1

Introduction

Ce chapitre introductif présente le contexte général de cette thèse, la problématique traitée et donne un aperçu global de la démarche suivie et du travail effectué. Nous décrivons tout d'abord l'activité de conception conjointe matériel/logiciel appelée usuellement co-design, son domaine d'application et les raisons de l'émergence de ce nouveau cycle de développement. Nous présentons ensuite succinctement l'approche particulière que nous avons adoptée pour résoudre deux problèmes sous jacents de l'activité de co-design: le partitionnement matériel/ logiciel et la co-simulation. Pour conclure ce chapitre, nous donnons l'apport de cette thèse et le plan de ce manuscrit.

1.1 L'ACTIVITE DE CO-DESIGN

La complexité croissante des systèmes pour lesquels la réalisation résulte de l'association d'une partie matérielle et d'une partie logicielle, la diversité des choix technologiques et les contraintes de coûts et délais de plus en plus sévères nécessitent l'utilisation de nouvelles méthodologies et outils logiciels associés pour diminuer leur durée de conception et accroître leur qualité. Le co-design est l'une de ces méthodologies.

1.1.1 Définition du terme co-design

Le terme "Hardware/Software Concurrent Design" souvent abrégé par "Hw/Sw Codesign" et qui se traduit par conception conjointe matériel et logiciel représente un processus de conception complet basé sur la trilogie: modèles, méthodes et outils ESDA (Electronic System Design Automation). Ce processus doit permettre aux concepteurs de transformer correctement du premier coup les spécifications d'un système en un produit industriel comportant une partie logicielle et une partie matérielle et satisfaisant les contraintes

fonctionnelles et non fonctionnelles de son cahier des charges. Il doit également permettre d'accroître la qualité de conception et de réduire le temps de développement.

Dans la trilogie, un modèle est une représentation formelle d'un système à un niveau d'abstraction donné. Il est bon de noter qu'une méthode efficace doit reposer sur un ensemble de concepts de modélisation restreint mais suffisant pour décrire n'importe quel système. Le terme méthode représente une procédure ou démarche bien définie et structurée permettant de résoudre un problème. Les outils ESDA se composent d'outils de capture de modèles, de simulation, d'analyse statique ou dynamique, de recherche de compromis, de synthèse et de co-vérification. Les spécifications non fonctionnelles représentent tout ce qui dans le cahier des charges n'est pas une exigence fonctionnelle. Il s'agit habituellement d'un ensemble de contraintes d'intégration avec l'environnement (taille, puissance consommée), de performances (temps de réponse, débit), d'ordre économique (coût, délai de fabrication), de qualité (durée de vie, MTBF), de sûreté de fonctionnement, etc.

Gajski donne la définition suivante [MCC/OMI-96]: "*CoDesign is defined as a methodology and technique for designing software and hardware concurrently, thus reducing the design time and time to market*". Dans cette définition, le terme le plus important est "concurrently" qui signifie que le développement simultané de la partie logicielle et de la partie matérielle du système s'effectue avec une interaction forte et permanente entre les deux parties. Cette approche diffère fondamentalement du cycle de conception conventionnel des systèmes qui repose sur un développement séparé des deux parties.

1.1.2 Les classes de systèmes concernées par le co-design

Le développement des systèmes électroniques composés d'une partie matérielle et d'une partie logicielle n'est pas un problème nouveau. Généralement, concevoir et réaliser de tels systèmes nécessite une compétence technique dans au moins 3 domaines: l'électronique analogique, l'électronique numérique et l'informatique. La nature spécifique du traitement à effectuer et le couplage du système avec son environnement nécessitent aussi des compétences complémentaires: en traitement de l'information (signal, image, parole...), en électronique de puissance lorsque l'environnement utilise des courants forts, en réseaux et télécommunications, etc.

Une analyse des types de systèmes conduit à la classification suivante [CALVEZ-90]:

- les *systèmes typiquement électroniques* qui impliquent essentiellement le développement de matériel. Ils ne font donc pas directement partie du domaine de l'activité de co-design même si l'apparition de composants matériels programmables, des langages de description de matériel (VHDL, Verilog) et des outils de synthèse tend à rendre de plus en plus floue la frontière entre matériel et logiciel.

- Les *systèmes interactifs* concernés par l'exploitation d'interfaces pour le dialogue homme-machine. Ces systèmes sont constitués principalement de logiciels (système d'information exploitant une base de données par exemple). Ils ne font donc pas partie du champs d'application de l'activité de co-design.

- Les *systèmes de communications* dominés par le transfert d'informations. Ces systèmes reçoivent, transforment et émettent des flots de messages. Ils sont représentatifs des systèmes pour lesquels les réalisations autrefois à dominante matérielle ont progressivement évoluées vers un partage matériel/logiciel avec un accroissement de la partie logicielle car les microprocesseurs sont devenus de plus en plus puissants. Mais, une partie matérielle est pour ce type de systèmes indispensable pour respecter les débits élevés des protocoles de communications (réseau ATM par exemple).
- Les *systèmes de traitement* concernés principalement par les traitements de toutes formes d'informations: signal, image, parole... Ces systèmes sont caractérisés par des techniques de conception orientées flot de données. Tout comme pour les systèmes de communications, les réalisations de ces systèmes étaient autrefois surtout matérielles et basées sur une partie opérative (ordonnancement ASAP ou ALAP en fonction du compromis surface de silicium/performance) et une partie contrôle. Puis peu à peu, l'arrivée de microprocesseurs spécifiques (DSP) de plus en plus performants a donné une part de plus en plus importante au logiciel. Aujourd'hui, le développement de ces systèmes est bien maîtrisé et repose sur l'utilisation de méthodes et outils efficaces (COSSAP, SPW, DSP WorkStation,...) mais limités uniquement à leur domaine d'application (approche algorithmique, CFG/DFG, synthèse haut niveau).
- Les *systèmes de contrôle/commande* dominés par des problèmes de suivi et de commande pour des applications incluant des procédés physiques en tous genres à piloter. Ils font partie d'un ensemble (système+environnement) composé d'actionneurs qu'ils contrôlent en fonction des événements perçus par des capteurs.

Les travaux présentés dans ce manuscrit concernent surtout les systèmes de contrôle/ commande, les systèmes de communications et partiellement les systèmes de traitement. Ces systèmes dit dédiés sont généralement conçus pour répondre à un besoin spécifique et entrent dans la catégorie des systèmes électroniques embarqués et temps-réel (Real-Time Embedded Systems).

Un *système temps-réel* réalise ses activités en respectant des contraintes de temps de nature externe (fréquence d'événement, débit de sortie) ou interne (temps de réaction). Lorsque les contraintes de temps sont assez faibles, la réalisation est logicielle avec ou sans l'emploi d'un exécutif temps-réel (utilisation du mécanisme de gestion des interruptions du processeur). Dans le cas contraire, le concepteur doit trouver le meilleur compromis entre une implantation tout en matériel aux performances dynamiques et coût élevés et une implantation complètement logicielle aux performances et coût faibles, ce qui en co-design correspond à la problématique du *partitionnement matériel/logiciel*.

Les *systèmes électroniques embarqués* sont caractérisés par de fortes contraintes d'intégration avec leur environnement: taille physique réduite, consommation faible, résistance aux chocs et aux variations climatologiques, sûreté de fonctionnement, etc. Aujourd'hui, ces systèmes intègrent systématiquement du matériel et du logiciel et sont de plus en plus présents dans une grande variété de produits. Lorsqu'ils sont produits en grande série, ils doivent présenter un coût de fabrication minimum. Le temps de mise sur le marché (time-to-market), qui est un facteur clef pour le succès d'un produit, doit également être le plus court possible. Enfin, tout comme les systèmes temps-réel, ils doivent généralement respecter des contraintes dynamiques (temps de réaction faible, débit élevé). Ils constituent donc la cible privilégiée de l'activité de co-design: "*Creating an embedded computer system which meets its*

performance, cost and design time goals is a hardware/software co-design problem. The design of the hardware and software components influence each other" [WOLF-94].

1.1.3 La nécessité d'un nouveau cycle de développement

Sachant que l'implantation matérielle permet d'obtenir des performances dynamiques plus élevées qu'une implantation logicielle mais à un coût plus élevé, depuis l'avènement de la technologie VLSI et des premiers microprocesseurs, les équipes de conception recherchent le meilleur partitionnement matériel/logiciel permettant de respecter les contraintes de coûts et de performances imposées. A l'époque où les possibilités d'implantation matérielle et logicielle étaient limitées, l'expérience seule des concepteurs pouvait suffire à les guider vers une solution proche de l'optimum.

Les progrès réalisés aussi bien dans le domaine du génie logiciel (environnement de programmation, langages de haut niveau, méthodes et langages orientés objet) que dans le domaine du génie matériel (technologie VLSI, outils de synthèse logique, outils de synthèse de haut niveau) permettent de réaliser des systèmes de plus en plus complexes. Jusqu'à encore peu de temps, la réalisation des systèmes électroniques embarqués de grande complexité reposait sur une approche d'*ingénierie système* caractérisée par une séparation nette de la conception et réalisation des parties logicielles et matérielles. Les ingénieurs système avaient en effet la responsabilité de concevoir l'architecture du système et d'identifier et spécifier les parties matérielles et logicielles dont les réalisations étaient ensuite à la charge d'équipes de conception distinctes ou étaient sous-traitées. Cette approche a entraîné un certain cloisonnement et une absence de dialogue entre les concepteurs des deux catégories professionnelles de culture différente: les concepteurs d'architectures matérielles et les informaticiens. Ce manque de dialogue conduit ensuite à des difficultés importantes découvertes tardivement durant l'intégration du logiciel sur le matériel. Malgré la malléabilité du logiciel, les problèmes rencontrés lors de l'intégration se traduisent généralement par un surcoût financier et temporel important.

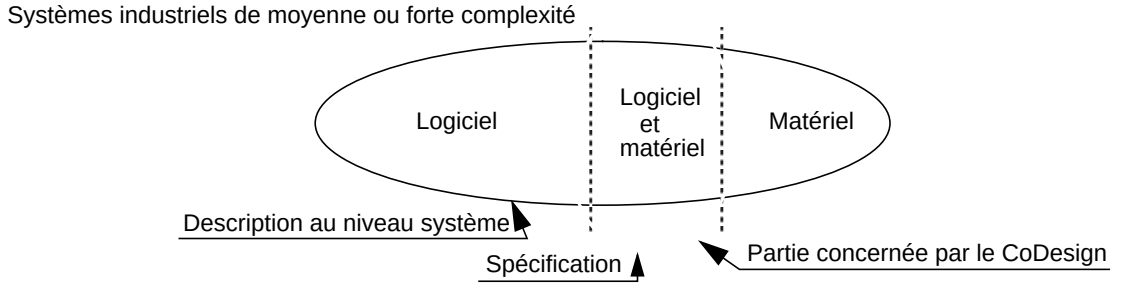
L'approche d'ingénierie système est indispensable pour la conception des systèmes hétérogènes complexes et/ou caractérisés par une sous-traitance importante (avionique, automobile par exemple). Mais l'utilisation d'un tel cycle de développement pour les systèmes électroniques embarqués s'est révélé très pénalisante et inadaptée pour répondre à des contraintes de qualité, de coûts et délais de plus en plus sévères.

Tout d'abord, pour faire face aux problèmes de l'intégration tardive des parties matérielles et logicielles, les concepteurs ont exploité la *simulation conjointe* de la partie logicielle et la partie matérielle du système (co-simulation). Ils pensaient en effet que la possibilité de détecter les erreurs de spécification et de conception en simulant le système relativement tôt dans le cycle de développement, suffisait à obtenir une conception sans erreur et de qualité.

Ils se sont très vite aperçus que leurs besoins allaient bien au delà de la co-simulation et que pour réduire la durée des développements et accroître la complexité des systèmes et la qualité de conception, il fallait utiliser une *méthodologie de conception complète* et les outils supports associés permettant de développer conjointement la partie matérielle et logicielle tout au long du cycle de développement. Le terme méthodologie représente ici un ensemble structuré et cohérent de modèles, de règles bien définies (méthodes), guides (démarche) et outils permettant de déduire la manière de résoudre un problème [CALVEZ-90]. Dans le court terme, les efforts concernent principalement le développement d'une telle méthodologie. Par

exemple, le projet américain RASSP développe actuellement une méthodologie de co-design en se basant sur l'utilisation d'un ensemble d'outils commerciaux et universitaires pour couvrir toutes les phases de développement d'un projet et en utilisant principalement le langage VHDL comme format d'échange entre les différents outils [SCHAMING-96].

Naturellement, une méthodologie de co-design apparaît comme un enrichissement d'une méthodologie de conception système existante. En effet, il est plus judicieux d'adapter à la problématique du co-design une méthodologie de conception système éprouvée que de redéfinir entièrement une nouvelle méthodologie. Une méthodologie de conception système est habituellement organisée selon quatre étapes principales: l'élaboration des spécifications, la conception fonctionnelle ou préliminaire, la conception architecturale ou détaillée et la réalisation. L'enrichissement d'une méthodologie système pour l'activité de co-design concerne principalement l'étape de conception architecturale. A ce stade, les concepteurs effectuent facilement en tenant compte de différents critères (répartition géographique, performance, coût, sûreté de fonctionnement, flexibilité, testabilité, etc.) une répartition de la solution fonctionnelle en une partie purement logicielle, une partie purement matérielle et une partie plus délicate où une variation est possible entre le matériel et le logiciel. Notons que dans les systèmes actuels, la part du logiciel est croissante. Des chiffres de plus de 70 % sont cités, ce pourcentage étant bien entendu très dépendant du problème traité.



-Figure 1.1- Exemple de sous-système concerné par le co-design.

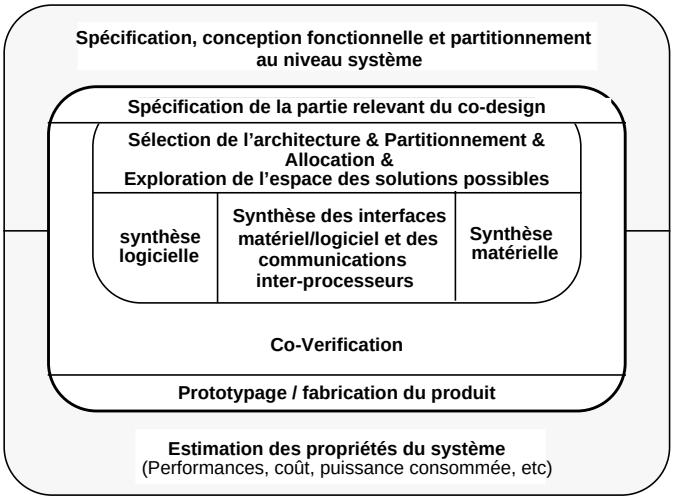
La description fonctionnelle détaillée de la partie spécifique ou sous-système sert de point de départ de l'activité de co-design.

L'activité de co-design concerne également les systèmes de complexité moyenne à faible pour lesquels une approche système n'est pas nécessaire. Aujourd'hui, le concepteur bénéficie d'une grande diversité de choix pour la réalisation d'un système. L'offre en matière de microprocesseurs (MPU, ASSP, ASIP, DSP), de langages de programmation et compilateurs associés est vaste. Parallèlement, on peut aussi implanter de plus en plus de fonctions dans le silicium (ASICs, FPGA, system-on-a-chip). Même pour les petits systèmes, cette diversité rend de plus en plus difficile la conception intuitive basée sur l'expérience du concepteur.

1.1.4 Méthodologie pour le co-design

Dans le rapport [MCC/OMI-96], le processus de co-design est représenté par la figure 1.2. L'activité de co-design (partie en blanc de la figure 1.2) est intégrée dans la démarche d'ingénierie système. Les spécifications de la partie relevant du co-design résultent de l'approche système qui consiste en la spécification, la conception fonctionnelle et le partitionnement au niveau système. A partir de ces spécifications, le concepteur doit définir l'architecture matérielle constituée de processeurs matériels (ASICs, FPGA, composants

standards) et de processeurs logiciels (MPU, ASIP, DSP) et l'allocation des éléments de la solution fonctionnelle sur cette architecture: c'est la problématique communément appelée *partitionnement matériel/logiciel*. Il est bon de noter pour éviter toute confusion qu'un sens différent est parfois donné au terme allocation. Par exemple dans [GAJSKI-95], le terme allocation correspond à la sélection d'une architecture et le terme partitionnement à la répartition des éléments de la description fonctionnelle sur l'architecture: "Allocation is the problem of finding a set of system components to implement the system's functions. Given a functional specification and an allocation of system components, we need to partition the specification and assign each part to one of the allocated components".



-Figure 1.2- Représentation du processus de co-design.

Un balayage rapide de l'espace des solutions conduit à sélectionner la solution la plus appropriée. Ce parcours de l'espace des solutions possibles peut être automatique ou interactif et repose toujours sur une estimation des propriétés résultantes du partitionnement choisi. Les propriétés retenues sont généralement des contraintes de coût et de performances statiques (taille, puissance consommée) ou dynamiques (temps de réponse, charge du processeur).

Une fois le partitionnement matériel/logiciel terminé, les descriptions fonctionnelles sont transformées en code machine pour une implantation en logiciel et en un ensemble de portes logiques et de bistables pour une implantation en matériel. C'est le rôle de la phase de co-synthèse qui concerne la synthèse des parties matérielles et logicielles et la synthèse des interfaces matériel/logiciel. Avant et après synthèse, une vérification fonctionnelle détaillée (co-verification) est effectuée généralement par une technique de co-simulation. Un prototype ou directement le produit est le résultat en sortie de l'activité de co-design et s'intègre alors dans le système global.

Plusieurs aspects de la conception conjointe des systèmes matériels et logiciels sont actuellement étudiés:

- la *co-spécification* qui concerne les étapes de spécification et de conception fonctionnelle et pour lesquelles beaucoup de formalismes existent [GAJSKI-95] [GAJSKI-96] [MICHELI-95] [CALVEZ-96a] [CALVEZ-96d]. Ces formalismes (modèles et langages) servent à décrire principalement l'aspect fonctionnel. Les spécifications non-fonctionnelles telles que les performances, la fiabilité, la sécurité, le

coût, sont beaucoup plus délicates à formaliser et pourtant elles sont essentielles pour décider d'un partitionnement approprié et justifié. A ce stade, les différences de culture et de jargon entre le monde du génie logiciel et du génie matériel ont nécessité des efforts d'uniformisation de la terminologie [RTWG-95].

- le *partitionnement matériel/logiciel* qui concerne l'étape de conception architecturale. Différentes techniques de partitionnement sont présentées dans le chapitre 2. Dans le cas général, l'architecture matérielle peut être quelconque, c'est-à-dire hétérogène et distribuée. Plus l'architecture est particularisée, plus les concepts et les outils associés sont actuellement avancés. Souvent, l'architecture générique cible est choisie. Dans un ensemble de travaux, par exemple [GUPTA-93], [THOMAS-93], on retrouve très souvent l'emploi d'une architecture du type maître/esclave utilisant un microprocesseur conventionnel ou un microcontrôleur comme maître associé à un ou plusieurs ASICs comme esclaves. Il résulte que la solution est fortement dépendante de l'architecture et des composants standards utilisés pour la réalisation [WOLF-94].
- la *co-simulation* qui concerne l'étape de conception architecturale pour la co-simulation fonctionnelle et l'étape de réalisation pour la co-vérification détaillée. La co-simulation est une simulation de la description mixte matériel-logiciel résultant d'un partitionnement: "Hardware-software cosimulation is a means of verifying the functionality of mixed hardware-software descriptions" [THOMAS-93]. La difficulté de la co-simulation est liée à la différence de modèles et/ou de niveaux d'abstraction des représentations du logiciel et du matériel. Les solutions de co-simulation reposent sur l'emploi d'un langage de représentation unique et exécutable ou pour une simulation hétérogène sur l'utilisation d'un mécanisme de communications entre différents simulateurs. Différentes techniques de co-simulation sont présentées dans le chapitre 2.
- la *co-synthèse* (étape de réalisation) qui se charge de transformer les descriptions fonctionnelles en descriptions directement implantables sur les processeurs matériels et logiciels de l'architecture cible. Considérés séparément, le développement des constituants logiciels (debuggeur symbolique, générateur d'IHM, profiler, approche objet, exécutif temps-réel) et matériels (synthèse logique et haut niveau) sont aujourd'hui bien maîtrisés. Les outils de synthèse logique sont très robustes. Les outils de synthèse comportementale même s'ils manquent encore un peu de maturité, sont aussi maîtrisés par les compagnies EDA/ESDA. Pour s'ouvrir de nouveaux marchés, celles-ci s'intéressent de plus en plus à la conception au niveau système. La synthèse logicielle souffre cependant de deux problèmes: le manque d'outil de compilation efficace pour des microprocesseurs spécifiques, et le manque d'outil de génération de code exécutable pour une description multi-tâches (ordonnancement des tâches compris). Beaucoup de travaux de co-synthèse concernent aussi la génération des interfaces matériel/logiciel [GUPTA-93] [YEN-95] [DAVEAU-95] [BORIELLO-96] [FISCHER-96] [MULLER-96].

1.2 MOTIVATIONS DE L'EQUIPE

La plupart des techniques de partitionnement automatique se limitent à une architecture matérielle constituée d'un microprocesseur, d'un ensemble de composants matériels programmables et éventuellement d'une mémoire commune. Cette architecture est intéressante pour son aspect générique mais elle correspond de moins en moins à la réalité industrielle. La

communauté du co-design a donc pris peu à peu conscience que le partitionnement automatique plus ou moins rejeté par le milieu industriel [TUCK-97], n'est efficace que sur une classe limitée de systèmes et que les recherches doivent s'orienter vers des solutions plutôt de nature semi-automatique guidées par le concepteur et basées sur des outils d'estimation rapides.

Le travail présenté ici est une contribution à cette nouvelle orientation. Comme la plupart des techniques de partitionnement reposent sur des estimateurs statiques et que de nombreux travaux ont déjà été effectués dans ce domaine, nous avons opté pour une technique différente et complémentaire: *l'évaluation des performances dynamiques d'un système par co-simulation*.

La technique de partitionnement proposée repose sur la démarche itérative suivante: compte tenu des contraintes diverses qui lui sont imposées, le concepteur définit une première architecture matérielle et l'allocation des fonctions sur les processeurs matériels et logiciels de cette architecture. La méthodologie MCSE fournit la démarche à suivre pour cette première recherche de solution. Puis, l'évaluation des performances dynamiques du modèle du système résultant donne en retour des estimations de performances tel que le temps de latence de messages, le débit sur un bus ou encore le taux d'occupation d'une ressource. L'analyse de ces performances permet de vérifier le respect des contraintes de performances et d'identifier les ressources critiques. Une estimation des propriétés de la solution est alors possible et sert à guider le concepteur qui peut alors modifier les ressources jugées critiques et l'implantation des fonctions et réévaluer le modèle de sa solution.

Pour l'évaluation des performances dynamiques d'un système, nous avons retenu une *technique de co-simulation macroscopique et non interprétée* basée sur la transcription en VHDL d'un modèle de performance et sa simulation à l'aide d'un simulateur VHDL du commerce. Le terme macroscopique signifie que le système n'a pas besoin d'être entièrement détaillé et le terme non-interprété indique que seul les temps des opérations et les dépendances temporelles sont prises en compte.

Cette approche se distingue donc par:

- l'utilisation de la simulation au lieu de l'évaluation analytique pour extraire les performances dynamiques (temps de réponse, taux d'occupation d'une ressource) d'un système. Contrairement à l'évaluation analytique, la simulation a l'avantage de pouvoir analyser tout type de systèmes. Même si celle-ci souffre encore de temps de simulation relativement longs, l'élévation du niveau d'abstraction des modèles et l'accroissement régulier de la puissance de calcul des ordinateurs tend à minimiser cet inconvénient. De plus, contrairement aux estimateurs statiques utilisés dans le partitionnement automatique, notre approche permet de faire une analyse assez fine des comportements et des performances très tôt dans le cycle de développement et éventuellement dès l'étape de spécification (étude de faisabilité du projet). Durant l'étape de conception fonctionnelle, elle permet aussi de dimensionner les éléments internes d'un modèle (taille d'un port de communication par exemple); ce qui en général ne sera pas possible ou très difficile avec l'utilisation d'un estimateur statique (théorie des réseaux de files d'attente et des réseaux de Petri stochastiques).
- une modélisation du système à un niveau d'abstraction plus élevé que celui habituellement utilisé pour la co-simulation qui généralement nécessite une description détaillée de la solution.

Dans le projet RASSP, la méthodologie de co-design mise en oeuvre repose sur l'utilisation successive de plusieurs outils et modèles. Bien qu'elle soit hiérarchique et incrémentale, nous pensons que cette approche basée sur un multi-formalisme entraîne un morcellement du processus de conception qui a au moins deux conséquences:

- le passage d'une phase de conception à l'autre nécessite une transcription de modèle qui peut entraîner des erreurs ou des déformations,

- il n'y a pas de traçabilité simple des informations et donc il est plus difficile de vérifier l'adéquation de la solution obtenue par rapport aux spécifications du système.

Notre approche méthodologique basée sur la méthodologie MCSE [CALVEZ-90] pour l'approche système est aussi hiérarchique et incrémentale et permet donc de faire face à la complexité des systèmes en procédant par décomposition. Pour la phase de spécification, un multi-formalisme s'impose face à la diversité des informations concernées qui sont de nature fonctionnelles et non fonctionnelles. La phase de spécification [CALVEZ-96a] concerne en effet l'analyse des entités de l'environnement du système (modélisation des informations échangées, des activités et du comportement), la description du comportement du système vis à vis des entités de l'environnement (automate à états finis, SpecChart, etc.) et la définition des spécifications opératoires (précision des calculs par exemple) et technologiques (contraintes de répartition, de temps, d'interfaces physiques, de sûreté de fonctionnement, etc.). Mais notre approche méthodologique se distingue très nettement par l'utilisation d'un modèle unique à partir de la phase de conception fonctionnelle. Le modèle de performance utilisé est en effet la composition du modèle fonctionnel et du modèle exécutif préconisés par la méthodologie MCSE et d'une vue comportementale qui permet de décrire le comportement de chaque fonction de la vue fonctionnelle sous forme d'une composition d'activités. L'unicité du modèle facilite la transition d'une phase de conception à l'autre: le modèle d'une phase de conception donnée s'obtient par raffinement et enrichissement du modèle de l'étape précédente. La continuité du modèle facilite aussi la conception sans erreur (pas de déformation ou perte d'information liées à une transcription de modèle) et améliore la traçabilité.

1.3 CONTEXTE DE CETTE THESE

Cette thèse effectuée au sein de l'équipe MCSE de l'IRESTE a été financée par le CNET dans le cadre de la CTI 9 (contrat n^{o} 941B CNET France-Télécom).

Le programme proposé des travaux était le suivant:

1- Montrer l'adéquation de la méthodologie MCSE pour la spécification, la conception et l'évaluation des performances d'un système complexe. Ce travail a été effectué sur la base d'un exemple d'application et en collaboration avec le CCETT de Rennes (G. Babonneau) et l'équipe de J.M Bergé du CNET de Meylan. L'exemple d'application fourni par le CCETT de Rennes est le serveur vidéo temps-réel décrit dans le chapitre 7.

2- Obtenir un modèle VHDL simulable pour l'extraction de performances. Pour cela, nous avons défini les règles de transcription du modèle MCSE en une description VHDL. Ces règles de transcription ont été validées à l'aide de deux exemples: le serveur vidéo et un système de communication réparti. Puis elles ont été implantées dans un générateur de code.

3- Développer une technique de génération de code automatique (modèle non-interprété pour l'évaluation des performances) ou semi-automatique (modèle interprété pour la vérification fonctionnelle et la synthèse) permettant de cibler sur différents langages.

4- Développer une démarche de partitionnement interactif basée sur une évaluation rapide des performances dynamiques du partitionnement choisi.

Ces travaux s'intègrent au développement d'un support informatisé pour la méthodologie MCSE. Ce développement constitue actuellement l'activité principale de l'équipe MCSE. En amont des travaux effectués lors de cette thèse qui portent sur la modélisation des performances et la génération de code pour la simulation, l'équipe développe des outils graphiques de description (saisie des structures fonctionnelle et exécutive, saisie du comportement des fonctions, configuration et allocation). En aval, le travail concerne la génération de code exécutable [CALVEZ-93c], la synthèse des interfaces logiciel/matériel [CALVEZ-94] [CALVEZ-96c], l'analyse des performances en différé (analyse de traces obtenues par simulation) ou en temps-réel (monitoring) [CALVEZ-95b] [CALVEZ-95c] [CALVEZ-98a].

1.4 CONTRIBUTIONS

Le modèle de performance utilisé pour la co-simulation est le résultat d'une réflexion de l'équipe sur la modélisation des performances qui a débuté en 1992 [CALVEZ-93b]. Le travail effectué lors de cette thèse a débouché sur:

- l'enrichissement et la validation des concepts du modèle de performance.
- la définition d'une méthode d'évaluation des performances dynamiques d'un système basée sur une transcription en VHDL du modèle de performance et l'utilisation d'un simulateur VHDL du commerce.

Concernant l'aspect développement d'outils, ce travail a conduit à la réalisation d'un générateur de générateur de code ou méta-générateur et d'un générateur de code VHDL comportemental pour l'évaluation des performances et la vérification fonctionnelle. Le principe de génération de code développé permet de transcrire facilement le modèle MCSE (ou tout autre langage source) vers d'autres langages cibles. Le travail effectué sur la génération de code a aussi amené l'équipe MCSE à revoir sa stratégie de développement des outils comme support pour la méthodologie MCSE. En effet, l'expérience de génération de code effectuée lors des travaux de cette thèse a permis d'appréhender les concepts de générateurs d'analyseurs syntaxiques et de méta-structures sur lesquels repose entièrement la nouvelle "philosophie" de développement des outils MCSE. Ces deux concepts et le nouveau principe de développement de la plate-forme d'outils pour MCSE sont détaillés dans le chapitre 5 sur le méta-générateur.

1.5 PLAN DE LA THESE

Ce manuscrit de thèse est découpé en 6 chapitres principaux numérotés de 2 à 7. Dans le chapitre 2, nous présentons la méthodologie de co-design, le principe de partitionnement et la technique de co-simulation préconisés par l'équipe MCSE. Ce chapitre contient également un état de l'art sur les techniques de partitionnement matériel/logiciel, les techniques de co-simulation et la modélisation des performances. Le modèle de performance de MCSE est approprié pour le co-design car il représente simultanément la partie matérielle et la partie logicielle d'un système, est décrit dans le chapitre 3. Transcrit en VHDL en appliquant les règles décrites dans le chapitre 4, ce modèle de performance permet de faire une co-simulation

et d'extraire les performances dynamiques d'un système. La technique de génération de code que nous avons utilisée pour transcrire le modèle textuel du modèle de performance en VHDL est particulière. Nous avons en effet développé un générateur de générateur de code ou méta-générateur nommé MetaGen et qui fait l'objet du chapitre 5. Avec cet outil, un générateur particulier s'obtient par l'écriture d'un script qui définit les manipulations à effectuer sur les structures de données pour mener à bien la transcription. Le chapitre 6 décrit le générateur de VHDL comportemental. Le chapitre 7 valide les concepts du modèle de performance et les méthodes de co-simulation et de partitionnement développées en présentant deux exemples d'application: le serveur vidéo temps-réel fourni par le CCETT de Rennes et un système de communication distribué basé sur l'interconnexion d'un ensemble de cartes identiques avec un bus série du type anneau à jeton. En guise de conclusion, le chapitre 8 rappelle les points essentiels du travail présenté et donne un ensemble de perspectives possibles.