Apprendre avec Alliance

M-M PAGET, M. M. ROSSET-LOUERAT, A. DERIEUX
Laboratoire MASI, Institut Blaise Pascal
Université P. et M. Curie - CNRS - URA 818
4 Place Jussieu, F75252 Paris Cedex 05, E-mail : paget@masi.ibp.fr

Abstract
We present the Alliance package, a learning environment dedicated to VLSI circuit design. Alliance has two targets : a pedagogical one since it is used for the teaching of circuit design and a research one since it provides a basic platform for formalizing new ideas and for making new CAD tools. It is extensively used both in the universitary world and in the industrial one, but we are here asking ourselves about what is learned with Alliance and how it is done, given that it represents special successful learning strategies.

Résumé
Nous présentons la chaîne Alliance, un environnement d'apprentissage dédié à la conception de circuits VLSI. Alliance a deux objectifs : l'un de type pédagogique puisqu'elle est utilisée pour l'enseignement dans le cadre d'une formation des étudiants à la conception de circuits intégrés, l'autre de type prospectif en recherche puisqu'elle sert de base pour la formalisation de nouvelles idées et la réalisation de nouveaux outils logiciels de CAO. Elle est déjà très utilisée telle quelle dans le monde universitaire et industriel, mais nous nous interrogeons ici sur ce qu'elle enseigne et comment elle le fait, puisqu'il s'agit là de la mise en place réussie de stratégies particulières d'apprentissage.

Keywords : CAO, CAD, VLSI circuits, Méthodologie de conception

Ce papier a été présenté à la Cinquième Rencontre francophone sur la Didactique de l'Informatique, Monastir, Tunisie, 10-12 Avril 1996

1. Introduction

Alliance est une chaîne complète d'outils et de bibliothèques portables destinée à la CAO (**C**onception **A**ssistée par **O**rdinateur) de circuits intégrés VLSI (**V**ery **L**arge **S**cale **I**ntegration) [GRE92-1, BAZ94-2]. Alliance est disponible gratuitement sur réseau Internet et peut être installée sur des stations de travail UNIX ou sur des PC 386 sous Linux .
C'est plus qu'un ensemble d'outils et de bibliothèques, c'est aussi une pédagogie adaptée à un apprentissage particulier. En effet pour concevoir de gros circuits intégrés (quelques centaines de milliers de transistors) il faut suivre une méthode qui s'apparente à la programmation structurée d'un gros logiciel. C'est un travail d'électronique et d'informatique à la fois, qui a pour but de produire le dessin optimisé des masques nécessaires pour la fabrication du circuit. Le procédé technologique de fabrication consiste à utiliser les masques dessinés par le concepteur pour graver dans le silicium plusieurs niveaux (couches) selon une séquence bien définie.

2. Méthodologie de conception d'un circuit intégré

Nous allons d'abord décrire brièvement les étapes et la méthodologie de conception d'un circuit intégré. Ceci devrait nous permettre déjà de répondre à la question: "Qu'apprend-t-on avec Alliance ?".
Un circuit intégré possède trois types de description (Figure 1) : une vue comportementale qui décrit la fonction qu'il doit réaliser, une description structurelle hiérarchique (ou netlist) qui représente le schéma en transistors et une vue physique ou layout qui fait apparaître le dessin des masques de fabrication. La vue comportementale donne les sorties en fonction des entrées. La vue structurelle utilise des schémas de base dont on dispose dans des bibliothèques.

Pour passer automatiquement de la vue comportementale en langage VHDL (**V**ery High Speed Integrated Circuit **H**ardware **D**escription **L**anguage) à la vue structurelle, on fait une synthèse logique. Entre le schéma en portes et la vue physique une opération de placement routage est nécessaire. Des outils permettent les opérations inverses, un extracteur de netlist de la vue physique vers la vue structurelle, un abstracteur fonctionnel pour établir le comportement à partir d'une netlist.

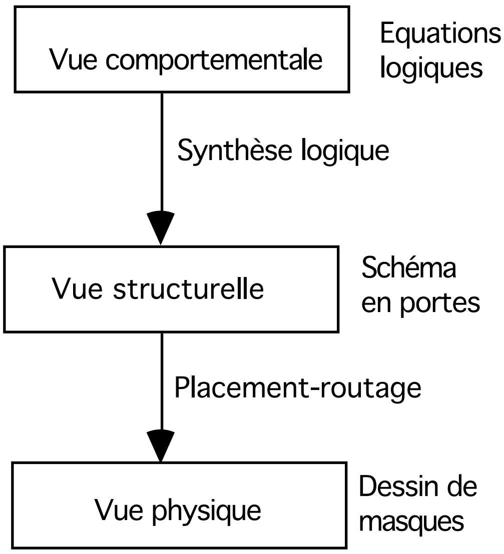


Figure 1 : Concept des vues d'un circuit intégré

Alliance est une chaîne ouverte. Chaque outil peut fonctionner indépendamment des autres. Les formats de fichiers utilisés sont des formats standard de l'industrie (VHDL, EDIF, SPICE, GDSII). Il y a interopérabilité avec d'autres chaînes de conception du commerce comme par exemple la chaîne Compass en ce qui concerne les représentations layout et netlist. Une structure de données internes commune, de type orienté objet permet de représenter les trois vues [PET94].
Signalons une conséquence du succès d'Alliance et de sa capacité à supporter des formats différents des siens : l'entreprise BULL a décidé d'interfacer son propre logiciel de routage `cheops` avec Alliance. Ce logiciel fait actuellement partie de la version 3.0 de la chaîne Alliance.

La méthodologie de conception est un ensemble séquencé d'opérations utilisant une bibliothèque de cellules prédéfinies dans une approche hiérarchique et structurée, où un objet sera décrit par un nombre limité d'objets standardisés plus simples. Pour réaliser un circuit "zéro défaut" on utilise une conception descendante, et une validation ascendante [LUC95].

3. Les outils d'Alliance

Les outils d'Alliance qui représentent plus de 500.000 lignes de langage C sont le fruit d'une réalisation collective, qui a d'ailleurs donné lieu au prix spécial d'équipe du concours Seymour Cray en 1994.
Le découpage modulaire permet une intégration plus facile des nouveaux outils par étapes successives. La Figure 2 qui complète la Figure 1,

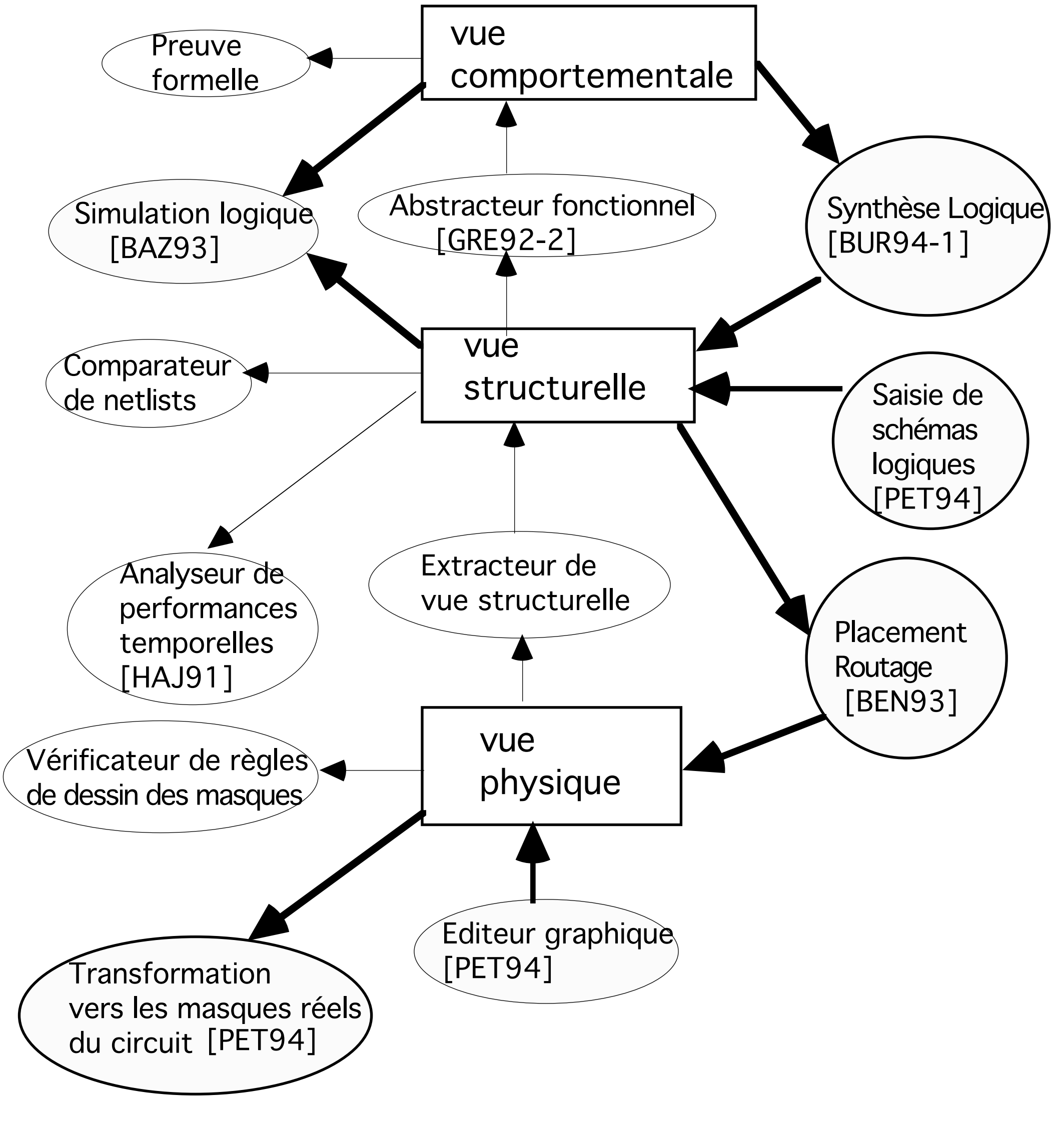
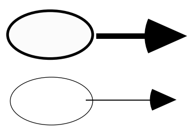


Figure 2 : Formalisation de la démarche conceptuelle [LUC95]



illustre les interactions entre les outils et les structures de données et fait apparaître les étapes essentielles de la méthode de conception et de validation d'un circuit avec Alliance. Une description succinte des outils d'Alliance est donnée en Annexe 2.

Un circuit est représenté par un cœur entouré par une couronne de plots. Les plots constituent l'interface entre le cœur qui réalise la fonction du circuit et l'extérieur du circuit. Les plots sont reliés aux broches du boitier dans lequel sera encapsulé le circuit. Le cœur d'un processeur peut être décomposé en une ou plusieurs parties opératives ou chemins de données, et une ou plusieurs parties contrôles. Ces éléments ont des caractéristiques propres et nécessitent des outils de conception différents.

4. Les tutoriels d'Alliance

Une fois décrite la chaîne Alliance, ses outils et la méthode qui a été adoptée pour mener à terme la conception d'un circuit depuis sa spécification jusqu'aux masques, nous allons présenter les aides qui lui sont associées afin de préciser "Comment on apprend avec Alliance".
Les choix pédagogiques pris pour cet environnement le placent selon Mendelsohn [MEN92] à la fois parmi les logiciels ouverts et parmi les systèmes spécialisés construits pour l'acquisition d'une expertise bien définie.

Les aides sont des tutoriels car ce sont des séquences d'apprentissage au sens le plus complet du terme, qui sont mises à la disposition de l'apprenant comme un tuteur, surtout quand il y a une part d'autoformation [DEM92]. Ces tutoriels, en langue anglaise, se classent en deux catégories, ceux qui concernent la méthodologie et ceux qui s'appliquent à un outil ou bien à une fonction du cycle complet.

4.1 Tutoriels concernant la méthodologie

4.1.1 Rôle des tutoriels concernant la méthodologie

Le but de ces tutoriels est d'apprendre la méthodologie de conception et de validation d'un circuit intégré avec Alliance. Trois circuits de complexités différentes ont été choisis. Les tutoriels correspondants peuvent être utilisés pour prendre en main les outils de la chaîne et suivre la réalisation du circuit en partant de sa spécification fonctionnelle.
Dans chacun des cas tous les fichiers sources sont fournis avec le tutoriel. Un ordre de grandeur du temps nécessaire pour la construction du circuit est indiqué (deux heures pour l'`amd2901` et six heures pour le `dlx` sur une station Sparc). Ces temps représentent le temps passé à exécuter le tutoriel et non le temps de réflexion et de compréhension.

Il est possible de faire la construction ou bien pas à pas en tapant les commandes successives indiquées ou bien automatiquement en exécutant le fichier `Makefile` fourni, qui gère l'ensemble de la construction du circuit considéré. Tels qu'ils existent, ces trois tutoriels ne sont pas seulement trois exemples différents mais trois exemples de plus en plus difficiles à aborder, mettant en œuvre de plus en plus d'outils d'Alliance, pour permettre à l'apprenant une familiarisation de plus en plus approfondie avec la chaîne .
Au cours des opérations, la seule interactivité possible de l'apprenant est localisée dans le choix d'une option plutôt qu'une autre au niveau des commandes.

4.1.2 Tutoriel 1 : Additionneur 4 bits

Le circuit `addacu`, qui devrait être étudié en premier avec Alliance, est très simple (environ 500 transistors); il est composé d'un additionneur 4 bits, d'un registre 4 bits et d'un multiplexeur. Ce tutoriel fait partie d'Alliance depuis sa première version (Alliance 1.0, septembre 1992).

4.1.3 Tutoriel 2 : AMD2901 4 bits

Le second circuit est déjà plus complexe (environ 3 000 transistors), c'est le processeur 4 bits `amd2901` [AMD77] . La conception est faite par saisie textuelle des schémas en portes logiques. Ce tutoriel fait partie d'Alliance depuis sa première version (Alliance 1.0, septembre 1992).

4.1.4 Tutoriel 3 : Microprocesseur 32 bits DLX

Quant au troisième circuit, c'est le microprocesseur 32 bits `dlx` [HEN92] d'environ 30 000 transistors. La conception est faite en tenant compte du découpage du microprocesseur en un bloc de contrôle et un chemin de données (partie opérative). Le contrôle est décrit sous forme d'automate. Les outils d'Alliance mis en œuvre sont les mêmes que pour le Tutoriel 2, avec en plus les outils de synthèse . Pour faciliter la simulation et le développement de nouveaux vecteurs de simulation, un environnement particulier a été créé. Il s'agit d'une carte, décrite en VHDL comportant le microprocesseur `dlx`, un décodeur d'adresse et des mémoires ram et rom. Un assembleur est utilisé pour transformer les instructions du `dlx` en fichier interprétable par le simulateur logique d'Alliance. Ce tutoriel fait partie d'Alliance depuis sa deuxième version (Alliance 2.0, février 1993).

Une partie du Tutoriel 3 décrivant la synthèse de l'automate par utilisation de l'outil `syf`, est donnée ci-dessous :

3.2 Structural Design

3.2.1 Design Hierachy

In this step, the structural descriptions of the chip **(dlxm.vst)**, and the core **(core.vst)** are used. The chip is described as a core surrounded by pads. The core is divided into two structural blocks: control and data path, the control block being also divided into two structural blocks : sequencer and status, each of which must be represented by its own behavioural description. The following source files are provided :

- **dlxm_chip.vst** : VHDL structural model of the dlxm chip instantiating core and pads.
- **dlxm_core.vst** : VHDL structural model of the core instantiating the data path, and the control.
- **dlxm_ctl.vst.h** : VHDL structural model of the control instantiating the sequencer and the status (the control model **dlxm_ctl.vst** being saved in the file **dlxm_ctl.vst.h** to prevent future erasing).
- **dlxm_seq.fsm** : VHDL finite state machine model of the sequencer.
- **dlxm_sts.vbe** : VHDL model of the status.
- **dlxm_dpt.vbe** : VHDL model of the data path.

3.2.2 Sequencer state assignment

The sequencer is written using a subset of VHDL specifically designed for the description of finite state machines. You must therefore compile this into a VHDL data-flow behavioural model (vbe) using the ALLIANCE tool syf :

>syf -s dlxm_seq -of dlxm_seq -scan -save

- **-s** : uses a vertical encoding algorithm
- **dlxm_seq** : fsm source file (**dlxm_seq.fsm**)
- **-of dlxm_seq** : output behavioural description (**dlxm_seq.vbe**)
- **-scan** : adds a scan path to the state register
- **-save** : saves encoding result in **dlxm_seq.cod** file

Ce tutoriel a l'avantage d'illustrer la méthodologie d'Alliance sur un circuit complexe et pourtant accessible et "connu" puisque le processeur `dlx` a été créé par Hennessy et Patterson comme un modèle typique de processeur RISC pour l'enseignement. Il est cependant manifeste que cet exemple est trop complexe pour qu'un utilisateur d'Alliance qui ne connaîtrait pas au préalable le `dlx` puisse comprendre, en exécutant le tutoriel, le contenu des fichiers sources, en particulier les fichiers décrivant l'automate de contrôle et le chemin de données.

4.2 Tutoriels concernant les outils

L'utilisateur dispose à la fois de la documentation en ligne et d'aides spécifiques concernant une fonction particulière du cycle de conception.

4.2.1 Documentation en ligne

Depuis la création d'Alliance il existe une documentation en ligne pour chaque outil de la chaîne Alliance, utilisant la commande `man` du système UNIX. En entrant la commande `man nom_de_l'outil` on obtient les informations concernant l'outil correspondant d'Alliance. Actuellement plus de mille pages de documentation de ce type sont consultables par le développeur. Un exemple tel qu'il est disponible, c'est à dire en anglais, est donné en Annexe 1 pour l'outil `syf` de synthèse d'automate. Ce type de documentation n'est que le mode d'emploi de l'outil.

4.2.2 Tutoriels spécifiques

Dans la première version d'Alliance, seuls existaient les tutoriels 1 et 2 présentés précédemment. L'arrivée des outils de synthèse logique et des outils de conception orientés chemin de données a fait apparaître la nécessité d'un nouveau tutoriel illustrant la méthodologie accompagnant ces nouveaux outils. Ainsi fut créé le Tutoriel 3. Le circuit qu'il présente est un microprocesseur 32 bits microprogrammé. A l'usage, ce tutoriel s'est révélé inefficace, car trop complexe pour enseigner la méthode de conception d'un circuit. Il est apparu qu'il fallait utiliser des circuits de fonctionnalité plus simple pour mettre l'accent sur une certaine étape de la conception.
C'est ainsi que de nouveaux tutoriels ont été conçus pour la dernière version (Alliance 3.0, mai 1995) qui présentent séparément les outils de conception, tout en permettant la conception totale du circuit et sa validation. Sont ainsi disponibles :

- un tutoriel décrivant la conception d'un circuit digicode à partir de sa spécification sous forme d'une machine à états finis, pour insister sur les outils de synthèse logique [BUR94-1].
- un tutoriel décrivant la conception de la partie opérative d'un circuit à partir de sa spécification structurelle, pour insister sur les outils de conception d'un chemin de données optimisé [BEN93].
- un tutoriel décrivant la réalisation du circuit digicode sur FPGA (**F**ield **P**rogrammable **G**ate **A**rray) par l'outil `Alligator` (**ALLI**ance **GAT**e Array **O**ptimize**R**). `Alligator` [BUR94-2] prend en entrée une vue comportementale VHDL et fournit une netlist de portes disponibles sur matrice prédiffusée, au format particulier du fabricant. L'intérêt des FPGA est de permettre la réalisation rapide et à faible coût d'un prototype du circuit puisque, une fois la matrice disponible, seuls les connexions entre portes sont à réaliser.

Malheureusement il n'existe pas un tutoriel pour chaque outil et il reste du travail à faire pour que les tutoriels permettent réellement un apprentissage autonome.

En fait la chaîne est surtout utilisée pour des étudiants de maîtrise ou DEA, qui ne sont pas livrés à eux mêmes et auxquels des enseignants donnent les explications nécessaires. Si l'on souhaite voir Alliance utilisé par des apprenants moins avertis, il faudrait qu'ils disposent d'exercices pratiques supplémentaires avec questions et réponses pour favoriser leur apprentissage. Chaque tutoriel serait composé d'une série d'unités comprenant chacune une présentation d'informations et de commandes à faire exécuter, accompagnée de sollicitation de l'apprenant, saisie de sa réponse et traitement du message correspondant avec réaction à ce message pour contrôler le bon acquis des notions à faire passer. En plus de compétences spécifiques sur l'expertise du contenu, ceci suppose des compétences didactiques pour déterminer les activités à proposer aux apprenants, types d'exercices et de commentaires, avec des niveaux différents allant du très débutant au débutant avancé. Ainsi faire d'Alliance un environnement de simulation pédagogique plus structuré et beaucoup plus interactif et convivial, reste une amélioration délicate, qui n'est pas du ressort des chercheurs en CAO et VLSI.

Dans les tutoriels un effort de présentation des écrans et l'utilisation de plusieurs fenêtres de visualisation apporteraient de la convivialité et éviteraient une lecture passive d'écrans successifs. On sait bien que l'ergonomie joue un rôle important dans la relation apprenant-machine. Celle-ci a par exemple été particulièrement soignée dans [EDEC95], un ensemble de cours interactifs concernant la conception à tous les niveaux de circuits électroniques, mis en place dans le cadre de la collaboration de plusieurs universités du Royaume Uni.
Une synthèse générale de l'ensemble de la chaîne Alliance permettrait aussi de se repérer au milieu des nombreux outils, sans avoir recours à un exposé préalable. Ceci s'est d'ailleurs déjà fait sentir puisqu'il existe quelques documents comme par exemple celui qui précise les structures pour l'outil de simulation logique `Asimut` [P. BAZARGAN SABET et H.N VUONG "Les nouvelles structures pour Asimut", Mai 1995], qui n'est pas un tutoriel mais un document d'information sur un point particulier.

Malgré ces aides plus ou moins tutorielles, il reste aux apprenants des difficultés à surmonter. Par exemple, l'un des utilisateurs a écrit (traduit de l'anglais) "J'ai essayé d'apprendre Alliance en exécutant les trois tutoriaux. Après avoir utilisé les exemples de `addacu` et `amd2901` il ne me semble pas très clair d'écrire les fichiers C nécessaires à l'outil de saisie textuelle de schémas `Genlib`. Les informations des man correspondants n'aident pas assez à comprendre comment manipuler l'écriture du code". Une trop grande liberté d'action rend souvent l'apprentissage difficile.

4.3 Expérience d'enseignement à l'Université Pierre et Marie Curie

Alliance est un environnement évolutif paramétrable par les enseignants. Il sert de base autour de laquelle se construit un enseignement dispensé aux étudiants. Il est utilisé à l'Université Pierre et Marie Curie en maîtrise d'informatique, en maîtrise EEA

(Electronique, Electrotechnique et Automatique), au DEA ASIME (Architecture des Systèmes Intégrés et Micro Electronique) et au DESS CIMI (Circuits Intégrés et Micro Informatique). Depuis plusieurs années, nous avons mis au point une formation de deux mois (octobre, novembre) permettant aux étudiants provenant de cursus différents (Informatique, Electronique, Physique), l'apprentissage de la chaîne de CAO Alliance. Cette formation a été conçue pour les étudiants désirant suivre le DEA ASIME ou le DESS CIMI . Chaque année se joignent à ces étudiants des auditeurs libres, qui sont en général des professeurs étrangers (deux à trois par an, en provenance du Brésil, d'Espagne et d'Egypte) désirant suivre cette formation afin d'utiliser la chaîne de CAO Alliance lors de leur enseignement de la microélectronique.

Des cours magistraux sont dispensés le matin du premier mois, sur les thèmes suivants:

- Méthodologie de conception d'un circuit
- Présentation des outils de CAO
- Circuiterie CMOS
- Architecture du processeur DLX.

Les après-midis sont consacrés à des travaux pratiques d'utilisation de la chaîne Alliance. Des exercices sont proposés aux étudiants qui travaillent en binôme :

- La première semaine, le travail consiste en un apprentissage du langage VHDL : la représentation comportementale et structurelle d'un additionneur accumulateur.
- La deuxième semaine, le travail demandé concerne les outils de synthèse logique et le routage de cellules standard, ainsi que le dessin des masques d'une cellule de la bibliothèque . L'exemple de circuit utilisé est un digicode. Les étudiants ont à décrire l'automate de ce digicode sous la forme d'un automate d'états finis qui est ensuite synthétisé en cellules standard, puis routé, et validé selon la méthodologie explicitée en cours (Figure 2).
- Durant la troisième semaine, on réalise l'AMD2901 en deux blocs distincts : la partie contrôle est conçue en cellules standard et la partie opérative est obtenue à l'aide du compilateur de chemin de données.
- La quatrième semaine est consacrée aux problèmes de simulation électrique et de vérification temporelle. La simulation électrique est obtenue par utilisation du logiciel SPICE [HMU90] tandis que l'analyse temporelle peut être obtenue à l'aide de l'outil d'Alliance TAS [HAJ91], ce qui permet d'aborder les notions de stabilité et en particulier l'étude des contraintes de type "temps maximal d'établissement d'un signal" et de "temps minimal de maintien d'un signal" à respecter pour le bon fonctionnement d'un circuit avec points mémorisants. On met en évidence les différences existantes entre un simulateur électrique

(précision, nécessité de stimuli, mais lenteur) et un analyseur temporel (moins précis, plus rapide et sans nécessité de stimuli).

Cette formation allie le contenu des tutoriels précédemment décrits à des explications sur les concepts nécessaires à la conception de circuits VLSI digitaux (Figure 1 et Figure 2).

Après ce mois de formation, nous considérons que les étudiants sont alors capables de réaliser un projet important le microprocesseur 32 bits DLX, ainsi le mois de novembre est entièrement consacré à la réalisation de ce circuit en respectant la méthodologie d'Alliance. Pour ce faire, par groupes de cinq à six personnes, les étudiants partent d'une description fonctionnelle qui constitue la spécification du processeur et réalisent le dessin des masques du circuit. Ce projet permet de faire évoluer le Tutoriel 3 DLX [BAZ94-1] d'une année à l'autre.

5. Autres objectifs

Au delà de l'acquisition des savoir-faire liés à la description d'actions à effectuer pour atteindre un but, la chaîne Alliance a aussi été utilisée pour la conception de circuits de très haute complexité. Trois circuits d'une complexité de plusieurs centaines de milliers de transistors ont été conçus avec Alliance : il s'agit d'un routeur de message utilisant des liens séries à 1 giga-bit/s [ZER96], d'un coprocesseur à architecture SIMD pour le traitement du génome [ARC94] et d'un microprocesseur superscalaire à mot d'instruction très large [GRE94, LUC95, WAJ95] .
Certains outils d'Alliance sont particulièrement originaux et appréciés des industriels, qui n'ont pas forcément à leur disposition un outil industiel équivalent plus performant. C'est le cas :

- du compilateur de chemin de données [BEN93, BEN94]
- des outils de synthèse logique [BUR94-1], [BUR94-2]
- des outils de vérification TAS [HAJ91], DESB [GRE92-2].

Nous ne pouvons citer ici qu'une partie des publications concernant la chaîne, elles sont en nombre déjà supérieur à 40...

6. Les utilisateurs d'Alliance hors l'Université P. et M. Curie

6.1 Diversité des utilisateurs d'Alliance

Alliance n'est pas seulement utilisée à Paris ! Comme le montre le Tableau 1, Alliance est utilisée dans plus de 150 universités françaises et étrangères dans le monde. Pour cet objectif d'apprentissage, elle peut être installée sur des machines peu coûteuses de type PC sous Linux.

Allemagne	16	Etats Unis	48	Portugal	2
Arabie Saoudite	1	France	18	Roumanie	2
Australie	3	Grèce	1	Royaume Uni	9
Belgique	1	Hong Kong	1	Russie	2
Brésil	6	Inde	1	Singapour	1
Bulgarie	1	Italie	6	Slovenie	3
Canada	2	Iran	2	Suède	3
Colombie	1	Israel	2	Suisse	2
Corée	1	Japon	7		
Danemark	1	Mexique	1		
Egypte	1	Norvège	2		
Espagne	3	Pays Bas	2		
Estonie	1	Pologne	4	TOTAL	157

Tableau 1 : Sites utilisateurs d'Alliance (Janvier 1995)

De plus la diffusion gratuite des fichiers sources sur le réseau Internet par ftp depuis 1992 donne la possibilité aux utilisateurs d'apporter leur réelle participation pour modifier ou corriger chaque nouvel outil. Une vingtaine de messages arrivent chaque semaine, la grande majorité pour demander des compléments d'information, et dont certains ont déjà permis la correction de plusieurs bugs.

6.2 Expériences d'utilisation

A la demande de deux instituts européens, des enseignants du Laboratoire MASI ont été enseigner la méthodologie de conception de circuits intégrés VLSI avec Alliance en Italie et en Pologne.
En novembre 1994, N. Abdallah a fait des cours à l'Université des Nations Unies dans le cadre du "Third Course on Basic VLSI Design Technics". On peut signaler que les organisateurs du cours avaient réalisé pour cette occasion un document composé de sept exercices de difficulté croissante à faire exécuter l'un après l'autre par l'apprenant [ICTP94]. Ces exercices ressemblent beaucoup par leur structure aux différents tutoriels d'Alliance, leur différence porte sur les circuits qui sont étudiés. La réussite de ce cours, proposé à des apprenants de pays très variés, a fait que les organisateurs ont demandé à renouveler l'expérience en 1995.
En mai 1995, E. H. Rejouan est allé installer et enseigner Alliance à l'Université de Gliwice en Pologne où Alliance a été choisie comme logiciel d'enseignement de la conception VLSI . Une coopération entre notre université, l'Université de Silésie et l'Université de Tübingen fait d'ailleurs l'objet d'un projet Européen pour l'enseignement TEMPUS.

Conclusion

D'une façon analogue à la définition donnée pour le Génie logiciel, la chaîne Alliance est un ensemble d'outils et de procédures relatives aux différentes phases de l'évolution d'un circuit intégré depuis sa spécification jusqu'à sa production avec les tests, la documentation, etc... . Elle comprend une trentaine d'outils et elle est bien particulière, parce que d'une part elle a le but pédagogique concret de servir pour l'enseignement de toutes les étapes de conception d'un circuit intégré et d'autre part elle comprend des outils avancés utilisés par les industriels. La méthode et les outils proposés sont donc "à l'état de l'art" du point de vue industriel, tout en permettant aux universitaires de montrer le cycle complet de conception ou bien une étape du cycle, qu'ils ont choisi de détailler.

Alliance permet donc à la fois l'acquisition des concepts méthodologiques conduisant à la conception des circuits intégrés et le développement de nouveaux concepts. C'est le type même d'environnement technologique et pédagogique approprié à un milieu spécifique, celui des concepteurs de circuits VLSI. Les difficultés rencontrées par les apprenants sont bien évidemment liées au milieu concerné, celui des développeurs.

En conclusion il ressort qu'Alliance n'est pas du tout conçue comme un véritable produit CBL (Computer Based Learning) analogue au cours d'électronique du Consortium d'Universités anglosaxonnes EDEC [EDEC95] et que pourtant il est très utilisé (150 universités dans le monde), peut-être comme le dit PONTA [PON94] parce que : "Students like much more working with a kind of tool instead of following a straight CBL package. Playing with a tool yields more than reading a text".

S'il est vrai que dans sa version actuelle, utiliser Alliance nécessite une formation préalable à la conception de circuit, il faut aussi souligner qu'un tel enseignement ne peut pas se concevoir sans l'utilisation conjointe d'une chaîne de CAO. Or ces chaînes "commerciales" sont chères, complexes d'utilisation et souvent liées à un fabricant de circuit particulier. Elles s'adressent toujours à un public déjà compétent en architecture de circuits. Dans ce contexte, l'intérêt d'Alliance est de rendre à tous complètement accessibles (sources, structures de données et bibliothèques de cellules) des outils professionnels de conception VLSI ainsi que la méthode pour les utiliser.

Remerciements

Les auteurs remercient TOUS les membres de l'équipe CAO-VLSI du laboratoire MASI (**M**éthodologies et **A**rchitectures des **S**ystèmes **I**nformatiques), qui ont développé la chaîne Alliance et qui l'ont testée en concevant des circuits complexes.

Bibliographie

[AMD77] "Data sheet of the amd2901", 1977
[HMU90] L.V. HMURCIK, M. HETTINGER, K.S. GOTTSCHALCK, F.C. FICHTEN : "SPICE Applications to an Undergraduate Electronics Program, IEEE Trans. Educ., 1990, vol. 33, n.2, pp 183
[HAJ91] A. HAJJAR, R. MARBOT, A. GREINER, P. KIANI : "TAS. An Accurate Timing Analyser for CMOS VLSI", The European Conference on Design Automation, Feb 1991, pp 261-265
[DEM92] F. DEMAIZIERE, C. DUBUISSON : "De L'EAO aux NTF. Utiliser l'ordinateur pour la formation", Ophrys 1992
[MEN92] P. MENDELSOHN : "L'ordinateur dans l'enseignement", Actes de la troisième rencontre francophone de didactique de l'informatique, Sion, Suisse, Juillet1992, pp 53-62
[GRE92-1] A. GREINER, F. PECHEUX : "Alliance. A complete Set of CAD Tools for Teaching Digital VLSI Design", Proc Third Eurochip Workshop on VLSI Design Training, Grenoble, Septembre 1992, pp 230-237
[GRE92-2] A. GREINER, M. LAURENTIN, R. MARBOT : "DESB. A functionnal Abstractor for CMOS VLSI circuits", EURO-DAC 92, European Design Automatisation Conference, Hamburg, Germany, Septembre 1992, pp 22-27
[HEN92] J. L. HENNESSY, D. A. PATTERSON : "Architecture des ordinateurs - Une approche quantitative", McGraw Hill pour l'édition française, Paris, 1992
[ALL93] P. ALLEGRE : "Synthèse logique de circuits intégrés VLSI : Utilisation d'un générateur de système expert pour la phase de projection structurelle", Thèse de l'Université Pierre et Marie Curie, Paris 6, Décembre 1993
[BAZ93] P. BAZARGAN-SABET, A. GREINER, H N. VUONG : "Asimut A Public Domain VHDL Simulation Tool",Proc. Fourth Eurochip Workshop on VLSI Design Training, Toledo, Spain, Septembre 1993, pp 62-65
[BEN93] L. BEN AMMAR, A. GREINER : "FITPATH A Process-Independant Datapath Compiler Providing High Density Layout", IFIP Transactions, Synthesis for Control dominated Circuits, G. SAUCIER and J. TRILHE Editors, IFIP North-Holland, 1993, pp 133-151
[BUR93] L.BURGUN : "Synthèse Logique de Circuits Intégrés VLSI : Utilisation de Graphes de Décision Binaire dans la Phase d'Optimisation Comportementale", Thèse de l'Université Pierre et Marie Curie, Paris 6, Décembre 1993
[ARC94] D. ARCHAMBAUD, P. FAUDEMAY, A. GREINER : "Rapid-2 an Object-Oriented Associative Memory Applicable to Genome Data Processing", 27th Hawaii International Conference on System Sciences, Hawaii, Janvier1994, vol.5, pp 150-159
[BAZ94-1] P. BAZARGAN-SABET, J. DUNOYER, A. GREINER, M.M. ROSSET LOUERAT : " A tutorial for Advanced VLSI Course Designing the 32 bit DLX Microprocessor with the Alliance CAD system", Proc Fifth Eurochip Workshop on VLSI Design Training, Dresden, Germany, Octobre 1994, pp 120-127
[BAZ94-2] P. BAZARGAN SABET, L. BURGUN, A. GREINER, F. PETROT : "Methodology of Development of a Complete CAD System for Digital VLSI Design", VIII Simposio Brasilerio de Concepcao de Circuitos Integrados, Gramado, Bresil, Novembre 1994, pp 11-15
[BEN94] L. BEN AMMAR : "Compilation de chemins de données optimisés pour circuits VLSI", Thèse de l'Université Pierre et Marie Curie, Paris 6, Mai 1994
[BUR94-1] L. BURGUN, N. DICTUS, A. GREINER, E. PRADO LOPES, C. SARWARY : "Multilevel Logic Optimization of very high complexity circuits", EURO-DAC, Grenoble, Septembre 1994, pp 14-19

[BUR94-2] L. BURGUN, A. GREINER, E. PRADO LOPES : "A Consistent Approach in Logic Synthesis for FPGA architectures", ASICON, Pekin, Octobre 1994, pp 104-107
[GRE94] A. GREINER, L. LUCAS, F. WAJSBURT, L. WINCKEL : "Design of a high complexity superscalar microprocessor with the portable IDPS Asic Library", The European Conference on Design Automation, EDAC EURO-ASIC ETC, Mars 1994, pp 9-13
[ICTP94] Third Course on basic VLSI Design Techniques, International Atomic Energy Agency, United Nations Educational Scientific and Cultural Organization,International Centre for Theoretical Physics, Microprocessor Laboratory, Trieste, Italie, Novembre et Décembre 1994
[LAU94] M. LAURENTIN : "Abstraction Fonctionnelle de circuits intégrés CMOS", Thèse de l'Université Pierre et Marie Curie, Paris 6, Décembre 1994
[PET94] F. PETROT : "Outils d'Aide au Développement de Bibliothèques VLSI Portables", Thèse de l'Université Pierre et Marie Curie, Paris 6, Juillet 1994
[PON94] D. PONTA, G. DONZELLINI : "Learning Electronics with Hypermedia and Computer Tools", International Conference on Computer Aided Learning and Instruction in Science and Engineering, CALISCE'94, Telecom Paris, France, Aout 1994, pp 7-23
[EDEC95] Electronic Design Education Consortium, "Computer based courseware for the interactive learning of electronic design", ED&TC95, University booth demonstrations, Paris, France, Mars 1995
[LUC95] L. LUCAS : "Conception et réalisation d'un microprocesseur VLIW : Méthodologie de conception et implantation VLSI", Thèse de l'Université Pierre et Marie Curie, Paris 6,Décembre 1995
[SAR95] C. SARWARY : "Synthèse d'automates d'états fini pour les circuits intégrés VLSI : Optimisation du codage par évaluation du coût après synthèse et synthèse d'automate à pile", Thèse de l'Université Pierre et Marie Curie, Paris 6, Février 1995
[WAJ95] F. WAJSBURT : "Conception et réalisation d'un microprocesseur VLIW : Architecture interne", Thèse de l'Université Pierre et Marie Curie, Paris 6, Décembre 1995
[ZER96] B. ZERROUK, V. REIBALDI, F. POTTER, A. GREINER, A. DERIEUX : "The HIC serial link technology and associated router", Real Time Magazine,1996

Annexe 1 : Documentation en ligne fournie pour l'outil de synthèse d'automate syf.

SYF(1) CAO-VLSI Reference Manual SYF(1)

NAME SYF
- Finite State Machine synthesis.

ORIGIN

This software belongs to the ALLIANCE CAD system from the CAO-VLSI team at MASI laboratory, University P. and M. Curie. 4, place Jussieu ; 75252 PARIS Cedex 05 FRANCE
FAX: (33-1) 44.27.62.86 ; E-mail cao-vlsi@masi.ibp.fr

SYNOPSIS

syf [-sl-ml-al-hl-r] FILE [-of output_file_name] [-stack] [-pseudorom] [-scan] [-save]

DESCRIPTION

syf is a Finite State Machine synthesizer. syf allows a fast generation of VHDL Data_Flow description (for further information about this VHDL subset, read the "vbe" manual) from a VHDL Finite State Machine description (see fsm(5)). The input FSM specification can use an internal STACK. Both MOORE and MEALEY FSM can be synthesized. For a MOORE FSM, a timing-optimized implementation that emulates a ROM with microsequencer is possible. A scan-path for the state registers can be also implemented.

ENVIRONMENT VARIABLES

MBK_WORK_LIB gives the work directory path.

OPTIONS

-s Uses a "vertical" encoding algrithm.

-m Uses a "horizontal" encoding algrithm.

-a Uses a random encoding.

-h Uses an encoding given by user through <file>.cod file. In this file, all line started by a # character is not interpreted. A valid line contains one state name followed by his hexadecimal code.

-r Uses simulated annealing as state assignment algorithm.

-scan With this option syf implements a scan-path for the state registers. Refer to fsm(5) for information about scan-path description.

-of output_file_name
With this option the output file generated by SYF will be out_put_file_name. Else the output file name will be the input file name added to the encoding option. If the -save option is used the encoding result is saved in <input_file_name.cod>.

-stack This option is only useful for STACK FSM. With this option SYF optimizes the state encoding in order to minimize the stack area. This option being time-consuming is optional.

-pseudorom
This option is only available for MOORE FSM. With this option, syf emulates a ROM with micro-sequencer implementation : there is no combinatorial logic between the state registers and the FSM outputs, can be mandatory for external timing constraints.

-save With this option syf saves the encoding result in the <file(encoding algorithm identifier)>.cod. This file has the same syntax as <file>.cod file which is used by -h option.

EXAMPLE

Environment variables: setenv MBK_WORK_LIB /users/Michel/fsm

syf is called as follow (the dlx_ctrl.fsm is already created in /users/Michel/fsm) :

syf -s dlx_ctrl -save

Two files will be generated, a states encoding file dlx_ctrls.cod and a VHDL data flow file
/users/Michel/beh/dlx_ctrls.vbe

SEE ALSO vhdl.

DIAGNOSTICS

Please, report problems or bugs to cao-vlsi@masi.ibp.fr

Annexe 2 : Les outils d'Alliance.

1. Les outils de conception

1.1 L'outil de simulation logique

ASIMUT : simulateur logique acceptant en entrée une description VHDL comportementale ou structurelle du circuit. Cet outil est utilisé aussi pour valider le comportement du circuit extrait après routage [BAZ93].

1.2.Les outils de synthèse logique

SYF : outil de synthèse d'automates d'états finis. Il utilise en entrée une description VHDL, de type automate et délivre en sortie une description comportementale VHDL utilisable par la synthèse logique [BUR94-1, SAR95].

LOGIC : outil de synthèse logique . Il utilise en entrée une description comportementale VHDL et fournit une description structurelle utilisant des portes logiques d'une bibliothèque, dont le comportement logique a été préalablement spécifié [ALL93, BUR93, BUR94-1].

NETOPTIM : outil d'optimisation de netlist. Il permet d'optimiser la puissance des portes utilisées par **LOGIC** en fonction du temps de cycle souhaité , de la puissance électrique du circuit ou de la surface maximale admise pour le circuit [BUR94-1].

ALLIGATOR : outil de synthèse logique . Il utilise en entrée une description comportementale VHDL et fournit une description structurelle utilisant des portes logiques disponibles sur matrice prédiffusée[BUR94-2].

1.3.Les outils de saisie textuelle des schémas logiques

GENLIB : langage de description d'une vue structurelle ou physique d'un circuit [PET94].

GENVIEW : outil graphique de débogage pour GENLIB [PET94].

FPGEN : langage de description d'une vue structurelle d'un chemin de données (partie opérative) [BEN94].

1.4 Les outils de placement routage

SCR : routeur pour les cellules standard des bibliothèques.

DPR : routeur pour chemin de données [BEN94].

CHEOPS : routeur du cœur aux plots.

GRAAL : éditeur graphique symbolique [PET94]. Le dessin symbolique consiste à placer les objets sur les noeuds d'une grille. Il permet la portabilité technologique des bibliothèques de cellules ou de macro-cellules, entre différents fabricants, ou entre générations successives de procédés de fabrication chez un même fondeur et facilite la saisie du dessin des cellules.

1.5 La génération du dessin des masques réels

S2R : fournit la vue physique réelle d'un circuit (dessin des masques) à partir de la vue symbolique en utilisant des règles d'homothétie et de translation fonctions de la technologie considérée [PET94].

2 . Les outils de validation

DRUC : vérificateur de règles de dessin. Il prend en entrée une description de la vue physique symbolique du circuit et vérifie la conformité du dessin des masques aux règles de dessin symboliques issues des règles technologiques du fabricant de circuit.

LYNX : extrateur de vue structurelle. Il prend en entrée une vue physique symbolique, éventuellement hiérarchique et délivre une vue structurelle au niveau de hiérarchie souhaité, avec la valeur des capacités parasites d'interconnexion.

LVX : comparateur de netlists , il est utilisé pour comparer les netlists d'un circuit avant et après extraction.

DESB : abstracteur fonctionnel. Il prend en entrée une description structurelle au niveau transistor d'un circuit et fournit la description comportementale simulable par ASIMUT [GRE92-2, LAU94].

PROOF : vérificateur d'équivalence entre deux descriptions comportementales .

TAS : analyseur des performances temporelles du circuit. Sa vue d'entrée est une netlist extraite, avec les valeurs des capacités d'interconnexion à partir de laquelle il fournit en sortie un fichier de résultats sur les temps de propagation entre toutes les entrées et toutes les sorties du circuit. Il fournit en particulier les portes logiques qui constituent la chaîne critique du circuit [HAJ91].