Table des matières

Table des figures

Liste des tableaux

INTRODUCTION GENERALE

Les concepteurs de systèmes modernes doivent gérer des projets associant plusieurs disciplines et plusieurs technologies. En particulier, depuis des années, les systèmes électroniques ne sont plus conçus isolément : ils intègrent des préoccupations de systèmes et de microsystèmes, dans divers secteurs d'applications scientifiques et industrielles. En raison de la complexité et de l'hétérogénéité de ces systèmes, il est nécessaire de mettre en place des méthodes et des outils facilitant l'intégration de solutions analogiques, numériques, mixtes, matérielles et logicielles. Ce problème et le besoin d'optimiser le processus de conception pour réduire le "time to market", ont conduit au développement de techniques telles que la modélisation et la validation à haut-niveau, la modélisation fonctionnelle, la réutilisation et la génération de modules de propriété intellectuelle (IP)… Ces composantes doivent être considérées dès les premières étapes de la conception. Nos propositions auront à tenir compte de nombreuses exigences, et donc à s'appuyer sur des langages et des procédures standardisés.

Les méthodes actuelles doivent aussi prendre en compte la conception coopérative et la réutilisation des acquis. Elles doivent donc être basées sur des procédures et des langages normalisés ou standardisés facilitant les échanges. Le but est alors de proposer des outils généraux et des méthodes capables de soutenir le travail coopératif entre divers participants d'un projet de conception. Dans une première étape, les méthodes utilisées doivent s'appuyer sur des modèles de haut-niveau, fonctionnels, exécutables, que nous appelons ici des Prototypes Virtuels. Ces prototypes permettent de vérifier, par simulation, leur conformité fondamentale avec le cahier des charges, avant d'entamer les démarches de matérialisation et de réalisation technologique.

Les motivations techniques, commerciales et l'influence de la concurrence créent des intérêts forts chez les industriels, dans le développement, l'utilisation et l'optimisation de technologies permettant d'arriver aussi loin que possible dans les extensions de la conception système [Ham01]. Le développement des prototypes virtuels et la nécessité de valider leur cohérence avec les spécifications, demande l'appui d'outils informatiques permettant de modéliser, dès les niveaux d'abstraction les plus élevés, les aspects suivants:

· Les interactions du système avec son environnement opérationnel.

· L'évaluation et la définition des entrées / sorties.

· L'étude et le développement des modèles comportementaux des constituants.

· La représentation graphique des relations fonctionnelles proposées.

· L'exploration architecturale.

· L'estimation des performances et les états critiques de fonctionnement.

Les exigences générales se situent au niveau de la gestion de ces aspects et de la recherche de techniques permettant de réduire le temps de développement des produits et d'accroître les performances de la conception sur des points essentiels comme la robustesse, la sûreté de fonctionnement et la vérification. Plusieurs axes de réflexion et de développement devront être explorés :

La réutilisation, autant que possible, des acquis et des modèles précédemment validés.

Dans ce contexte, l'extension de l'utilisation des outils de CAO pour l'électronique vers d'autres domaines peut représenter une source de progrès important.

L'utilisation de techniques telles que le co-design, la Co-Simulation, la création et la gestion de modules de propriété intellectuelle ont amené le monde de l'électronique numérique au sommet de ce qu'il est convenu d'appeler l'EDA (Electronic Design Automation) .

La partie analogique qui pourtant reste en retard par rapport à ces techniques à cause de la complexité du problème et de la difficulté à mettre en place une vraie politique de standardisation des méthodes, et des langages **:** Il n'est pas facile d'y établir une base commune de ressources informatiques et méthodologiques, à l'image de la modélisation VHDL ou de la gestion des IPs dans le domaine de l'électronique numérique.

La plus grande difficulté de l'approche est de généraliser les méthodes et de considérer la conception système comme un tout. De cette manière, en partant des spécifications ou des cahiers des charges, les concepteurs pourraient établir et valider des modèles fonctionnels et proposer des solutions architecturales. Une étape essentielle de cette problématique est la traduction ou l'interprétation des spécifications sous forme de modèles fonctionnels.

En plus, l'approche consiste à utiliser un seul langage pour la spécification complète du fonctionnement du système. Cela suppose qu'il possède une sémantique consistante et assez riche pour qu'il puisse supporter l'hétérogénéité des modules constituant le modèle entier.

L'inconvénient majeur de cette approche réside dans la difficulté de trouver un langage capable de couvrir la plupart des domaines impliqués dans les systèmes actuels. Cela amène à la

définition de nouveaux langages et par suite à des temps d'apprentissage important et à la construction de nouvelles bibliothèques.

Dans ce contexte, nous trouvons pas mal de langages et outils : pour la simulation analogique, nous avons comme exemple PSPICE. Pour la simulation mixte analogique, numérique nous trouvons principalement SABER, Verilog-AMS et VHDL-AMS. Une bonne partie des outils destinés à la conception des systèmes mixte comme Spec C et System C. Dans une autre catégorie, nous citons des outils qui abordent le problème de simulation à un niveau plus élevé que la partition logicielle /matérielle. Parmi eux, nous trouvons POLIS qui permet de créer une méthodologie formelle unifiée. Et nous terminons par des approches généralistes; il s'agit des plus célèbres outils de modélisation qui est MATLAB produit par The MathWorks. Il permet de réaliser une modélisation détaillée des algorithmes de contrôle des systèmes dans des domaines multiples.

Comme nous l'annoncions, le grand défit consiste maintenant à offrir un environnement qui ne soit pas réservé exclusivement à des experts en un seul domaine. Les spécialistes du traitement de données, habitués à manipuler le langage C/C++ ou Simulink trouveraient un intérêt certain à pouvoir accéder directement au matériel. Des bibliothèques de circuits logiques dédies prêts à être utilisés existent déjà. Des synthétiseurs de circuits logiques, sur base de quelques paramètres, existent également et sont développés par des firmes spécialisées.

Notre objectif consiste à identifier une méthode efficace de validation fonctionnelle de système embarqué dans son environnement unifié. Il s'agit de mettre en place un environnement de Co-Simulation hétérogène. Nous présentons le travail en deux parties : la première présente la simulation MATLAB/VHDL tandis que la deuxième consiste à envisager la simulation MATLAB/FPGAs. Afin de tester les plateformes mises en place, nous avons considéré l'exemple LMS-based adaptive equalization (Synthesizable RTL implementation using M-code Block). Ce modèle montre qu'a T/2 l'adaptive FSE (Fractionally Spaced Equalizer) fonctionnant sur un point d'émission 16-QAM avec le bruit et filtrer présenté dans le modèle.

Dans le chapitre 1, nous commençons par présenter les systèmes enfouis (embarqués), nous réalisons un bilan des outils et des méthodes existantes pour la conception du système. Les outils de CAO et les méthodes de conception sont en constante évolution et leur développement reflète l'état de l'art des pratiques. Au niveau de la conception des systèmes électroniques, une fois que les différentes approches et méthodes dépassent le niveau purement théorique, et que les

langages sont adoptés, des processus de simulation et de Co-Simulation sont engagés. Au cours de cette étude, nous sommes restés attentifs aux nouveautés provenant des fournisseurs commerciaux, et aux innovations proposées par le monde académique et scientifique et nous terminons par présenter l'Approche classiques des validations logicielles.

Le chapitre 2 porte sur la mise en place des langages et des outils de notre plateforme de conception système à haut niveau. Nous détaillons les apports de chacune d'elles pour assurer la Co-Simulation.

Dans chapitre 3, nous abordons la Co-Simulation MATLAB/VHDL, nous expliquons les différentes interactions entre les logiciels formant la plateforme et nous présentons un exemple pour mieux éclaircir cette approche.

Le chapitre 4 est consacré à la Co-Simulation MATLAB/FPGAs. Nous expliquons les différentes interactions entre les logiciels formant la plateforme. Le but de cette dernière partie est de traiter complètement un cas d'application en utilisant cette approche afin d'illustrer, de la meilleure façon possible, son fonctionnement.

Enfin, nous terminons par une conclusion générale, dans laquelle nous présentons les avantages des environnements proposés dans ce mémoire.

LANGAGE ET ENVIRONNEMENT DE SIMULATION : ETAT DE L'ART

Chapitre I: LANGAGE ET ENVIRONNEMENT DE SIMULATION : ETAT DE L'ART

I.1 Introduction

Les outils de CAO sont devenus des appuis incontournables pour les ingénieurs et les scientifiques, au moment d'exécuter tous types de projets, particulièrement ceux dont la complexité et le temps de développement sont importants. Nous avons conduit, pendant toute la durée du mastère, une veille technologique des outils disponibles sur le marché et des techniques émergentes des laboratoires de recherche et autres organismes universitaires. Malheureusement, nous n'avons pas eu la possibilité de tout tester et nos analyses ne pourront qu'être fortement influencées par les standards et les tendances d'utilisation au niveau de l'industrie et de la recherche.

Pour ce chapitre d'analyse, nous avons commencé par présenter les systèmes enfouis. Ensuite nous avons classé les outils selon trois champs d'application : la conception électronique en général, la conception de haut-niveau des systèmes à base d'électronique et la gestion de l'information des projets de conception système. Cela permet de prévoir les langages et les outils que nous allons utiliser pour attendre toujours notre objectif qui est en premier lieu réduire le temps de conception et utiliser un seul langage pour toute la simulation.

I.2 Les systèmes enfouis (embarqués)

Afin de mieux situer le cadre des travaux présentés dans ce mémoire, il convient de définir plus précisément ce que sont les systèmes enfouis (également appelés systèmes embarqués). S'il n'existe pas de définition "officielle", il est communément admis que, d'une manière très générale, un système enfoui peut être vu comme un système électronique dont le fonctionnement repose sur un microprocesseur (avec ou sans système d'exploitation), mais qui n'est pas un ordinateur (au sens Personale Computer) [Aas 04]. Pour être plus précis, nous pouvons dire qu'un système enfoui est un système électronique dédié à une ou à un ensemble d'applications prédéfinies et dont la mise à jour ne peut être que très limitée (par exemple chargement de nouveaux logiciels ou reprogrammation de matériel reconfigurables). De plus, un système enfoui peut être éventuellement multi-domaines, c'est à dire analogique et numérique : le domaine

analogique comprenant par exemple la partie RF d'un téléphone mobile alors que le domaine numérique comprenant le traitement numérique du signal et l'interface utilisateur.

La partie numérique est souvent hétérogène, puisqu'elle fait appel à des composantes logicielles et matérielles. Nous pouvons distinguer dans cette partie les systèmes fixes et les systèmes mobiles. Pour ces derniers, la maîtrise de la consommation est un facteur essentiel à leur réussite commerciale ; c'est pour cela que la recherche sur cet aspect est très dynamique. Enfin, la puissance du calcul des systèmes enfouis peut être très variable.

Les systèmes les moins complexes intègrent de simples microprocesseurs et microcontrôleurs (de nombreux produits font appel aux 8051, 68HC05, ou autres dérivés de 68000) alors que les systèmes plus évolués font appel aux dernières générations de microprocesseurs et de DSP (Digital Signal Processor), avec aussi des accélérateurs matériels dédiés.

Comme indiqué plus haut, les systèmes enfouis sont généralement implantés sous la forme de composantes logicielles et matérielles. Les composantes logicielles peuvent être implantées sur des processeurs à usage général (GPP) ainsi que sur des processeurs à usage spécifique (DSP, ASIP). Les composantes matérielles sont implantées soit sur des composants dédiés (ASIC), soit sur des composants reprogrammables (FPGA). Il est également important de souligner les cas des systèmes sur-une-puce (SoC), qui intègrent sur une seule puce de silicium les composantes logicielles et matérielles. En effet, bon nombre de processeurs (ou cœur de processeur) sont disponibles sous la forme de "propriétés intellectuelles" (IP, spécification, à un niveau d'abstraction donné, d'un composant matériel ou logiciel, et destinée à être réutilisée par une tierce partie) synthétisables, ce qui permet de les intégrer à coté des parties matérielles, soit sur ASIC soit sur FPGA de dernière génération.

Ces derniers intègrent un nombre considérable de portes (1,5 millions pour le modèle Altera EP20K1500E en technologie 0.13 µM), des blocs de traitement du signal numérique (modules matériels de type multiplieur, additionneur, soustracteur, accumulateur, registres pipelines) comme la famille Stratix chez Altera. On peut enfin citer les FPGA intégrant d'origines des processeurs telle la famille VirteX-II Pro (pouvant disposer de 0 à 4 processeurs de type PowerPc405).

D'où la nécessité des langages et des outils pour assurer la simulation ou la Co-Simulation des systèmes hétérogènes.

I.3 Simulateur analogique

Nous commençons ce tour d'horizon des outils et langages d'aide à la conception des systèmes à base d'électronique par l'un des outils pionniers de la simulation purement analogique, qui, au long des années, est devenu un standard industriel et académique : **PSPICE**. Suite à l'apparition de Spice 1 en 1972 [Mal02] à l'Université de Berkeley, Pspice [Cds03a] est devenu le standard de fait pour la simulation électronique analogique. Sa version, PSpice 9.2.3 est un des modules fonctionnels d'OrCad. Le premier simulateur de PSpice a été introduit en 1985. Depuis cette date, il a été constamment mis à jour en fonction de la technologie des ressources informatiques et des systèmes d'exploitation jusqu'au point de devenir un outil universellement utilisé dans l'industrie, dans les universités et dans les laboratoires de recherche. La plupart des fabricants de composants électroniques fournissent aujourd'hui des modèles écrits en PSpice.

PSpice est un simulateur complet pour la conception analogique. Avec ses modèles internes et ses bibliothèques largement répandues et développées, dés les systèmes à haute fréquence jusqu'aux circuits intégrés de basse puissance, tout peut être simulé. Dans la bibliothèque de PSpice, des modèles peuvent être édités mais les utilisateurs peuvent également créer des modèles pour de nouveaux dispositifs à partir des fiches techniques. « PSpice A/D Basics » est un simulateur de signaux mixtes. C'est une version plus élaborée de PSpice qui peut être employée pour simuler des systèmes mixtes sans limite théorique de taille, contenant des parties analogiques et des éléments numériques.

Malheureusement, quand il s'agit de grands systèmes, les simulations deviennent trop lourdes et demandent un temps d'exécution prohibitif.

I.4 Simulateur mixte (analogiques, numériques)

Les exigences de la technologie et du marché ont imposé le développement d'outils plus puissants capables de traiter simultanément les domaines analogiques et numériques. La plupart des systèmes électroniques actuels comportent des combinaisons de circuits analogiques et numériques. Ce besoin a entraîné depuis la fin des années 90, l'apparition de langages de description matérielle de systèmes à signaux mixtes [Coo01] MSHDLs. Ces types de langages offrent un grand intérêt dans une approche de conception système.

I.4.1 SABER

SABER [Ham01] est un outil très utilisé, développé par la société Analogy (aujourd'hui Synopsys) et orienté vers la conception système. Il offre la possibilité de faire des simulations de signaux et technologies mixtes : analogiques et numériques, grâce à l'existence de passerelles avec d'autres outils. Les algorithmes de simulation de SABER, fournissent une capacité de convergence qui permet à l'utilisateur d'arrêter et de relancer la simulation pour regarder les résultats intermédiaires et/ou changer certains paramètres des composants sans quitter l'environnement de simulation. La liste d'analyses disponibles sur SABER inclut : l'analyse de Monte Carlo, l'analyse de sensibilité, l'analyse en fréquence, l'analyse du bruit, l'analyse de distorsion, le calcul de fonctions transfert, transformées de Fourier et simulation des tensions d'alimentation. Tous les modèles (numériques, analogiques et mixtes) de la bibliothèque SABER standard sont codés en langage MAST.

L'interface de Co-Simulation Saber/Verilog-XL [Ana01a] combine les capacités de SABER avec le simulateur pour la conception numérique Verilog-XL de Cadence Systems. Cette interface donne à SABER l'avantage de pouvoir co-simuler avec Verilog dans presque tous les principaux environnements de conception, y compris SaberSketch, des environnements de Mentor Graphics, de Cadence ou Viewlogic. La sortie de la simulation est combinée et synchronisée en temps pour afficher et corréler les données analogiques et numériques. L'interface de Co-Simulation de SABER/ModelSim [Ana01b] incorpore les simulateurs numériques VHDL de ModelSim.

L'interface Saber/Fusion [Ana01c] (ancienne STI) fournit un service efficace pour la simulation mixte analogique / numérique dans l'environnement de conception Powerview. Le STI combine le simulateur AHDL de SABER avec de ViewSim structural, de VHDL Speedwave et des simulateurs numériques de VCS Verilog. Le Saber/Fusion STI se combine également avec le logiciel d'intégration Frameway. Le résultat est une interface graphique qui fournit la Co-Simulation rapide de circuits conçus avec de nombreux modèles composants de bibliothèque disponibles dans les simulateurs de SABER et de FUSION.

L'inconvénient majeur de Saber, est son langage propriétaire, MAST, qui ralentit sa diffusion. Dernièrement, suite au rachat de l'outil par la société Synopsys et grâce à une réaction commerciale naturelle face à la monter en puissance de VHDL-AMS, une nouvelle initiative a été

lancée. Il s'agit d'une proposition OpenMAST™ [Syn04] dont l'objectif est de faciliter l'accès au code des modèles écrits en MAST.

I.4.2 Verilog –AMS

Verilog-AMS [Acc98] a été créé sous la tutelle d'Accellera (Organisation de standards EDA) afin de mettre en place les extensions analogiques mixtes de Verilog (IEEE-1364). La première version était Verilog-A LRM sortie en juin 1996 puis Verilog-AMS LRM en août 1998. Le langage Verilog-AMS permet de faire la description comportementale des systèmes analogiques et mixtes.

Ainsi que VHDL-AMS (1.3.2.3), Verilog-AMS peut être applicable aux systèmes électriques et non électriques. Le langage permet de faire des descriptions de systèmes, en utilisant des concepts comme des nœuds, des branches, et des ports. Les signaux de type analogique et numérique peuvent être présents dans le même module. Au contraire de VHDL-AMS, Verilog-AMS n'est pas un standard IEEE.

I.4.3 VHDL-AMS

VHDL-AMS est une norme IEEE [Ieee99] (1076,1 de 1999) qui élargit la définition du VHDL pour inclure la description des systèmes analogiques mixtes. Avec VHDL-AMS les systèmes qui étaient décrits en utilisant plusieurs outils tels que MATLAB, VHDL et SPICE peuvent être tous modélisés en utilisant un seul langage.

VHDL-AMS inclut toutes les propriétés du VHDL standard, avec en plus, la capacité de décrire les systèmes mixtes, analogiques et numériques par le biais de modèles multi abstractions, multidisciplinaires, hiérarchiques à temps continu et à événements discrets [Her02]. Au niveau conception système, VHDL-AMS peut être utilisé pour faire des descriptions de haut-niveau comme la description comportementale, le RTL (Register transfert level), les fonctions de transfert avec les transformées Z et de Laplace, des convertisseurs numérique/analogique, analogique/numérique, « phase-lock-loops » comportementaux, et les filtres analogiques et numériques. En revanche, VHDL-AMS ne permet pas de résoudre ni des systèmes à équations différentielles partielles ni des descriptions de caractéristiques géométriques des systèmes.

Pour la conception électronique détaillée, VHDL-AMS permet :

· Des simulations au niveau des portes logiques.

· Des modélisations de circuits analogiques et de modèles au niveau transistor SPICE / VHDL-AMS.

· Des descriptions de systèmes par des équations simultanées, non linéaires, différentielles et algébriques.

· De la modélisation et de la simulation des effets physiques liés au fonctionnement numérique.

Dans notre démarche, VHDL-AMS présente l'avantage [Her02] de proposer un langage commun indépendant des fournisseurs et de la technologie. Du point de vue technique, il permet une haute modularité facilitant les descriptions hiérarchiques. Cependant, le langage est complexe et les premières impressions de l'utilisateur peuvent être relativement décourageantes. Cette sensation est accentuée par le fait de ne pas pouvoir compter sur le support total d'une norme récente. Actuellement, une nouvelle version est en cours de préparation. Par rapport à la synthèse, VHDL-AMS inclut tous les sous-ensembles synthétisables de VHDL pour la partie numérique. Pour la partie analogique, des premiers travaux ont été réalisés [Dob03] [Dv03].

Quelques simulateurs sont déjà disponibles sur le marché :

· AdvanceMS™ [Mg01a] ANACAD (Mentor Graphics).

· System Vision™ [Mg03] version 8.3.0 de Mentor Graphics.

· SIMPLORER® 7.0 [Ans03] développé par ANSOFT.

· SMASH™ 5.1.3 [Dg03] de DOLPHIN Integration.

· TheHDL d'AVANTI.

· Hamster [Sim03], un simulateur gratuit pour PC, de SIMEC. Cet outil a disparu mais il s'utilise encore pour guider les « premiers pas » des utilisateurs de VHDL-AMS.

· SaberHDL™ [Syn03], de chez Synopsys propose l'option d'un simulateur intégré pour la simulation mixte. Le fabricant offrira un outil capable de supporter les langages : VHDL-AMS, MAST, HSPICE et Verilog AMS. SaberHDL pourra fonctionner sur Sun Solaris 2.6.8, Windows 2000 et RedHat Linux.

En conclusion, hormis les soucis d'implémentation de la part des fabricants d'outils. L'intégration naturelle de modèles de plusieurs disciplines permet d'avoir une vraie approche système en adéquation à notre problématique générale de conception.

I.5 La simulation matérielle / logicielle

Le co-design est l'une des techniques les plus intéressantes car elle s'efforce de mettre en place une vraie méthode de conception simultanée du matériel (100% numérique) et du logiciel. Son émergence est due à la grande et croissante ressemblance de la conception des systèmes numériques avec la conception du logiciel. L'objet du co-design [Dmg97] est de réaliser les objectifs de la conception au niveau système en regroupant et exploitant la synergie du matériel et du logiciel par le biais d'une conception concourante. Voici quelques exemples d'outils conçus pour le co-design.

I.5.1 La conception basée sur le langage C

Une bonne partie des outils destinés à la conception de systèmes mixtes numériques utilisent l'approche « C-Based System design », c'est-à-dire, les systèmes sont écrits sous la forme du code C ou C++. Nous commençons notre analyse avec SpecC :

- **SpecC** : Créé à l'Université de Californie Irvine par l'équipe de travail du professeur Daniel Gajski au CADLAB, SpecC [Dgg01] est plus une extension ou une adaptation du C, il est un langage de co-design «hardware/software» basé sur le langage C et proposé par UCI CADLAB. C'est une version élaborée d'ANSI-C dont le niveau d'abstraction le plus haut est décrit à base de machines à états finis. SpecC propose des spécifications comme canaux de communication, des représentations hiérarchiques, de la simultanéité et de l'abstraction de la synchronisation. Il est conçu pour être un langage unique qui peut être utilisé dans toutes les étapes du processus de co-design matérielle / logicielle. Un projet SpecC se compose d'un ensemble de déclarations comportementales, déclarations de canaux et d'interfaces. Un comportement est une classe avec un ensemble de ports : L'ensemble des comportements secondaires, l'ensemble des canaux, l'ensemble des variables et des fonctions « privées » et une fonction principale« publique ». Par ses ports, un comportement peut être relié à d'autres comportements ou canaux afin de communiquer. La fonctionnalité d'un comportement est indiquée par ses déclarations de fonction. Un canal est une classe qui contient la transmission. Il se compose d'un ensemble de variables et de fonctions appelées méthodes, qui définissent un protocole de communication. La version SpecC 2.0 a été développée par l'équipe du professeur

Masahiro FUJITA, à l'Université de Tokyo. Elle intègre des améliorations dans la gestion des événements concurrents, des interruptions, et du parallélisme. Il est intéressant de noter que récemment l'équipe du professeur FUJITA a été contactée par des concepteurs de satellites japonais qui veulent aborder la conception des micro-systèmes embarqués du point de vue système. Pour l'instant, SpecC ne comporte pas d'options pour la conception analogique et mixte. Le groupe du travail pour la version 3 a été lancé à Tokyo le 8 octobre 2002. Il étudie la faisabilité d'une extension analogique du langage. A notre avis, il reste encore du travail pour arriver à un langage système général. Le développement de l'outil a été pénalisé par la grande partie de marché couverte par son grand concurrent System C.

- **System C :** Peut être le plus utilisé des approches « C based »**.** Les origines de System C remontent au milieu des années 90, dans les travaux de l'Université de California Irvine et du groupe Synopsys. Le premier produit était appelé « Scenic » puis « Fridge ». La première version System C 0.9 est sortie en 1999 avec l'incorporation des éléments récupérés de N2C –Coware.

System C est un ensemble de bibliothèques crées en langage C++, permettant de faire la description d'un système logicielle / matérielle par le biais de spécifications exécutables et de plusieurs niveaux d'abstraction pour un même système. System C fournit la possibilité de créer des modules, processus fonctionnels et portes logiques. Le compilateur Co-Centric SystemC synthétise la description « hardware » écrit en System C au niveau des portes logiques (gate-level netlist) ou en Verilog ou VHDL pour faire de la synthèse sur des FPGAs. Les modèles System C sont écrits sur le formalisme des FSM « Finite State Machines ».

Fabriquant	Outil
Axys Design	MaxCore developer Suite
Axys Design	MaxSim
Cadence	SPW
CoFluent Studio	Cofluent Design
CoWare	CoWare N2C Design System
Forte Design System	Cynlib Tool Suite
Innoveda	Visual Elite-Architect
Mentor Graphics	Vstation TBX
Synopsys	CoCentrics System Studio
Synopsys	SCC Synopsys Cocentric SystemC compiler
Veritools	SuperC
Virtio	Virtual Prototyping to SystemC
WHDL Language	Rule Checker & Rule Generator

Figure 1 : Quelques outils qui supportent System C.

Cette approche très intéressante est devenue l'une des standards du fait pour la conception et la synthèse de systèmes numériques mixtes matériels et logiciels. Tel que l'illustre la Figure 1, la plupart des fabricants d'outils CAO proposent System C parmi leurs produits. Une initiative commence à prendre de l'ampleur ; il s'agit d'étendre l'utilisation du System C aux systèmes électroniques mixtes matériels et logiciels. Des travaux de mise en forme d'une proposition System C-AMS a été proposés par [Gev04]. L'approche système préconisée est intéressante car elle peut représenter une alternative pour le traitement des systèmes avec tout type de composantes : Analogiques, numériques et logicielles.

- **Handel-C** [Cel02] est un langage écrit sur la base d'ISO/ANSI-C destiné à l'implémentation d'algorithmes sur « hardware », à l'exploration architecturale et au co-design. Handel-C permet la conception de matériel en utilisant des méthodes de conception de logiciel élargies avec des particularités pour le développement de matériel. Elles incluent des largeurs variables des structures (vecteurs) de données, le traitement parallèle des communications et des événements. Handel-C n'utilise pas de machines à états finis, grâce à une méthode propriétaire de description des écoulements périodiques et parallèles.

Les modèles Handel-C peuvent être insérés dans une méthodologie de réutilisation car des fonctions peuvent être compilées dans des bibliothèques et être employées dans d'autres projets. Des noyaux écrits sous Handel-C peuvent être exportés comme boîtes noires d'EDIF, de VHDL ou de Verilog pour leur réutilisation.

D'après le fabricant, Celoxica, les points forts de Handel-C sont :

· Un langage de haut-niveau basé sur ISO/ANSI-C pour l'exécution des algorithmes sur « hardware ».

· Le langage ne demande pas de grands investissements en temps de formation des utilisateurs.

· Handel-C permet de faire des appels directs sur des fonctions externes écrits sous C/C++ et vice-versa.

· Des extensions spécifiques pour le matériel incluant la gestion du parallélisme et des communications.

· Construction des noyaux spécifiques pour la suite Celoxica DX.

Nous trouvons intéressante la compatibilité de l'outil et l'utilisation de leur propre modèle de représentation des états des systèmes ; mais il est trop focalisé sur la réalisation matérielle. Il manque la généralité requise par notre approche.

- **N2C** [Cow01] a été développé par la société CoWare. Cet outil permet de capturer les spécifications d'un système numérique dans un modèle exécutable et implantable à partir de langage C/C++. Avec N2C, l'utilisateur peut faire une spécification concourante, visant deux objectifs :

· Une implémentation et vérification de matériel et logiciel embarqué spécifique à l'application.

· L'évaluation et l'intégration de la propriété intellectuelle (IP) de matériel et de logiciel vers des nouveaux produits ou dérivés.

CoWare N2C est conçu pour co-exister avec la plupart des outils commerciaux : Simulateurs de HDL, simulateurs de positionnement d'instruction (ISS), outils intégrés de l'environnement de développement de logiciel (IDE), et des systèmes d'exploitation temps réel. Les outils de synthèse reçoivent la sortie de N2C pour démarrer la déclinaison et synthèse du système. Une version universitaire de N2C est disponible pour les membres d'Europractice Software Service [Eur04].

I.5.2 Les outils de haut-niveau

Dans une autre catégorie, nous citons des outils qui abordent le problème du co-design à un niveau plus élevé que la partition logicielle/matérielle. Parmi eux, le projet **POLIS** [Clo01] de l'Université de Californie Berkeley qui a été développé afin de créer une méthodologie formelle unifiée pour la modélisation complète des systèmes embarqués. Cette méthodologie inclue la partition matérielle/logicielle, la synthèse automatique et la vérification. POLIS a été développé sur le modèle de calcul formel CFSM ou « Co-design Finite State Machine ». Il est un logiciel expérimental.

Bien qu'il ait été testé sur plusieurs exemples de dimension industrielle [Fll+98] [San96], il ne peut être applicable qu'à certains domaines spécifiques. Eagle*i* [Syn00] est un outil pour la co-vérification Logicielle/Matérielle depuis la post-partition jusqu'au prototype physique. Eagle*i* supporte des outils EDA de haut rendement, la simulation cycle à cycle, les accélérateurs de matériel et l'émulation de matériel pour la conception multiprocesseur. Avec Eagle*i,* il est possible de distribuer la simulation à travers un réseau pour améliorer la vitesse de simulation. Il fournit une plate-forme d'interopérabilité UNIX/PC. Cette caractéristique peut le rendre intéressant pour son intégration dans des plates-formes de conception.

Seamless CVE [MG00a] est un outil de Mentor Graphics pour la conception électronique. Grâce à son interface « Plug-In » (SPI), il est capable de réaliser la simulation multiprocesseur. Seamless CVE est compatible avec plusieurs outils de vérification et description [Mg00b] comme ModelSim VHDL, Verilog XL, VSC et avec plus de 70 microprocesseurs des différents fournisseurs.

A cause de l'exécution du modèle complet d'un microprocesseur, la vitesse du simulateur peut être six ou sept fois plus lente que l'exécution temps réel. Seamless CVE accélère la Co-Simulation grâce à la séparation fonctionnelle du microprocesseur de son interface électronique. La suppression sélective de certains cycles dans la simulation matérielle est facultative. Les simulations matérielles et logicielles sont divisées en un simulateur d'instructions ou «Instructions Set Simulator » et un modèle d'interface ou « Bus Interface Model » pour le comportement électronique des entrées/sorties du processeur. L'arbitrage entre l'exécution des simulations est réalisé par le « Co-Simulation Kernel ».

Nous trouvons que cet outil demande de connaître préalablement l'architecture du système et donc il est utile seulement lorsque les choix de conception ont été réalisés. Nous le classons

dans la catégorie d'outils système car il permet de vérifier le fonctionnement complet de l'application.

L'outil **Co-Fluent Studio SDE,** proposé récemment par la société Co-Fluent Design. Il est orienté vers la conception de systèmes électroniques numériques comportant des implémentations matérielles et logicielles. Les origines de ce logiciel se trouvent à l'Ecole Polytechnique de Nantes sous la direction de Jean-Paul CALVEZ. Le principe d'utilisation [Per04] est de distinguer clairement les représentations fonctionnelles et architecturales du système à concevoir. En effet, l'outil permet de réaliser ces descriptions indépendamment. Trois niveaux d'abstraction ont été identifiés pour l'analyse des performances :

- Le niveau Système.
- Le niveau composant.
- Le niveau des communications entre composants.

La Méthodologie de Conception de Systèmes Electroniques [Cof03] (MCSE ou CoMES en anglais) est utilisée avec l'outil afin de gérer les différents niveaux de complexité d'un projet de conception.

La description finale du comportement des systèmes est générée sous deux formes automatiques de code. La première, destinée à la modélisation Système en C/C++, la deuxième en VHDL synthétisable orientée à l'implémentation des systèmes sous la forme de circuits intégrés. Nous trouvons cet outil particulièrement intéressant car il propose une approche générale pour la conception de haut niveau du domaine électronique, en considérant la modélisation du comportement des systèmes avec des modèles formels de calcul, de plus une issue vers la matérialisation est proposée. Dans de futurs travaux, il sera convenable d'approfondir cette démarche afin d'envisager des ouvertures vers une généralisation orientée VHDL-AMS donnant la possibilité de gérer des projets pluridisciplinaires.

I.6 Les méthodes pour la conception logicielle

Nous continuons maintenant avec les méthodes pour la conception logicielle. Ces méthodes ont été pionnières dont la façon d'aborder le problème de la conception. En effet, les approches de la conception logicielle ont fait émerger des nombreux concepts également intéressants pour la conception amont.

I.6.1 SADT et SA/RT des méthodologies à l'origine de la réflexion système

Nous trouvons dans la seconde moitié du siècle dernier où la complexité croissante des systèmes avait déjà demandé des efforts des scientifiques et des ingénieurs pour l'établissement d'outils et/ou des méthodes permettant d'alléger la tâche de spécification et de conception de ces systèmes. Parmi ces premiers travaux, ROSS avec SADT [Ros72] est une des pionniers dans la recherche d'une solution au problème de la spécification et de la conception des systèmes à haut-niveau. Développée à partir de 1972, cette approche avait pour objectifs de couvrir l'analyse de besoins, la spécification, la conception et la documentation en facilitant le partage de l'information entre les utilisateurs. Le modèle utilisé propose une description en blocs (d'activités ou de données) relié par quatre types de liens : Entrée, sortie, contrôle et mécanisme. Les blocs SADT peuvent être décomposés en niveaux hiérarchiques. L'approche propose deux formes possibles de diagrammes : Les « actigrams » et les « datagrams » qui représentent deux vues différentes d'un même système. L'utilisation de SADT été basée sur des principes de délimitation du contexte du système et de la limitation de taille de l'information. Les décompositions étaient limitées à sept blocs ±2 (plus ou mois deux) par feuille. Cette approche simple et compréhensible n'est pas exclusive à un métier spécifique. Un banquier, un fonctionnaire,… peuvent lire un diagramme de leur domaine sans connaître la méthode. Efficace en spécification des exigences, elle présente un certain nombre d'inconvénients dès que des phases de conception sont abordées notamment ses insuffisances pour l'expression des algorithmes de contrôle.

D'un autre côté, **SA/RT** (Structured Analysis with Real-time-Extensions) [Wm85] propose, à partir d'une analyse établie sur une représentation graphique, une modélisation système dans laquelle deux facettes sont clairement différentiables : un modèle du processus statique qui lui est attaché, et un modèle de contrôle dynamique qui en permettra l'utilisation. L'originalité de cette méthode est la prise en compte de l'aspect dynamique du système. SA/RT est donc bien adapté aux applications temps réel à fort comportement dynamique.

I.6.2 UML : le langage unifié de modélisation

UML est un langage qui émerge comme un standard de fait pour la conception de systèmes logiciels à haut-niveau, il a été proposé par l'OMG [Omg03] en 1997, avec comme objectif quatre activités principales [Bro03] du processus de conception :

- Une description du système selon plusieurs points de vue.
- La spécification des besoins et de la mise en œuvre.
- La visualisation pour faciliter la compréhension et la communication parmi les partenaires de la conception avant la réalisation du système.
- La représentation de systèmes complexes.
- La documentation de la totalité du projet, dès les spécifications jusqu'aux tests de fonctionnement.

L'application d'UML exige l'adoption d'une méthodologie claire et l'utilisation d'un bon outil logiciel mettant en œuvre le langage. Malgré cette volonté d'unification, UML n'est pas une solution totale pour la conception système, car elle n'établie pas la façon dont les diagrammes doivent être employés et moins encore un principe d'intégration ou d'interopérabilité entre eux. L'utilisation du langage perd beaucoup d'efficacité sans une méthodologie et sans le support d'un outil.

Dans le langage UML, plusieurs types de représentations graphiques [Mil03] sont possibles. Cinq modèles de représentation (chacun avec un ou plusieurs types de diagrammes) conforment la sémantique UML 1.5, à savoir :

Modèle d'utilisateur :

- « Use case diagrams » : ces diagrammes représentent le fonctionnement du système du point de vue d'un observateur externe. Leur but est de montrer ce que le système fait sans détailler le « comment ».

Modèle structurel :

- Diagrammes de classes : ces diagrammes décrivent les états statiques du système et leurs connexions.
- Diagrammes à objets : il s'agit d'une simplification des diagrammes de classes. Ces diagrammes décrivent les objets avec leurs interactions.

Modèle comportemental :

- Diagrammes de séquence : il s'agit de diagrammes d'interaction qui montrent le séquencèrent des opérations du système. C'est une vue temporelle.

• Diagrammes de collaboration : Ces diagrammes d'interaction comportent la même information que les diagrammes de séquence mais se focalisent sur les rôles au lieu des temps.

• Diagrammes à états et leurs extensions les « Statechart diagrams » : ils illustrent les états possibles du système et les transitions qui provoquent les changements des états.

• Diagrammes d'activités : Il s'agit essentiellement de « flowcharts ».

Modèle d'implémentation :

• Diagrammes de composants : ces diagrammes représentent l'équivalent matériel et ou logiciel des diagrammes de classes.

Modèle d'environnement :

• Diagrammes de déploiement : Ce dernier type de diagramme illustre les configurations physiques du matériel et du logiciel.

D'une façon générale, ces outils permettent de décrire les nœuds de distribution et leurs interactions dans le cas de systèmes distribués. Hormis l'approche objet, l'avantage majeur du langage UML est la multiplicité de diagrammes qu'il offre. Il permet au concepteur de créer différentes représentations du fonctionnement du système.

I.7 Autres approches métiers et mixtes

D'autres approches ont essayé de se hisser à un niveau plus amont pour aborder le problème de la conception système. Ici, nous avons identifié deux volets : d'un part, les outils issus des initiatives des communautés indépendantes d'aborder le « haut-niveau » relatif à leur domaine, tels que l'automatique, le logiciel et l'électronique. D'autre part, les initiatives récentes qui « joignent » la conception système avec des approches généralistes et hétérogènes.

La communauté des automaticiens utilise depuis des années M**atlab® et Simulink®** comme leurs outils de base pour le calcul scientifique. Il s'agit, peut-être des plus célèbres outils de modélisation mathématique globale, Matlab® et Simulink® de chez **MathWorks**. Il dispose en 2004 des versions 7 et 6 respectivement. Traditionnellement, ils sont utilisés pour faire de la modélisation générale de systèmes par des fonctions de transfert, avec une forte orientation vers les systèmes de contrôle et commande. Ils permettent de réaliser une modélisation détaillée des

algorithmes de contrôle des systèmes dans des domaines multiples. Les versions actuelles comportent plusieurs « toolboxes » permettant de participer à la conception système à différents niveaux. Parmi les applications les plus intéressantes, nous trouvons l'utilisation de Matlab®/Simulink® en combinaison avec des outils VHDL pour réaliser le test de modèles pour leur implémentation matérielle. Le lien de Co-Simulation avec ModelSim®1.1 [Tmw03] permet de co-simuler et de vérifier du VHDL et du Verilog.

Cet outil permet de réaliser des vecteurs de test « logiciels » en intégrant les solutions HDL avec les algorithmes, ceci permet de vérifier le fonctionnement du HDL par rapport au modèle original ainsi que de donner des caractéristiques comportementales aux « testbench ». Ce principe est aussi répandu pour réaliser des essais de type « harwdare in the loop », décrits de façon plus détaillée par [Gom01].

Parmi les offres de **MathWorks** nous trouvons des applications pour la spécification et la modélisation des systèmes automobiles [Tmw04a], la conception électronique mixte et la modélisation des composants. Ils proposent aussi des solutions pour la conception des systèmes embarqués [Tmw04b] et de certaines applications aérospatiales, notamment [Tmw04c] pour la conception des systèmes de commande et la validation de leurs interfaces homme/machine, la modélisation des systèmes mécaniques, des sources d'énergie et pour la modélisation détaillée de l'environnement de l'appareil en considérant des aspects tels que le vent et la gravité.

Dans la mesure où les outils de simulation progressent, nous pourrons envisager des modélisations couplées VHDL/MATLAB dans les quelles les équipes de conception pourront combiner leurs « savoir-faire » en matière de systèmes de commande sous MATLAB, pour élaborer des « testbenchs », pour vérifier les modèles écrits en VHDL-AMS ou bien pour modéliser et synthétiser des lois de commande incluses dans ces systèmes.

Du côté informatique, de nombreux outils ont été proposés : Esterel Technologies a créé **Esterel Studio** [Dd00], pour la spécification et le développement des systèmes numériques et logiciels temps réel en utilisant la représentation hiérarchique graphique SyncCharts, donc, une notation graphique conçue par Charles ANDRE [And96] à l'Université de Nice Sophia-Antipolis. Le langage de programmation synchrone et son compilateur ont été conçus à Ecole des Mines de Paris et à l'INRIAi. Les travaux de recherche [Cla01] sur le langage, pour la plupart d'origine française, sont à la base de la création, en avril 2000, de la société d'Esterel Technologies SA.

L'approche synchrone d'Esterel studio pour la modélisation et la programmation a été retenue afin d'éviter les erreurs et les difficultés propres de la conception de ce type de systèmes, par le biais de méthodes traditionnelles. Esterel Studio utilise une sémantique de type FSM (Finite State Machine) très pure, idéale pour concevoir des systèmes indépendants de l'implémentation et dominés par des commandes.

I.8 Approche classique de la validation logicielle

Le flot de conception classique de circuits intégré propose une étape de validation complète du logiciel embarqué, apparaissant relativement tard dans le processus du développement. Notamment, un modèle matériel du processeur dédié doit être développé et validé. Cette section présente successivement le flot de conception classique dans lequel s'insère la validation logicielle, puis décrit l'interfaçage entre le matériel et le logiciel, et enfin identifie les limites de son utilisation dans le cas de systèmes complexes.

La figure 2 représente le flot de conception classique d'un système contenant un processeur et son logiciel embarqué. A partir des spécifications complètes du système, les descriptions de haut niveau des parties matérielles et logicielles sont produites (manuellement dans le cas général). Pour la partie matérielle, un langage tel que VHDL ou VERILOG est le plus souvent utilisé. Nous nous restreignons ici au langage VHDL, couramment utilisé en Europe. La partie logicielle utilise un flot de compilation C, relativement classique.

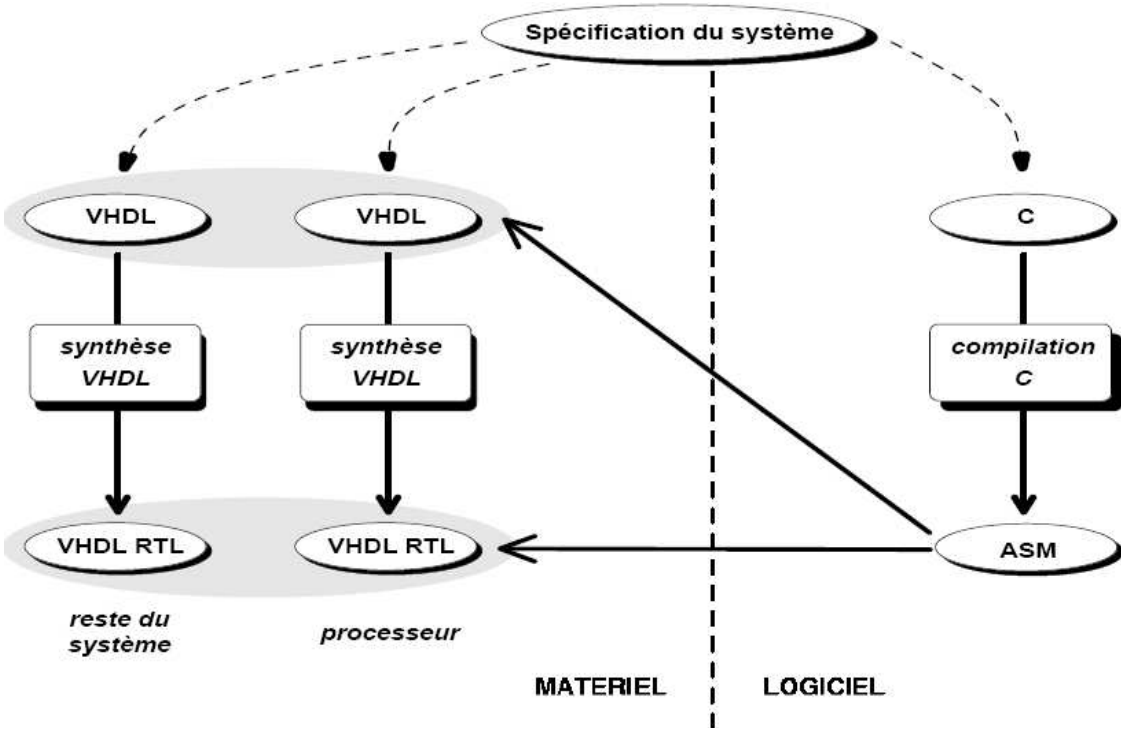

Figure 2 : Flow de conception classique.

La partie matérielle (à gauche) est constituée de deux parties : un processeur embarqué pour simplifier, nous ne considérons qu'un seul processeur et le reste du système constituer d'un ensemble de blocs matériels divers. Nous représentons ici un flot de synthèse comportementale du matériel, générant des modèles VHDL-RTL à partir de modèles comportementaux VHDL.

Le logiciel écrit en C (à droite) est compilé en code assembleur (ASM), par un compilateur développé spécialement pour ce processeur. Ce code assembleur, sous sa forme binaire, est ensuite chargé dans la mémoire programme du modèle VHDL du processeur (comportemental ou RTL), lequel est simulé conjointement avec le reste du système par un unique simulateur VHDL. Cette simulation est appelée simulation au niveau assembleur. Grâce à l'exactitude temporelle des simulations VHDL, la validation du logiciel embarqué atteint un niveau de précision au cycle près [Hag 93]. Le prix à payer est une vitesse de simulation relativement faible, puisqu'elle ne dépasse pas quelques instructions par seconde pour un seul processeur [Mei 97].

L'interface entre le logiciel et le matériel se situe entre le logiciel applicatif et le processeur, et plus précisément entre les instructions en code machine et le décodeur d'instructions figure 3.

Le code assembleur (ou code machine) de l'application, comprenant également les entrées/sorties avec le reste du système, est directement traité par le décodeur du processeur. Les opérations d'entrée/sortie destinées au reste du système sont transmises à l'interface bus, via le bloc d'entrée/sortie du processeur. L'interface bus gère la transmission des données et signaux de contrôle sur le bus physique, partagé avec les autres opérateurs du système. L'interface matérielle -logicielle est donc entièrement définie par le jeu d'instructions du processeur.

Nous distinguons deux types de validations : la validation fonctionnelle et la validation temporelle. La première vise simplement à valider le comportement du logiciel, communiquant éventuellement avec le reste du système. La seconde s'attache à vérifier que les différents signaux de contrôle et données sont traités dans un intervalle de temps compatible avec les exigences de performances, et qu'ils sont disponibles à la date prévue.

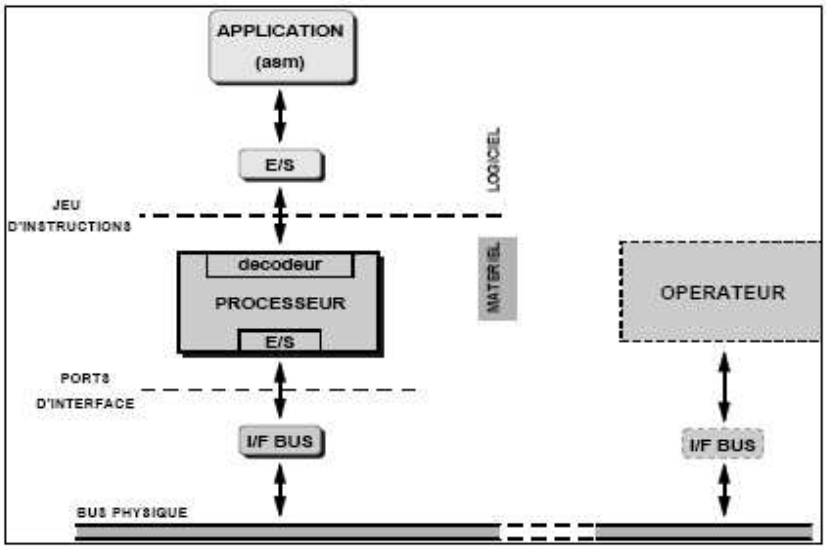

Figure 3 : Interface logiciel-matériel en simulation VHDL.

Cette approche de validation présente un inconvénient majeur : la simulation du logiciel avec le matériel intervient tard dans le flot de développement. La figure 4 représente les étapes nécessaires pour aboutir à la validation du logiciel embarqué.

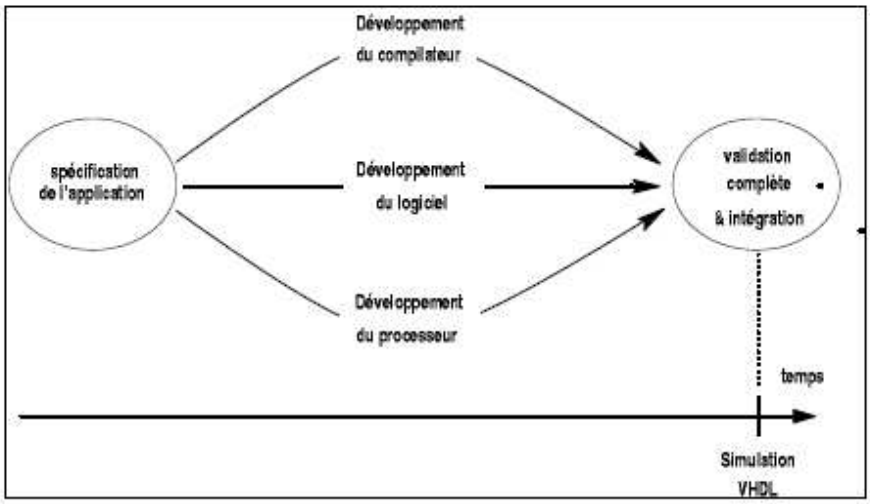

Figure 4 : Étapes pour la validation du logiciel dans un flot classique.

Le logiciel doit évidemment être développé (en C), de même que le compilateur complet (comprenant au minimum le générateur de code, l'assembleur et l'éditeur de liens) doit être développé et validé. De plus, le processeur lui-même doit être disponible, et donc ses spécifications figées (interface et jeu d'instructions). Il est remarquable que plus la validation intervient tôt dans la conception d'un système, plus une erreur est corrigée rapidement (voir figure 5) [Cal 95].

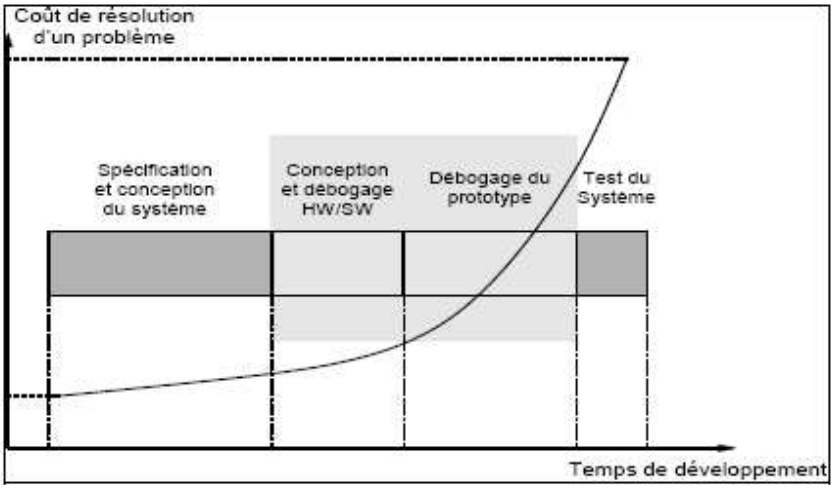

Figure 5 : Temps de conception et coût de débogage.

Dans ce contexte, la moindre modification des spécifications (processeur, interface, jeu d'instructions) implique la régénération du modèle du processeur ainsi que le compilateur. Le temps du cycle est alors trop long pour espérer explorer plusieurs solutions [Ver 94] [Bol 97].

I.9 Outils existants

Comme nous venons de le voir, les systèmes enfouis font appel à des architectures hétérogènes logicielles/matérielles [Aas 04]. Afin de concevoir de tels systèmes, les méthodes de conception conjointe logicielle/matérielle sont utilisées. Elles permettent de définir les sous-ensembles du système à intégrer et d'effectuer leur partitionnement sur les cibles logicielles et matérielles. Cependant, la complexité sans cesse grandissante de ces systèmes et les diverses contraintes auxquelles doivent faire face les concepteurs, font qu'il est nécessaire de faire appel à des nouvelles langages et outils, ce que nous allons montrer maintenant. Afin d'assurer le succès commercial de tels systèmes il est primordial que ceux-ci répondent aux attentes des consommateurs parmi les quelles nous pouvons citer : le délai de mise sur le marché (time-to-market) des nouvelles générations de produits, nouvelles fonctionnalités, petite taille, poids léger, faible consommation en énergie, simplicité d'utilisation et bien sûr coût acceptable. Les outils sont arrivés à un degré de maturité assez important restent cependant trop lents et ne permettent donc pas d'explorer, dans des délais raisonnables, les gigantesques espaces de conception inhérents aux systèmes actuels. Nous allons citer ainsi l'outil Design Trotter et plus particulièrement le module d'estimation système qui permet l'exploration rapide des fonctions complètes décrites en langage de haut de niveau (actuellement le langage C). Le résultat obtenu grâce à la méthode proposée, est aidé le concepteur lors de la phase initiale de la conception d'un système mais il reste limiter sur un nombre d'application et peut contenir des fautes. Le concepteur peut avoir une approche plutôt logicielle (approche "informatique", exemple : C) ou plutôt matérielle (approche "électronique", exemple : VHDL). Le choix du langage de spécification qu'il utilise est d'ailleurs souvent guidé par le type d'approche auquel il est habitué. Ensuite, il existe plusieurs algorithmes pour décrire un même traitement, quelque soit le langage choisi. C'est surtout ce deuxième point qu'il serait intéressant d'intégrer à cette méthodologie et à l'outil Design Trotter.

Et pour le plus récent, nous trouvons l'outil proposé par M. Juan-Carlos Hamon qui est HileS .l'outil HiLeS Designer 0 est arrivé au stade opérationnel [Moc05]: il est en version 0V6. Il s'agit d'un outil de conception amont permettant de représenter les spécifications d'un produit en un modèle formel basé sur les Réseaux de Pétri et le langage VHDL-AMS. Ce modèle permet par une relation à l'outil TINA une certaine vérification des spécifications et la validation d'une

architecture temporisée du système sous la forme de blocs fonctionnels interconnectés. L'outil est encore faible en matière de vérification. Le premier exemple, le calculateur ECP, a permis de tester la première version de cet outil dans un environnement industriel. Un certain nombre de limites sont apparues pour les quelles ils ont fait des propositions de nouveaux développements :

- Lecture de spécifications avec un guide d'interprétation. Ce dernier projet nous a permis d'envisager le rôle de UML dans notre démarche ainsi que de profiter de l'importante avis de l'expert sur la construction de la modélisation HiLeS.
- La mise en place des observateurs définis par l'utilisateur.
- L'amélioration des interfaces avec des outils de simulation VHDL-AMS. Pour incrémenter la flexibilité de HiLeS Designer.
- L'utilisation ou la proposition d'une méthode facilitant l'écriture du code VHDL-AMS associé aux blocs fonctionnels HiLeS.

I.10 Conclusion

Ce premier chapitre définit tout d'abord le domaine de la conception dans un contexte où l'innovation ne provient plus seulement des nouveautés en termes de matériaux ou de technologies :

La conception et la simulation des systèmes électroniques ont été un argument fort de motivation pour le développement d'outils, de méthodes et langages permettant de gérer la complexité croissante des circuits et des systèmes. La CAO électronique, prise en compte à tous ses niveaux, industriel, académique et recherche, est devenue un moteur incontournable d'innovation et de développement d'outils dédiés à la conception. Les objectifs sont toujours les mêmes, réduire le temps de production« time to market », anticiper les possibles sources d'erreur, réduire les coûts de fabrication, réaliser des prototypes virtuels les plus représentatifs de la réalité et en général, réaliser la conception sans faute.

Malgré, le haut degré de spécialisation de la plupart des outils, le besoin de réduire le coût et le temps de fabrication a imposé la nécessité de trouver des solutions permettant d'avoir une vision globale des systèmes en prenant compte de leur complexité et de leur pluridisciplinarité. L'objectif est d'assurer la cohérence du système dès les étapes les plus en amont du processus de

conception. Dans ce contexte, l'utilisation de langages standardisés et des outils de simulation s'imposent comme une pratique incontournable qui sera expliqué dans le chapitre suivant.

PLATEFORMES DE SIMULATION

Chapitre II: Plateformes de simulation

II.1 Introduction

La conception et la simulation des systèmes électroniques ont été un argument fort de motivation pour le développement d'outils, de méthodes et langages permettant de gérer la complexité croissante des circuits et des systèmes.

Pour cela, nous avons mis en place des plateformes de simulation qui sont composées principalement de MATLAB, ModelSim 6.0 et FPGA. Pour le kit de la carte FPGA, nous trouvons principalement : ISE 7.1i, Chip Scope pro, PlanAhead, Synplify 7.3.4 et System Generator qui permettent d'assure l'interfaçage entres plateformes. En plus, nous présentons les nouveautés de chaque outil et les résultats qu'il produit pour la Co-Simulation des systèmes hétérogènes tous en conservant le même objectif qui est en premier lieu la réduction du temps de conception.

II.2 Simulation VHDL (ModelSim)

Le VHDL est un langage général de description de matériel permettant un grand niveau d´abstraction. Un système aussi complexe peut être décrit sous forme d´un ensemble de blocs interconnectés [Pn 03].

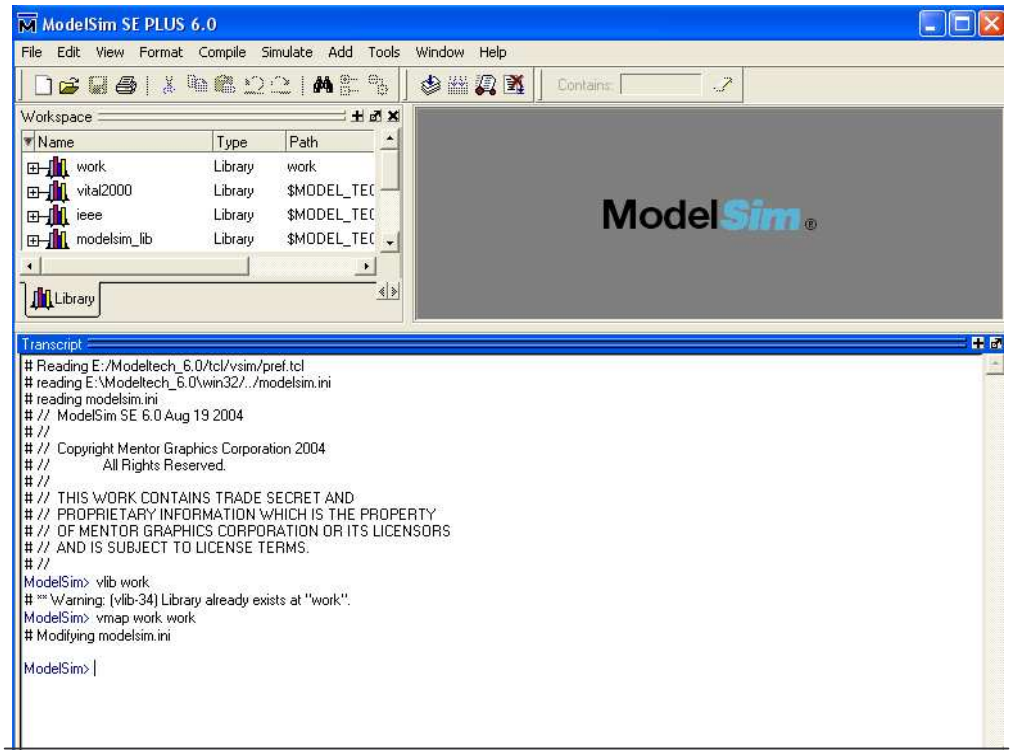

Figure 6 : Le simulateur VHDL.

Lorsque nous désirons réaliser un circuit de type programmable, ASIC, ou bien de type circuit imprimé, nous sommes soumis à une double contrainte technologique. Il faut savoir représenter un certain nombre de primitives [Sdc03] : nous parlons alors de niveau portes avec des fonctions logiques de base (ET, OU etc...), des fonctions combinatoires MSI (addition, multiplexeur etc...), des fonctions séquentielles simples (bascules et registres), (mémoire et latch). Le synthétiseur est l´outil capable d´interpréter une certaine description VHDL et d´en déduire le schéma niveau porte correspondant. La description acceptable par le synthétiseur sera dite niveau RTL (Register Transfer Logic). Le synthétiseur est très efficace au niveau porte pour des tâches telles que l'implantation des fonctions combinatoires, le calcul, le codage du compteur, le séquenceur, l'implantation des échanges par bus etc...Par contre, les choix d´architecture, la structuration d´un système sont des tâches de trop haut niveau et restent à la charge du concepteur.

Le VHDL synthétisable est soumis à des limitations propres au synthétiseur (ceux-ci évoluent sans-cesse et essaient d´intégrer des fonctions de plus en plus complexes). Au moment de la description VHDL, nous devons avoir une idée de ce qui va être généré :

- Il faut que la table de vérité d´un circuit combinatoire soit complètement définie.
- Dans un système synchrone, les fronts d´horloge doivent être parfaitement identifiés.
- Le nombre de bits d´un mot soit optimisé etc.

Donc, nous pouvons dire que le VHDL est un outil incontournable lors d´un cycle de conception. Il permet de passer d´un niveau très abstrait à un niveau circuit comme l'ullistre la Figure 7. Tout le long du cycle, certaines parties resteront inchangées car non synthétisables, ce sont les dispositifs de test du circuit. Le circuit en cours de conception passera de l´état de modèle abstrait à celui de description synthétisable.

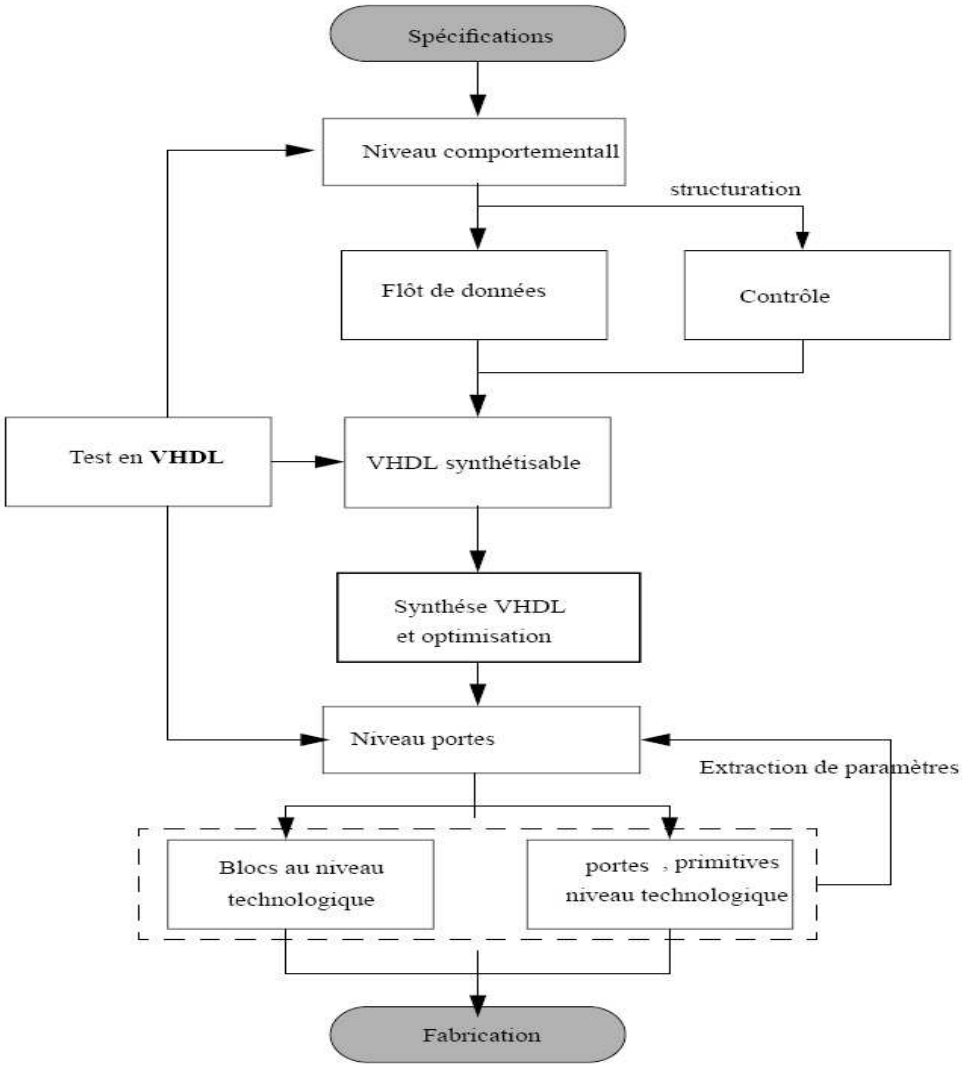

Figure7 : Utilisation du VHDL pour les niveaux de conception.

II.3 Simulation MATLAB

The MathWorks Release 14 inclut tous les produits de l'environnement MATLAB® et Simulink® [Tmw04c].

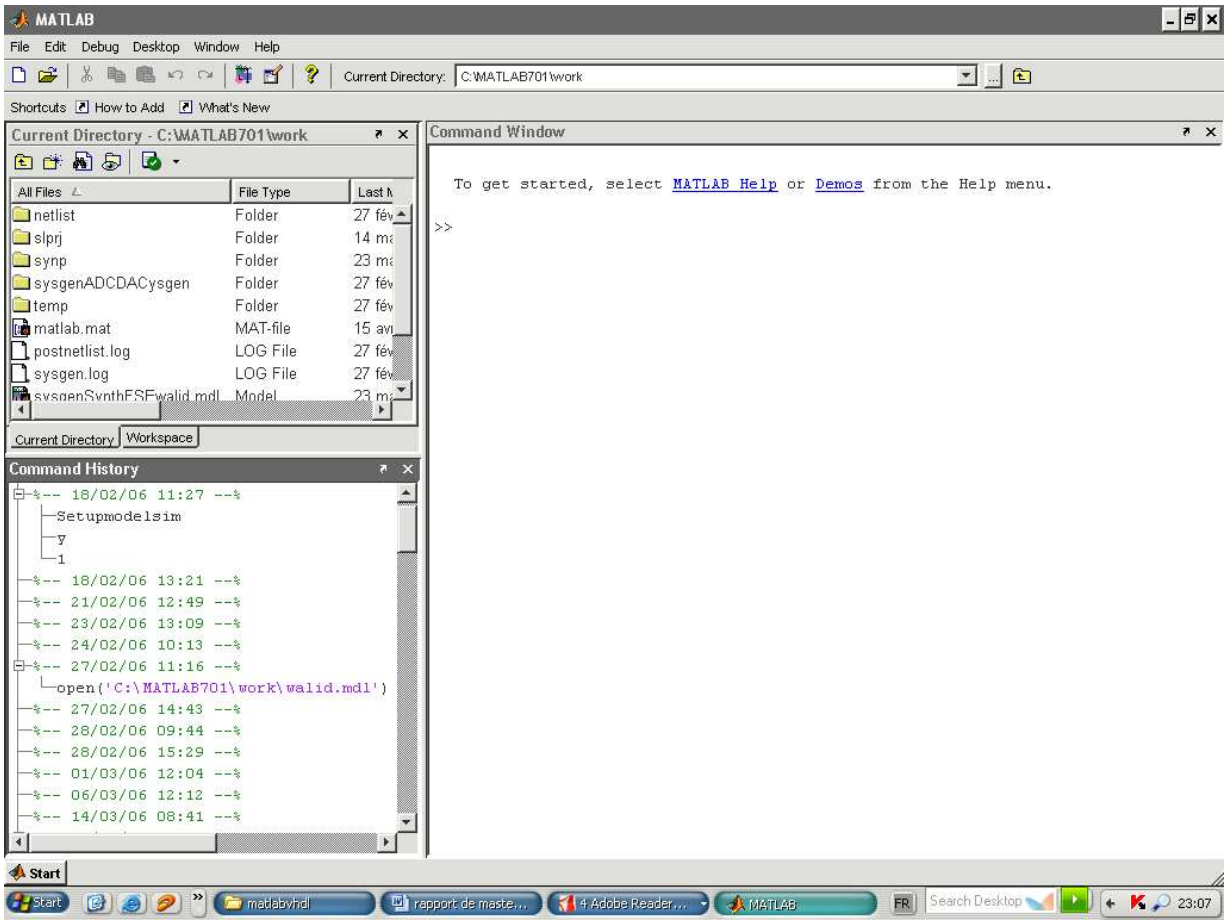

Figure 8 : L'interface de MATLAB 7.

Nouveautés pour le calcul scientifique avec MATLAB 7 :

Cette version inclut de nouveaux outils de programmation, la prise en charge des calculs entiers et simple précision, la possibilité de manipuler des ensembles de données plus importants et apporte des améliorations au niveau des performances. De plus, MATLAB Compiler 4 prend en charge la totalité du langage MATLAB, permettant ainsi le déploiement d'un plus grand nombre d'applications.

MATLAB est un langage de calcul scientifique de haut niveau et un environnement interactif pour l'analyse de données et le développement d'algorithmes et d'applications. Elle inclut aussi de nouvelles fonctionnalités majeures dans les domaines de la programmation et de la génération automatique du code, du graphe et de la visualisation, des mathématiques, de l'accès aux données. Les nouvelles fonctionnalités sont les suivantes :

Nouveautés pour la conception basée sur des modèles avec Simulink 6 :

Cette version prend en charge les projets de développement de très grande largeur et la conception d'applications dont la performance est un point critique. Simulink 6 inclut de nouvelles fonctionnalités pour gérer les modèles de grande taille et augmenter l'efficacité des flux de travail pour les systèmes de contrôle, de traitement du signal et de communication.

Simulink est une plate-forme pour la simulation multi domaine et la conception basée sur des modèles de systèmes dynamiques. Il fournit un environnement graphique interactif et un ensemble de bibliothèques de blocs personnalisables qui nous permettent de concevoir, de simuler, de mettre en œuvre et de tester de façon précise des systèmes de contrôle, de traitement de signal, de communication et d'autres systèmes qui varient dans le temps. Simulink 6 améliore les performances, la réactivité, la fidélité de la modélisation et l'efficacité des flux de travaux lors de la modélisation de grands systèmes. Les nouvelles fonctionnalités sont les suivantes :

Modélisation basée sur des composants pour les grands systèmes :

- Possibilité de segmenter un modèle en plusieurs fichiers, où chaque fichier est un modèle séparé.
- Possibilité de modéliser, de simuler, de tester et de mettre en œuvre individuellement chaque composant de la conception avant de l'incorporer dans un modèle de système.
- Meilleure intégration de nous modèles dans un logiciel existant de gestion de configuration et de contrôle de version basé sur des fichiers.
- Chargement et génération de code incrémentiels.
- Meilleurs diagrammes de mise à jour et simulations plus rapides pour les modèles de grande taille.

- Fonction Model Workspaces fournit des espaces de travail séparés pour le stockage et la gestion des paramètres et des variables de chaque modèle.
- Meilleure prise en charge des bus pour la définition des interfaces, la prise en charge des opérations sur les signaux de bus et la spécification des bus comme structures pour la génération de code.

II.4 XilinX ISE 7.1i

La nouvelle version d'ISE [XilinX05], la version 7.1, ajoute une technologie plus innovatrice pour aider à minimiser la logique de développement et le coût de production. ISE 7.1i fournit une exécution plus rapide que n'importe quel autre PLD, il permet une conception à grande vitesse, et plus facile à employer par rapport au logiciel de conception disponible. Elle permet au client de lancer leurs produits sur le marché avant leurs concurrents.

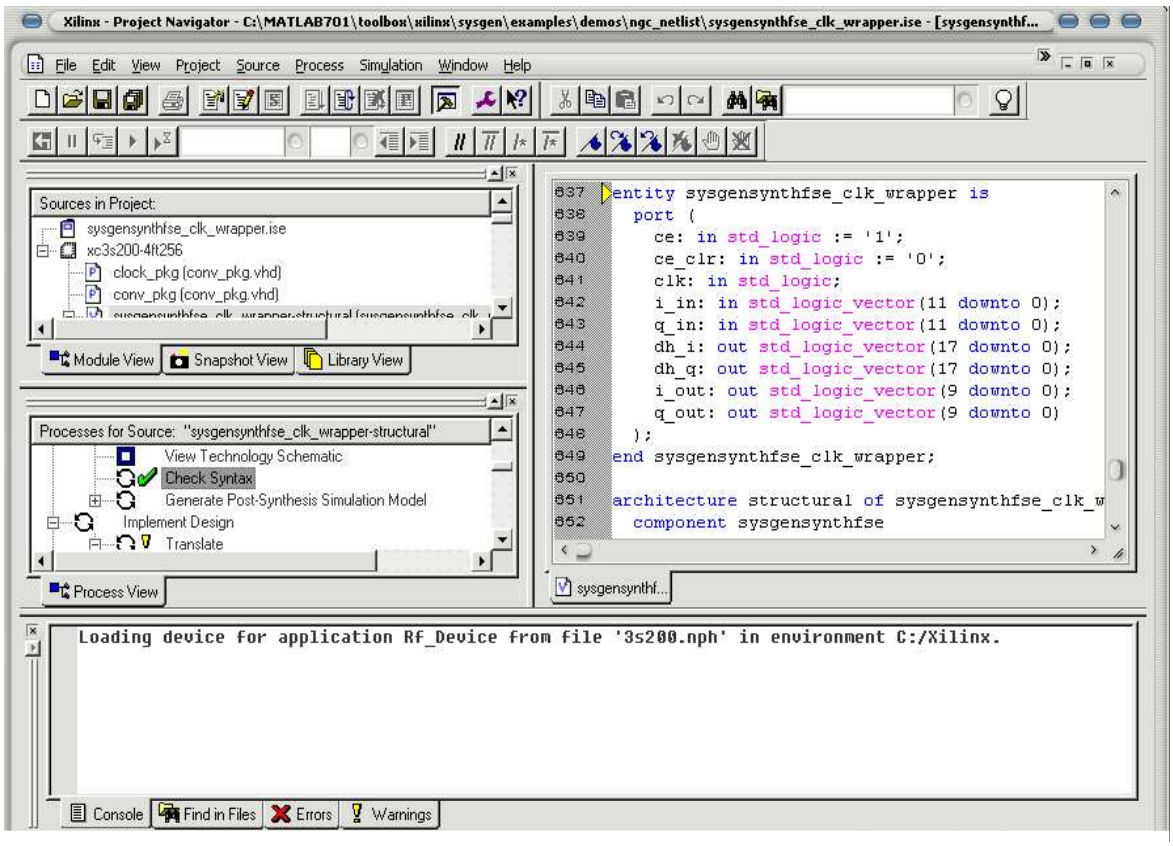

Figure 9 : L'interface ISE 7.1i de XilinX.

La facilité d'utilisation est un grand objectif de XilinX. Avec le dégagement d'ISE 7.1i, XilinX peut maintenant offrir aux clients une solution intégrée de simulation avec le nouveau simulateur d'ISE. Le client de chaque base d'ISE BaseX et d'ISE accède à une version limitée appelée le simulateur « Lite d'ISE ». Les clients de *base d'*ISE peuvent améliorer le simulateur d'ISE.

ISE 7.1 fournit l'appui pour la nouvelle famille de Spartan-3[Xap 05]. Ce dernier fixe une nouvelle norme pour réduire le prix de FPGA. Cette famille d'industrie, présente les dispositifs les moins chers de la deuxième génération sur les technologies de 90nm.

La nouveauté se prolongeant aux FPGAs VirteX-4, ISE 7.1i offre plus de dispositif d'appui additionnel dans ISE *BaseX* et ISE. Ce nouvel appui lui facilite plus encore pour les clients de commencer des conceptions visant FPGAs par des logiques avancées, rendement plus élevé, densité la plus élevée, et une grande capacité de mémoire.

Configurations

Le logiciel XilinX ISE 7.1i offres trois configurations de conception, tous fournies sous le logiciel intégré de l'environnement (ISE) de conception:

Figure 10 : ISE Foundation 7.1i de XilinX.

ISE *Foundation* 7.1i : *La base d'*ISE est l'exécution de conception de la logique programmable la plus complète de l'industrie. Cette configuration d'ISE soutient toutes les familles de logique de « fil » de XilinX et fournit tout requis pour remplir n'importe quelle conception de XilinX, intégrant « seamlessly » avec les produits de la vérification les plus

avancés de l'industrie. XilinX offre également les outils facultatifs de productivité conçus pour collaborer avec la base d'ISE.

Figure 11 : ISE *BaseX* 7.1i de XilinX.

ISE *BaseX* 7.1i : ISE *BaseX* est l'environnement programmable de conception de logique le plus rentable, le plus complet. La configuration d'ISE *BaseX* fournit toutes les possibilités contenues dans ISE WebPACK en plus des outils additionnels. Ils aident les concepteurs à rendre leur conception logique programmable, plus rapidement et minimiser le coût de conception.

Figure 12: ISE *WebPACK* 7.1i de XilinX.

ISE *WebPACK* 7.1i : la configuration d'ISE la plus facile à obtenir, il est sur le Web et libre ! ISE *WebPACK* fournit tout requis pour remplir des conceptions logiques programmables visant tout le principal XilinX CPLDs et FPGAs à basse densité. ISE WebPACK combine l'entrée de HDL, la synthèse et les possibilités de vérification avec les outils de l'exécution les plus puissants de l'industrie. ISE *WebPACK* est disponible pour le Microsoft Windows XP, le Windows 2000, et maintenant Linux 3.

Pour augmenter la productivité nous devons aussi avoir les Logiciels facultatifs qui sont :

- **Chip Scope Pro On-chip Debugging Tools:** les conditions de taille [XilinX05], de vitesse et de conseil de la situation actuelle d'aujourd'hui FPGAs le rendent difficile de corriger des conceptions en utilisant des méthodes traditionnelles d'analyse de logique. Les outils de Chip Scope fournissent un outil puissant et précis pour aider à vérifier et corriger des conceptions de FPGAs, en temps réel et sur option des travaux directement avec des analyseurs de logique d'Agilent pour encore une analyse plus profonde de signal de FPGA. Chip Scope pro permet l'insertion des noyaux low-profile d'analyseur de logique et d'autobus dans des conceptions. Ces noyaux logiques permettent à l'utilisateur de regarder tous signaux et nœuds internes dans un FPGA. Déclencher les conditions et l'installation en temps réel par l'intermédiaire du port de JTAG sans affecter la logique d'utilisateur ou exiger la recompilation de la conception.

- **PlanAhead Hierarchical Floorplanner :** PlanAhead fournit une conception hiérarchique, bloque basée et accroissement méthodologie, permettant à des concepteurs de changer seulement une partie de la conception et de laisser la conception intacte. Les possibilités hiérarchiques de planification de conception de PlanAhead incluent une interface utilisateur graphique avancée (GUI) qui le rend facile à utiliser pour même les concepteurs inexpérimentés. L'affichage intuitif des ressources de dispositif, la hiérarchie de connectivité, logique et physique laisse des concepteurs visualiser et fixer rapidement les problématiques. Les concepteurs peuvent créer et mettre en œuvre l'hiérarchie physique indépendamment de l'hiérarchie logique, et simultanément projeter et analyser les réalisations physiques multiples, maximisant l'exploration de l'espace de conception en identifiant plus rapidement la réalisation optimal.

- **ModelSim XilinX :** ModelSim XE est un environnement complet de simulation du PC HDL qui permet aux concepteurs de vérifier le code source de HDL , les modèles fonctionnels et de synchronisation de leurs conceptions. MXE réalise la simulation de HDL et l'environnement de correction fournissant l'assurance de 100% VHDL et de langue de Verilog.

- **System Generator for DSP** : est l'outil logiciel du ministre de l'industrie pour concevoir, simuler, et mettre en application les systèmes à base FPGA de rendement élevé

de DSP. L'utilisant du System Generator raccourcit considérablement le chemin du concept de construction du matériel en fonction de la simplicité, la flexibilité, la vitesse, la puissance, et la ponctualité.

II.5 Chip Scope Pro 7.1i

Suite à l'augmentation de densité des dispositifs de L' FPGA [Csp05], nous aurons des problèmes pour tester ces dispositifs. Les outils de Chip Scope™ intègrent la clef logique des composants matériel d'analyseur avec la cible XilinX : Virtex™, Virtex-E,Virtex-II, Virtex-II pro, Virtex-4, Spartan™-II, Spartiate-IIE, Spartan-3 et Spartan-3^E dispositifs (including the QPro ™ variants of these families). Les outils de Chip Scope™ communiquent avec ces composants et fournissent au concepteur une logique complète d'analyse.

Nous allons présenter les outils de Chip Scope dans un tableau :

L'OUTIL	Description
ChipScope Pro Core Generator.	Fournit des Netlists et des calibres d'instantiation pour : • Pro (ICÔNE) noyau intégré de contrôleur. • Noyaux intégrés d'analyseur de logique pro (ILA). • Noyau de trace d'Agilent (ILA/ATC). • Analyseur intégré d'autobus pour IBM CoreConnect Noyau périphérique de l'autobus de Sur-Morceau (IBA/OPB). • Analyseur intégré d'autobus pour des gens du pays de processeur de CoreConnect. Noyau de l'autobus (IBA/PLB). • Noyau virtuel de l'entrée-sortie (VIO). • trace d'Agilent de Noyau 2 (ATC2).
ChipScope Pro Core Inserter	Insère automatiquement l'ICÔNE, les noyaux ILA, ILA/ATC, et ATC2 dans la conception synthétisée de l'utilisateur.
ChipScope Pro Analyzer	Fournit la configuration de dispositif, l'installation de déclenchement, et la trace affichage pour l'ILA, ILA/ATC, IBA/OPB, IBA/PLB, et VIO noyaux. Les divers noyaux fournissent le déclenchement, commande, et possibilités de capture de trace. Le noyau ICON communique au pins.
Tcl/JTAG Scripting	Le TCL/ JTAG l'interface scriptable de commande permet d'agir l'un sur l'autre avec des dispositifs dans lachaineJTAG et un TCL shell.

Tableau 1 : Outils de Chip Scope.

L'outil d'analyse Chip Scope soutient les câbles suivants de téléchargement pour la communication entre le PC et les dispositifs de JTAG [Csp05] Figure 13:

• USB de câble de plateforme.

• Câble parallèle IV.

• Câble parallèle III.

• MultiPRO (mode de JTAG seulement)

• MultiLINX™ (mode de JTAG seulement)

• Agilent E5904B Option 500, analyseur de port de trace de FPGA (Agilent E5904B TPA)

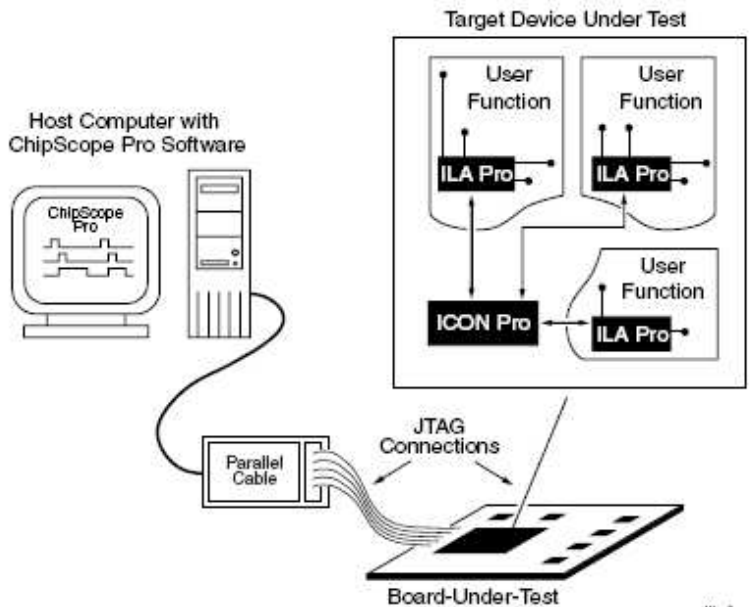

Figure 13 : Connections JTAG.

II.5.1 Flot de conception

Les outils de Chip Scope conçoivent des fusions figure 14 d'écoulement facilement avec n'importe quel FPGA standard. Ils conçoivent aussi l'écoulement qui emploie un outil standard de synthèse de HDL et l'outil d'exécution XilinX ISE 7.1i [Csp05].

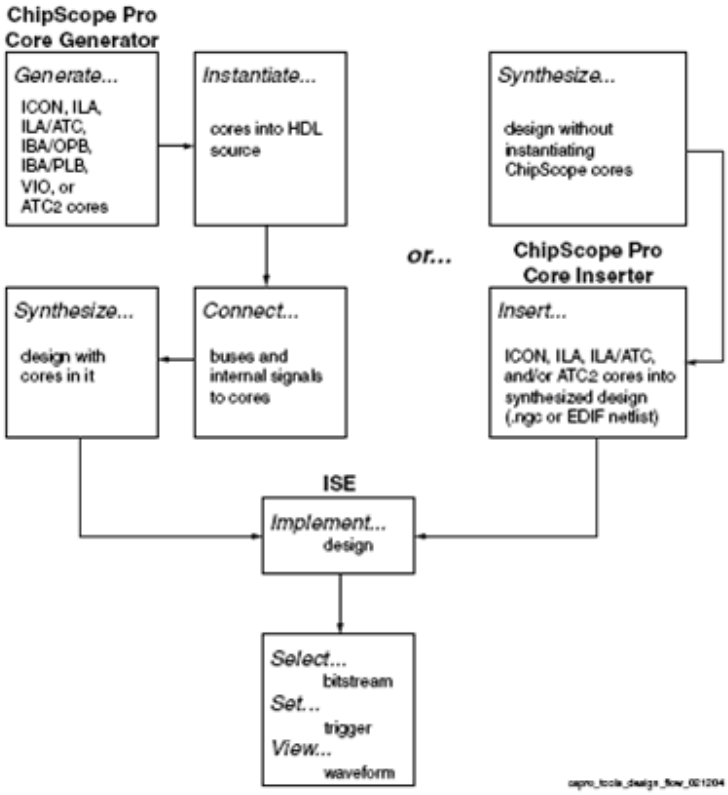

Figure 14 : Flot de conception de Chip Scope.

II.5.2 Conclusion

Cette interface simple fournit l'accès Scripting de TCL aux câbles de téléchargement de XilinX JTAG par l'intermédiaire du Bibliothèque de communication de Chip Scope JTAG. Le but du TCL du JTAG est de fournir un simple système Scripting pour accéder à des fonctions de base de JTAG. Dans quelques lignes de manuscrit de TCL, nous pouvons balayer et manœuvrer la chaîne de JTAG par les câbles standards de XilinX.

II.6 PlanAhead 7.1.10

Les concepteurs de FPGA peuvent relever différents défits d'exécution de multiple selon leur application cible, les buts de projet et ces priorités. La fonctionnalité de PlanAhead peut soutenir des écoulements multiples efficaces pour différents aspects d'exécution de FPGA [Pa05]. PlanAhead emploie les dossiers synthétisés de contrainte de Netlist et de conception pour ses possibilités d'analyse puissantes. De PlanAhead, les utilisateurs peuvent exporter un Netlist d'extension EDIF et concevoir le dossier de la contrainte UCF pour conduire les outils de XilinX P&R. PlanAhead soutient des Netlists supérieurs dans le format d'EDIF ou de NGC. Les Netlists plus bas peuvent être dans le format d'EDIF, de NGC ou de NGO. Si des dossiers de NGC ou de NGO sont employés, l'environnement de placement et routage doit être établi pour que PlanAhead appelle NCG2EDIF d'ISE.

PlanAhead permet de lire les noyaux de conception. L'outil pourra analyser les noms d'UCF qui se dirigent dans les noyaux et fournissent une synchronisation et une utilisation plus précises de ressource. Des contraintes de conception peuvent être incorporées à partir du dossier d'un ou plusieurs UCF ou de NCF (s). Quand des noyaux de format de NGC ou d'O.N.G. sont importés dans PlanAhead, ils sont employés pour la planification seulement. Pour l'exécution, les modules de NGC et d'O.N.G. sont filtrés créant dehors les boîtes noires pour les noyaux. Ceci permet d'employer les dossiers originaux de NGC et d'O.N.G. pendant l'exécution.

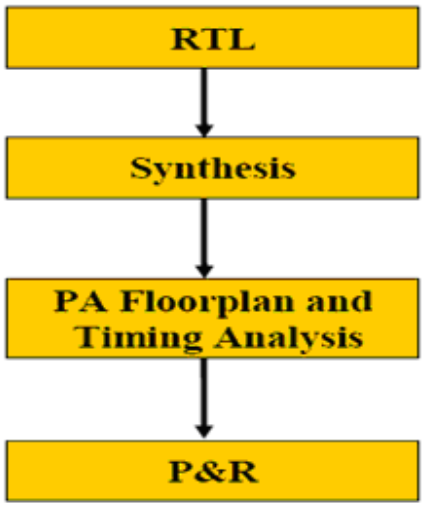

Figure 15 : Flot de conception de Plan Ahead pour les FPGAs.

II.7 Synplify 7.3.4

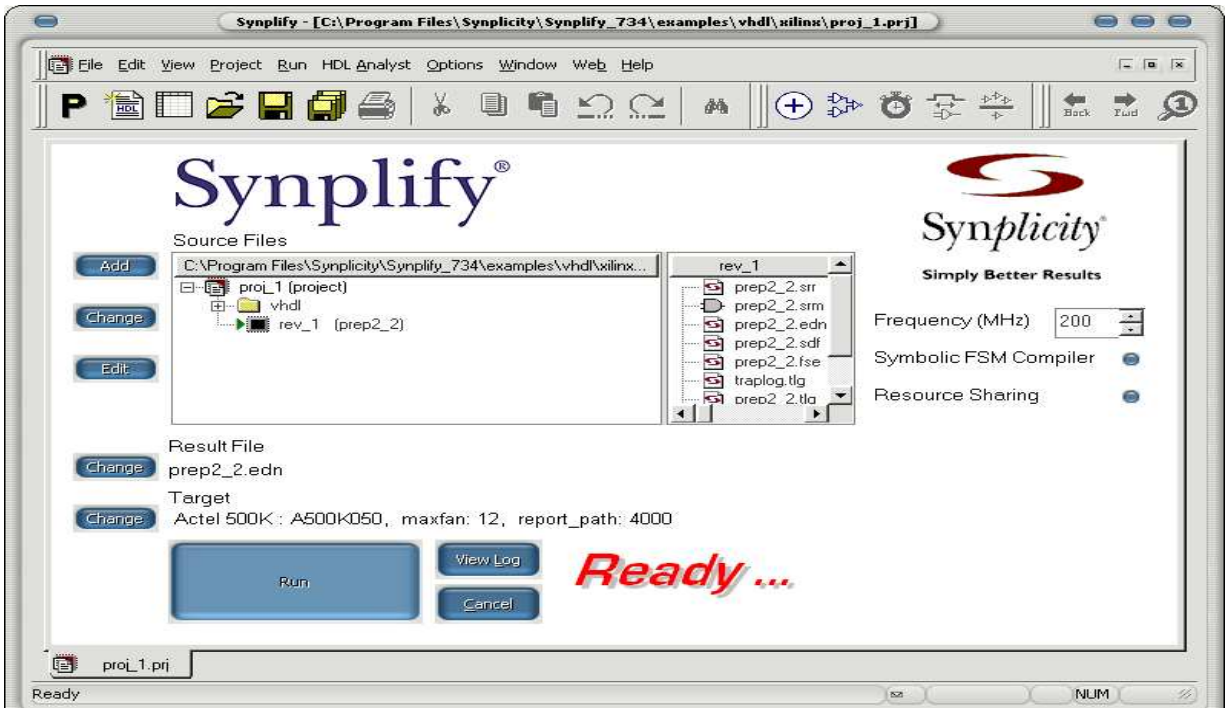

Figure 16 : Interface de Synplify 7.3.4.

Synplify® est un logique synthesistool pour les FPGAs (Field Programmable Gate Arrays), développé par Synplicity ® de Sunnyvale, Californie [Sug 03].

Il a comme entré des conceptions à niveau élevé écrites dans des langages de description de matériel de Verilog et de VHDL (HDLs). Il emplois aussi la classe des propriétaires B.E.S.T (Behavior Extracting Synthesis Technology®), l'outil optimise le code HDL, il a un rendement élevé. Le logiciel peut écrire post-synthesis VHDL de Verilog que nous pouvons employer pour vérifier la fonctionnalité par la simulation.

Le logiciel intègre les dispositifs suivants :

• L'outil de HDL Analyst®, une interface graphique pour l'analyse et crossprobing.

• La fenêtre d'éditeur de texte de Synplify pour écrire et éditer le code de HDL.

• L'interface de SCOPE® (Synthesis Constraint Optimization Environment®), que nous utilisations dans la conception comme le bilan de contrôler pour les contraintes de synchronisation et d'attribut.

• Un compilateur de FSM symbolique, qui exécute la machine avancée d'optimisations.

La figure suivante 17 contient un écoulement générique de conception montrant les étapes typiques du concepteur en prenons comme une application FPGA. La partie ombre montre les étapes que nous pouvons accomplir avec la synthèse de Synplify. Cet écoulement générique de conception complète l'écoulement spécifique de conception utilisé pour le cours d'instruction.

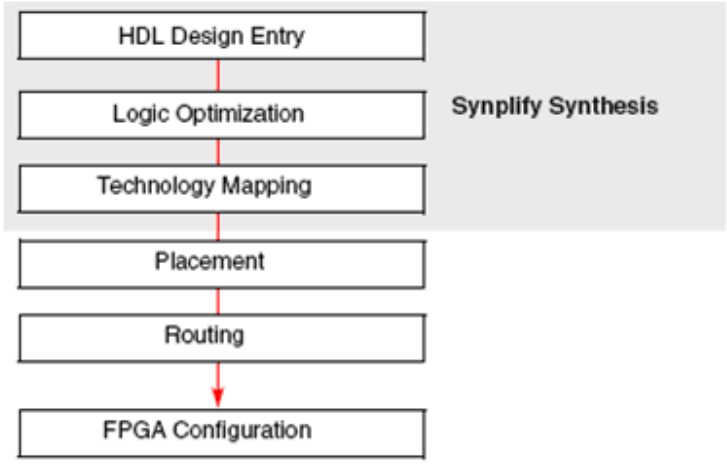

Figure 17 : Les différentes étapes de conception supporter par Synplify.

II.7.1 Entrer de conception VHDL

La logique du circuit de FPGA à mise en application est le point de départ pour la conception de l'FPGAs. Nous pouvons faire ceci en dessinant un schéma, écrivant une description HDL, ou indiquant des expressions booléennes. Pour l'écoulement de Synplify, l'entrée de conception est l'étape où nous produisons de l'entrée pour l'outil. L'entrée doit être des descriptions de Verilog ou de VHDL. Le logiciel nous fournit un environnement où nous pouvons écrire ou éditer des descriptions de HDL.

II.7.2 Logique d'optimisation (Compilation)

C'est le premier étage de la synthèse, dans laquelle le logiciel réorganise le travail original dans un ensemble de fonctions combinatoires [Sug 03]. Dans l'écoulement de Synplify, les fonctions combinatoires sont représentées comme un travail booléen. Ceci nous permet de spécifier dans le processus de conception et de modifier la conception initiale de logique pour optimiser les secteurs ou accélérer la vitesse du circuit final, ou toutes les deux. L'optimisation est calculée à partir du Netlist et est indépendante de la technologie cible. Il inclut des opérations comme le déplacement, la redondance et l'élimination commune des expressions secondaires.

II.7.3 Technologie de traçage

La technologie de traçage est la deuxième phase d'optimisation, où la logique est optimisée à une technologie spécifique. Pendant cette phase, la conception compilée se transforme en circuit des blocs optimisés en tenant compte de la logique de L' FPGA. Selon notre chois de priorités, nous pouvons se concentrer sur l'optimisation de secteur (minimizing the total number of blocks), retarder l'optimisation (minimizing the number of logic block stages in time-critical paths), ou toutes les deux.

L'outil de Synplify emploie des techniques de traçage spécifiques à l'architecture pour tracer les conceptions logiques. Elle a les outils intégrés pour analyser les chemins critiques, « crossprobe » et la vérification au niveau RTL. Le logiciel produit des Netlists dans les formats appropriés, placement et routage, pour les outils qui suivent.

II.7.4 Placement

Le placement est la première étape du processus physique de conception. Pendant cette étape, les blocs logiques sont placés dans une rangée de L'FPGA [Sug 03]. En ce moment, la densité d'interconnexion devient importante.

C'est le point auquel le logiciel de Synplify remet la commande de conception à un autre outil. Cependant si nous avons l'optimisation physique d'amplification, nous pouvons employer les résultats d'un premier passage de placement pour optimiser plus loin notre logique de conception.

II.7.5 Routage

Le routage est l'étape finale du processus physique de conception. A ce stade, nous employons l'outil placement et routage pour relier les blocs placés en tenant conte de la logique de notre carte FPGA et pour le choix des commutateurs programmables.

II.7.6 Configuration FPGA

Dans cette phase de conception, nous configurons le morceau final de FPGA et le mette en application.

II.8 System Generator for DSP

Concernant System Generator, nous devons installer les logiciels que nous avons cites précédemment. Il est mis en place à partir de la commande « run »de MATLAB [Xsg05]. Il permet comme présenter dans la figure 18 de communiquer les différents logiciels constituant la plateforme.

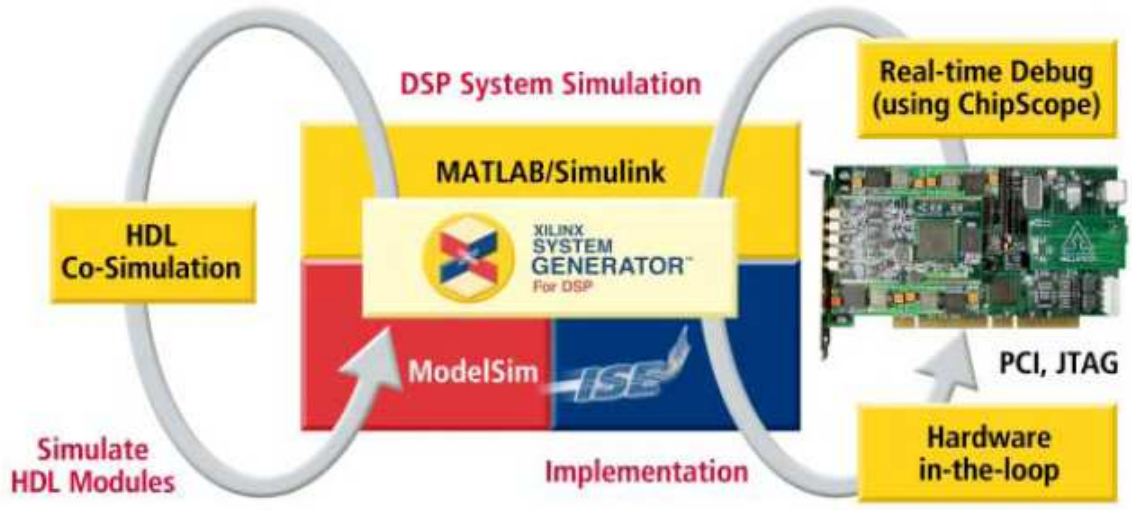

Figure 18 : communication de System Generator avec la plateforme.

Flot de conception du System Generator

System Generator peut être utile dans beaucoup d'arrangements [Xsg05]. Parfois nous voulons explorer un algorithme sans traduire la conception dans le matériel. D'autres fois nous voulons employer une conception de System Generator en tant qu'élément de quelque chose de

plus grand. Une troisième possibilité est qu'une conception de System Generator doit être employée dans le matériel de FPGA. Dans cette section, nous expliquons d'avantages chacune des trois possibilités :

- Exploration d'algorithme : System Generator est particulièrement utile pour l'exploration d'algorithme, le prototypage de conception et l'analyse modèle. L'outil est utilisé en dehors d'un algorithme afin d'observer les problèmes de conception qui sont susceptibles d'être faits face et peut estimer le coût et l'exécution dans le matériel. Un concepteur peut assembler les parties principales de la conception sans s'inquiéter des points fins ou de l'exécution détaillée. Les blocs de Simulink et le code de MATLAB .m fournissent des stimuli pour des simulations et pour des résultats d'analyse.
- Mettre en application une partie d'une plus grande conception : System Generator est souvent utilisé pour mettre en application une partie d'une conception plus grande. Par exemple, le System Generator est un bon arrangement dans lequel nous pouvons mettre en application des circulations et les commandes de données, mais il est bien moins adapté pour les interfaces externes sophistiquées qui ont des conditions strictes de synchronisation. Dans ce cas, il peut être utile de mettre en application des parties de la conception à l'aide du System Generator, met en application d'autres pièces dehors, et les combine alors dans un fonctionnement entier.
- Mettre en application une conception complète : Pour une telle conception, nous appuyons sur le bouton génerate de system Generator il permet la traduction de toute la conception en HDL, d'écrire les dossiers requis pour traiter le HDL à l'aide des outils. Les dossiers écrits incluent ce qui suit : HDL qui met en application la conception elle-même ; Un wrapper d'horloge qui joint la conception. Elle utilise les signaux dont la conception a besoin. Un test bench HDL qui enferme l'emballage d'horloge. il permet à des résultats des simulations de Simulink d'être comparés contre ceux produites par ModelSim.

II.9 Simulation FPGA

II.9.1 Les méthodes de conception

Afin de satisfaire les avancées technologiques actuelles : densité d'intégration de plus en plus élevée et conception de circuits toujours plus complexes, les concepteurs proposent d'utiliser des approches méthodiques pour maîtriser le flot de développement. Une méthodologie peut se considérer comme une « boite à outils » dans laquelle le concepteur trouve une variété d'outils : modèles, solutions, méthodes. Reste au concepteur à trouver pour chaque situation l'outil approprié pour une résolution efficace de son problème.

Nous nous focaliserons sur la présentation de la méthode descendante ***top down***. En effet, cette approche est la méthode la plus populaire parmi les concepteurs de circuits intégrés ASICs et FPGAs. Par ailleurs, c'est une méthode « générique » où les approches de conception viennent s'imbriquer. Par exemple, la synthèse comportementale permet dans le flot ***top down*** de générer automatiquement une architecture.

II.9.2 La méthode descendante « top-down »

Comme nous l'avons vu précédemment, les méthodes de développement des circuits FPGAs et ASICs ont beaucoup profité des récentes avancées de la microélectronique. Ainsi lors de la phase de conception, l'utilisation de langages de description HDL se généralise [Jerr97]. Ces langages présentent un aspect convivial ; ils aident au développement d'un ensemble de couches d'abstraction du circuit et à la division du flot de conception en sous problèmes simplifiés.

Ce type d'approche de conception, qui s'applique aussi bien aux circuits ASICs qu'aux circuits FPGAs, s'effectue selon une méthode hiérarchique descendante, appelée aussi conception **top-down** [Ries99]. L'approche descendante part du système en circuits puis sous circuits et évolue ainsi jusqu'au schéma composé de transistors. La figure 19 représente le flot hiérarchique de cette méthode de conception avec les actions et les modèles associés à chaque niveau d'abstraction.

Les quatre niveaux d'abstraction de la méthode de conception descendante sont présentés dans la figure 19 peuvent être définis comme suit :

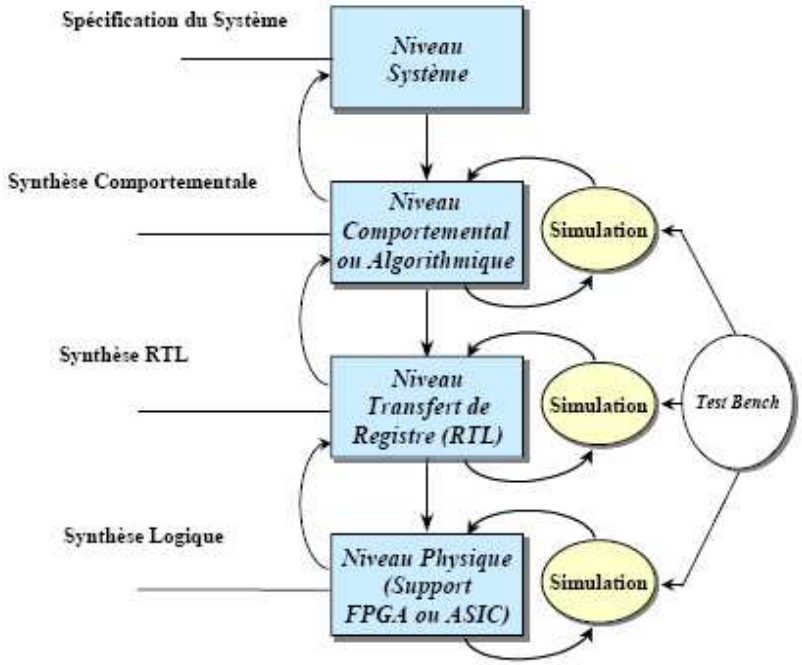

Figure 19 : Schéma hiérarchique de la méthode de conception descendante.

Le niveau système ou spécification système (system level en anglais) : est le niveau d'abstraction le plus élevé. A ce niveau, aucune architecture et aucun séquencèrent des opérations sont définis.
Le niveau comportemental et algorithmique : A ce niveau d'abstraction, le circuit est spécifié en terme de pas de calcul, séparés par des points de synchronisation ou des lectures/écritures des entrées/sorties. Nous parlerons de synthèse comportementale dont l'objectif est de découper ces pas en ensembles de cycles d'horloge pour fournir une architecture synchrone.
Le *niveau transfert de registre*, ou bien RTL : A ce niveau, les opérateurs sont associés aux composants de librairie, les variables aux points mémoires pour obtenir une représentation en transfert de registre. Nous étudierons à ce niveau l'architecture de la fonctionnalité et la synthèse RTL, cette synthèse transforme un circuit spécifié pour chaque cycle d'horloge en un ensemble d'équations booléennes.
Le niveau physique : est le niveau le plus bas, il prend en considération les informations électriques du système, il possède le plus haut degré de précision dans le modèle. Nous parlerons de synthèse logique, qui permettra la configuration du circuit.
Par ailleurs pour compléter le flot de la méthode descendante, il faut ajouter des étapes de validation. Comme nous l'avons présenté sur la Figure 19 précédente le circuit modélisé peut être

validé par mode de simulation à tous les niveaux de description. Cette opération est possible à partir d'un fichier de test que l'on nommera testbench dans le cas de l'utilisation du langage VHDL.

. La synthèse comportementale : La synthèse comportementale est un processus qui transforme un algorithme en une architecture tout en préservant la même fonctionnalité. L'architecture obtenue est donnée sous forme d'une description au niveau transfert de registre RTL et se compose généralement d'un contrôleur et d'un chemin de données [Jerr97] [Cesa99].

. La méthode de conception modulaire : Dans le contexte de développement actuel, où les critères prédominants sont la rapidité de développement et la maîtrise des coûts. La modularité est la méthode la plus fiable à long terme .Elle consiste à dégrossir un problème de conception en le décomposant en une somme de problèmes simples correspondant à des cas plus rudimentaires déjà connus.

. La méthode d'adéquation algorithme architecture : L'adéquation algorithme architecture consiste à étudier simultanément les aspects algorithmiques et architecturaux en prenant en compte leurs interactions, en vue d'effectuer une implantation optimisée de l'algorithme tout en réduisant les temps de développement et les coûts de l'application étudiée Cette méthode est basée sur l'optimisation des graphes flots de données afin d'aboutir à une architecture respectant les contraintes de conception : temps de calcul, surface du circuit, …etc.[Sore01].

II.9.3 Les contraintes actuelles de conception

Bien que la méthode de conception descendante, dite **top-down**, offre une approche de conception générique, elle présente néanmoins certaines limites. En effet, bien que la division du flot de conception en couches d'abstraction ait permis de simplifier l'implantation sur cibles matériel ASICs et FPGAs, la méthode **top-down** est très peu adaptée à l'implantation d'algorithmes complexes comme ceux des commandes des systèmes électriques. Dans ce cas, le passage du niveau comportemental au niveau RTL est très critique puisque, l'architecture, qui détermine les performances du circuit et de la commande, est définie lors de cette transition. Par ailleurs, cette méthode de conception ne prend pas en considération les contraintes inhérentes aux systèmes électriques telles que : l'intégration mixte, l'optimisation de l'architecture (temps de calcul, surface utile du circuit, …), l'intégration conjointe, …etc.

De plus, face à la complexité croissante des systèmes et aux exigences techniques, budgétaires et temporelles toujours plus contraignantes, de nouveaux enjeux apparaissent.

Actuellement, le concepteur se retrouve face à des spécifications qui ne sont pas parfaitement définies au début du cycle de conception (étape de spécification du système) et dont la définition va dépendre des résultats intermédiaires de l'intégration. Ce cas se présente souvent, et en particulier pour les spécifications non techniques : coût global du système, disponibilité des composants, dépendance vis à vis d'un fondeur ou d'un fabricant, durée de vie et évolution du produit, délai de conception, etc.

Aussi, ayant à traiter à la fois des contraintes mal définies ou ayant changé un éventail de solutions techniques très vastes en constante évolution, le concepteur peut être amené à vouloir tester plusieurs stratégies d'intégration après avoir défini un modèle fonctionnel unique du système. Sans outils appropriés, une telle technique n'est pas envisageable. Ainsi, une forme d'automatisation de la conception permettrait d'une part de réduire les délais de conception, mais aussi de traiter des complexités que l'approche manuelle interdit et de diminuer un taux d'erreur dû au facteur humain.

Enfin, il faut ajouter aux contraintes précédemment citées, la problématique de la validation du circuit. Valider le bon fonctionnement d'un circuit est impératif durant tout le long du processus de conception, surtout avant l'envoi en fabrication. Pour s'en convaincre, il suffit de se rappeler le dysfonctionnement du processeur Pentium. Cependant, la simulation n'étant pas en mesure de traiter les fortes complexités, il faut trouver des techniques permettant soit d'apporter la preuve mathématique du bon fonctionnement du circuit, soit d'en émuler le fonctionnement.

Par ailleurs, face à des domaines très contraignants comme celui des systèmes électriques, la validation numérique est insuffisante. Une commande implantée dans un circuit intégré n'est jamais conçue pour fonctionner seule. Elle est toujours au sein d'un système global : capteurs, CANs, onduleur, redresseur, moteur, etc. Ainsi, le fonctionnement d'une commande implantée dans un ASICs ou bien un FPGA sera étudié de façon plus précise si, dans les simulations, le système complet est pris en compte. Le concepteur actuel est alors confronté au problème crucial de la conception d'un système multidisciplinaire pour lequel, il devra effectuer des simulations mixtes (analogique/numérique) du circuit intégré en tenant compte de son environnement effectif.

Pour arriver à satisfaire toutes les contraintes évoquées ci-dessus, il est nécessaire de trouver des méthodes avancées de conception. C'est à cette problématique que nous allons essayer de répondre.

II.10 Conclusion

Nous avons présenté tout au long de ce chapitre les outils et les langages que nous allons adopter, en décrivant le rôle de chacune d'eux et en précisant les résultats qui peuvent être produite pour la simulation.

Comme nous l'avons constaté, ces outils permettent de simuler une seule technologie à la fois. L'objectif est de pouvoir simuler un système composé de plusieurs modules hétérogène. Notre but est l'utilisation d'un environnement unifié pour la simulation.

Dans ce qui suit, nous présentons le mécanisme de communication entre MATLAB et ModelSim et nous terminons par un dernier chapitre pour la Co-Simulation MATLAB / FPGAs tous en présentant les apports de cette plateforme de simulation.

CO-SIMULATION MATLAB / VHDL

Chapitre III: CO-SIMULATION MATLAB / VHDL

III.1 Introduction

Après avoir présenté, dans le chapitre précédent, les outils et les langages de simulation, nous introduisons dans ce chapitre la Co-Simulation MATLAB/ VHDL. En effet ce chapitre présente trois parties : en premier temps, nous commençons par un aperçut sur les travaux antérieures qui se résume à la conversion de code écrit en VHDL vers MATLAB et inversement. Ensuite, nous présentons une vue générale sur la synchronisation et la communication entre ModelSim et MATLAB, en utilisant le mode de communication TCP/IP et nous terminons par un exemple de Co-Simulation afin d'accélérer la simulation MATLAB/VHDL et la rendre automatique.

III.2 VUE D'ENSEMBLE DE BOÎTE À OUTILS DE CONVERSION

Aujourd'hui, beaucoup de concepteurs systèmes utilisent des outils logiciels comme MATLAB pour modeler une conception, mais ils trouvent des grandes difficultés à la conversion entre les deux langages : MATLAB et VHDL [Wvc00]. Cette approche peut permettre à un utilisateur de développer et de simuler un algorithme de commande numérique en utilisant MATLAB. La routine de conversion doit être considérée comme élément d'un large flot de conception et développement système.

La vue d'ensemble du processus de conception et présenter dans la figure 20, il est constitué de trois étapes principale : (1) Système initiale modelé et simulé, (2) Conversion des données et synthèses et (3) exécution du matériel.

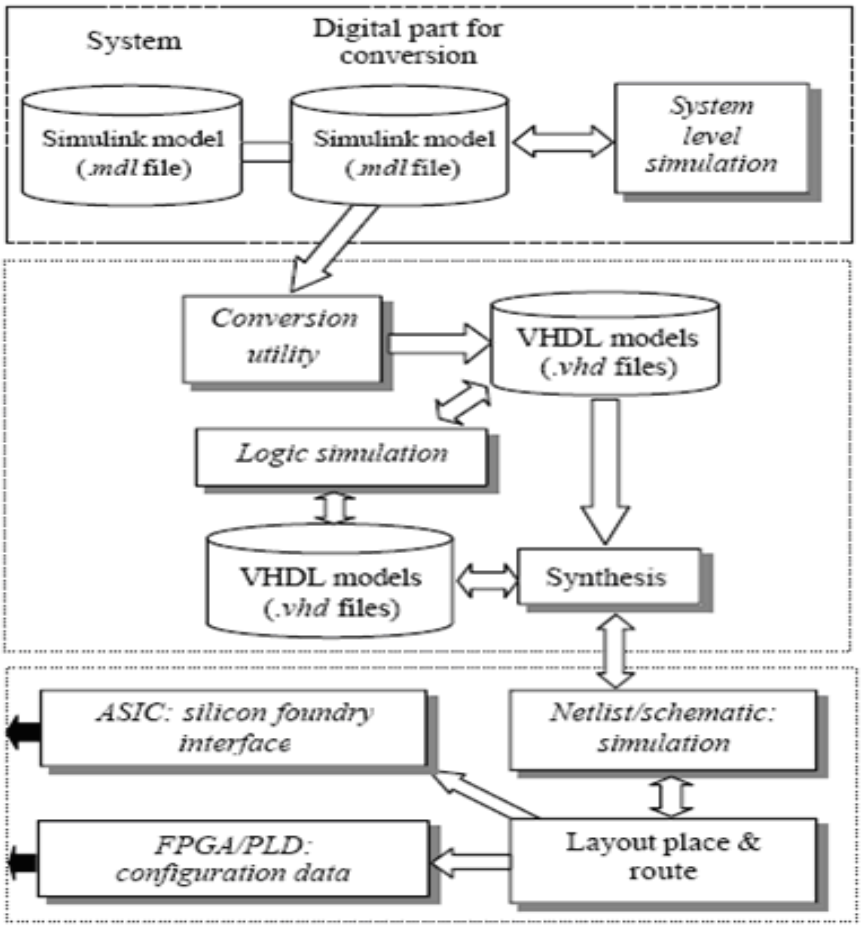

Figure 20 : Routine de conversion dans le processus de conception.

Le programme de conversion peut être actionné directement par des commandes de l'UNIX, ou par l'intermédiaire d'une interface graphique figure 21.

Figure 21 : interface graphique.

Ceci permet à l'utilisateur de spécifier les paramètres de conversion (conversion périodique/parallèle d'entrée-sortie, conditions d'optimisation, type arithmétique interne, l'outil de synthèse cible). Mais la notion du temps et la synchronisation reste un glot d'étranglement pour les concepteurs. Avec la conception étant synchrone en nature, la commande des signaux doivent être produits dans une horloge externe. Elle ne permet pas l'optimisation du code qui exécutera les fonctions exigées tout en prévoyant pour réduire le nombre requis de portes.

III.3 Link for ModelSim

MATLAB Simulink fournit une méthode puissante de développement logique à l'aide d'un outil de conception à haut niveau, il intègre le matériel dans l'environnement de MATLAB Simulink [Ug05]. Traitant des signaux complexe, des conceptions peuvent être développées rapidement en utilisant les schémas fonctionnels de Simulink agissant l'un sur l'autre avec le matériel en temps réel. Les passages entre MATLAB Simulink et le matériel permettent à des données de couler entre le matériel et MATLAB, apportant la puissance de MATLAB à la logique procédée de développement.

Des schémas fonctionnels de Simulink sont directement traduits en logique à l'aide de l'outil de System Generator de XilinX. Pour chaque produit soutenu, une couche d'interface de matériel de composants de Simulink est à condition que permet au matériel d'être employé dans la conception de Simulink. Les diverses bibliothèques composants Simulink ont été fourni par The MathWorks, XilinX et l'interface innovatrice du matériel connecte la couche pour établir la logique d'application sur le produit.

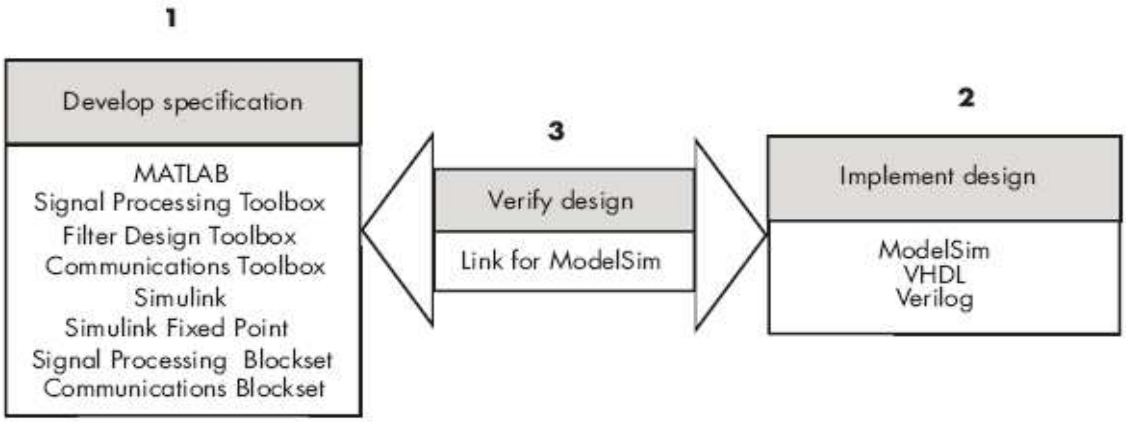

Figure 22 : Le rôle de Link for ModelSim pour communiquer MathWorks et ModelSim.

La figure 22 montre le scénario d'adaptation des produits de ModelSim et de The MathWorks pour de conception matériel. Le link for ModelSim relie les outils qui sont traditionnellement employés discrètement pour accomplir des étapes spécifiques dans le processus de conception.

En reliant les outils, le link for ModelSim simplifie la vérification près en permettant au co-simulate l'exécution et les spécifications originales directement. Il permet aussi l'économie significative de temps de conception et l'élimination d'erreurs inhérentes en comparaison avec l'inspection manuelles. En plus du scénario précédent de conception, le link for ModelSim nous permet d'employer :

• MATLAB ou Simulink pour créer des signaux de testes et des tests bench pour des Code de HDL.

• MATLAB ou Simulink pour fournir un modèle comportemental pour une simulation de HDL.

• MATLAB nous permet la possibilité d'analyser et de visualisation en temps réel d'une exécution de HDL.

• Simulink permet la traduction des descriptions du HDL en des vues au niveau système.

III.4 Environnement de Co-Simulation

Le link for ModelSim est un client/serveur pour le test bench et la Co-Simulation des applications. Le rôle que ModelSim joué dans un lien pour l'environnement de simulation dépend du fait est ce que ModelSim lie à MATLAB ou à Simulink.

Une fois lié avec MATLAB, ModelSim fonctionne comme client, comme la montre la figure 23.

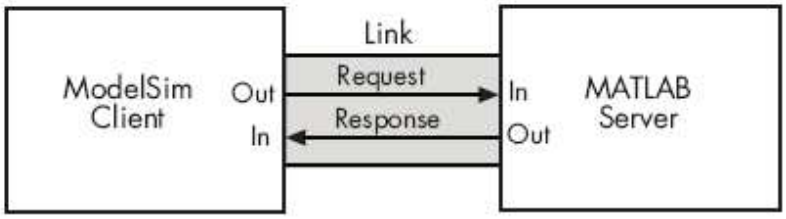

Figure 23 : Mode de communication entre MATLAB et ModelSim.

Dans ce scénario, MATLAB Server lance des fonctions d'attentes jusqu'à qu'il reçoive des « Requests » du ModelSim. Après réception de ces derniers, le serveur établit un lien de communication et appelle des fonctions spécifique de MATLAB qui permet le calcule des données, vérifie, ou visualise des model HDL qui sont sous ModelSim.

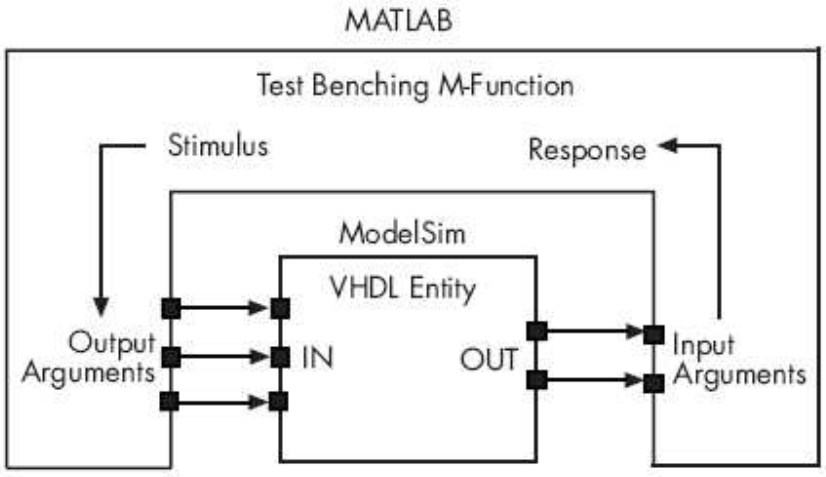

Figure 24 : Principe de communication de MATLAB et ModelSim pour la phase de test.

La figure 24 montre comment une fonction de MATLAB englobe et communique avec ModelSim pendant une session test bench de simulation.

Le serveur MATLAB peut entretenir simultanément des multiples sessions de ModelSim et des entités de HDL. Cependant, nous devrions adhérer aux directives recommandées à assurer le serveur peut dépister l'entrée-sortie liée à chaque entité et session. La figure 25 montre un scénario de multiple-client se reliant au serveur au port 4449 du socket TCP/ IP.

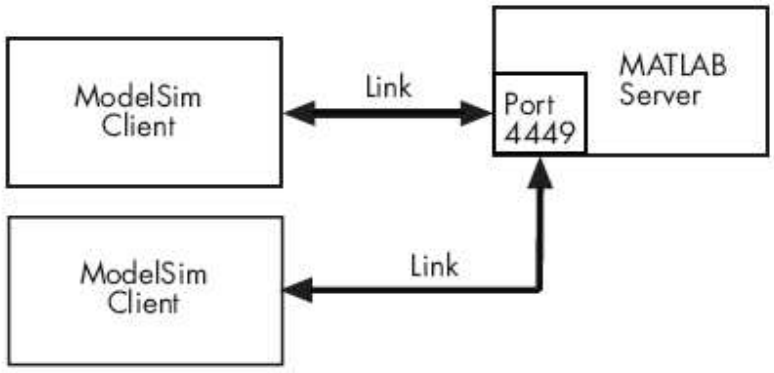

Figure 25 : Multiple-client communique avec MATLAB simultanément.

Le mode de communication que le link for ModelSim emploie pour un lien entre ModelSim et MATLAB ou Simulink dépend légèrement de si notre application de simulation fonctionne dans un aspect local, dans une configuration de simple-système ou dans une configuration réseau. Si ModelSim et les produits de MathWorks peut exécuter localement sur le même système et notre application exige seulement un voie de transmission, nous avons l'option du choix entre partagé mémoire et communication de « sochet TCP/IP. » La communication partagée de mémoire fournit l'exécution optimale et elle est le mode de communication par défaut.

Le mode de « sochet TCP/IP » est plus souple. Nous pouvons l'employer pour le simple-système et la configuration réseau. C'est le choix optimal pour les applications qui ont un potentiel de croissance important.

Pour les configurations dans les quelles ModelSim et les produits de MathWorks résident différents systèmes, chaque système doivent être configurés pour l'Ethernet et c'est nous que nous devons employer la communication « sochet de TCP/IP ».

Une fois lié avec MATLAB, ModelSim fonctionne comme client, Le serveur de MATLAB, par lequel nous commençons fournie une fonction de MATLAB, attend des demandes de raccordement des exemples de ModelSim fonctionnant sur la mêmes ou les différents ordinateurs. Quand le serveur reçoit une demande, il exécute une fonction spécifique de MATLAB qui nous permet d'accomplir des tâches au nom d'une entité dans notre conception VHDL.

III.5 Chois du port TCP/IP

Pour employer la communication du « sochet TCP/IP », nous devons choisir un nombre de « sochet TCP/IP » qui est disponible dans notre environnement de calcul à l'usage du link for ModelSim des composants client et serveur. Les deux composants emploient ce nombre pour établir un raccordement de TCP/IP. Les nombres sont particulièrement importants pour des applications qui interconnectent des multiples clients et serveurs. Les nombres identifient uniquement chaque client et serveur et permettent des raccordements seulement entre les composants partageant le même nombre.

III.6 Exemple de Co-Simulation MATLAB/ VHDL

Concernant la simulation MATLAB/VHDL, nous avons mis en place MATLAB v7.0.1/Simulink v6.1 (R14) Service Pack (R14SP1), nous avons téléchargé ModelSim 6.0 pour la famille XilinX [XilinX05].

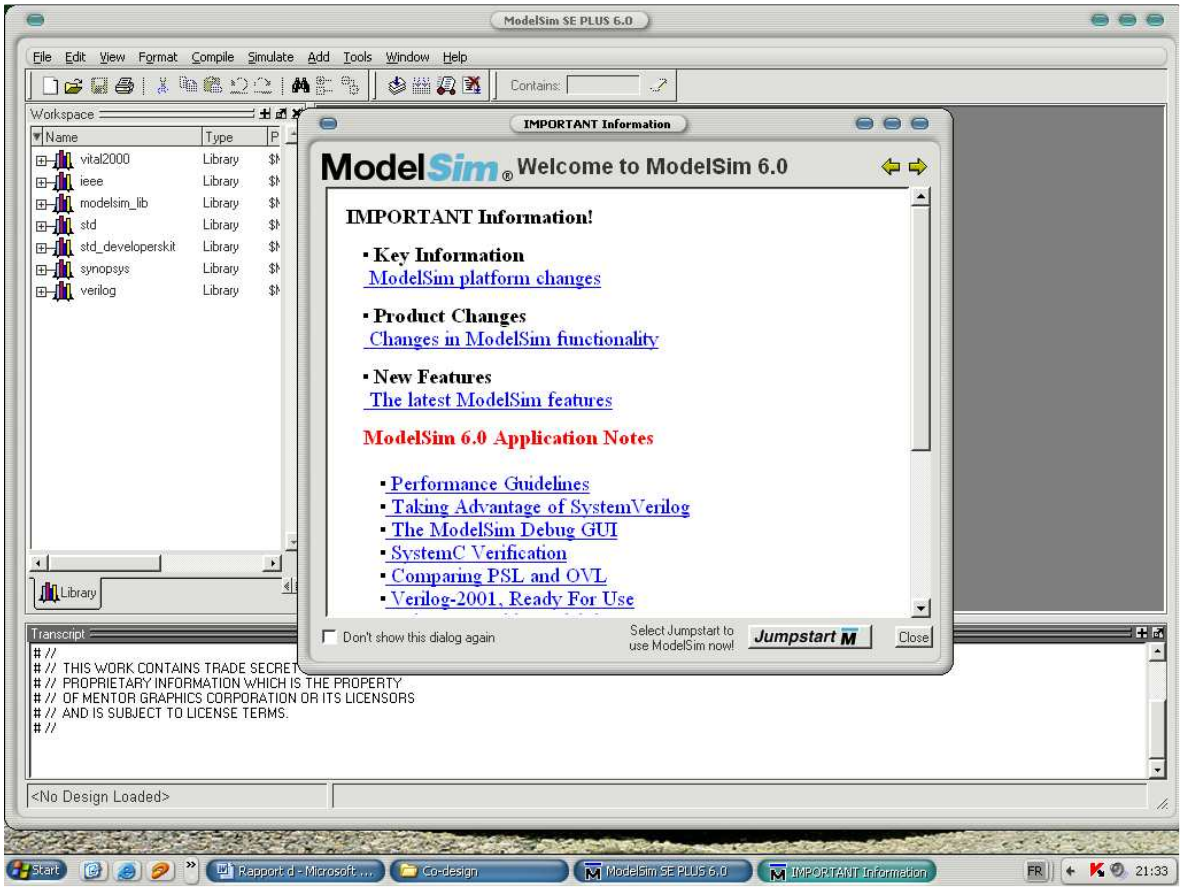

Figure 26 : ModelSim 6.0 de XilinX.

Nous commençons par écrire un petit programme en VHDL qui sera compilé et simulé dans ModelSim, nous lançons MATLAB et nous suivrons les instructions pour la simulation VHDL/MATLAB :

- Setupmodelsim : cette commande nous permet de configurer MATLAB avec la version installer de ModelSim sur le PC, qui dans notre cas ModelSim SE 6.0.
- Nous passons ensuite à MATLAB, nous ouvrons un nouveau projet a partir de Simulink de MATLAB, nous construirons notre circuit en ajoutant tous les blocs nécessaires et surtout le bloc VHDL Co-Simulation Figure 27.

Figure 27 : bloc de Co-Simulation VHDL de MATLAB.

Nous devons configurer tous les champs de ce bloc qui sont :

- **Ports** : qui seront relié au programme écrit en VHDL.
- **Connection** : Nous devons choisir le port de communication entre Simulink et ModelSim.
- **Clocks** : comme nous avons ajouté les ports déjà cité, nous ajoutons le « clock » qui sera relié au programme.
- **TCL** : dans le bloc Pre- simulation command, nous écrivons la commande

echo "Running inverter in Simulink!" dont inverter est le programme écrit en VHDL et dans le bloc Post- simulation command, nous écrivons echo "Done" Figure 28.

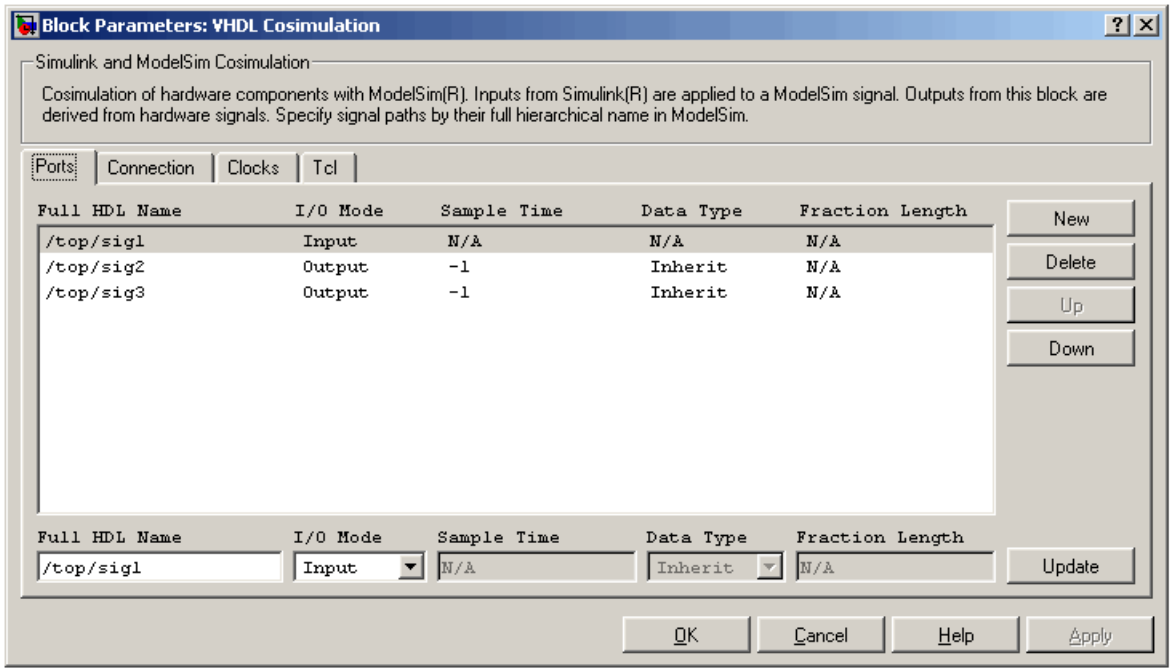

Figure 28 : Bloc de Co-Simulation.

Nous fermons ce bloc et nous passons au Configuration Paramètres de Simulation dont nous devons configurer tous ces champs qui son principalement le délai de la simulation qui est dans notre exemple entre 0.0 et 10 .0 Ms et le type de « Solver » qui peut être « Variable-step » ou « Fixed-step », nous terminons par OK pour conserver le changement figure 29.

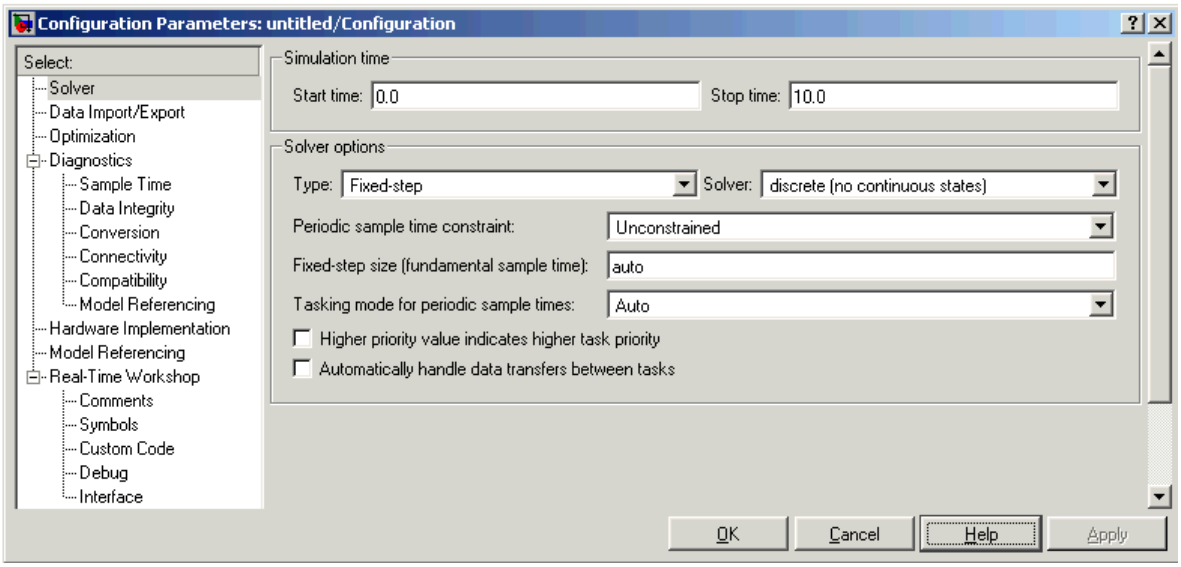

Figure 29 : Bloc de Configuration Paramètres de Simulation.

Nous Avons maintenant une représentation de VHDL d'un inverseur et d'un modèle de Simulink qui s'applique à l'inverseur.

Pour lancer ModelSim, nous devons écrire : vsim ('socketsimulink', 4449) cette commande permet de lancer ModelSim à partir de MATLAB.

Nous passons maintenant à travailler sur ModelSim, nous commençons par changer la direction de notre fichier d'extension VHDL en écrivant la commande :
ModelSim> cd C:/MyPlayArea, nous passons ensuite a simuler notre exemple par écrire la commande : ModelSim> vsimulink work.inverter. Enfin en écrivant la commande VSIM nn> add wave /inverter/, nous aurons l'ajout des ports et du clock dans le WAVE
Figure 28.

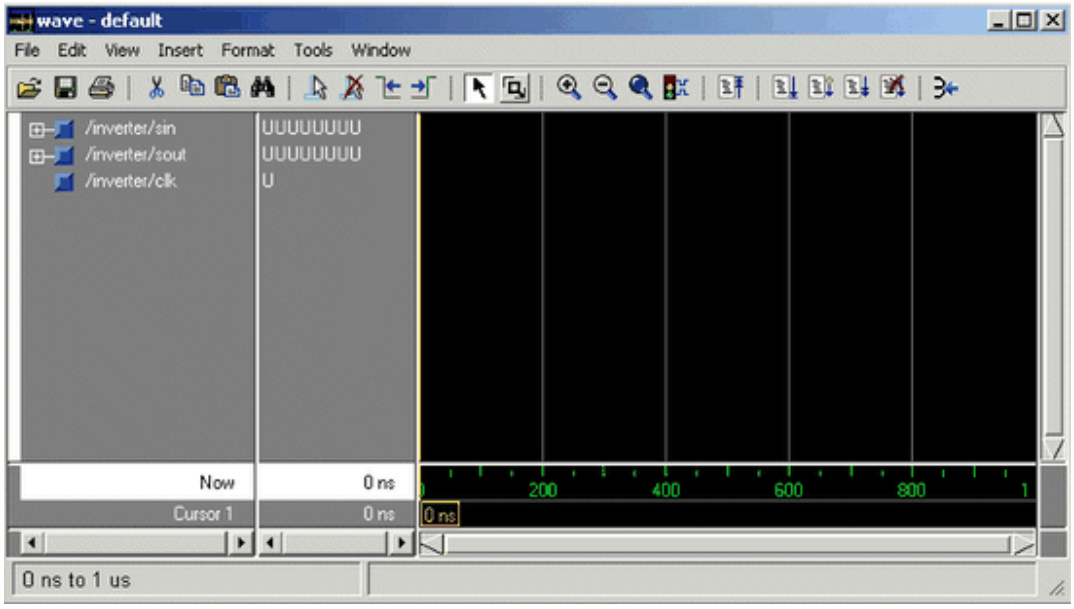

Figure 30 : Affichage des pots de l'algorithme a simulé dans ModelSim.

Nous retournons une autre fois au MATLAB et exactement a notre exemple et nous cliquons sur « START » simulation, nous obtiendrons les résultats sur ModelSim figure 30.

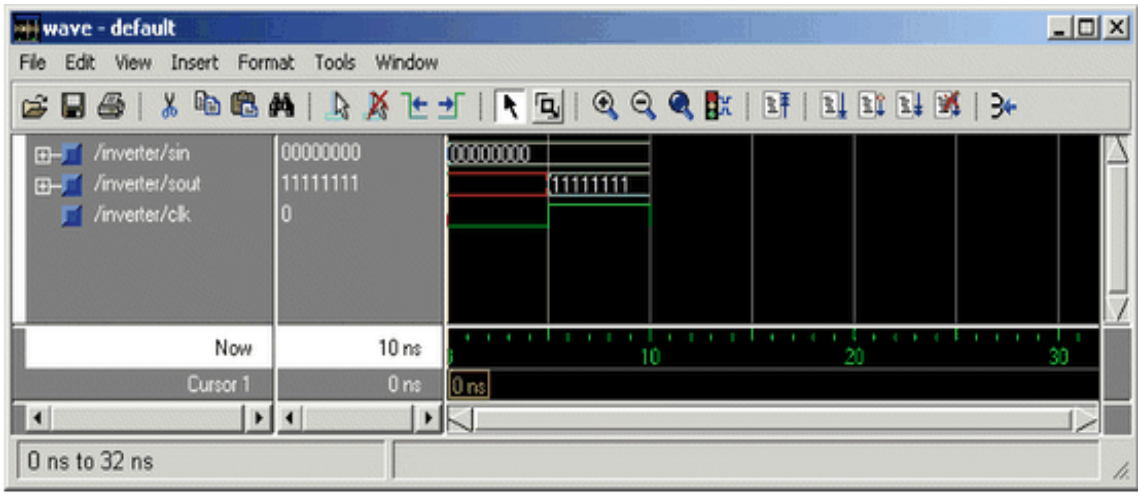

Figure 31 : Affichage des résultats sur ModelSim.

Avec cet exemple, nous avons montré toutes les étapes à suivre pour assurer la Co-Simulation MATLAB/VHDL.

III.7 Conclusion

Link for ModelSim est l'interface de Co-Simulation crée par The MathWorks pour la conception des ASICs et les FPGAs. Cette interface permet une communication entre ModelSim et le produit MATLAB de MathWorks et Simulink. En outre, une bibliothèque des blocs de Simulink est disponible pour inclure des conceptions de ModelSim HDL dans des modèles de Simulink pour le Co-Simulation. Dans ce chapitre, nous avons assuré la synchronisation et la communication entre MATLAB et ModelSim à fin d'accélérer la Co-Simulation MATLAB/VHDL. Mais cette interface n'est pas suffisante pour la Co-Simulation MATLAB/FPGAs.

Dans le dernier chapitre, nous entamons la Co-Simulation MATLAB/ FPGA tout en précisant le rôle de chaque langage et outil, qui constituent la plateforme, et les fichiers qui vont être produit par System Generator qui est crée par The Math Works.

CO-SIMULATION MATLAB / FPGA

Chapitre IV: CO-SIMULATION MATLAB / FPGA

IV.1 Introduction

Nous nous sommes intéressés dans le chapitre précédant au mode de communication et synchronisation de MATLAB et ModelSim en utilisant l'interface produit par The MathWorks qui est Lin for ModelSim. Mais cette interface n'est pas suffisante pour entamer le terme Co-Simulation MATLAB/ FPGAs. Dans ce contexte, nous allons voire le reste des langages et outils présenté dans le chapitre 3, qui vont nous permettre de résoudre ce problème et attendre notre objectif qui est l'accélération de la Co-Simulation et réduire le taux d'erreurs.

Dans ce chapitre, nous commençons par la présentation des nouveautés de MATLAB, quelque caractéristique des FPGAs, ensuite nous étudions le mode de communication des langages constituant la plateforme et surtout System Generator. Nous terminons ce chapitre par un exemple illustratif de toute l'approche de Co-Simulation.

IV.2 Interface MATLAB/FPGA

MATLAB est un logiciel de calcul numérique produit par MathWorks (le site web http://www.mathworks.com/) [Edm 04]. Il est disponible sur plusieurs plateformes. Il est un langage simple et très efficace, optimisé pour le traitement des matrices, d'où son nom. Pour le calcul numérique, il est beaucoup plus concis que les "vieux" langages (C, Pascal, Fortran, Basic). MATLAB contient également une interface graphique puissante, ainsi qu'une grande variété d'algorithmes scientifiques.

Nous pouvons enrichir MATLAB en ajoutant des "boîtes à outils" (toolbox) qui sont des ensembles de fonctions supplémentaires, profilées pour des applications particulières (traitement de signaux, analyses statistiques, optimisation, etc.).

Si les langages sont identiques pour les ASICs et les FPGAs, les outils de synthèse et de placement routage sont différents. En effet, lors de la conception sur des cibles FPGAs, le concepteur sera amené à utiliser des outils propres aux fabricants : XilinX, Altera, Actel, etc, qui est de ce fait adaptés à leurs composants [Gros97] [Alte02].

Par ailleurs, les avantages des FPGAs par rapport aux ASICs sont directement liés à leurs architectures programmables. En effet, ces composants sont constitués d'une matrice de cellules

préconçues dont les interconnexions et les cellules de base sont programmables. Cette disponibilité du matériel dans la puce permet au concepteur de s'affranchir de certaines contraintes de conception telle que la définition d'un arbre d'horloge dans le cas d'un circuit synchrone, du padring…*etc*. Cette spécificité architecturale permet un gain considérable en temps de conception ainsi qu'une plus grande souplesse et facilité de développement. Pour toutes ces raisons les concepteurs d'ASICs utilisent souvent les FPGAs comme cible de prototypage.

Par ailleurs, nous présenterons dans ce qui suit les trois principaux types de composant FPGA existant sur le marché [Yk 02] :

- Les FPGAs à forte densité d'intégration : La force est de constater que les évolutions des FPGAs durant ces dix dernières années permettent désormais de disposer de composants de plusieurs milliers de cellules logiques avec des possibilités de mémorisation de plus en plus importantes. Nous pensons par exemple aux composants tels que les APEXII d'Altera qui offrent une densité allant de 16.640 à 67.200 éléments logiques et une capacité mémoire entre 416 Kbits et 1.1 Mbits [Alte02].

- Les FPGAs avec cœur de processeur : La tendance actuelle montre l'émergence de nouvelles technologies FPGA qui associent les performances d'un processeur et celle d'une cible matérielle programmable. Il offre ainsi, la possibilité d'une programmation haute niveau, et une optimisation de l'intégration pour les parties implantées dans l'architecture FPGA classique. Ces circuits sont destinés au prototypage ASIC ainsi qu'à un volume de production faible [Pana02]. A titre d'exemple, nous citons les composants VirtexII de XilinX.

Par ailleurs, l'apparition récente de nouveaux logiciels de conception et de nouvelles familles de FPGAs, que nous associons aux IPs (*Intellectual Properties*) de DSPs ainsi qu'à *MATLAB/*Simulink, permet de fournir un flot de conception semblable à celui des DSPs. [Pana02].

- Les FPGAs reconfigurables dynamiquement : L'apparition dans les récentes années des FPGAs reprogrammables dynamiquement ouvre de nouvelles perspectives dans

différents domaines d'applications tels que le traitement du signal, les télécommunications, les systèmes électriques, etc. Ces composants permettent une reconfiguration partielle ou complète du circuit pendant son fonctionnement. Cette caractéristique permet dans certains cas critique, comme par exemple la phase de démarrage d'un moteur, d'exécuter des programmes spécifiques.

IV.3 Mode de communication de la plateforme

Suite a une inscription dans le cite de XilinX, nous avons pu avoir :

- Product ID
- Registration ID pour les version (6.3i, 7.1i et 8.1i)

Nous avons téléchargé et mise en place, tous d'abords, ISE 7.1i de XilinX figure 32.

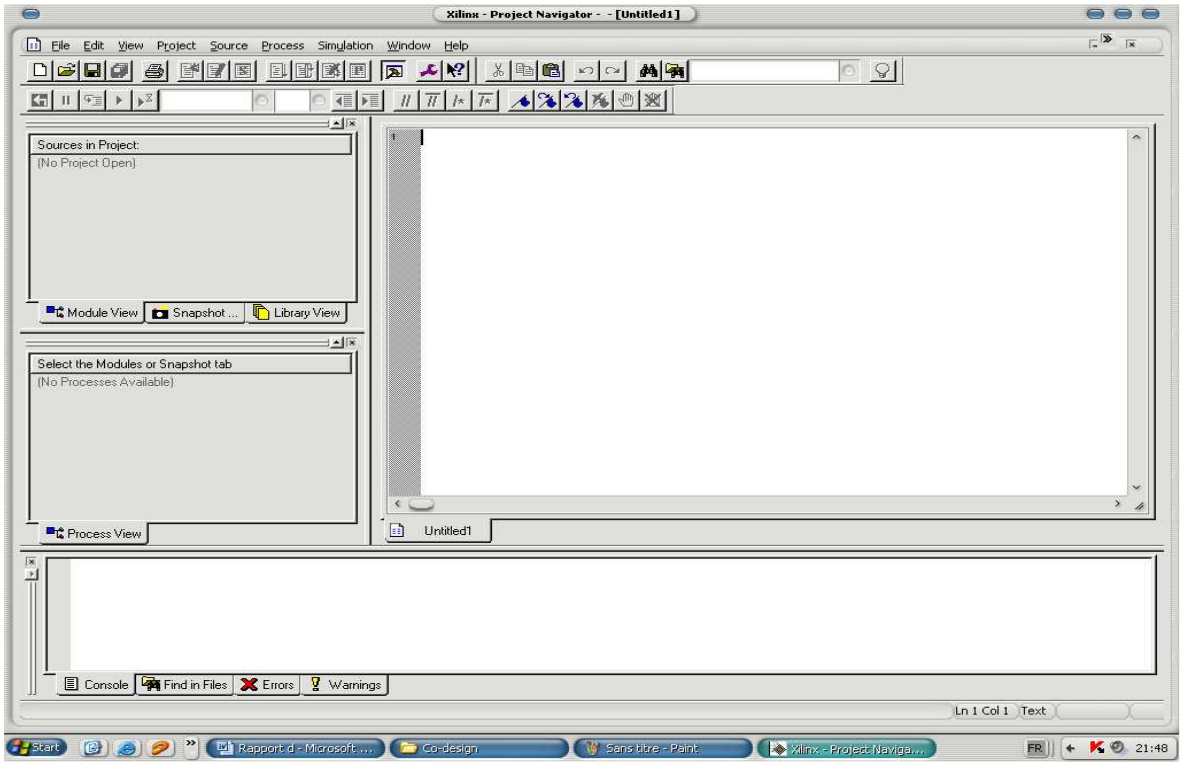

Figure 32 : ISE 7.1i de XilinX.

Le chipScop pro 7.1i, PlanAhead 7.1i et des bibliothèques pour la mise à jour de ModelSim qui contiennent des bibliothèques de XilinX.

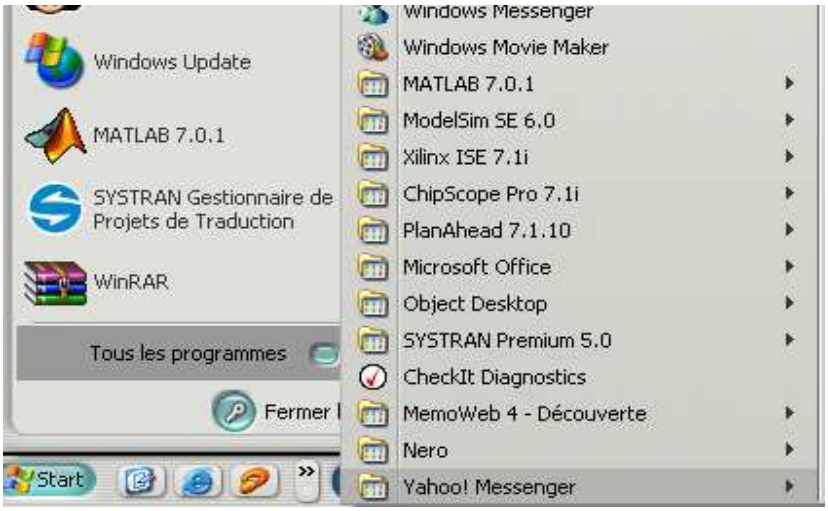

Figure 33 : Plateforme installé.

Et pour installer System Generator, nous devons télécharger la mise à jour de l'ISE 7.1i de XilinX et la version 7.0 de System Generator figure 34.

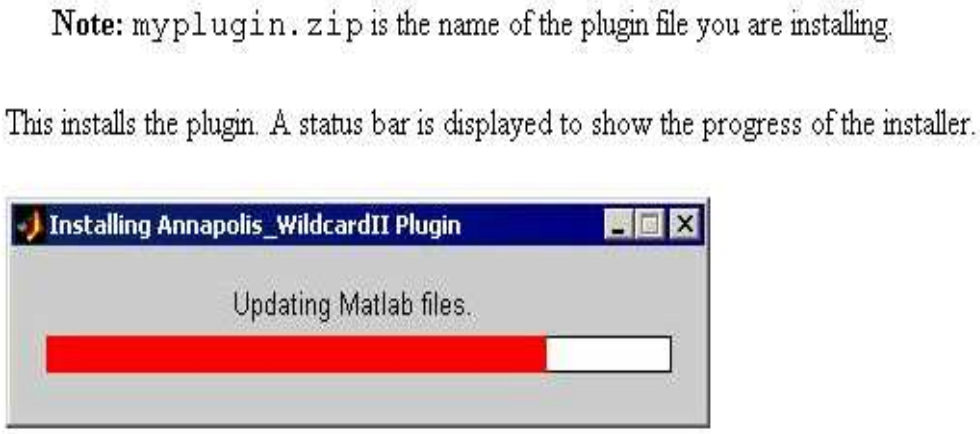

Figure 34 : Installation de System Generator.

Nous aurons l'ajout des bibliothèques de XilinX dans Simulink de MATLAB qui sont :

- XilinX Blockset.
- XilinX Reference Blockset.
- XilinX XtremeDSP Kit.

Celles-ci contiennent des blocs pour la Co-Simulation du matériel.

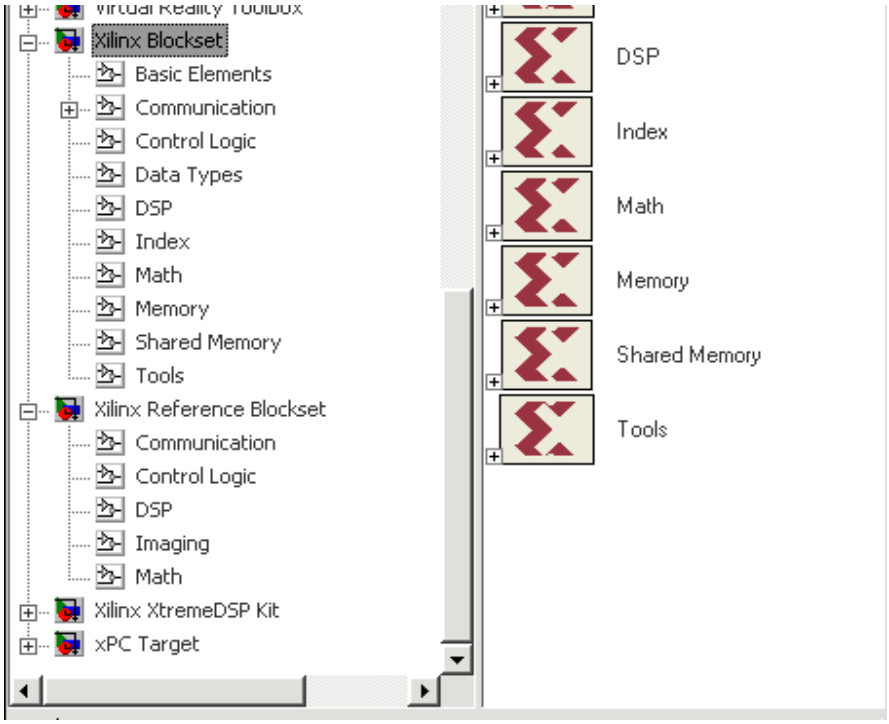

Figure 35 : Ajout des bibliothèques de XilinX dans le Simulink de MATLAB.

Nous citons dans cette partie le fonctionnement des différents logiciels composant la plateforme [XDSP05] :

IV.3.1 Model de conception système

Le model de conception système comporte essentiellement :

- Simulink : Simulink (du MathWorks) est une plateforme pour multi domaine de simulation et de conception des systèmes dynamiques. Il fournit un environnement graphique interactif et un ensemble de bibliothèques de bloc qui nous laissent exactement modeler et simuler le traitement des signaux, de communications, et d'autres systèmes à temps variables.
- Platform Studio : La Platform Studio (de XilinX) est un environnement qui intègre le développement contenant une grande variété d'outils de conception, d'IP, de

bibliothèques et de générateurs incorporés de conception pour accélérer et faciliter la création de notre plateforme.

IV.3.2 Algorithme de développement

Il comporte essentiellement :

- MATLAB : MATLAB (du MathWorks) est une langue de calcul technique à niveau élevé et un environnement interactif pour le développement d'algorithme, la visualisation de données, l'analyse de données, et le calcul numérique. En utilisant MATLAB, nous pouvons résoudre des problèmes de calcul techniques plus rapidement qu'avec des langages de programmation traditionnels, tels que C, C++.

- Accelchip : Accelchip DSP fournit un lien direct entre MATLAB et System Generator de XilinX ou le logiciel d'ISE. Il fournit un environnement unifié de conception qui produit automatiquement des modèles synthétisable RTL et des testbench dans MATLAB.

IV.3.3 Simulation et génération VHDL

Il comporte essentiellement :

- ISE 7.1i : ISE 7.1i est le logiciel de base (de XilinX), il nous permet de programmer essentiellement les FPGAs. Les concepteurs de matériel peuvent concevoir en utilisant VHDL ou Verilog. À l'aide du System Generator, des outils de conception d'ISE peuvent être appelés.

- Synthesis : XST de XilinX et Synplify pro de Synplicity sont des outils de synthèse qui permettent de concevoir peu coûteux et très efficace du matériel de XilinX.

- ModelSim : Si nous avons déjà des produits déjà prêts de HDL, System Generator fournit les interfaces nécessaires pour nous permettre de connecter au ModelSim. Nous pouvons Co-simuler notre HDL en utilisant ModelSim et importer des résultats simulés vers le Simulink/la simulation System Generator en temps réel.

IV.3.4 Verification

Il comporte essentiellement de :

- Chip Scope Pro : Il permet de vérifier les conceptions des FPGAs pour expédier l'étape de correction. Des sondes de Chip Scope peuvent être insérées dans Simulink/ System Generator. Elles sont automatiquement insérées dans le matériel pendant l'étape de génération de HDL.

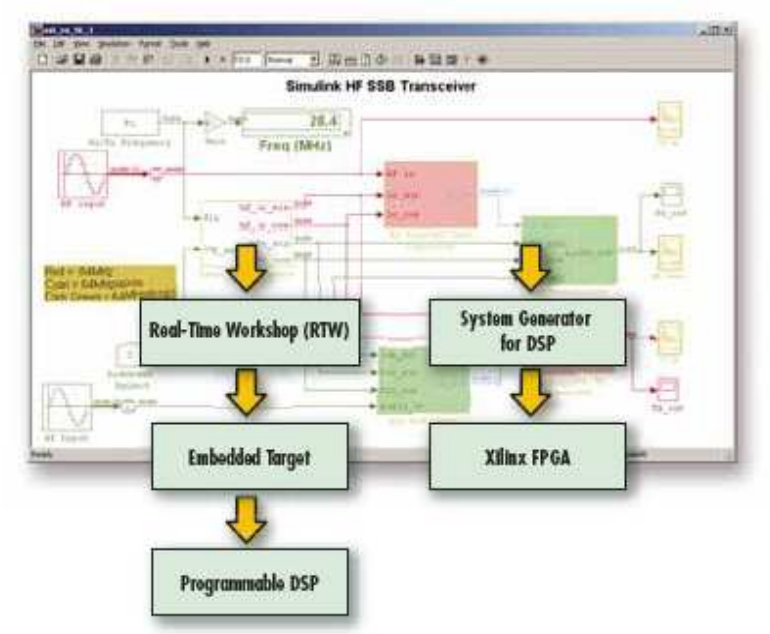

Figure 36 : Exemple de Co-Simulation MATLAB/ FPGAs.

MILPITAS, CA, 9 janvier 2006 [Aem05], les fournisseurs de l'industrie qui sont principalement DSP semi-conducteur de l'IP et « algorithmic synthesis » software pour la conception modèle-basée, ont annoncé aujourd'hui la disponibilité dans sa version 2006.1 des outils Accelchip® DSP Synthesis and AccelWare® IP toolkits figure 36. La chose nouvelle dans 2006 est M2C-Accelerator™, une option à la synthèse d'Accelchip DSP qui prolonge la solution modèle-basée de la conception en ajoutant la génération automatique de C++ dans des modèles de vérification de MATLAB. Avant le M2C-Accélérateur, des compagnies conçoivent des algorithmes dans MATLAB. Pour la conversion, elle était manuelle des modèles de MATLAB en C. Maintenant, ce processus est rendues automatiques, rapides et sans erreur avec le M2C-Accélérateur. Les équipes de conception peuvent maintenant développer des algorithmes plus

rapidement et explorer une gamme des solutions architecturales dans moins de temps. Les modèles de C++ produits par le M2C-Accélérateur peuvent être employés dans MATLAB®, Simulink®, System Generator de XilinX et environnements autonomes de vérification de C.

Les clients de M2C-Accélérateur du d'Accelchip travaillent sur des algorithmes pour des applications telles que 802.11 et des satellites de positionnement globaux (GPS) , ils ont amélioré des exécutions de vérification jusqu'à 1000X en utilisant le M2C-Accélérateur dans leurs suites C-basées de vérification et jusqu'à 150X dans des simulations de MATLAB une fois comparée aux temps d'exécution à point fixe courants de MATLAB.

Le M2C-Accélérateur nous permet ainsi d'accélérer le processus de conception à point fixe, plus d'itérations de conception par jour dans notre choix d'environnement Modèle-Basé de conception et de réduire le temps de mise sur le marche (time –to-market).

IV.4 Expérimentations et résultats

Nous avons pris un exemple de LMS-based adaptive equalization (Synthesizable RTL implementation using M-code Block), Cette conception montre comment employer le bloc de M-Code pour créer des conceptions entièrement synthétisable du niveau de transfert de registre (RTL) dans System Generator. La conception a été dérivée directement de la conception de démonstration de sysgenFSE.mdl, remplaçant des blocs de non-RTL par des blocs de M Code équivalents. Les cibles de RTL sont particulièrement utiles quand la source simple doit viser différentes familles de FPGA. Cette conception montre qu'a T/2 l'adaptive Fractionally Spaced Equalizer (FSE) fonctionnant sur un point d'émission 16-QAM avec le bruit et le filtre présenté dans le modèle de canal ullistré dans la figure 37.

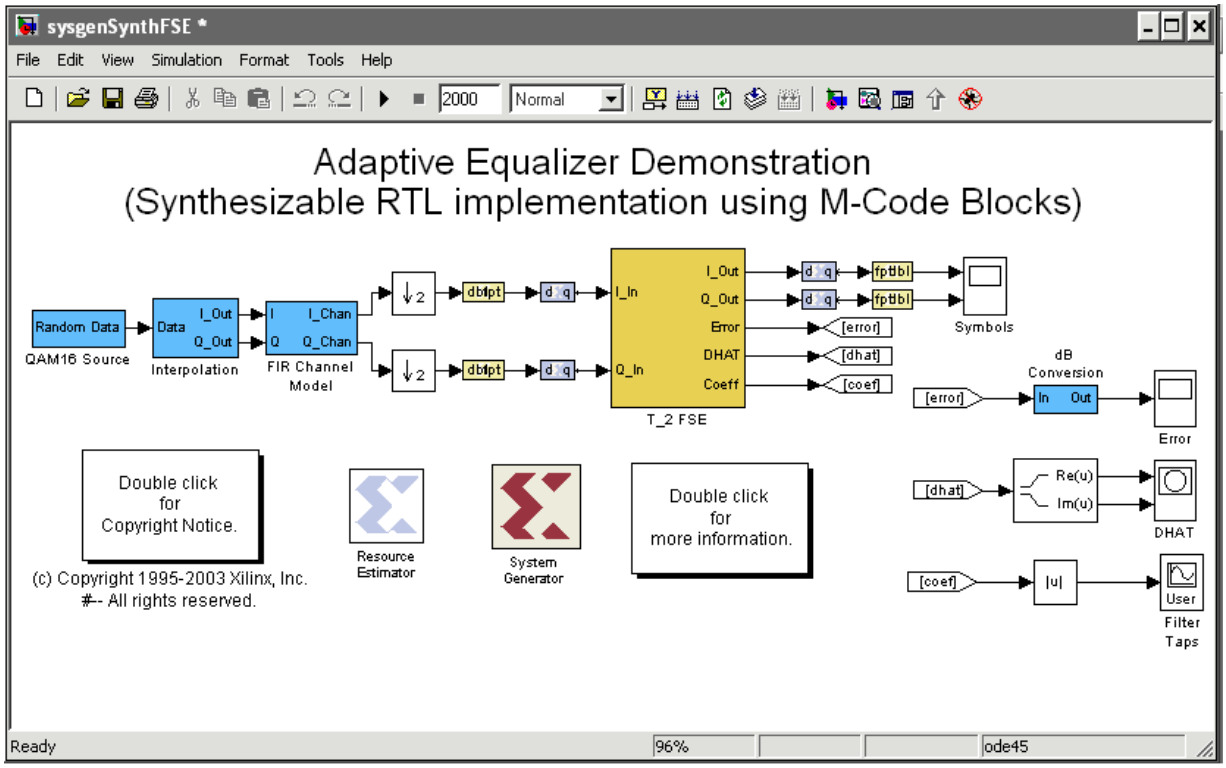

Figure 37 : Notre exemple pour tester la plateforme.

Lorsque nous cliquons sur le bloc de System Generator de notre exemple, nous aurons l'ouverture d'une fenêtre de dialogue dans la quelle nous devons configurer tous les champs.

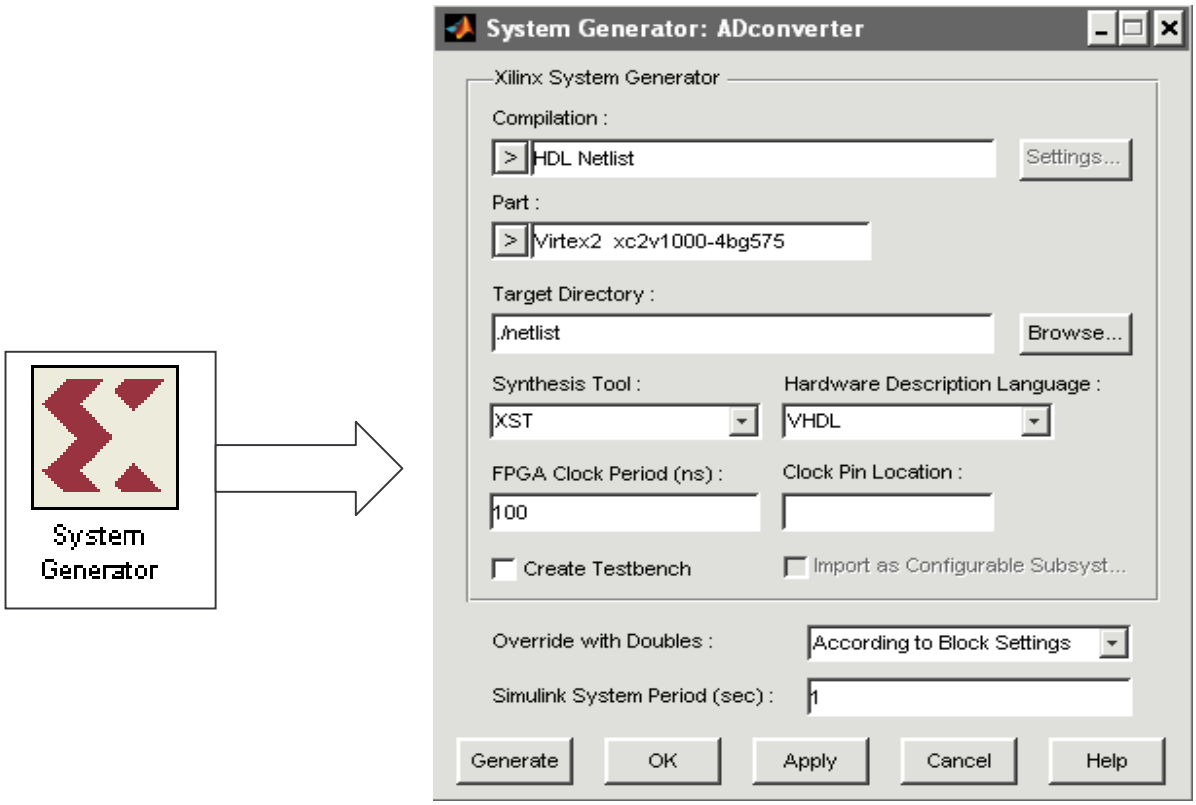

Figure 38 : Configuration du bloc de System Generator.

Nous citons dans ce qui suit tous les champs du bloc de System Generator figure 38 :

- **Part** : Définit l' FPGA à employer.
- **Target Directory** : Définit où le System Generator devrait écrire des résultats de compilation.
- **Synthesis Tool** : Indique l'outil à employer pour synthétiser la conception.
- **Hardware Description Langage** : Indique la langue à employer pour le Netlist de HDL de la conception. Les possibilités sont VHDL et Verilog.
- **FPGA Clock Period** : Définit la période en nanosecondes de l'horloge du matériel.
- **Clock Pin Location** : Définit l'endroit des pins pour l'horloge du matériel.

- **Create Testbench** : Ceci demande au System Generator de créer un testbench de HDL.
- **Import as Configurable Subsystem** : System Generator est sensé de faire deux choses : 1) Construisez un bloc auquel les résultats de la compilation sont associés et 2) la construction d'un bloc qui se compose de sous-ensemble configurable.

Le bloc d'estimateur de ressource de XilinX fournit des évaluations rapides des ressources de l' FPGA exigées et ceci pour mettre en application un sous-système ou un modèle de système figure 39.

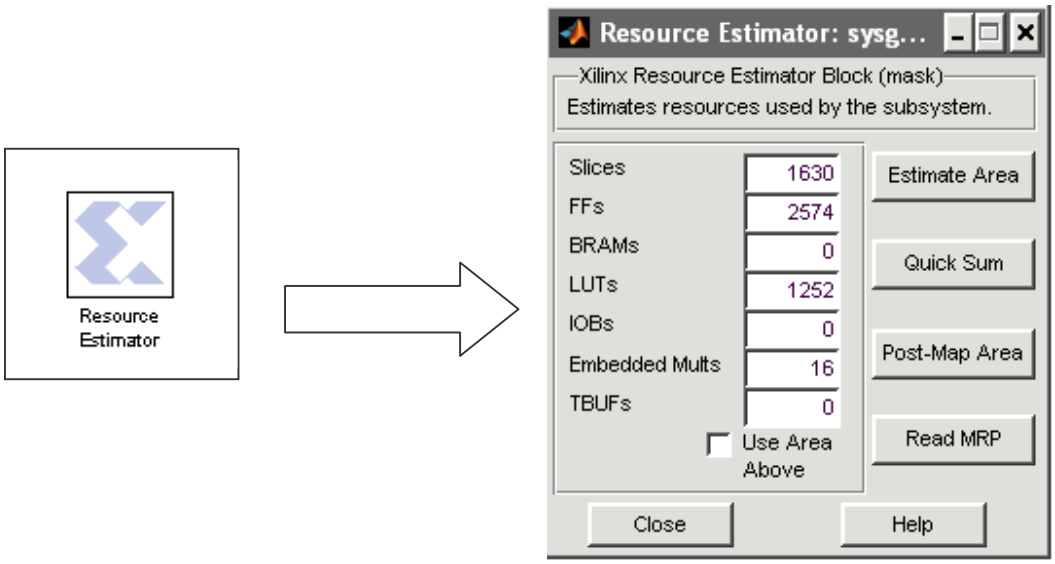

Figure 39 : Présentation des différents champs du bloc Resource Estimator.

- **Slices**: indique les nombres de Slices utilisées par bloc.
- **FFs** : Bascules électroniques utilisées par bloc.
- **BRAMs** : blocs de RAMs utilisées par bloc.
- **LUTs** : Tableaux de consultation utilisés par bloc.
- **IOBs** : Blocs d'entrée-sortie consommés par bloc.
- **Embedded Mults**: Multiplicateurs inclus utilisés par bloc.
- **TBUFs**: Buffers utilisés par block.

- **Use Area Above :** Quand cette case est collectionnée, toute évaluation de ressource effectuée sur ce sous-ensemble renverra les nombres écrits dans les boîtes d'édition de la zone de dialogue.

IV.4.1 Différents types de compilation du System Generator

Il y a différentes manières dont le System Generator peut compiler notre conception. La manière dont une conception est compilée dépend des arrangements dans la fenêtre de dialogue du System Generator. En supportant différents types de compilation, nous avons la liberté de choisir une représentation qui convient à l'environnement que la conception sera employée pour notre exemple, un HDL ou NGC Netlist est une représentation appropriée quand votre conception est employée comme composant dans un plus grand système. Si le système complet est modelé à l'intérieur du system Generator, nous pouvons choisir de compiler notre conception Bitstream. Parfois nous voulons compiler notre conception dans un module à niveau élevé équivalent qui exécute une fonction spécifique dans les applications externes au System Generator (par exemple, Co-Simulation de matériel avec ModelSim) :

- HDL Netlist Compilation : Le générateur de système emploie le type de compilation de HDL Netlist comme cible de génération de défaut.
- NGC Netlist Compilation : La NGC Netlist compilation nous permet de compiler notre conception dans un dossier binaire autonome de Netlist de XilinX NGC. Le dossier de Netlist de NGC que le System Generator produit contient l'information logique et facultative de contrainte pour notre conception. Ceci signifie que les HDL, les noyaux, et les contraintes classent l'information qui correspondent à une conception de System Generator sont dans un seul bloc dans un simple dossier. Le System Generator produit le dossier de Netlist de NGC en exécutant les étapes suivantes pendant la compilation : génerate un HDL Netlist pour la conception, exécute l'outil choisi de synthèse pour produire un Netlist plus bas. Le type du Netlist (par exemple, EDIF pour Synplify et de Leonardo, NGC pour XST) dépend de quel outil de synthèse est choisi pour la compilation.

- Bitstream Compilation : Le type de compilation Bitstream nous permet de compiler notre conception dans un dossier de Bitstream de configuration de XilinX qui convient à la carte FPGA choisie dans la zone de dialogue de System Generator. Le dossier de Bitstream est appelé <design>_clk_wrapper.bit et il est placé dans l'annuaire de la conception, où le <design> est dérivé de la partie de la conception étant compilée. Le System Generator produit le dossier de Bitstream en exécutant les étapes suivantes pendant la compilation : Produire d'un Netlist de HDL pour la conception ; Courir l'outil choisi de synthèse pour produire un Netlist plus bas. Le type du Netlist (par exemple, EDIF pour Synplify et de Leonardo, NGC pour XST) dépend de quel outil de synthèse est choisi pour la compilation. exécution du XFLOW pour produire un Bitstream de notre configuration.
- EDK Export Tool : L'outil d'exportation d'EDK permet à une conception de System Generator d'être exporté vers un projet du kit de développement inclus par XilinX (EDK). L'outil d'exportation d'EDK simplifie le processus de création d'un périphérique en produisant automatiquement des dossiers exigés par l'EDK.
- Hardware Co-Simulation Compilation : System Generator peut compiler des conceptions dans l'FPGA qui peut être utilisé dans la boucle avec des simulations de Simulink. Nous pouvons choisir une cible de Co-Simulation de matériel et en choisissant la plateforme désirée de Co-Simulation de matériel. Si nous avons une plateforme de FPGA qui n'est pas énumérée comme cible de compilation, nous pouvons créer une nouvelle cible de compilation de System Generator qui emploie le JTAG pour communiquer avec le matériel de FPGA.

IV.4.2 Résultats de System Generator

Dans cette section, nous discutons les fichiers produite par System Generator bas niveau quand HDL Netlist est choisit comme type de compilation. Les dossiers se composent de HDL et d'EDIF qui met en application la conception. En outre, le System Generator produit les dossiers auxiliaires qui simplifient en aval le traitement, par exemple, introduisant la conception dans le navigateur de projet, simulant en utilisant ModelSim, et le synthétisant à l'aide de divers outils de synthèse. Tous les dossiers sont écrits à l'annuaire de cible indiqué sur le bloc de System

Generator. Si aucun testbench n'est demandé, alors les dossiers principaux produits par System Generator sont les suivants :

- <design>_files.vhd/.v : Ceci contient la majeure partie du HDL pour la conception.
- <design>_clk_wrapper.vhd/.v : C'est un emballage de HDL pour <design>_files.vhd/.v. Il conduit des horloges.
- conv_pkg.vhd/.v : Ceci contient des constantes et des fonctions utilisées dans <design>_files.vhd/.v.
- .edn files : system Generator exécute le NOYAU de générateur (coregen) pour mettre en application des parties de la conception. Coregen écrit les dossiers d'EDIF dont les noms semblent typiquement quelque chose comme l'edn de multiplier_virtex2_6_0_83438798287b830b.
- Globals : Ce dossier se compose de la clef/ valeur qui décrivent la conception. Le dossier est organisé comme table de brouillage de Perl de sorte que les clefs et les valeurs puissent être rendues disponibles aux manuscrits de Perl en utilisant des evals de Perl.
- <design>.xcf (or .ncf) : Ceci contient la synchronisation et les contraintes d'endroit. Celles-ci sont employées par l'outil XST de synthèse de XilinX et les outils d'exécution de XilinX. Si l'outil de synthèse est placé à quelque chose autre que XST, alors le suffixe est changé en un ficher d'extension .ncf.
- <design>.ise : Ceci permet au HDL et l'EDIF à introduire dans le navigateur de XilinX d'outil de gestion de projet.
- hdlFiles : Ceci indique la liste de dossiers de HDL écrits par System Generator. Les dossiers sont énumérés dans l'ordre habituel de dépendance de HDL.
- synplify_<design>.prj, xst_<design>.prj, or spectrum_<design>.tcl : Ces dossiers permettent à la conception d'être compilée par l'outil de synthèse que nous avons indiqué.
- vcom.do : Ce manuscrit peut être employé dans ModelSim pour compiler le HDL pour une simulation comportementale de la conception.

- sysgen.log, postnetlist.log : system Generator emploie ces dossiers pour rapporter si la traduction a réussi, et pour enregistrer ce qui a mal tourné quand la traduction échoue, le fichier devient d'extension. nterf
- Various ace and interface.txt files. : System Generator écrit des dossiers d'interface pour décrire ses résultats de traduction. Ce sont les dossiers binaires, les dossiers du compagnon interface.txt qui contiennent la même information, mais exprimé sous format le texte.
- <design>_config.m : C'est une configuration de fonction M, elle permet au HDL et l'EDIF pour que la conception soit apportée de nouveau dans le générateur de système comme boîte noire.

IV.4.3 Résultats expérimentaux

Nous retournons à notre application, nous cliquons sur Generate dans la fenêtre de dialogue de System Generator, nous aurons, comme l'ullistre la figure 40, le déclanchement de la création des fichiers que nous avons déjà cité dans la section présidente.

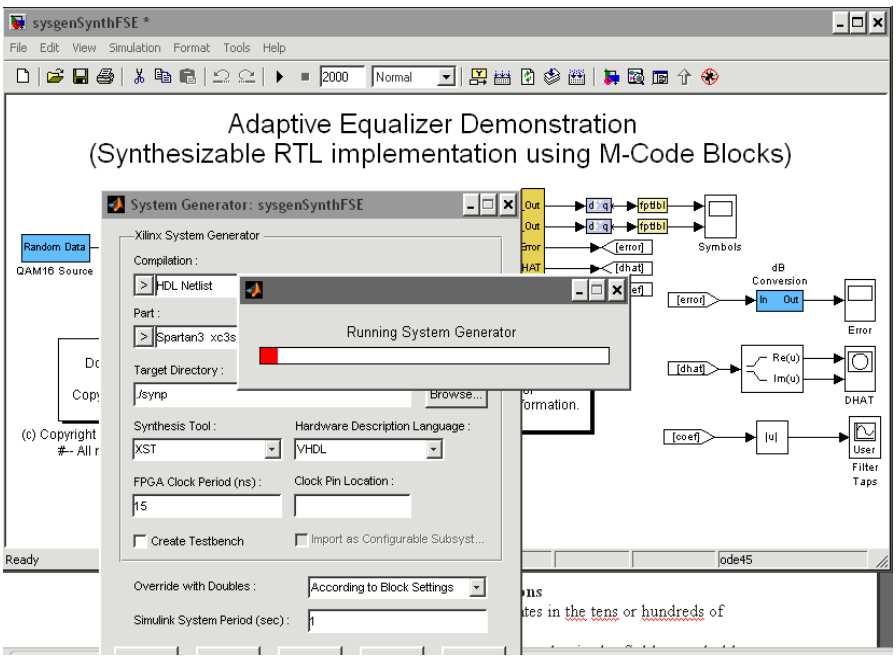

Figure 40 : Lancement du System Generator.

Une fois la génération est terminée, nous aurons l'apparition de la fenêtre « Generation complited » comme le montre la figure 41.

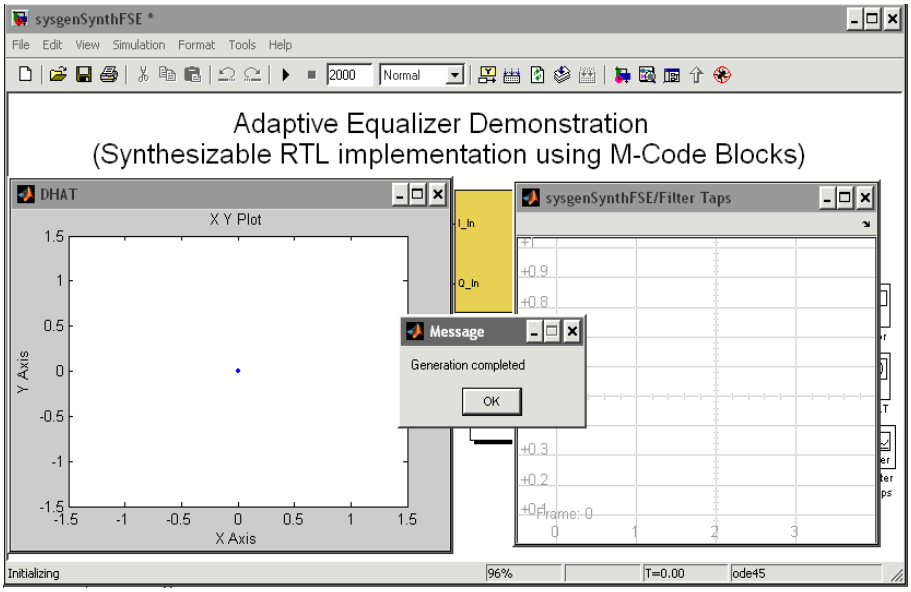

Figure 41 : Fin de la génération.

Nous pouvons maintenant accéder à l'emplacement ou nous avons crée le fichier de conception pour interpréter les résultats comme le montre la figure 42.

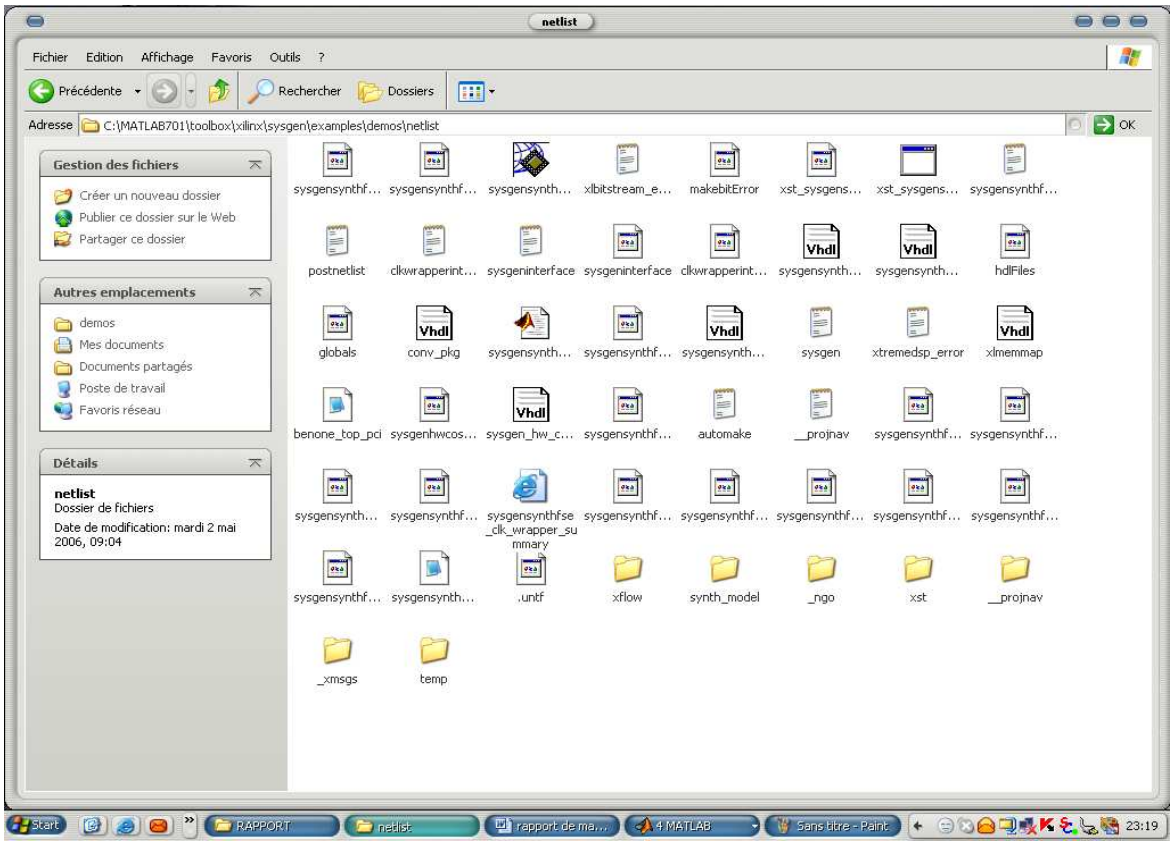

Figure 42 : Résultat de Co-Simulation.

Nous ouvrons le fichier sysgensynthfse_clk_wrapper_summary.html , nous trouvons des tableaux déjà remplis. Le premier tableau, comme le montre la figure 40, contient essentiellement :

- l'emplacement du projet qui est dans notre cas c:\matlab701\toolbox\xilinx\sysgen\examples\demos\netlist.
- Le type de la carte qui est ici xc3s200, c'est une information lié à la carte qui est dans notre cas une SPARTAN 3.
- Le dossier sysgensynthfse_clk_wrapper.ucf qui contient des informations sur les pins que nous devons utiliser pour notre exemple.
- La date du dernier changement pour ce fichier.
- Une version imprimable du rapport présenté en fichier HTML.

Property	Value
Project Name:	c:\matlab701\toolbox\xilinx\sysgen\examples\demos\netlist
Target Device:	xc3s200
Constraints File:	sysgensynthfse_clk_wrapper.ucf
Report Generated:	Tuesday 05/02/06 at 09:55
Printable Summary (View as HTML)	sysgensynthfse_clk_wrapper_summary.html

Tableau 2 : principal information du projet.

Ce tableau conclut le taux d'utilisation des ressources de la carte et est ce que le projet ou l'application peut être exécuté sur notre l'FPGA.

Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slices:	1883	1920	98%	
Number of Slice Flip Flops:	2817	3840	73%	
Number of 4 input LUTs:	1307	3840	34%	
Number of bonded IOBs:	83	173	47%	
Number of MULT18X18s:	19	12	158%	Resource Overuse
Number of GCLKs:	1	8	12%	

Tableau 3 : Tableau d'estimation de ressources.

Pour des informations sur le déroulement de la conception, nous trouvons dans le dernier tableau deux liens pour accéder aux fichiers de synthèse de la conception et le fichier de translation vers la carte que XFLOW génère.

Report Name	Status	Last Date Modified
Synthesis Report	Current	Tuesday 05/02/06 at 09:55
Translation Report	Out-of-Date	Tuesday 05/02/06 at 09:31

Tableau 4 : Plus d'information sur le rapport produit par System Generator.

IV.5 Conclusion

Dans ce dernier chapitre, nous avons présenté l'utilité et le rôle de chaque langage et outil formant la plateforme. Nous nous sommes intéressés au System Generator, crée par The MathWorks et responsable de l'interconnexion des différents langages pour la conception des systèmes hétérogènes.

Son rôle se résume principalement en quatre points qui sont :

- Rendement élevé : développer des systèmes qui exigent des taux d'échantillon et de prototypage très élevé.
- Une grande flexibilité : pour une architecture configurable de matériel qui est également champ extensible, c'est-à-dire que nous pouvons changer a tous moment dans notre architecture.
- Productivité élevée : les concepteurs matériel et logiciel peuvent utiliser les mêmes écoulements de conception.
- Le délai de mise sur le marché : il permet de réduire le temps de mise sur le marché.

Nous avons testé toute la plateforme sur l'exemple LMS-based adaptive equalization (Synthesizable RTL implementation using M-code Block) .Avec cette interface, nous avons accéléré la simulation et produit les fichiers nécessaires pour la Co-Simulation des systèmes hétérogènes.il nous a permit aussi d'identifier une méthode efficace de validation fonctionnelle de système embarqué dans son environnement unifié.

CONCLUSION ET PERSPECTIVE

Conclusions et perspectives

Les systèmes enfouis sont de plus en plus présents dans la vie quotidienne, que ce soit pour un usage professionnel ou personnel. Nous pouvons citer par exemple les téléphones mobiles, les assistants personnels (PDA), les consoles de jeux vidéos portables, les lecteurs multimédias portables (MP3 et consorts). Nous trouvons aussi de plus en plus de systèmes enfouis dans les automobiles, les appareils domestiques "intelligents" etc. Les fonctions qui peuvent être intégrées dans ce type de système peuvent être, par exemple, de type traitement de signal numérique (filtrage, compression décompression audio-vidéo,...), de type télécommunication (protocole réseau,...) ou bien encore contrôle/commande (domotique...).

La complexité grandissante des applications fait qu'il est nécessaire de pouvoir aborder leurs conceptions à des niveaux d'abstractions élevés. En effet, il est très intéressant de travailler à ces niveaux (par exemple au niveau système) car les gains (en surface/temps/consommation/coût) qu'il est possible d'obtenir par diverses transformations (tant algorithmiques qu'architecturales) sont proportionnels au niveau d'abstraction auquel on se situe. De plus, les décisions prises au niveau système peuvent avoir un impact très important en termes de développement industriel.

En effet, une mauvaise adéquation application/architecture (architecture sur/sous-dimensionnée ou mal adaptée aux caractéristiques de l'application) peut imposer, soit de mettre sur le marché un produit trop cher ou peu performant, soit de relancer un cycle de conception entraînant des délais pouvant être rédhibitoires.

Notre travail de recherche s'inscrit dans la préoccupation générale de conception de systèmes hétérogènes. Il y a un besoin urgent de proposer des méthodes et des outils permettant au chercheur et à l'ingénieur de créer des nouveaux objets au plus vite et « sans faute ».

Nous avons réalisé un état de l'art des pratiques de conception de systèmes, à base d'électronique, en nous appuyant sur notre interprétation de la démarche générale de conception. Pour ce faire, nous avons parcouru les outils de conception électronique analogique, numérique et mixte. Nous nous sommes intéressés principalement au produit de MathWorks qui est précisément Link for ModelSim. C'est une interface de Co-Simulation crée par The MathWorks pour la conception des ASICs et les FPGAs. Cette interface permet une communication entre

ModelSim et le produit MATLAB de MathWorks et Simulink. En outre, une bibliothèque des blocs de Simulink est disponible pour inclure des conceptions de ModelSim HDL dans des modèles de Simulink pour le Co-Simulation. Nous avons assuré la synchronisation et la communication entre MATLAB et ModelSim à fin d'accélérer la Co-Simulation MATLAB/VHDL. Mais cette interface, n'est pas suffisante pour la Co-Simulation MATLAB/FPGAs. Ensuite, Nous nous sommes intéressés au System Generator qui est crée aussi par The MathWorks qui est responsable de l'interconnexion des différents langages pour la conception des systèmes hétérogènes.

Son rôle se résume principalement en quatre points qui sont :

- Rendement élevé : développer des systèmes qui exigent des taux d'échantillon et de prototypage très élevé.
- Une grande flexibilité : pour une architecture configurable de matériel qui est également champ extensible, c'est-à-dire que nous pouvons changer a tous moment dans notre architecture.
- Productivité élevée : les concepteurs matériel et logiciel peuvent utiliser les mêmes écoulements de conception.
- Le délai de mise sur le marché : il permet de réduire le temps de mise sur le marché.

Avec c'est deux interface link for ModelSim et System Generator, nous avons identifié une méthode efficace de validation fonctionnelle de système embarqué dans son environnement unifié, accéléré la tache de Co-Simulation MATLB/ VHDL et Co-Simulation MATLAB /FPGAs et minimiser le taux d'erreurs dans les conceptions.

Mais des questions qui se posent : est ce que nous pouvons le faire pour plusieurs carte FPGAs en mêmes temps, est ce que sa reste valable pour d'autre carte sur tous les nouvelles comme VirteX-5 qui a apparu très ressèment pour le 17 mai 06.

Bibliographie

[Aas 04] Y. Le Moullec « développer un estimateur système dont les résultats puissent être utilisés par le concepteur très tôt dans le flot de conception »2004.

[Acc98] Accellera, Accellera Verilog Analog Mixed-Signal Group, “Velilog-AMS Home”. [En ligne].Adresse URL : http://www.eda.org/verilog-ams/. 1998

[Aem05] « AccelChip Enhances Model-Based Design Tool Suite » Barbara Marker 2006 barbara@hipcom.com

[Alte02] Altera « Altera device » on : http://www.altera.com , 2002.

[Ana01a] Analogy, Inc. « Saber® / Verilog-XL® Co-Simulation Interface ». Analogy, Inc, Beaverton, Oregon, Etats Unis d’Amérique, 2001.

[Ana01b] Analogy, Inc. « Saber® / ModelSim TM Co-Simulation Interface ». Analogy, Inc, Beaverton, Oregon, Etats Unis d’Amérique, 2001.

[Ana01c] Analogy, Inc. « Saber® / ViewSim ® Co-Simulation Interface ». URL : http://www.analogy.com/Products/simulation/simulation.htm#ViewSim. Analogy, Inc,

[And96] C. ANDRE. « Representation and Analysis of Reactive Behaviors: A Synchronous Approach » CESA'96, IEEE-SMC, Lille, France, 9 au 12 juillet, 1996.

[Ans03] ANSOFT Corporation, « System Modeling » [En ligne]. Adresse URL : http://www.ansoft.com/products/em/simplorer/. ANSOFT Corporation. Etats-Unis d’Amérique d’Amérique 2003.

[Bol 97] BOLSENS I., « Specification, co-simulation and Hardware/Software Interfacing for Telecom Systems », *Leuwen Codesign Course*, February, 1997.

[Bro03] D. BROWN. « A beginners guide to UML ». University of Kansas, Department of Electrical Engineering and Computer Science. [En ligne]. Adresse URL : http://consulting.dthomas.co.uk. Dustan Thomas Consulting 2003.

[Cal 95] CALVEZ J., HELLER D., PASQUIER O., « System Performance Modeling and Analysis with VHDL : Benefits and Limitations », *Proceedings of VHDL-Forum Europe Conference*, April, 1995.

[Cel02] Celoxica Ltd. « HANDEL-C language Overview ». Celoxica Ltd, août 2002.

[Cesa99] W. O. Cesario « *Synthèse architecturale flexible* » Thèse de doctorat, Institut National Polytechnique de Grenoble INPG, 1999.

[Cds03a] Cadence Design Systems Inc, « Using PSpice » [En ligne]. Adresse URL :

http://www.orcadpcb.com/pspice/default.aspbc=F. Cadence 2003.

[Cla01] P. CLARKE. « ESTEREL system-level language emerges from the lab ». EE TIMES, EE Times Network 2001.

[Clo01] F. CLOUTE, « Etude de la conception des systèmes embarqués sur silicium : Une approche de codesign matériel / logiciel », Thèse doctorat, Institut National Polytechnique de Toulouse, 2001.

[Cof03] Cofluent Design. « The MCSE Methodology Overview ». Cofluent Design 2003.

[Coo01] R. Scott COOPER. "The Designer's Guide to Analog & Mixed-Signal Modeling". Avant Corp. ISBN 0-9705953-0-1. 2001.

[Cow01] CoWare, Inc. « CoWare N2C Design System », Coware N2C Data Sheet, Coware, Inc, Santa Clara, Californie, Etats Unis d'Amérique 2001.

[Csp05] ChipScope Pro Software and Cores User Guide *(ChipScope Pro Software v7.1i)* UG029 (v7.1) February 16, 2005 www.xilinx.com

[Dd00] B. DION, S. DISSOUBRAY, "Modeling and implementing critical real-time systems with Esterel Studio", Esterel Technologies 2000.

[Dgg01] R. DOMER, A. GERSTAULER, D. GAJSKI, « SpecC Language Reference Manual, Version 1.0 », Université de Californie, Irvine,Mars 6, 2001.

[Di03] Dolphin Integration. "Dolphin Medal. New Features in SMASH™ 5.0.0 – 5.1.3". Dolphin Integration 2003

[Dmg97] G. DE MICHELLI, R. GUPTA. « Hardware / Software Co-Design ». Proceedings of the IEEE, Vol 85, No 3, March 1997.

[Dob03] A. DOBOLI. «Towards Automated Synthesis of Analog and Mixed-Signal Systems from High-Level Specifications». University of New York. FDL 2003. Frankfurt, Germany, September 2003.

[Dv03] A. DOBOLI, R. VEMURI. "A VHDL-AMS Compiler and Architecture Generator for Behavioral Synthesis of Analog Systems", Proceedings of DATE'99, pp.338-345, 1999.

[Edm 04] Eléments de MATLAB *Alfred A. Manuel* Département de la Physique de la Matière Condensée *alfred.manuel@physics.unige.ch* 15 October 2004

[Eur04] EUROPRACTICE Software Service, « RAL-EUROPRACTICE Software Service Home Page » [En ligne]. Adresse URL : http://www.te.rl.ac.uk/europractice/. 2004.

[Fll+98] E. FILIPPI, L. LAVAGNO, L. LICCIARDI, A. MONTANARO, M. PAOLINI, R.

PASSERONE, M. SGROI, A. SANGIOVANNI-VINCENTELLI. « Intellectual Property Reuse in Embedded System Co-design: an Industrial Case Study ». Proceedings of *International Symposium System Synthesis*, Hsinchu, Taiwan, December 1998.

[Gev04] C. GRIMM, K EINWICH et A. VACHOUX. « Analog and Mixed-Signal System Design with SystemC ». FDL'04 Tutorial. Septembre 16 2004. Lille, France.

[Gom01] M. GOMEZ. « Hardware-in-the-loop Simulation ». Embedded System Programming ». Etats Unis d'Amérique, novembre 2001.

[Gros97] C. Gross « *La conception système d'ASIC et de FPGA* » Electronique, n°75,pp.95-103, Novembre 1997

[Hag 93] HAGEN K., MEYR H., « Timed and Untimed Hardware/Software Cosimulation : Application and Efficient Implementation », *Proceedings of CODES*, 1993.

[Ham01] J.C HAMON. « Plate-forme de Prototypage Virtuel, Conception Système et « Codesign » micro électronique », Stage DEA CCMM, Institut National Polytechnique de Toulouse, 2001.

[Her02] Y. HERVE. « VHDL-AMS : «Applications et enjeux industriels». Dunod-Université - collection : Sciences-sup - préface d'Alain Vachoux. ISBN : 2-10-005888-6 - mars 2002.

[Ieee99] IEEE 1076.1-1999 standard, Language Reference Manual. "VHDL Analog and Mixed Signal extensions". ISBN 0-7381-1640-8.

[Jerr97] A. A. Jerraya, H. Ding, P. Kission, M. Rahmouni « *Behavioral synthesis and component reuse with VHDL* » Kluwer Academic Publishers, 1997.

[Mal02] D. MALINIAK. « From CAD to CAE to EDA, Design tools have wrestled with complexity ».ED Online ID #2311. Penton Media, Inc. juin 2002.

[Mei 97] MEIER W. ET AL., « Design of Multimedia Systems : Anatomy of an MPEG2 Decoder », *Leuwen Codesign Course,* February, 1997.

[Mil03] R. MILLER. « Practical UML™: A Hands-On Introduction for Developers ». BorlandDeveloper Network. [En ligne]. Adresse URL:http://community.borland.com 0,1410,31863,00.html. Borland SoftwareCorporation, Inc, 2003.

[Mg00a] Mentor Graphics Corporation, « Seamless CVE User's Reference and Manual, software version 4.0 », Mentor Graphics 2000.

[Mg00b] Mentor Graphics Corporation, « Getting Started with Seamless CVEl, software version 4.0 », Mentor Graphics 2000.

[Mg01a] Mentor Graphics Corporation, « AdvanceMS Datasheet » Mentor Graphics Corporation.2001

[Mg03] Mentor Graphics Corporation, « System Modeling » [En ligne]. Adresse URL : http://www.mentor.com/systemvision/. Mentor Graphics Corporation.2003.

[Moc05] « Méthodes et outils de la conception amont pour les systèmes et les microsystèmes » M Juan-Carlos HAMON Soutenue le 1 février 2005,

[Omg03] Object Management Group, Inc., "OMG Unified Modeling Language Specification, Version 1.5", Object Management Group, Inc. Etats Unis d'Amérique 2003.

[Pa05] PlanAhead User Guide *Release 7.1* 6/2/2005. www.xilinx.com

[Pana02] J. Panattoni « News & Views » Newsletter for Altera Custemers, Third Quarter 2002.

[Per04] V. PERRIER. "System Architecting complex designs". Embedded Systems Europe. Janvier/février 2004. pp. 24 – 26.

[Pn 03] Patrice NOUEL « Langage VHDL et conception de circuits » Dernières mise à jour Juillet 2003

[Ros72] DT, ROSS. "Structured Analysis and Design Technique (SADT)". The Massachusetts Institute of Technology. Cambridge, Massachusetts, Etats Unis 1972.

[Ries99] T. Riesgo, Y. Torroja, E. De la Torre « Design methodologies based on hardware description languages » IEEE Transaction On Industrial Electronics, Vol. 46, n°1, February 1999.

[San96] A. SANGIOVANNI-VINCENTELLI. « Trends in Electronic Systems ». Proceedings of Mediterranean Electrotechnical Conference on Industrial Applications in Power Systems, Computer Science, and Telecommunications, mai 1996.

[Sdc03] Le langage VHDL P.N ENSEIRB « Synthèse des circuits » 2003

[Sim03] SIMEC GmbH & Co KG. « hAMSter The High Performance AMS Tool for Engineeringand Research ». [En ligne]. Adresse URL : http://www.hamster-ams.com/. SIMEC GmbH & Co KG, 2003.
[SUG 03] Synplify® User Guide June 2003

[Sore01] Y. Sorrel « *Méthodes et architectures pour le TSI en temps réel, chapitre méthodologie AAA d'adéquation algorithme architecture* » Kluwer Academic Publishers, 2001.

[Sug 03] Synplicity, Inc. 600 West California Avenue Sunnyvale, CA 94086 User Guide June 2003

[Syn00] Synopsys, « Hardware/Software Co-Verification with Synopsys Eaglei Tools », Synopsys, Inc, Etats Unis d'Amérique 2000.

[Syn03] Synopsys®. « Saber HDL: Language-Independant Mixed-Signal Multi-Technology Simulator ». Synopsys® Etats Unis d'Amérique 2003.

[Syn04] Synopsys®. « OpenMAST Overview ». [En ligne]. Adresse URL : http://www.openmast.org/overview/overview.html. Synopsys® Etats Unis d'Amérique 2004.

[Tmw04a] The MathWorks. « System Specification and Modeling ». [En ligne]. Adresse URL : http://www.mathworks.com. The Mathworks, Inc. 2004.

[Tmw04b] The MathWorks. « Embedded System Design ». [En ligne]. Adresse URL : http://www.mathworks.com. The Mathworks, Inc. 2004.

[Tmw04c] The MathWorks. « Aerospace and Defense – Engineering Tasks ». [En ligne]. Adresse URL : http://www.mathworks.com. The Mathworks, Inc. 2004.

[Tmw03] The MathWorks. « Link for ModelSim® 1.1 ». The mathWorks, Inc. 2003.

[Ug05] User's Guide Link for ModelSim For Use with MATLAB® and Simulink www.mathworks.com

[Ver 94] VERMA D., « Very Large Scale Integrated Circuit Architecture Performance Evaluation Using SES Modelling Tools », rapport technique, VLSI Technology, 1994.

[Wm85] P. WARD, J. STEPHEN J. MELLOR. "Structured Development for Real-Time Systems".Vol. 1-3, Yourdon Press, Englewood Cliffs, 1985.

[Wvc00] I.A. Grout , K. Keane «A MATLAB TO VHDL CONVERSION TOOLBOX FOR DIGITAL CONTROL» 2000

[Xap 05] « Using the ISE Design Tools for Spartan-3 Generation FPGAs » XAPP473 (v1.1) May 23, 2005 : http://www.xilinx.com/ise/ise_promo/ise_spartan3.htm

[Xdsp05] Xtreme DSP Sélection Guide Fourth Quarter, 2005 www.xilinx.com/dsp.

[XilinX05] Xilinx 7.1i Design Tools *Product Backgrounder* February, 2005 www.xilinx.com.

[Xsg05] Xilinx System Generator v 7.1 User Guide 2006.

[Yk 02] « Développement d'une Méthodologie de Conception Matériel à Base de Modules Génériques VHDL/VHDL-AMS en Vue d'une Intégration de Systèmes de Commande Electriques » **Youssef KEBBATI** 16 Décembre 2002