Le systeme LUCIFER d'aide a la conception de circuits integres

J. Chailloux, J.M. Hullot, Jean-Jacques Levy, J. Vuillemin

IRIA

CENTRE DE ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél. (3) 954 90 20

Rapports de Recherche

N° 196

LE SYSTÈME LUCIFER D'AIDE À LA CONCEPTION DE CIRCUITS INTÉGRÉS

Jérôme CHAILLOUX
Jean-Marie HULLOT
Jean-Jacques LEVY
Jean VUILLEMIN

Mars 1983

LE SYSTEME LUCIFER D'AIDE A LA CONCEPTION DE CIRCUITS INTEGRES

Jérôme CHAILLOUX
Jean-Marie HULLOT
Jean-Jacques LEVY
Jean VUILLEMIN

Résumé :

Nous décrivons ici le système Lucifer d'aide à la conception, description et vérification de plans de masques de circuits intégrés. Ce système, entièrement écrit en Lisp, comprend un éditeur de structures, un éditeur graphique couleur, un vérificateur de règles de dessin, un extracteur de schémas logiques et un simulateur digital.

Lucifer a été réalisé à l'INRIA, avec la participation des chercheurs suivants : Louis Audoire (INRIA), Gérard Baudet (Brown University), Jérôme Chailloux (INRIA), Pascal Delamotte (CII-HB), Laurence Gallot (stagiaire INRIA), Christian Heintz (Thomson-LCR), Jean-Marie Hullot (INRIA), Jean-Jacques Levy (INRIA), Wing Kin Luk (stagiaire Hong-Kong), Louis Monier (Carnegie Mellon), Bertrand Serlet (INRIA), Jean Vuillemin (INRIA).

Abstract :

We present Lucifer, a CAD system for representing and checking mask level specifications of integrated circuits. This system written in Lisp, consists of a structured editor, a color graphic editor, a design rule checker, a circuit extractor and a digital simulator.

It has been developped with the participation of : L. Audoire (INRIA), G. Baudet (Brown University), J. Chailloux (INRIA), P. Delamotte (CII-HB), L. Gallot (stagiaire INRIA), C. Heintz (Thomson-LCR), J.-M. Hullot (INRIA), J.-J. Levy (INRIA), W.K. Luk (stagiaire Hong-Kong), L. Monier (Carnegie Mellon), B. Serlet (INRIA), J. Vuillemin (INRIA).

1 Introduction

L'essor de l'intégration à haute densité des circuits sur silicium (VLSI) a des implications économiques et sociologiques telles, qu'on l'a qualifié de seconde révolution industrielle. Son impact scientifique n'est pas moins important, et une mutation profonde est en cours dans :

- Les technologies de fabrication des VLSI : circuits à haute densité d'intégration.

- L'architecture de ces circuits.

- Les techniques et outils d'aide à la conception de circuits : CAO-VLSI.

1.1 Présentation du projet VLSI-INRIA

Les activités et les recherches du projet VLSI de l'INRIA, démarrées fin 1980, portent sur :

1. Le développement d'un poste de travail intégré pour la CAO-VLSI.

2. L'architecture, la conception et la réalisation de circuits expérimentaux.

3. La formation et l'encadrement de chercheurs extérieurs.

Nous renvoyons à (Vlsi-Inria 82) pour une description des réalisations du groupe concernant les points 2 et 3.

L'objectif de ce rapport est de décrire le sytème Lucifer d'aide à la conception de circuits intégrés réalisé à l'INRIA par le groupe Vlsi.

1.2 Objectifs du système Lucifer

Le système LUCIFER que nous développons vise à fournir un environnement complet d'aide à la conception de circuits. Ses caractèristiques générales et ses objectifs sont :

- Une prise en compte, dans une description hiérarchique unique, de tous les niveaux de représentation du circuit, de la spécification fonctionelle au dessin des masques.

- Un interface utilisateur commun à tous les outils de traitement, grâce à un éditeur structuré en modes graphique et alphanumérique simultanés.

- Un environnement complet d'aide à la programmation Lisp : interprète, compilateur, outils d'édition de programmes et de traces. Un système portable , sur des postes de travail économiques, à base de micro-processeurs.

- Un système ouvert et extensible, dans lequel sont construits les mécanismes permettant l'incorporation de programmes ultérieurs d'assemblage et de compilation de silicium.

1.3 Plan de cette présentation

Nous présentons en(2)l'organisation matérielle et logicielle du système, à la fin 1982. Lucifer est opérationel sur les ordinateurs et dispositifs graphiques décrits en 2.1. L'ensemble du système est écrit dans le langage Le_lisp (2.2). La description et la manipulation interactive de structure se fait au moyen de l'extension Ceyx (2.3) de ce langage.

La partie du sytème spécifique aux circuits intégrés est décrite en(3). Lucifer se compose actuellement de:

Un langage hiérarchique de description de masques (3.1), et du mode correspondant de l'éditeur de structures Ceyx.

L'éditeur graphique couleur Luciole (3.2)

D'un vérificateur de règles de dessin (3.3) complètement paramétré par la technologie.

D'un programme Extract (3.4) reconstruisant le schéma logique à partir du plan des masques.

D'un simulateur digital Mossim (3.5) de niveau interrupteur, pouvant opérer sur les données de l'extracteur.

2 Organisation matérielle et logicielle des postes de travail

2.1 Ordinateurs et sytèmes graphiques

2.1.1 Matériels opérationnels en 1982

Le système est à la fois implémenté sur Multics et sur machine Motorola 68000 Exormacs. Le choix de l'Exormacs fut simple : c'était la seule machine à base de 68000 achetable en France au début 1981. Le désavantage essentiel de l'Exormacs est son logiciel de base (Versados).

La machine Exormacs a un microprocesseur 68000, 1 Méga octets de mémoire, 6 portes séries, 2 x 16 Méga octets de disques durs. La connexion avec Multics s'effectue par une liaison série à 1200 bauds.

Nous avons également une tablette Bitpad Summagraphics, un moniteur couleur Mitsubishi, et un contrôleur Colorix connecté sur le bus de L'Exormacs qui a été conçu et développé par Louis Audoire à l'Université de Vincennes Saint-Denis. Colorix a une résolution 512 x 256 x 12. La communication entre l'Exormacs et Colorix est i/o mappée. L'avantage de Colorix est en grande partie son faible prix. De plus, il nous était facile d'interagir avec son concepteur.

2.1.2 Développements en cours

Notre expérience avec le processeur graphique couleur Colorix (L. Audoire 1980) guide le développement d'un nouveau processeur graphique, Colorix 90. C'est un élément du poste de travail de CAO VLSI qui peut, soit être intégré dans un ordinateur type SM90, soit constituer un système graphique autonome, de faible coût, connectable à n'importe quel ordinateur. Ses caractéristiques techniques sont :

- un microprocesseur 68000 réalisant les fonctions graphiques propres à l'application ;

- une mémoire-graphique d'au moins 16 plans de 1024 x 1024 points (PIXELS), faisant partie de l'espace d'adressage du microprocesseur et de ce fait, pouvant ainsi être utilisée également comme mémoire-programme (2 M octets):

- l'image visible sur le tube couleur est une fenêtre de la mémoire-graphique de 768 x 575 pixels au maximum : des fonctions permettent de changer la taille de cette fenêtre et de la déplacer dans tout l'espace de la mémoire-graphique :

- une fonction permet d'effectuer des zooms câbles dans un rapport de 1 a 16 ;

- la sortie au standard video 625 lignes a 25 images/sec. en trois composantes Rouge,Vert,Bleu (RVB).

Un prototype du système COLORIX 90 est réalisé sur quatre cartes électroniques :

- une carte processeur 68000 développée par le CNET pour la machine SM90 ;

- une carte contrôleur graphique ;

- une carte mémoire-graphique comportant 4 plans de 1024 x 1024 pixels ; il est possible de connecter plusieurs de ces cartes sur le contrôleur-graphique (le prototype fonctionne avec 4 de ces cartes, soit 16 plans) ;

- une carte table de transcodage et convertisseurs numériques/analogiques avec sortie video RVB.

2.2 Le système Le_Lisp

L'objectif d'un poste de travail intégré de CAO VLSI fonctionnant sur un micro-processeur 16 bits de type MC68000 de Motorola, et l'absence (encore aujourd'hui) sur ce type de processeur de systèmes Lisp, nous ont amené à concevoir et à réaliser complétement un nouveau système Lisp. Ce système, débuté l'année précédente (cf. le rapport d'activité du groupe VLSI en 1981), a été achevé cette année sur MC68000. Son transport sur d'autres machines a été entrepris, et sa distribution à grande échelle tant dans le monde de la recherche que dans le monde purement industriel, a été amorcée.

Ce système contient l'interprète, le compilateur, l'éditeur et les outils de développement d'un nouveau dialecte du langage Lisp, appelé Le_Lisp, fils spirituel de Vlisp (Greussay 77, Chailloux 80) auquel il a emprunté sa concision et sa rapidité dans l'interprétation, fils naturel de Maclisp (et plus précisément des "post-Maclisp Lisps" tels que le Lisp de la machine Lisp du M.I.T. (Weinreb et Moon 79), Spice Lisp (Steele 81), NIL (White 79) et Franz Lisp (Foderaro 79)) à qui il doit ses qualités en tant que langage d'écriture de systèmes expérimentaux très performants.

2.2.1 Principales caractéristiques du système Le_Lisp

La conception et la réalisation du système ont été guidées par les grandes lignes suivantes :

1 - souplesse et généralité d'un système qui doit rester au service des concepteurs de systèmes expérimentaux,

2 - transport sur des machines différentes du MC68000,

3 - efficacité tant à l'interprétation qu'à la compilation,

4 - puissance et confort des outils de développement,

5 - compatibilité avec les autres systèmes Lisp.

2.2.1.1 Souplesse et Généralité

Le_Lisp, langage pour le développement de systèmes expérimentaux, se devait d'être le plus souple possible à la fois au niveau des spécifications du langage et à celui des extensions souhaitées par les différentes catégories d'utilisateurs. La maîtrise d'oeuvre complète du système alliée à une interaction extrêmement rapide et fructueuse entre les implémenteurs et les utilisateurs nous ont permis d'atteindre cet objectif. De fait, aujourd'hui, l'utilisation du système Le_Lisp dépasse le strict cadre de la CAO VLSI et permet le développement et la réalisation d'autres systèmes de haut niveau axés sur la conception assistée par ordinateur, la simulation, la synthèse d'images colorées, l'informatique musicale, et d'une manière générale les problèmes d'intelligence artificielle et les domaines qui y sont rattachés.

2.2.1.2 Transportabilité

En vue d'assurer son transport, l'interprète Le_Lisp a été presque entièrement écrit dans le langage de la machine virtuelle LLM3 (Chailloux 82a) dérivée de la machine VCMC1 (Chailloux 78) et il est donc très proche d'un langage machine. Outre son pouvoir descriptif et la concision de son langage source, cette approche permet d'avoir une version du système "de référence". Actuellement l'interprète Le_Lisp est décrit au moyen de 11.000 lignes LLM3. Le transport proprement dit, sur MC68000 s'est effectué par macro-génération simple (au moyen du macro-assembleur fourni par Motorola). L'écriture des "macros" et de la bibliothèque des fonctions utilisées durant l'exécution a demandé de l'ordre de 4 000 lignes d'assembleur MC68000. Tout le reste du système, chargeur, compilateur, éditeur et utilitaires est écrit en Lisp et donc "bootstrappé" par l'interprète. Cette technique de macro-génération va être utilisée pour les prochains transports sur Perkin-Elmer et Intel8086.

2.2.1.3 Efficacité de l'interprète et du compilateur

L'essentiel de l'effort a porté sur l'efficacité tant au stade de l'interprétation qu'à celui de la compilation. Cette efficacité doit être mesurée en terme de place, de temps et de clarté de fonctionnement.

Au niveau de l'interprète les options suivantes ont été prises :

- liaisons dynamiques des variables en utilisant le "schéma superficiel"

- description et liaison des arguments au moyen d'arbres (Chailloux, Saint-James 1982)

- allocation optimale de la mémoire (mesurée en terme d'appel au module CONS) pour les besoins propres de l'interprète.

- classification des fonctions par type permettant un lancement immédiat de ces fonctions.

- non consommation de ressource de type mémoire ou pile dans l'interprétation des fonctions récursives terminales.

Et au niveau du compilateur :

- conservation totale de la sémantique de l'interprète

- déclarations optionnelles

- production de code LLM3, assurant le transport du compilateur.

- optimisations globales et locales (Chailloux 79)

Les résultats actuels sont très encourageants. Voici un tableau comparatif des temps d'exécution de l'interprète et du compilateur de différents systèmes Lisp. Ce test consiste en l'évaluation de (fib 20), la fonction fib étant définie de la manière suivante :

```
(def fib (n)
     (cond ((= n 1) 1)
           ((= n 2) 1)
           (t (+ (fib (- n 1)) (fib (- n 2))))))
```

nom du Lisp	Machine	tp(cy)	ti	tc
Le_Lisp 68K	(EXORmacs)	32(2)	12.0	0.56
Le_Lisp 80	Micral 80-22	16(2)	22.0	——
Maclisp	HB68 - Multics	72(2)	13.0	0.38
Lisp Machine	Symbolic LM2	36(1)	46.0	0.78
Vlisp 10.3	PDP10 (KI)	18(1)	3.2	0.45

"tp" est la taille d'un pointeur en bit, "cy" le nombre d'accès à la mémoire nécessaire à la lecture d'un pointeur, "ti" le temps à l'interprétation et "tc" le temps d'exécution compilé de (fib 20).

2.2.1.4 Puissance et confort de l'environnement

L'environnement de programmation, sans lequel les avantages d'un système hybride interprété - compilé sont complètement perdus, se compose de :

- un ensemble de macro-caractères facilitant l'entrée des commandes au niveau terminal

- un paragrapheur d'expressions Lisp

- un pisteur et un dispositif d'exécution incrémentale

- une provision de nouvelles fonctions manipulant des "buffers" dans lesquels peuvent être redirigés toutes les entrées-sorties.

- un éditeur video pleine page compatible au niveau des clés et de la sémantique attachées aux clés avec le célébre éditeur Emacs.

- un chargeur assembleur LLM3

2.2.1.5 Compatibilité

Le problème de la compatibilité avec les autres systèmes Lisp fut une préoccupation constante et majeure, tout au long de la conception du nouveau dialecte Le_Lisp. L'utilisation parallèle de Maclisp sur Multics et l'arrivée prochaine d'un VAX qui possède Franz Lisp nous ont conduit à éviter toute synonymie dangereuse au niveau des noms des fonctions, à utiliser le même mode de liaison des variables, à documenter dans le manuel de référence du système Le_Lisp (Chailloux 82a) les différences et les incompatibilités avec les autres dialectes Lisp et enfin à écrire, en Maclisp Multics, un ensemble de macro-fonctions Lisp permettant de simuler la plupart des fonctions Le_Lisp manquantes pour les Lisps de la famille MacLisp. Cet ensemble de mesures nous a permis de développer la plupart des logiciels à la fois sur MacLisp Multics et sur Le_Lisp 68K.

Ce travail a été réalisé conjointement avec le projet Formel à qui se posait également le problème du transport de gros systèmes écrits en Lisp.

2.2.2 Etat actuel et développements futurs

Le_Lisp a été transporté aujourd'hui sur les Unités Centrales de type MC68000 (Le_Lisp 68K) et Intel8080/Zilog80 (Le_Lisp 80).

2.2.2.6 Le_Lisp 68K

Ce système, le premier à avoir été terminé, fonctionne ou va fonctionner sur les machines suivantes :

- EXORmacs (de Motorola) : c'est la machine qui supporte aujourd'hui l'ensemble du logiciel "poste de travail intégré CAO VLSI"

- SM90 (du CNET) : le transport sur cette machine du noyau de l'interprète est terminé. Le transport de l'environnement et du logiciel VLSI doit être achevé à la fin de l'année 1982.

- MicroMega (Thomson) : les accords passés avec la Thomson DTI (Mr. Frehel) doivent permettre de terminer l'ensemble du transport à la fin de l'année 1982.

- d'autres accords sont en discussion en particulier sur la machine APOLLO (Brown University), Plexus 68000 et SUN.

2.2.2.7 Le_Lisp 80

Cette version du système, fonctionnant sur 8080, est une version réduite du système Le_Lisp et n'a pu être transportée automatiquement à partir de la version "de référence" LLM3 du fait de la taille extrêmement réduite de la mémoire centrale disponible et du peu de puissance du jeu d'instructions. Toutefois cette version du système est promise à une vaste distribution en particulier par l'Education Nationale qui va en équiper les micro-ordinateurs installés dans les lycées et collèges dès la rentrée prochaine. Cet accord fait l'objet d'une convention entre l'INRIA et le CNDP.

Le_Lisp 80 fonctionne aujourd'hui sur les machines :

- Micral 80-22 G (R2E) sous système CP/M

- Z89 (Zenith) sous système CP/M

- TRS80 Model I et III (Tandy), sous système TRSDOS

de nombreuses autres machines sont candidates pour recevoir ce système, entre autres, toutes les machines équipées du système d'exploitation CP/M.

2.2.2.8 Le_Lisp 86 / Le_Lisp 32

Deux autres transports sont prévus pour l'année prochaine sur les machines :

- Intel 8086 (Le_Lisp 86) grâce à un accord avec l'ADI pour en équiper les machines des DEUGs,

- Perkin Elmer 32-40 en collaboration avec l'école des Mines de Sophia.

2.3 Le système Ceyx

Le système Ceyx, conçu comme une extension structurée à Le_Lisp, fournit un environnement complet pour la représentation, la génération, le traitement et l'édition de structures.

2.3.1 Une extension à LISP : les RECORDS.

Un logiciel de base pour la création d'objets structurés en LISP a été mis au point. Il s'agit d'un précompilateur pour LISP qui permet la définition d'objets analogues aux RECORDS de PASCAL. Ce logiciel d'utilité générale fait désormais partie de l'environnement standard sur Le_Lisp 68k et fonctionne également sur MacLisp Multics (Hullot 82a).

2.3.2 Production d'arbres.

Parce qu'elle faisait suite à des systèmes tels que MENTOR (Mentor) , nous avons retenu la structure d'arbre comme structure de base de notre environnement de programmation. Cette structure est implémentée comme un RECORD particulier. L'utilisateur peut définir des arbres particuliers à l'aide de constructions LISP spéciales, analogues à celle utilisée pour la définition de records. Par exemple, il est possible de définir ainsi tous les constructeurs d'un langage de programmation. C'est de cette manière, que nous avons implémenté sous notre système, des langages tels que FLIP (Flip) et LUCIFER (cf. 3.1).

2.3.3 CEYX : un éditeur d'arbres.

Notre environnement de programmation pour la CAO VLSI a été enrichi d'un éditeur d'arbres appelé CEYX. Il permet d'éditer de manière uniforme toutes les structures arborescentes définies par les constructions précédentes. Il peut donc être utilisè, à la fois comme un éditeur de programmes à la MENTOR, et comme un éditeur de bases de données arborescentes, comme les répertoires, les bibliothèques, la documentation et toutes structures définies par l'utilisateur. Comme tous les autres processeurs du système, CEYX est écrit en LISP, et fonctionne tant sur Le_Lisp que sur MacLisp.

CEYX est fortement inspiré de l'éditeur de texte EMACS et de l'éditeur de programmes MENTOR. Il possède une fonctionalité analogue à celle de MENTOR en donnant accès à un ensemble de fonctions de base de manipulation d'arbres. Cet ensemble de fonctions définit le noyau de l'éditeur, de même qu'en EMACS le noyau est constitué par un ensemble de fonctions de manipulation de listes de chaînes de caractères. L'interface utilisateur est de type EMACS en ce sens que :

- le langage de commande est LISP : l'éditeur est ainsi extensible par l'utilisateur qui peut définir de nouvelles commandes de l'éditeur, comme des fonctions LISP appelant les fonctions du noyau de l'éditeur ;

- les clés du terminal sont attachées par programmes à des fonctions LISP ;

- l'écran, rafraîchi à chaque interaction avec l'utilisateur, montre une décompilation holophrastée de l'arbre en cours

d'édition, le curseur étant attaché au noeud courant de cet arbre;

- au point de vue ergonomique, les clefs standards ont été choisies en calquant celles de EMACS, par transposition de la notion de caractères sur celle de noeuds d'arbre. Ainsi l'apprentissage est immédiat pour un utilisateur habitué à EMACS.

Enfin CEYX et EMACS bénéficient d'un mode de communication privilegié. Un arbre, en cours d'édition sous CEYX, peut être décompilé dans un buffer texte de EMACS, édité sous cette forme textuelle et, quand un parser a été défini pour ce type de structures, être récupéré sous forme d'arbre dans un buffer CEYX. Ceci permet un type d'édition mixte arbres/forme textuelle.

3 Le système LUCIFER d'assemblage et d'analyse de circuits

Nos outils d'assemblage de silicium sont construits autour de Ceyx pour la gestion hiérarchique et l'édition sous forme alphanumérique de telles structures. Nous disposons aussi d'un éditeur graphique de la représentation des masques associés aux mêmes structures.

3.1 Le langage Lucifer de description de plans de masques

Le mot Lucifer provient de CIF (Mead-Conway 80) et Lucie (Lucie 81). Ceci indique que Lucifer est un langage de bas niveau, visant à décrire la géométrie des masques de circuits intégrés. Cependant ce langage est structuré. Un des éléments importants de structuration est la juxtaposition de rectangles : ceci permet de rendre modulaires les différents processeurs géométriques travaillant sur des structures Lucifer.

Lucifer permet la description des circuits au niveau des masques. Une expression <exp> Lucifer est caractérisée par une origine et une boite enveloppante. Elle peut être:

(masque x y w h)
(UN <exp1> <exp2> ... <expn>)
(JX <exp1> <exp2> ... <expn>)
(JY <exp1> <exp2> ... <expn>)
(TB m n <exp>)

(DS <symbol>)
(TR x y <exp>)
(ROT n <exp>)
(MX <exp>)
(MY <exp>)

Un programme Lucifer est une séquence de déclarations et une expression. Les déclarations sont de la forme:

(DEFS <symbol> <exp>)

et permettent de définir des symboles. Le plus simple pour comprendre la sémantique d'une expression Lucifer est de consulter la figure 1.

L'expression "masque" permet de définir des rectangles atomiques.

UN signifie union et permet le recouvrement de plusieurs expressions en faisant coincider leurs origines.

JX signifie la juxtaposition selon la coordonnée x. Donc l'origine est celle de la première sous-expression, et on met bout à bout les boites enveloppantes de toutes les sous-expressions.

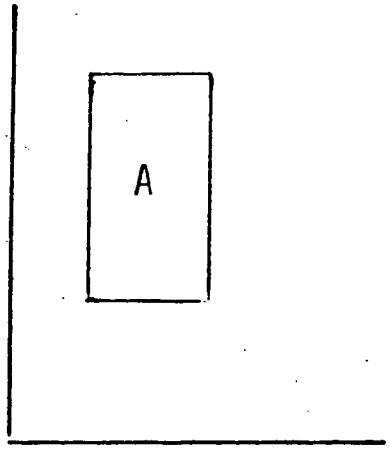
De même pour JY, mais le long de l'axe des y.

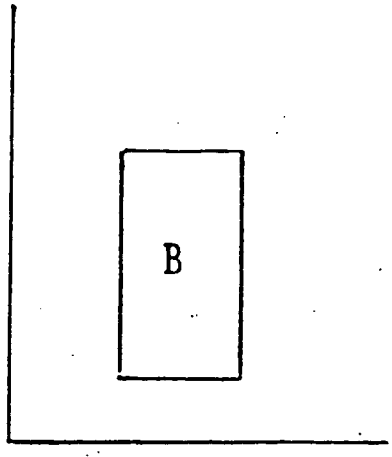
TB permet la construction de tableaux bidimensionnels.

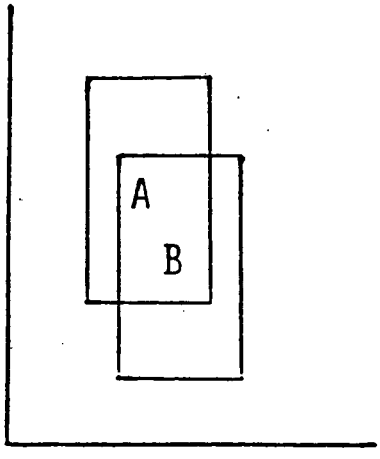
Les symboles permettent de donner des noms à une expression et d'être appelés plusieurs fois par DS (dessiner).

TR, ROT, MX, MY sont les transformations géométriques standards: translations, rotations (de multiples de 90 degrés), et opérations miroirs.

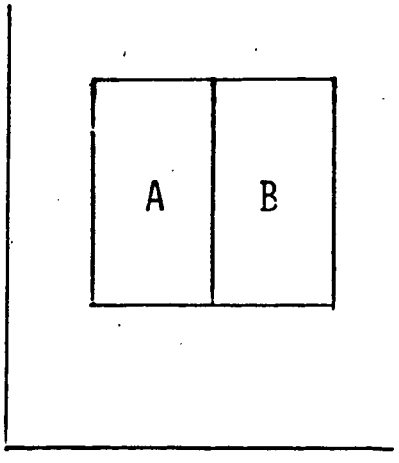
Des traducteurs de CIF en Lucifer et de Lucifer en Lucie existent. Deux circuits ont été ainsi traduits en Lucie pour être envoyés à la fabrication : un additionneur 8 bits qui est une version réduite de celui de (Vuillemin-Guibas 82), et qui a été codé en Lucifer par C. Heintz, et un circuit de calcul de fermeture transitive par B. Serlet.







(UN A B)



(JX A B)

Figure 1

3.2 L'éditeur graphique Luciole

Cet éditeur tourne actuellement sur Exormacs avec le contrôleur graphique couleur Colorix, une tablette Bitpad et une souris à quatre boutons. Le programme est entièrement écrit en Lisp, à l'exception de quelques sous-programmes d'affichage faits en assembleur et intégrés au système Le-Lisp. Les temps d'interaction obtenus dans la maquette actuelle sont relativement satisfaisants, et ceci même avec Lisp interprété. Luciole est inspiré d'Emacs Multics à la fois dans ses commandes et dans sa possibilité d'écrire des extensions. Par ailleurs, son implémentation en Lisp permet son intégration au reste du système Lucifer, et notamment à Ceyx. Cette intégration qui sera réalisée fin 82 permettra d'éditer simultanément les structures Lucifer géométriquement et sous forme arborescente. Autre extension : l'arrivée de Colorix 90 permettra à l'éditeur de travailler sur un écran de résolution supérieure (1000x1000x8 bits).

Actuellement, trois interfaces utilisateurs existent.

1) un menu fixe sur la tablette. Les boutons de la tablette sont attachés à l'asservissement d'un curseur sur l'écran et à la sé_lection d'un élément du menu. L'écran fonctionne alors en mode "flip", c'est-à- dire on calcule la nouvelle image dans la face cachée de Colorix et on bascule au dernier moment les deux faces de Colorix. (Colorix possède une mémoire 512x256 bits). On travaille donc avec un écran 256x256.

2) un menu arborescent qui est affiché sur le bord droit de l'écran. L'écran est alors en format 256x384, la partie dessin étant de 256x256. La tablette est alors partitionnée en deux. Sur la partie principale, les boutons de la souris sont attachés à l'asservissement du curseur, à l'insertion de nouveaux rectangles, et à la sélection du masque courant (qui est affiché dans le coin inférieur droit de l'écran). Dans la bande droite de la tablette, la souris permet de se déplacer dans le menu et de choisir l'élément du menu à activer. Le menu change donc dynamiquement.

3) une entrée clavier. Les commandes peuvent être alors activées par les clés du clavier d'un terminal alphanumérique standard, comme en Emacs. Par ailleurs, une commande Lisp peut à tout moment être déclanchée en restant sous Luciole.

Les commandes principales de l'éditeur permettent de manipuler différents "buffers" d'édition à partir des fichiers de l'Exormacs. La commande de base est de marquer des expressions Lucifer grâce au curseur et à une origine relative. Un cadre

apparaît sur l'écran autour des expressions marquées. La majorité des autres commandes s'effectuent par rapport à ces expressions marquées. Par exemple, la destruction de l'expression marquée sauve dans un "kill-ring" cette expression recalculée par rapport à son origine relative. L'insertion d'un nouveau rectangle s'effectue après l'expression marquée et fait coincider un des coins du rectangle avec l'origine marquée. De même pour l'insertion à partir du "kill-ring", ou l'insertion d'une expression Lucifer calculée par une évaluation Lisp.

3.3 Vérification des gardes technologiques

Nous mettons au point des programmes de vérification des règles de dessin des masques (VRD) qui soient totalement paramétrables par les règles de la technologie. Vu le volume des traitements à effectuer, ces programmes nécessitent une optimisation tout à fait poussée, du point de vue de leur efficacité algorithmique (Gallot 83).

3.4 Extraction du schéma logique à partir des masques

L'extracteur de (Heintz 82) prend en entrée une description du circuit sous forme de hiérarchie structurée de rectangles technologiques, correspondant aux couches du jeu de masques. Il produit en sortie une représentation structurée des composantes électriques (fils, connections, transistors) de ce circuit, sans en modifier la nature géométrique, qui est conservée en attribut de la structure électrique.

Un tel programme sert de base commune à trois types d'outils de CAO :

3.4.1 Adéquation du dessin au schéma électrique.

Il s'agit de vérifier que le schéma électrique extrait de la géométrie est bien identique à celui spécifié à priori par le concepteur : un processeur est à l'étude.

3.4.2 Adéquation à la technologie.

Il s'agit de vérifier que les règles électriques et géométriques imposées par la technologie sont bien respectées.

Règles électriques : connectivité, rapports d'aspects, dégradation du signal électrique.

Règles de dessin : vérifications de largeur et d'espacement, cf. supra.

3.4.3 Extraction des paramètres de simulation.

La représentation électrique issue de l'extracteur peut être transformée pour servir d'entrée à un simulateur. Ceci a été fait pour le simulateur digital MOSSIM de l'INRIA. La mèthode présente l'avantage de permettre le calcul exact des paramètres pour la simulation électrique et temporelle. La thèse présente deux méthodes de calcul des capacités des conducteurs et des rapports d'aspect des transistors.

3.5 Simulation digitale

En représentant de manière résolument digitale, le comportement d'un circuit, (chaque fil le composant est dans l'état 0 ou 1), on peut simuler une vaste partie du comportement fonctionnel. De telles simulations sont totalement hors de portée des simulateurs électriques, dont l'utilisation -nécessairement très lourde- doit être confinée aux parties électriquement ou temporellement cruciales du circuit. Ecrit en LISP par G. Huet et J-M. Hullot d'aprés (Bryant 80) , ce simulateur traite des simulations de quelques milliers de transistors en dix secondes, soit une milli-seconde par transition élémentaire du circuit simulé.

Mossim est un simulateur au niveau des transistors. Il ne fournit aucune indication temporelle. Les données du simulateur sont les sorties de l'extracteur. La simulation s'effectue en donnant différentes forces aux noeuds. Par exemple un fil "input" est plus fort qu'un noeud "pullup", qui lui-même domine un fil normal. Trois valeurs sont possibles pour chaque noeud : 0, 1 ou partage de charge (indéfini entre 0 et 1). Chaque étape de simulation consiste à geler l'état des grilles de transistors et àcalculer un état stationnaire en partant du principe que la valeur du plus fort l'emporte quand deux noeuds sont connectés.

Un traitement spécial est toutefois effectué pour les grilles de transistors à valeur indéfinie.

Une première réflexion a eu lieu sur un langage de description électrique des circuits, qui serait le langage de sortie de l'extracteur, et le langage d'entrée, à la fois des programmes de compilation de silicium, et des programmes de simulation.

Diverses idées sur un programme de simulation hiérarchique, et couvrant aussi bien le domaine de la simulation logique que de la simulation électrique, ont vu le jour, et elles devraient donner lieu à court terme à l'écriture d'un simulateur hiérarchique paramétrable.

4 Réferences

(Vlsi-Inria 82) "Rapport d'Activité 1982, I.N.R.I.A."

(Greussay 77) "Contribution à la définition interprétative et à l'implémentation des lambda-langages," P. Greussay, Thèse, Université de Paris VI, Novembre 1977.

(Colorix-90 82) "Description technique du système graphique COLORIX 90," L. Audoire, Rapport Technique, I.N.R.I.A ; (à paraître en 1983).

(Weinreb et Moon 79) "Lisp Machine Manual," D. Weinreb, D. Moon, Artificial Intelligence Laboratory, M.I.T., Cambridge, Mass., 79.

(Steele 81) "Spice Lisp Reference Manual," G. Steele, Spice Document S061, Carnegie Mellon University, July 1981.

(White 79) "NIL - a perspective," White, Proc. of the Macsyma User's conf., Washington D.C., June 1979.

(Foderaro 79) "Franz Lisp Manual," J.J.K. Foderaro, Univ. of California, Berkeley, Ca., 1979.

(Chailloux 78) "A Vlisp interpreter on the virtual VCMC1 machine," J. Chailloux, Lisp Bulletin 2, July 1978.

(Chailloux 79) "Etude d'un compilateur Lisp optimisant," J. Chailloux, Bulletin du Groupe Programmation et Langage, A.F.C.E.T., Division Théorique et Technique Informatique, No. 9, Octobre 1979.

(Chailloux 80) "Le modèle Vlisp : description, évaluation et interprétation," J. Chailloux, Thèse de 3ème cycle, Université de Paris VI, Avril 1980.

(Chailloux 82a) "Le Lisp 68K : manuel de référence," J. Chailloux, Rapport Technique I.N.R.I.A., Juillet 1982.

(Chailloux 82b) "Transport d'un système Lisp au moyen d'une machine virtuelle : spécifications et évaluations," J. Chailloux, Rapport de recherche I.N.R.I.A., 1982, (à paraître)

(Chailloux, Saint-James 82) "Du nouveau dans l'interprètation de Lisp." J. Chailloux, E. Saint-James, Rapport de recherche du L.I.T.P., à paraître, Novembre 1982.

(Teitelman 78) "Interlisp Reference Manual," W. Teitelman, XEROX PARC, October 1978.

(Hullot 82a) "Une extension à LISP : les records," J.-M Hullot, Rapport technique, I.N.R.I.A, 1982.

(Hullot 82b) "CEYX - Manuel Utilisateur," J.-M. Hullot, Rapport technique, I.N.R.I.A, 1982.

(Mead-Conway 80) "Introduction to VLSI Systems," C. Mead, L.A. Conway, Addison-Wesley, Reading, MA, 1980.

(Lucie 81) "LUCIE, Langage Universitaire de Conception de Circuits Intégrés pour l'Enseignement," A. Guyot, A. Jerraya, J. Raymond, lab. IMAG report, Grenoble, 1981.

(Vuillemin-Guibas 82) "On Fast Binary Addition in MOS Technologies," J. E. Vuillemin, L. Guibas, Proc. IEEE, Vol. 429, Sept. 1982.

(Mentor) "Programming Environments Based on Structured Editors : the Mentor Experience," V. Donzeau-Gouge, G. Huet, G. Kahn, B. Lang, Rapport de Recherche No 26, I.N.R.I.A. "Metal, un langage de spécification pour le système Mentor," B. Mélèze, TSI, AFCET, Vol. 1, No 4, Juillet-Août 82.

(Flip) "Manuel d'utilisation FLIP," G. Kahn, Rapport Technique No 2, I.N.R.I.A., Juin 81.

(Gallot 83) "Techniques informatiques dans la vérification des gardes technologiques des circuits intégrés" L. Gallot, Thèse de troisième cycle, 1983.

(Heintz 82) "Un extracteur de circuits intégrés," C. Heintz, Thèse de troisième cycle, Orsay, Oct. 1982.

(Bryant 80) "An Algorithm for MOS Logic Simulation," R.E. Bryant, LAMBDA, Vol.1, No 3, Fourth Quarter 1980.

Imprimé en France

par

l'Institut National de Recherche en Informatique et en Automatique