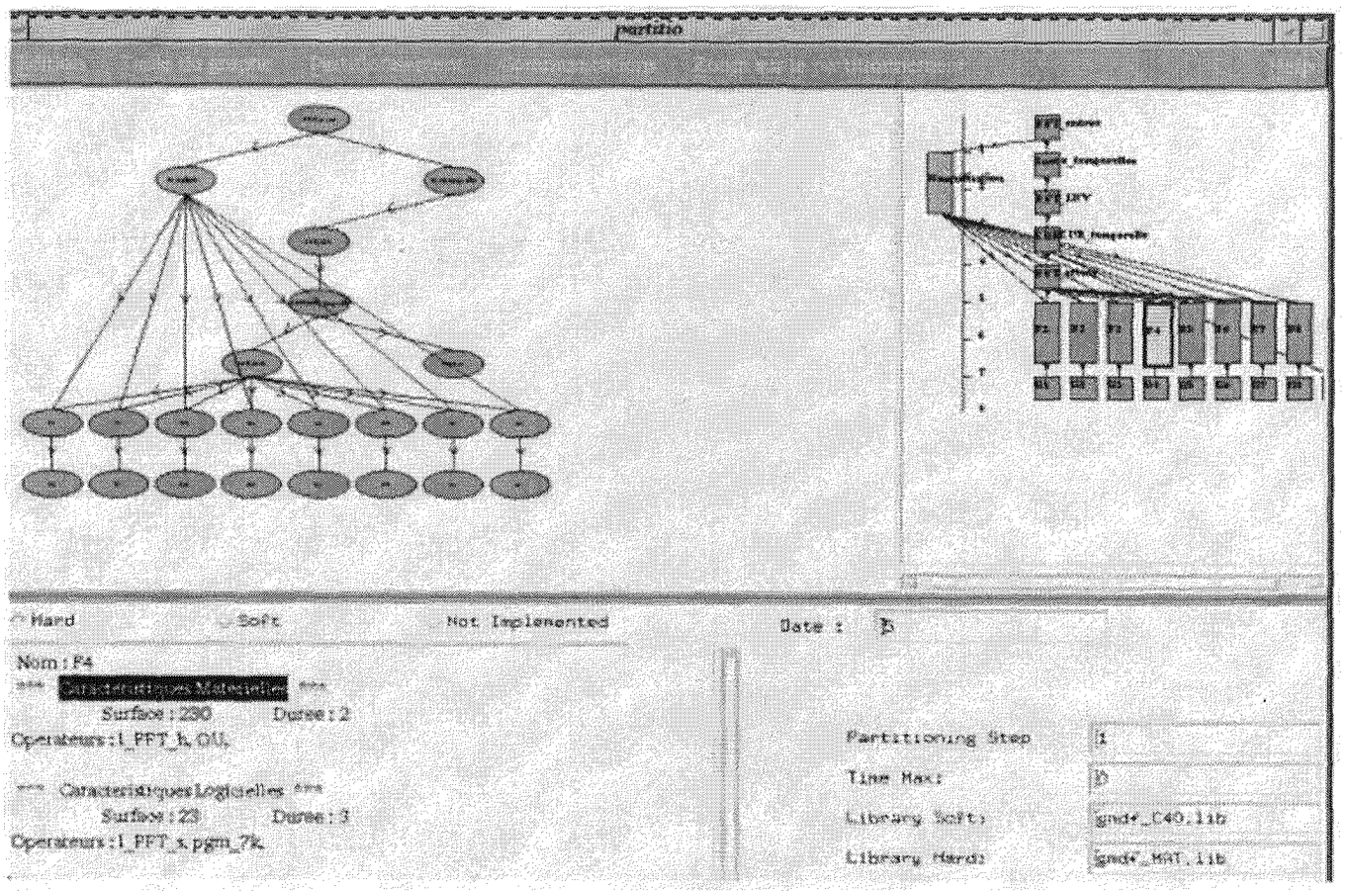


Figure 3. – Outil de visualisation graphique du partitionnement.

Le partage matériel/logiciel, qui fait appel à des techniques connues d'ordonnancement et d'allocation de ressources, doit intégrer les communications entre les différentes fonctions. Celles-ci revêtent une grande importance, car leur coût peut représenter 20

Nous pensons que la co-spécification et la co-conception de systèmes, plus particulièrement de systèmes spécifiques pour le traitement du signal, va prendre une grande importance dans un avenir proche. La co-spécification demande des compétences en architecture, en spécifications formelles, en graphe et en systèmes distribués. Il sera donc fondamental de pouvoir réunir ces différentes compétences pour pouvoir déboucher sur des systèmes et des méthodologies cohérentes et efficaces.

Remerciements

Les auteurs tiennent à remercier les équipes d'architectures et de spécifications du LAMI pour leur collaboration durant cette étude.

BIBLIOGRAPHIE

[1] G. Barrett «*Formal Methods applied to a floating point number system*», IEEE trans. on Software Eng. May 1989, pp 611-617.

[2] (Hardware C).

[3] Proceedings of the IEEE, Special issue on Hardware/Software co-design, Numéro de mars 1997.

[4] P. Paulin *et al.*, «Embedded software in real-time signal processing systems : application and architecture trends», Proc. IEEE, Vol. 85, No. 3, Mar. 1997.

[5] F. Moussa, Invited paper, Fall VIUF, Boston, 1995.

[6] S. Edwards *et al.*, «Design of Embedded Systems : Formal Models, Validation, and Synthesis», Proc. IEEE, Vol. 85, No. 3, Mar. 1997.

[7] Proceedings of the IEEE, Special issue, Numéro de septembre 1991.

[8] S. Kumar, J. H. Aylor, B. W. Johnson, and W. A. Wulf, «Exploring hardware/software abstractions and alternatives for codesign», in Proc. Int. Workshop on Hardware-Software codesign,1994.

[9] F. Vahid and D. G. Gajski, «Specification partittionning for system design», in Proc. Design Automat. Conf., June 1992.

[10] W. Wadge, E.A. Ashcroft, «Lucid, the dataflow programming language», New York, Academic, 1985.

[11] A. Davie, «An Introduction to functionnal programming using Hashkell», Cambridge University Pres, 1992.

[12] Aiguier M *et al*, «ECOS : A Generic Codesign Environment for the Prototyping of Real Time Applications, 'From Formal Specifications to Hardware-Software Partitioning'», Current Issues in Electronic Modeling (CIEM), Issue #8 : Kluwer Academic Publishers 1996.

[13] Aiguier M., Bernot G., «ETOILE-specifications : an Object Oriented Way of Specifying Systems», Proc. of the Workshop on Proof Theory of Concurrent Object-Oriented Programming, Linz, Austria, July 8, 1996, J-P. p.50-57, 10th European Conf. on Object-Oriented Programming (ECOOP 96), 1996.

[14] P. Chou, E. A. Walkup, and T. Benner, «Adaptatation of partitionning and high-level synthesis in hardware/software co-synthesis», in Proc. Int. Conf. On Computer-Aided Design, Nov 1994.

[15] E. Barros, W. Rosentiel, and X. Xiong, «Hardware/software partitionning using whit UNITY», in Proc. Int. Workshop on Hardware-Software codesign,1994.

[16] A. Kalavadde and E. A. Lee, «A global criticality/local phase driven algorithm for the constrained hardware/software partitionning probleme», in Proc. Int. Workshop on Hardware-Software codesign,1994.

[17] T. B. Ismail, M. Abid, and A.A. Jerraya, «COSMOS/ a codesign approach for communication systems», in Proc. Int. Workshop on Hardware-Software codesign,1994.

[18] R. K. Gupta, C. N. Coelho Jr., and G. De Micheli, «Program implementation schemes for hardware-software systems», IEEE computer, pp. 48-55, Jan 1994.

[19] R. Ernst, J. Henkel, T. Benner, «Hardware-Software Co-synthesis for microcontrôleur», IEEE Design and Test of Computer, Vol 10, N°4, Decembre 1993.

[20] L. Freund, D. Dupont, M. Israël, F. Rousseau : «Interface Optimization During Hardware-Software Partitioning», 5th International Workshop on Hardware/Software Co-Design, Codes/CASHE '97, Braunschweig, Germany - 24-26 March 1997.

[21] E. K. Pauer. Multiprocessor System Development for High Performance Signal Processing Applications in 8th IEEE International Workshop on Rapid System Prototyping, Chapel Hill June 1997.

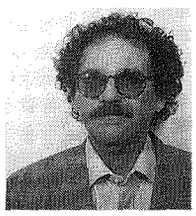
[22] I. Bolsens, H. De Man, B. Lin, K. Van Rompaey, S. Vercauteren, D. Verkest. Hardware/Software Co-Design of Digital Telecommunication Systems in Special Issue on HARDWARE/SOFTWARE CO_DESIGN, Proceedings of the IEEE, Vol. 85, NO. 3, March 1997.

[23] J.T. Buck *et al.* «PTOLEMY : «A framework for simulating and prototyping heterogeneous systems», in Int. J. Computer Simulation, Jan. 1994.

Manuscrit reçu le 23 juillet 1997.

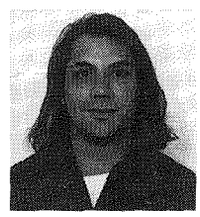
LES AUTEURS

Michel ISRAËL



Michel Israël est Professeur des Universités à l'université d'Evry Val d'Essonne où il est directeur du département informatique et responsable du département du LaMI. Son axe de recherche est la CAO de systèmes et d'architecture depuis 1982, il est depuis 1993 président du comité technique Design Automation de l'IEEE Computer Society. Il est l'auteur de nombreuses communications dans le domaine du Design Automation.

Denis DUPONT



Denis Dupont est Maître de conférences à l'université d'Evry Val d'Essonne. Son axe de recherche porte sur la CAO de systèmes et d'architecture.