Une nouvelle approche pour la vérification des masques des circuits intégrés

A.A. Jerraya

THESE

présentée à

l' Institut National Polytechnique de Grenoble

pour obtenir le grade de
DOCTEUR INGENIEUR
«Informatique»

par

JERRAYA Ahmed Amine

UNE NOUVELLE APPROCHE POUR LA VERIFICATION DES MASQUES DES CIRCUITS INTEGRES.

Thèse soutenue le 24 novembre 1983 devant la commission d'examen.

F. ANCEAU	Président
P. JORRAND	
G. MAZARÉ	Examinateurs
J.P. MOREAU	
J. VUILLEMIN	

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Année universitaire 1982-1983

Président de l'Université : D. BLOCH

Vice-Président : René CARRE
Hervé CHERADAME
Marcel IVANES

PROFESSEURS DES UNIVERSITES :

ANCEAU François	E.N.S.I.M.A.G.
BARRAUD Alain	E.N.S.I.E.G.
BAUDELET Bernard	E.N.S.I.E.G.
BESSON Jean	E.N.S.E.E.G.
BLIMAN Samuel	E.N.S.E.R.G.
BLOCH Daniel	E.N.S.I.E.G.
BOIS Philippe	E.N.S.H.G.
BONNETAIN Lucien	E.N.S.E.E.G.
BONNIER Etienne	E.N.S.E.E.G.
BOUVARD Maurice	E.N.S.H.G.
BRISSONNEAU Pierre	E.N.S.I.E.G.
BUYLE BODIN Maurice	E.N.S.E.R.G.
CAVAIGNAC Jean-François	E.N.S.I.E.G.
CHARTIER Germain	E.N.S.I.E.G.
CHENEVIER Pierre	E.N.S.E.R.G.
CHERADAME Hervé	U.E.R.M.C.P.P.
CHERUY Arlette	E.N.S.I.E.G.
CHIAVERINA Jean	U.E.R.M.C.P.P.
COHEN Joseph	E.N.S.E.R.G.
COUMES André	E.N.S.E.R.G.
DURAND Francis	E.N.S.E.E.G.
DURAND Jean-Louis	E.N.S.I.E.G.
FELICI Noël	E.N.S.I.E.G.
FOULARD Claude	E.N.S.I.E.G.
GENTIL Pierre	E.N.S.E.R.G.
GUERIN Bernard	E.N.S.E.R.G.
GUYOT Pierre	E.N.S.E.E.G.
IVANES Marcel	E.N.S.I.E.G.
JAUSSAUD Pierre	E.N.S.I.E.G.
JOUBERT Jean-Claude	E.N.S.I.E.G.
JOURDAIN Geneviève	E.N.S.I.E.G.
LACOUME Jean-Louis	E.N.S.I.E.G.
LATOMBE Jean-Claude	E.N.S.I.M.A.G.

.../...

LESSIEUR Marcel	E.N.S.H.G.
LESPINARD Georges	E.N.S.H.G.
LONGEQUEUE Jean-Pierre	E.N.S.I.E.G.
MAZARE Guy	E.N.S.I.M.A.G.
MOREAU René	E.N.S.H.G.
MORET Roger	E.N.S.I.E.G.
MOSSIERE Jacques	E.N.S.I.M.A.G.
PARIAUD Jean-Charles	E.N.S.E.E.G.
PAUTHENET René	E.N.S.I.E.G.
PERRET René	E.N.S.I.E.G.
PERRET Robert	E.N.S.I.E.G.
PIAU Jean-Michel	E.N.S.H.G.
POLOUJADOFF Michel	E.N.S.I.E.G.
POUPOT Christian	E.N.S.E.R.G.
RAMEAU Jean-Jacques	E.N.S.E.E.G.
RENAUD Maurice	U.E.R.M.C.P.P.
ROBERT André	U.E.R.M.C.P.P.
ROBERT François	E.N.S.I.M.A.G.
SABONNADIERE Jean-Claude	E.N.S.I.E.G.
SAUCIER Gabrielle	E.N.S.I.M.A.G.
SCHLENKER Claire	E.N.S.I.E.G.
SCHLENKER Michel	E.N.S.I.E.G.
SERMET Pierre	E.N.S.E.R.G.
SILVY Jacques	U.E.R.M.C.P.P.
SOHM Jean-Claude	E.N.S.E.E.G.
SOUQUET Jean-Louis	E.N.S.E.E.G.
VEILLON Gérard	E.N.S.I.M.A.G.
ZADWORNY François	E.N.S.E.R.G.

PROFESSEURS ASSOCIES

BASTIN Georges	E.N.S.H.G.
BERRIL John	E.N.S.H.G.
CARREAU Pierre	E.N.S.H.G.
GANDINI Alessandro	U.E.R.M.C.P.P.
HAYASHI Hirashi	E.N.S.I.E.G.

PROFESSEURS UNIVERSITE DES SCIENCES SOCIALES (Grenoble II)

BOLLIET Louis
Chatelin Françoise

PROFESSEURS E.N.S. Mines de Saint-Etienne

RIEU Jean
SOUSTELLE Michel

CHERCHEURS DU C.N.R.S.

FRUCHART Robert	Directeur de Recherche
VACHAUD Georges	Directeur de Recherche

.../...

ALLIBERT Michel	Maître de Recherche
ANSARA Ibrahim	Maître de Recherche
ARMAND Michel	Maître de Recherche
BINDER Gilbert	
CARRE René	Maître de Recherche
DAVID René	Maître de Recherche
DEPORTES Jacques	
DRIOLE Jean	Maître de Recherche
GIGNOUX Damien	
GIVORD Dominique	
GUELIN Pierre	
HOPFINGER Emil	Maître de Recherche
JOUD Jean-Charles	Maître de Recherche
KAMARINOS Georges	Maître de Recherche
KLEITZ Michel	Maître de Recherche
LANDAU Ioan-Dore	Maître de Recherche
LASJAUNIAS J.C.	
MERMET Jean	Maître de Recherche
MUNIER Jacques	Maître de Recherche
PIAU Monique	
PORTESEIL Jean-Louis	
THOLENCE Jean-Louis	
VERDILLON André	

CHERCHEURS du MINISTERE de la RECHERCHE et de la TECHNOLOGIRE (Directeurs et Maîtres de Recherches, ENS Mines de St. Etienne)

LESBATS Pierre	Directeur de Recherche
BISCONDI Michel	Maître de Recherche
KOBYLANSKI André	Maître de Recherche
LE COZE Jean	Maître de Recherche
LALAUZE René	Maître de Recherche
LANCELOT Francis	Maître de Recherche
THEVENOT François	Maître de Recherche
TRAN MINH Canh	Maître de Recherche

PERSONNALITES HABILITEES à DIRIGER des TRAVAUX de RECHERCHE (Décision du Conseil Scientifique)

ALLIBERT Colette	E.N.S.E.E.G.
BERNARD Claude	E.N.S.E.E.G.
BONNET Rolland	E.N.S.E.E.G.
CAILLET Marcel	E.N.S.E.E.G.
CHATILLON Catherine	E.N.S.E.E.G.
CHATILLON Christian	E.N.S.E.E.G.
COULON Michel	E.N.S.E.E.G.
DIARD Jean-Paul	E.N.S.E.E.G.
EUSTAPOPOULOS Nicolas	E.N.S.E.E.G.
FOSTER Panayotis	E.N.S.E.E.G.

.../...

GALERIE Alain	E.N.S.E.E.G.
HAMMOU Abdelkader	E.N.S.E.E.G.
MALMEJAC Yves	E.N.S.E.E.G. (CENG)
MARTIN GARIN Régina	E.N.S.E.E.G.
NGUYEN TRUONG Bernadette	E.N.S.E.E.G.
RAVAINE Denis	E.N.S.E.E.G.
SAINFORT	E.N.S.E.E.G. (CENG)
SARRAZIN Pierre	E.N.S.E.E.G.
SIMON Jean-Paul	E.N.S.E.E.G.
TOUZAIN Philippe	E.N.S.E.E.G.
URBAIN Georges	E.N.S.E.E.G. (Laboratoire des ultra-réfractaires ODEILLON)
GUILHOT Bernard	E.N.S. Mines Saint Etienne
THOMAS Gérard	E.N.S. Mines Saint Etienne
DRIVER Julien	E.N.S. Mines Saint Etienne
BARIBAUD Michel	E.N.S.E.R.G.
BOREL Joseph	E.N.S.E.R.G.
CHOVET Alain	E.N.S.E.R.G.
CHEHIKIAN Alain	E.N.S.E.R.G.
DOLMAZON Jean-Marc	E.N.S.E.R.G.
HERAULT Jeanny	E.N.S.E.R.G.
MONLLOR Christian	E.N.S.E.R.G.
BORNARD Guy	E.N.S.I.E.G.
DESCHIZEAU Pierre	E.N.S.I.E.G.
GLANGEAUD François	E.N.S.I.E.G.
KOFMAN Walter	E.N.S.I.E.G.
LEJEUNE Gérard	E.N.S.I.E.G.
MAZUER Jean	E.N.S.I.E.G.
PERARD Jacques	E.N.S.I.E.G.
REINISCH Raymond	E.N.S.I.E.G.
ALEMANY Antoine	E.N.S.H.G.
BOIS Daniel	E.N.S.H.G.
DARVE Félix	E.N.S.H.G.
MICHEL Jean-Marie	E.N.S.H.G.
OBLED Charles	E.N.S.H.G.
ROWE Alain	E.N.S.H.G.
VAUCLIN Michel	E.N.S.H.G.
WACK Bernard	E.N.S.H.G.
BERT Didier	E.N.S.I.M.A.G.
CALMET Jacques	E.N.S.I.M.A.G.
COURTIN Jacques	E.N.S.I.M.A.G.
COURTOIS Bernard	E.N.S.I.M.A.G.
DELLA DORA Jean	E.N.S.I.M.A.G.
FONLUPT Jean	E.N.S.I.M.A.G.
SIFAKIS Joseph	E.N.S.I.M.A.G.
CHARUEL Robert	U.E.R.M.C.P.P.
CADET Jean	C.E.N.G.
COEURE Philippe	C.E.N.G. (LETI)

.../...

DELHAYE Jean-Marc	C.E.N.G. (STT)
DUPUY Michel	C.E.N.G. (LETI)
JOUVE Hubert	C.E.N.G. (LETI)
NICOLAU Yvan	C.E.N.G. (LETI)
NIFENECKER Hervé	C.E.N.G.
PERROUD Paul	C.E.N.G.
PEUZIN Jean-Claude	C.E.N.G. (LETI)
TAIEB Maurice	C.E.N.G.
VINCENDON Marc	C.E.N.G.

LABORATOIRES EXTERIEURS

DEMOULIN Eric	C.N.E.T.
DEVINE	C.N.E.T. (R.A.B.)
GERBER Roland	C.N.E.T.
MERCKEL Gérard	C.N.E.T.
PAULEAU Yves	C.N.E.T.
GAUBERT C.	I.N.S.A. Lyon

à ma mère

et frère et soeurs

Je tiens à remercier

- Monsieur F. ANCEAU, professeur à l'ENSIMAG, qui m'a toujours accordé sa confiance et son soutien, je lui suis très reconnaissant d'avoir accepté de présider le jury.

- Monsieur J.P. MOREAU, délégué auprès du directeur technique d'EFCIS, qui a bien voulu juger mon travail et faire partie du jury.

- Monsieur P. JORRAND directeur de recherche au CNRS, de faire partie du jury, ainsi que pour l'attention qu'il a accordé à ce travail.

- Messieurs G. MAZARE et J. VUILLEMIN, qui m'ont honoré en acceptant de participer au jury.

- Monsieur Alain GUYOT, pour ses conseils et pour les discussions enrichissantes qui ont permis l'amélioration de ce travail.

- Monsieur Christian ANTONINO, pour sa patience à réviser le français de ce texte.

- Mes collègues Jean- Pierre, Thomas, Samuel, Yves, Gilles, ainsi que tous les autres membres de l'équipe d'architecture des ordinateurs qui ont participé, par de nombreurses discussions, à l'enrichissement de ce travail.

- Madame H. DIAZ, qui à contribué à la dactylographie de ce document.

- Les membres du service de reprographie de l'IMAG, pour l'exellente qualité de leur travail.

TITRE: UNE NOUVELLE APPROCHE POUR LA VERIFICATION DES MASQUES DES CIRCCUITS INTEGRES.

COMFOR: UN EXTRACTEUR DE SCHEMA ELECTRIQUE PARAMETRABLE PAR LA TECHNOLOGIE.

RESUME.

Ce travail présente une nouvelle approche pour la réalisation d'outils de vérification des masques de circuit integrés (CI). Le système COMFOR est un extracteur de schéma électrique paramétrable par la technologie. Ce système analyse des images de CI pour reconnaitre les composants électriques et calculer leurs caractéristiques.

Le système COMFOR est basé à la fois sur des notions de programmation logique et des techniques de reconnaissance syntaxique de formes.

Le systéme COMFOR est opérationnel à l'IMAG, il permet de générer une description électrique de CI pour des simulateurs électrique (MSINC et SPICE) à partir de leurs dessins de masques décrits en LUCIE.

MOTS-CLES:

Vérification des masques de C.I.
Programmation logique.
Reconnaisssance syntaxique des formes.

ABSTRACT

This work presents a new approach in solving the device recognition problem for VLSI circuits. The COMFOR system is a general electrical circuit extractor. It operates on a layout picture to recognize the electrical components, and compute their electrical characteristics. The COMFOR system handles several technologies and works on large circuit layouts. It is based on both logic programming notions and syntactic pattern recognition techniques.

The COMFOR system has been implemented to generate a complete description for an electrical simulator.

KEYWORDS

IC layout verification.
Logical programmation.
Syntactic pattern recognition.

TABLE DES MATIERES

INTRODUCTION 7

CHAPITRE I METHODES ET OUTILS DE CONCEPTION DES MASQUES DES CIRCUITS INTEGRES 11

I.1 INTRODUCTION 13
I.2 METHODES DE CONCEPTION DES MASQUES DE CI 14
I.3 VERIFICATION DES MASQUES DES CI 19
I.4 CADRE DE L'EXTRACTEUR 21
I.5 COUT D'UTILISATION DES EXTRACTEURS 24
I.6 PRESENTATION DU SYSTEME COMFOR 25

CHAPITRE II : SPECIFICATION DES CIRCUITS ET DE LA TECHNOLOGIE 29

II.1 INTRODUCTION 31
II.2 SPECIFICATION DES CIRCUITS 32
II.3 SPECIFICATION DE LA TECHNOLOGIE 39
II.4 CONCLUSION 44

CHAPITRE III LE LANGAGE DE DESCRIPTION DES FORMES LANDFOR 45

III.1 INTRODUCTION 47
III.2 NOTIONS TOPOLOGIQUE 47
III.3 CARACTERISTIQUES DU LANGAGE LANDFOR 50

III.4 DESCIPTION DES FORMES..... 55
III.5 LES ARBRES DE FORMES..... 66
III.6 CONCLUSION. 68
ANNEXE 69

CHAPITRE IV : SYNTAXE DE FORMES : LE LANGAGE SYNFOR..... 79

IV.1 INTRODUCTION 81
IV.2 RECONNAISSANCE SYNTAXIQUE DES FORMES..... 81
IV.3 AUTOMATES POUR LA RECONNAISSANCE DES FORMES PREDEFINIES..... 94
IV.4 LA FORME INTERMEDIAIRE SYNFOR.....103
IV.5 CODAGE DE LA DESCRIPTION LANDFOR.....108
IV.6 CONCLUSION. 114

CHAPITRE V : IMPLANTATION LOGICIELLE DU SYSTEME COMFOR.....115

V.1 INTRODUCTION.....117
V.2 ORGANISATION DES PROCESSUS. 120
V.3 GESTION DES PROCESSUS. 124
V.4 SORTIE DE L'EXTRACTEUR.....131
V.5 APPLICATION.....136
V.6 EVALUATION.....141

CONCLUSION.....143

INTRODUCTION

Ce travail porte sur l'étude et la réalisation d'un extracteur de schéma électrique capable de traiter de grands circuits VLSI, et qui soit paramétrable par la technologie.

Son but principal est de traiter le maximum de technologies différentes: pour cela nous avons réalisé un système général de reconnaissance de formes particulièrement adapté à l'extraction des schémas électriques des circuits intégrés.

Nous avons défini un langage de description logique des formes (LANDFOR) qui permet de spécifier les éléments de la technologie par des prédicats. Chaque élément est décrit par un arbre de formes.

Le système fonctionne comme un système expert spécialisé. Pour cela nous nous sommes inspiré du formalisme du langage PROLOG pour présenter LANDFOR: le mécanisme (processus) de reconnaissance d'une forme est déductible de la description de celle-ci. Il décrit la syntaxe de la forme. Il sera codé en une forme intermédiaire appelée SYNFOR.

L'interpréteur SYNFOR permet de reconnaître les formes par une méthode de balayage du circuit. Il cherche à "unifier" les règles de reconnaissance avec les formes rencontrées.

Le système fonctionne comme un compilateur (COMFOR = COMpilateur de FORme):

- La description SYNFOR spécifie la syntaxe des éléments à reconnaître de la même façon qu'un compilateur.

- Le circuit est représenté par une grille, qui sera balayée point par point par le compilateur. Ce dernier va retrouver toutes les parties du circuit qui correspondent à l'une des formes de la description.

- Pour chaque forme trouvée, le système génère une arborescence qui décrit les caractéristiques de la forme. Il produit aussi la liste des connexions, et construit les équipotentielles du circuit. Les connexions sont aussi considérées comme des formes.

L'originalité de COMFOR réside dans son universalité. Bien que le langage LANDFOR soit adapté à la spécification des éléments de circuits intégrés, il permet aussi de décrire d'autres formes qui peuvent être représentées par des arbres.

L'objet de ce travail étant de réaliser un outil d'extraction de circuit, dans cet exposé, nous nous limiterons à la description du fonctionnement du système en tant qu'extracteur de schémas électriques. Les problèmes généraux de reconnaissance des formes, et d'interprétation ne seront pas approfondis.

PLAN DE LA THESE:

La première partie (chapitre I) présente les méthodes de conception des masques de circuits intégrés afin de situer le cadre du problème.

La deuxième partie (chapitre II-III) est consacrée aux méthodes de spécification des circuits et des technologies. Le chapitre III

décrivant plus particulièrement le langage LANDFOR.

Les deux derniers chapitres décrivent le modèle d'interprétation et sa réalisation.
Le chapitre IV présente la méthode de reconnaissance syntaxique de forme utilisée. Dans cette partie seule une présentation intuitive de la méthode est exposée.
Quant au dernier chapitre, il décrit la réalisation logicielle du système.

CHAPITRE I
METHODES ET OUTILS DE CONCEPTION DES MASQUES DES CIRCUITS INTEGRES.

I.1 INTRODUCTION
I.2 METHODES DE CONCEPTION DES MASQUES DE CI
I.2.1 Méthode de dessin manuel anarchique
I.2.2 Méthode semi-automatique structurée
I.2.3 Méthode automatique
I.2.4 Méthode symbolique
I.2.5 Les compilateurs de silicium.
I.3. VERIFICATION DES MASQUES DES CI
I.3.1. Erreurs contenues dans les masques
I.3.2 les règles de dessin
I.3.3 Les connexions
I.3.4 Les erreurs de performances
I.4. CADRE DE L'EXTRACTEUR
I.4.1. Dessin des masques
I.4.2. La technologie
I.4.3. La sortie de l'extracteur
I.5 COUT D'UTILISATION DES EXTRACTEURS
I.6 PRESENTATION DU SYSTEME COMFOR
I.6.1 Description des éléments de la technologie.
I.6.2 Reconnaissance des formes
I.6.3 Sortie

I.1 INTRODUCTION

L'étape de dessin des masques constitue un goulot d'étranglement dans le processus de fabrication des circuits intégrés (C.I.). Cette étape est génératrice de défauts dans les circuits, vu qu'elle est réalisée en grande partie manuellenent.

En attendant les outils de génération automatique des masques, nous sommes obligés de supporter le coût de la vérification qui augmente plus vite que la complexité des circuits [DEM-80,PED-81]. Actuellement le vérificateur des règles de dessins et l'extracteur de schéma électrique (S.E.) sont les deux outils les plus utilsées pour la vérification. Le premier détecte les violations des règles de dessin dans les masques, et le deuxième fournit le S.E qui correspond au circuit dessiné. Ce dernier résultat, est utlisé par des outils de vérification tels que les simulateurs ou les comparateurs de S.E. Certains fabricants de C.I. qui n'ont pas d'extracteurs pour traiter les grands circuits font travailler des "décodeurs" qui vérifient manuellement la conformité du dessin des masques avec le S.E.

Le problème des vérificateurs à été maitrisé. Mais celui des extracteurs l'est beaucoup moins en raison de sa complexité et de sa liaison étroite avec la technologie de fabrication des C.I. Les problémes à résoudre pour réaliser un vérificateur de règles de dessin se ramènent souvent à mesurer des distances entre des éléments du dessin [BAK-80, BAK-82, GUP-83, WAN-82 ...].

Par contre un extracteur est trés différent d'une technologie à l'autre si on considère l'aspect géometrique du dessin des masques: Le dessin d'un transistor en technologie N-MOS est différent de celui d'un transistor en bipolaire ou de celui d'une porte M-DMOS. Le premier se présente comme une superposition de deux niveaux (polycristallin, désoxydation), le deuxième comme une disposition particulière de plusieurs niveaux, et le troisième comme une

structure complexe d'intersections et de recouvrements de niveaux.

I.2 METHODES DE CONCEPTION DES MASQUES DE CI

Il existe plusieurs méthodes pour la conception des masques. Elle sont en grandes parties manuelles. Chacune nécessite un certain nombre d'outils d'aide à la conception des masques.

I.2.1 Méthode de dessin manuel anarchique

Elle consiste à faire le circuit par petites parties. Chaque partie étant dessinée sur un écran (à l'aide d'un éditeur graphique) ou esquissée sur du papier et ensuite digitalisée. Elle est dite anarchique par opposition a la méthode structurée. Aucun plan de masse n'est étudié, les parties du circuit sont assemblées au fur et à mesure qu'elles sont prêtes. Cette technique est très utilisée actuellement, et ceci, faute d'outil de CAO approprié. Elle nécessite un grand effort humain pour la conception d'un grand circuit. Les vérifications et les corrections d'un circuit dessiné complètement à la main sont très coûteuses.

Il faut vérifier tout le circuit, une fois déssiné, à l'aide d'un extracteur et d'un vérificateur de règles de dessins (V.R.D).
L'extracteur doit être utilisé pour chaque bloc digitalisé afin de vérifier que le dessin du circuit correspond au schéma électrique de départ. L'extraction du circuit complet et sa simulation au niveau logique permettent de détecter les autres types d'erreurs. Bien que cette dernière utilisation puisse paraître coûteuse, elle permet généralement d'éviter la fabrication de prototypes qui ne fonctionnent pas.
Par contre, cette méthode permet d'optimiser la surface.

L'expérience nous montre que les erreurs de dessin et de connexion sont inévitables pour les grands circuits quand ils sont dessinés manuellement.

I.2.2 Méthode semi-automatique structurée

Elle consiste en une conception structurée du circuit. celui-ci est décomposé en plusieurs blocs fonctionnels. Chaque bloc est conçu indépendamment des autres. Le circuit est composé par des programmes de placement automatique des blocs. Les connexions sont aussi générées automatiquement par des programmes de routage. La conception d'un bloc peut être décomposée de la même manière.

Cette technique est appelée semi-automatique car la décomposition est effectuée manuellement. D'autre part, les programmes existant ne permettent pas d'effectuer un placement et un routage complet. Une partie doit être faite manuellement.

Dans cette démarche, le dessin manuel des masques est limité aux cellules. Certaines parties sont générées automatiquement comme les PLAs et les ROMs. Le circuit constitue une structure hiérarchique de cellules.

Cette méthode accélère la conception du CI. Par contre elle provoque une perte dans la surface du circuit. L'apport de cette méthode réside dans la structuration de la conception.

La représentation hiérarchique du circuit permet la réduction de la masse des données manipulées par les programmes de vérification. Elle permet surtout de développer des programmes qui utilisent cette hiérarchie.

La vérification des circuits est alors moins coûteuse. Elle consiste surtout à vérifier les cellules de base qui sont dessinées manuellement.

Un examen complet du circuit n'est pas nécessaire, car, le comportement global du circuit peut être déduit du comportement des cellules qui le composent.

I.2.3 Méthode automatique

Elle utilise une bibliothèque de cellules standards préconçues. Chaque cellule représente une fonction de base telle que multiplexeur, additionneur, latch, etc... Ces cellules sont testées et sumulées une à une. La bibliothèque peut contenir aussi, en plus des dessin des masques la description des connexions et des caractéristiques électriques des cellules.

Une description logique du circuit est utilisée pour générer automatiquement les dessins de masques par un algorithme de placement routage en utilisant la bibliothéque. Ceci évite toutes les interventions manuelles dans la phase d'assemblage du circuit. Aucun outil de vérification des masques n'est alors nécessaire pour la totalité du circuit.

Actuellement, il existe des systèmes qui utilisent cette méthode. Ces systèmes sont appliqués à la conception de petits circuits, car ils provoquent une diminution énorme dans la densité d'intégration.

I.2.4 Méthode symbolique

Les nouveaux systémes d'aide au dessin des masques ont une entrée symbolique. Les cellules sont saisies à partir d'un dessin de symboles. Cette entrée peut être une description STICK [MEA-80] ou même le schéma électrique de la cellule. Le système génère automatiquement le dessin des masques correspondant en applicant les règles de dessin de la technologie.

Cette technique concerne uniquement la mise au point des cellules. Certains outils offrent la possibilité d'optimiser topologiquement les dessin générés.

La conception des cellules est alors plus rapide, et si le système fonctionne bien, les cellules ne contiennent aucune erreur de

dessin. La classification des cellules dans une bibliothèque ou dans une base de données est alors plus facile. Ces outils peuvent générer automatiquement les caractéristiques électriques et géométriques (connexion) des cellules.

Ces nouveaux systèmes nécessitent une nouvelle catégorie d'outils de vérification capable de travailler sur une entrée symbolique. Les outils doivent être assez rapides pour être utilisés interactivement, ou même être intégrés au système de saisie.

I.2.5 Les compilateurs de silicium.

Les compilateurs de silicium constituent la nouvelle génération d'outils pour le dessin des masques. Ils produisent le dessin des masques à partir d'une description de haut niveau.
Actuellement les compilateurs de silicium appartiennent au domaine de la recherche, et nous sommes loin de voir les dessins de masques d'un CI VLSI complètement généré par ordinateur. Il existe des outils qui produisent des parties particulières des dessins des masques tels que les générateurs de PLA et de ROM.

Nous présentons brièvement le système CAPRI [ANC-82, ANC-83] qui est un compilateur de silicium en cours de développement par l'équipe de recherche en architecture des ordinateurs de l'IMAG.
CAPRI (Conception Assisté de PRocesseurs Intégrés) s'applique à tous les CI se décomposant en une partie opérative et une partie de contrôle.
La conception d'un CI à l'aide de la méthodologie CAPRI est décomposée en quatre étapes:

- Déterminer l'architecture du circuit. Cette étape définit la partie de contrôle et la partie opérative à partir d'une description comportementale du circuit.
- Générer le dessin des masques de la partie opérative.
- Générer le dessin des masques de la partie de contrôle.
- Assemblage des parties opératives, parties contrôle, et des plots d'E/S.

Le système CAPRI est constitué d'un langage de description de matériel appelé IRENE et d'un ensemble d'outils qui permettent de passer de la description IRENE au dessin des masques.
Les principaux outils du système CAPRI sont:

- Le langage de description IRENE qui permet la description structurelle et comportementale des circuit sans ambiguité.
- Un extraceur qui définit les structures de la partie opérative et de la partie de contrôle à partir de la description IRENE du circuit.
- Un générateur de la partie opérative.
- Un générateur de la partie de contrôle.
- Un optimisateur de PLA.
- Un évaluateur topologique.
- Un assembleur de cellules.

Ces outils définissent un ensemble de cellues de bases avec leurs caractéristiques topologiques et électriques. Certaines de ces cellules sont générées automatiquement tels que les PLA ou les ROM. Les autres sont dessinées manuellement à l'aide du système d'aide au dessin LUCIE [GUY-81].
Le système LUCIE contient aussi un ensemble d'outils qui sont essentiellement:

- Le langage LUCIE pour la description structurée des dessins de masques.
- Un éditeur graphique.
- Le langage LUCIEN qui est une extension du langage LUCIE.
- Un programme pour le dessin symbolique squelettisés LUSTICK.
- Un vérificateur des règles de dessin VERDICT.
- Nôtre extracteur de schéma électrique COMFOR.

Tous les outils du système LUCIE sont indépendant de la technologie de fabrication des CI dessinés.

La conception de circuits par la méthodologie CAPRI suit un modéle

descendant. Tandis que celle des masques suit un modèle ascendant. Les cellules de bases sont déssinée en premier, et puis assemblées pour former de nouvelle cellules jusqu'à l'obtention du circuit complet. L'assemblage des cellules est fait par le programme LUBRICK [SCH-83] qui est un outil de CAPRI.

I.3. VERIFICATION DES MASQUES DES CI

Comme nous l'avons dit ci-dessus, actuellement, le dessin des masques est en grande partie fait manuellement. Les interventions manuelles provoquent inévitablement des erreurs dans le dessin des masques. Le circuit peut aussi contenir des erreurs qui proviennent des étapes précédentes de la conception. Ces erreurs sont rares. C'est comme durant la phase de mise au point des programmes: Les erreurs sont généralement causées par un mauvais codage de l'algorithme et non à la conception de l'algorithme lui-même. Dans ce qui suit nous parlerons uniquement des erreurs qui surviennent durant l'étape de dessin.

I.3.1. Erreurs contenues dans les masques et outils de vérifications

Les erreurs sont classées en trois types, selon les outils qui les détectent. On distingue les erreurs provenant d'une violation des règles de dessin, les erreur de connexions, et les erreurs de fonctionnalité.

Les vérificateur de règles de dessin détectent les premières, les secondes sont trouvées par les comparateurs de schémas électriques, et les vérificateurs statiques, et enfin, les simulateurs sont nécessaire pour découvrir les dernières.

I.3.2 les règles de dessin

Chaque technologie de fabrication de CI impose un ensemble de contraintes géométriques auxquelles doit obéir les dessins des masques, appelées règles de dessin. Elles déterminent les tailles minimales des éléments d'un circuit et les espacements nécessaires entre ces éléments.

La violation de ces règles constitue l'erreur la plus courante dans les dessins des masques. Heureusement pour les concepteurs, elles sont les plus faciles à trouver. Il existe actuellement plusieurs programmes pour la vérification des règles de dessin [BAK-82,] qui sont assez performants.

I.3.3 Les connexions

Ces erreurs sont facilitées par les outils d'aide au dessin utilisés. Au moment de la digitalisation d'une cellule, il arrive souvent que l'on oublie un rectangle ou que l'on change un niveau pour un autre. Ou encore, en utilisant un éditeur graphique on peut déplacer un rectangle en dehors de la fenêtre éditée et ne pas le retrouver par la suite. Ce genre d'incident peut provoquer des violations des règles de dessin, mais surtout des erreurs de connexions telles que des courts-circuits et des circuits ouverts.

Pour détecter ces erreurs il faut retrouver le schéma électrique (ou logique) qui correspond au dessin des masques. Ce travail est réalisé par un extracteur. Ensuite on analyse le schéma électrique extrait pour déterminer ces erreurs. Il existe une autre pratique courante de vérification. Elle consiste à comparer le schéma électrique extrait avec le schéma électrique original, on considère alors que toutes les différences sont des erreurs.

I.3.4 Les erreurs de fonctionnalité

Ces erreurs n'empêchent pas le bon fonctionnement logique du circuit fabriqué résultant, mais ce dernier ne répond pas aux caractéristiques électriques demandées. Elles résultent d'un mauvais choix des paramètres de certains composants, tel que un dimensionnement trop faible.. Elles peuvent être détectées par la simulation du schéma électrique extrait.

Ces erreurs sont beaucoup plus rares mais leur détection est trés coûteuse vu le coût des simulateurs électriques. Nous assistons aujourd'hui au développement de nouveaux outils tels que les vérificateurs statiques [MAL-83] et les simulateurs à formalisme d'interrupteurs (switch level simulator en anglais) qui coûtent moins cher à l'utilisation et permettent de détecter la plupart des erreurs de performances.

I.4. CADRE DE L'EXTRACTEUR

Dans ce paragraphe nous décrivons les considérations qui doivent être prises en compte pour la conception d'un extracteur. Un extracteur permet de retrouver le schéma électrique à partir des dessins de masques. Le processus d'extraction dépend de la technologie de fabrication. D'autre part l'extracteur n'est pas un outil de vérification proprement dit, car il ne détecte pas les erreurs dans les masques. Par contre, la sortie de l'extracteur, soit le schéma électrique, est utilisée par des programmes de vérification.

I.4.1. Dessin des masques

La première tâche de l'extracteur consiste à lire le circuit. Cette opération suppose une certaine représentation interne du circuit, qui soit bien définie. La plupart des extracteurs actuels utilisent

une représentation du circuit qui est différente de celle utilisée dans les systèmes d'aide au dessin pour deux raisons. La première est que la représentation des systèmes de dessin n'est pas toujours bien adaptée pour l'algorithme de l'extracteur. De plus, elle contient beaucoup d'informations inutiles telle que la structure du dessin. La deuxième raison est la recherche de l'indépendance de l'extracteur par rapport aux systèmes de dessin. L'utilisation d'un format d'entrée simplifié par un extracteur facilite son utilisation avec plusieurs systèmes de dessin différents.

Le système COMFOR représente le circuit par une grille de points. Dans le chapître suivant nous présenterons cette méthode de représentation ainsi que les méthodes utilisées par les autres extracteurs.

I.4.2. La technologie

Elle définit les caractéristiques électriques des niveaux de masques et les formes des éléments qu'on doit retrouver dans le circuit. Les extracteurs sont classés en trois catégories selon la manière de représenter les spécifications de la technologie:

- l'extracteur spécialisé: Il utilise des mécanismes de reconnaissance fixes qui ne peuvent pas être modifiées. Il ne peut donc extraire que les circuits d'une technologie bien définie.

- l'extracteur souple: Il fonctionne pour une famille de technologies. Il fixe une partie des spécifications de la technologie, tout en laissant à l'utilisateur la possibilité de définir certains paramètres.
  Par exemple, si l'on suppose que tous les circuits contiennent des transistors et que chaque transistor est composé d'une grille, d'une source et d'un drain, et que l'on permet à l'utilisateur de définir les niveaux des grilles, des sources et des drains, on pourra traiter

uniquement la famille des circuits MOS.

- l'extracteur général: Il ne pose pas de contraintes sur la forme et le nombre d'objets à reconnaître. La technologie est spécifiée par un langage adapté. Dans cette catégorie nous pouvons distinguer deux classes: Les extracteurs dont le langage permet uniquement la spécification de la topologie des éléments de la tecnologie, et ceux qui utilisent un langage capable de décrire les interconnections et les paramètres électriques.

Nous verrons les langages de spécification de la technologie avec plus de détail dans le chapitre suivant.

I.4.3. La sortie de l'extracteur

Comme nous l'avons dit auparavant, l'extracteur fournit une entrée pour des outils de vérification. Sa sortie doit donc être adaptées à l'outil qui va l'utiliser.

Un comparateur de schéma électrique nécessite uniquement une description du réseau électrique. Un vérificateur statique éxige la taille des éléments en plus. Un simulateur électrique demande aussi un calcul précis des capacités et des résistances du circuit.
Chaque type d'information impose un traitement spécial qui peut coûter cher en temps CPU. La plupart des extracteurs travaillent en plusieurs étapes. La première consiste à calculer le réseau électrique et génère en même temps plusieurs fichiers contenant des données pour faciliter le calcul des autres informations. Les autres phases sont optionnelles, elles constituent des post-traitements pour le calcul des informations restantes. Cette décomposition implique une augmentation du temps de calcul car plusieurs opérations sont exécutées plusieurs fois (la lecture du circuit par exemple).

Le système COMFOR travaille en une seule passe qui calcule des

informations à la demande. En fait la spécification de la technologie en LANDFOR englobe la description des paramètres à calculer, et la description des connexions. Elle détermine même le format de la sortie.

I.5 COUT D'UTILISATION DES EXTRACTEURS

Les extracteurs sont réputés par leur coût d'utilisation qui est très élevé et parfois même inabordable. Ceci vient du fait que les gens qui en parlent, pensent toujours à extraire des circuits contenant des milliers de transistors, bien que la plupart des outils de vérification actuels soient incapables de traiter de tels circuits.
Aujourd'hui, seuls les comparateurs de schéma électrique sont capables de manipuler des gros circuits. Il faut noter aussi que les outils de vérification et les outils d'aide au dessins coûtent plus cher que l'extracteur lui-même. Par exemple, La simulation d'un circuit coûte beaucoup plus cher que son extraction.

Comme les méthodes de conception actuelles sont quasi-manuelles, les concepteurs sont engagés à utiliser les schémas électriques correspondant à toutes les parties du circuit dessinées manuellement, afin de les vérifier. Si on n'utilise pas un extracteur, ce schéma électrique doit être extrait manuellement, ce qui est non seulement coûteux et long, mais est aussi une grande source d'erreurs. Nous connaissons un constructeur qui a du extraire manuellement un circuit de 15.000 transistors pour le vérifier.

Nous n'avons pas assez d'informations sur les extracteurs qui sont utilisés par des industriels pour pouvoir en parler. Par contre il existe plusieurs publications sur ceux developpés dans des milieux universitaires. Leurs performances sont trés variés. Les extracteurs spécialisés les plus rapides peuvent traiter jusqu'a 40.000 transistors par heure CPU (sur un VAX/780) [GUP-83]. Nous

pensons que ces temps sont mal calculés car ces extracteurs super-rapides sont encore à l'état de prototype. Les temps declarés dans les publications correspondent au temps d'exécution d'une seule étape de l'extracteur qui, en général, fonctionne en plusieurs étapes. Par contre l'extracteur japonais [WER-82] qui fonctionne réellement nécessite 10 heures CPU pour extraire un circuit de 20.000 transistors (sur un ordinateur ACOS-700).

Le temps d'exécution d'un extracteur dépend essentiellement de deux facteurs:

- La taille ou la complexité du circuit selon l'algorithme de l'extracteur.
- La complexité des formes à reconnaitre.

Pour les extracteurs spécialisés le deuxième facteur est constant. Comme les formes des éléments sont connu d'avance, les mécanismes de reconnaissances des formes peuvent donc être optimimisés. Ce qui fait que les extracteurs spécialisés sont plus rapides que les extracteurs généraux pour une technologie donnée.

I.6 PRESENTATION DU SYSTEME COMFOR

Le système COMFOR analyse des images d'un CI pour identifier ses composants électriques et aussi calculer leurs caractéristiques. Il est indépendant de la technologie, et peut traiter les grands circuits.

Le système COMFOR est basé à la fois sur des notions de programmation logique et sur des techniques de reconnaissance syntaxique de formes.

COMFOR contient un language de description logique de formes (appelé LANDFOR) qui permet de décrire les éléments de la technologie par des prédicats. IL contient aussi un langage de description syntaxique de formes (appelé SYNFOR) qui définit des processus de reconnaissances pour les différents éléments de la

technologie décrits en LANDFOR.

I.6.1 Description des éléments de la technologie.

Nous nous sommes inspiré de la programmation logique, et surtout du langage PROLOG [COL-79,WAR-81], pour definir LANDFOR. Il permet la description des formes par des prédicats (ou termes). Un prédicat est une propriété topologique telle qu'une disposition particulière de couleurs. Par exemple un croisement des deux couleurs "rouge" et "vert" est une disposition de couleurs. Les couleurs "rouge" et "vert" sont des valeurs constantes, le croisement des deux couleurs est un terme (ou prédicat).

Une forme est décrite par un prédeicat comme suit:

```
B_R_V  debut
         BLEU; ROUGE; VERT
       fin
```

Le prédicat B_R_V décrit une forme ayant une couleur de fond bleue et contenant du vert et du rouge. Ce prédicat est localement vrai pour chaque région bleue contenant du vert et du rouge. Dans cette description les couleurs peuvent être remplacées par des prédicats plus complexes.

Une forme composée (qui inclut des sous-formes) est décrite en LANDFOR par un arbre, dans lequel la racine décrit la forme de fond (la couleur de fond), et les descendants (les sous-arbres) decrivent des sous-formes.

Nous considèrons qu'un circuit contient deux types d'éléments: Les conducteurs et les composants.
Un conducteur est un fil conducteur au sens électrique tel que un chemin de métal.
Un composant représente un élément complexe du circuit tels que un transistor ou une diode. Les contacts sont décrits comme étant des composants.

La description d'un élément contient la spécification de sa forme, et la liste des caractéristiques qu'on veut calculer pour cet élément, telle que sa résistance ou sa capacité. La description d'un composant peut aussi contenir la description des interconnexions. Tous ces éléments sont décrit par des prédicats LANDFOR.

Comme en PROLOG une description LANDFOR a une interprétation déclarative et une interprétation procédurale. La première définit quels sont les prédicats qui peuvent être déduits vrai dans une partie du circuit. La deuxième définit comment on reconnait une forme, autrement dit, les étapes à suivre pour prouver un prédicat. Cette dernière définit un processus de reconnaissance [WAN-82] qui est une suite d'opérations élémentaires à effectuer pour retrouver un forme donnée. Ce processus correspond au mécanisme d'unification de PROLOG.

I.6.2 Reconnaissance des formes

Le processus d'unification d'une forme est déduit de sa description LANDFOR. Pour le prédicat B_R_V ce processus est donné par la figure I.1.

L'identification d'une forme simple (qui ne contient pas de sous-formes) est faite par un automate. Ce dernier peut retrouver une forme simple en suivant une syntaxe donnée sur une représentation du ciruit par une grille de points. Toute les formes sont reconnues en un seul balayage du ciruit. Les processus de reconnaissance des différents éléments travaillent en parallèle.

Une forme composée (qui contient des sous-formes) est identifiée par plusieurs processus, chacun reconnait une forme simple. Le processus qui reconnait la forme de fond contrôle ceux qui font la reconnaissance des sous-formes.

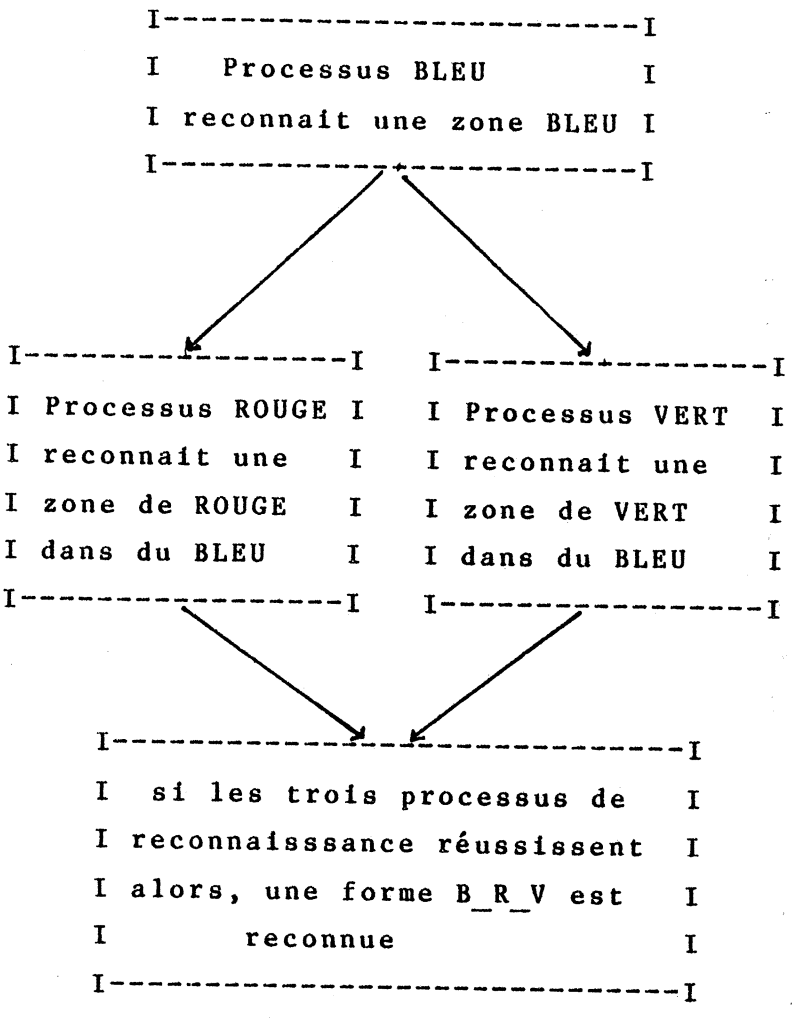


Figure I.1 Le processus de reconnaissance de le forme B_R_V.

Les processus de reconnaissance sont décrit par un langage appelé SYNFOR qui décrit la syntaxe des formes.

I.6.3 Sortie

Chaque processus qui réussit, génère une description de la forme qu'il a reconnu. A la fin du balayage on va avoir la liste des composants et des conducteurs du circuit ainsi que la liste des interconnexions. Cette sortie est traitée pour produire des descriptions du circuit pour les outils de vérifications, ou pour dessiner son schéma électrique.

CHAPITRE II
SPECIFICATION DES CIRCUITS ET DE LA TECHNOLOGIE

II.1 INTRODUCTION
II.2 SPECIFICATION DES CIRCUITS .
II.2.1 Représentation sous la forme de figures élémentaires
II.2.2 Représentation par une grille
II.2.3 Représentation par des vecteurs
II.2.4 La spécification des circuits dans COMFOR .
II.2.5 Décomposition du circuit
II.3 SPECIFICATION DE LA TECHNOLOGIE .
II.3.1 Pourquoi un LST
II.3.2 Les graphes de contours
II.3.3 Les arbres de décisions .
II.4 CONCLUSION .

II.1 INTRODUCTION

L'analyse des masques de CI se heurte à deux difficultées: la grande quantité d'informations à manipuler et le temps de calcul important qui est nécessaire a ce traitement. Ces difficultés deviennent de plus en plus grandes avec l'évolution technologique qui touche plusieurs aspects des CI, en particulier l'augmentation de la taille des puces et la densité d'integration.

Nous trouvons actuellement sur le marché des CI possédant plusieurs dizaines de millier de transistors pour une suface de quelques dizaines de millimètres carré. D'où l'augmentation de la précision necessaire au dessin des masques. La plupart des technologies actuelles, utilisées dans l'industrie, emploient comme unité de dessin le 1/2 micron.

L'exemple des trois derniers circuits IAPX fabriqués par INTEL montre bien cette évolution.

IAPX	IAPX43201	IAPX43202	IAPX43203
surface en micron2	66 000 000	74 000 000	75 000 000
nombre de transistors	110 000	49 000	60 000

tableau II.1 Tailles des trois circuits de l'IAPX432

L'ensemble de ces 3 circuits est dessiné par 500000 rectangles.

La taille et la complexité d'un CI poussent les programmes d'analyse des masques aux limites des ordinateurs actuels sur deux points: la capacité de stockage et le temps de calcul.

Les outils d'analyse des masques, doivent aussi résoudre le problème de l'évolution et la diversité des technologies de fabrications des C.I.
Jusqu'à la fin des anneés 70 chaque outil d'analyse était conçus pour une technologie particulière c.à.d. pour traiter des C.I. qui ont une certaine topologie. Toute modification de la technologie rend cet outil inutilisable. D'où la nécessité d'avoir des programmes plus généraux et indépendants de la technologie qui possèdent des langages pour la spécification de la technologie.
Il existe actuellement certains outils pour la vérification des règles de dessins qui sont indépendants de la technologie. Mais en ce qui concerne les extracteurs, quelques prototypes seulement sont en cours de développement dans des milieux universitaires. A nôtre connaissance aucun extracteur universel n'a été utilisé dans un contexte industriel.

Ce chapitre décrit les méthodes les plus utilisées pour la spécification des circuits et deux langages pour la spécification de la technologie. Nous exposerons aussi la méthode de spécification des circuits utilisée dans COMFOR.

II.2 SPECIFICATION DES CIRCUITS.

Il existe essentiellement trois méthodes de représentation des masques de C.I. Elles consistent, toutes les trois, en une "mise à plat" de la description du circuit, c.à.d. la transformation de la représentation du C.I en une liste d'objets élémentaires indépendamment de la manière dont le circuit à été structuré.
L'objet élémentaire peut être:
- une figure géometrique (rectangle ou polygône),
- un point sur une grille (appelé pixel),
- un vecteur.

II.2.1 Représentation sous la forme de figures élémentaires

Le dessin des différents niveaux de masques est décrit par un ensemble de figures élémentaires telles que des rectangles, des polygônes, des cercles ... .
Cette méthode permet d'obtenir une représentation compacte, actuellement, elle est la plus utilisée dans les programmes d'analyse de masques.

L'analyse des masques ainsi représentés, se fait par des algorithmes qui sont de l'ordre de N**K où N est le nombre de figures et $1 \leq K \leq 2$. Il existe des méthodes d'optimisation pour ce genre de traitement et l'on arrive à diminuer la valeur de K; On trouve dans [BAI-76] une discussion détaillée de ces méthodes. Ces algorithmes nécessitent un temps de calcul très important car les opérations sur des rectangles ou sur des polygones sont très coûteuses.

Ces algorithmes [HEI-82,WAN-82,YOS-81] sont limités au traitement des petits circuits qui ne comportent pas plus d'un millier de composants.

II.2.2 Représentation par une grille

Le dessin de masques est représenté par une grille fine de points (appelés pixels) [BAK-80] chacun représente une surface d'une unité carrée du circuit. L'unité dépend de la précision du dessin. Chaque point est décrit par l'ensemble des niveaux qu'il contient.

Cette représentation permet l'analyse des masques à l'aide d'algorithmes simples basés sur des traitements locaux dans le circuit. Ces algorithmes consistent en un balayage de la grille point par point. Le temps de calcul est alors proportionnel à la

taille du circuit. Cette méthode est également limitée aux petits circuits surtout à cause de la taille de la grille de plus elle s'adapte difficilement aux diagonales ou aux obliques. Elle peut être aussi utilisée pour des circuits de taille moyenne dessinés avec des règles standardisées (règles Mead et Conway) [MEA-80], qui sont surtout utilisées dans les milieux universitaires.

II.2.3 Représentation par des vecteurs

Les figures de base sont décomposées en vecteurs [BAI-76], ceux-ci sont éclatés pour tenir compte de la superposition des niveaux de masques, et ensuite, ils sont triés. L'algorithme d'analyse se traduit par un parcours des vecteurs triés, en calculant des propriétés géométriques. Ces opérations sont plus simples que celles effectuées sur les rectangles et polygônes mais plus complexes que les opérations sur les points. La complexité de l'algorithme de traitement est de l'ordre de C*E où E est le nombre de vecteurs. Cette méthode nécessite une grande mémoire de travail qui, elle aussi, est proportionnelle à E. D'autre part, la transformation des figures en vecteurs et leur tri sont coûteux pour de grands circuits.

Les deux dernières représentations (pixel et vecteurs) ne représentent que le masque. La première (rectangles) représente égalenent la façon dont dont le masque à étè conçu: Elle contient des informations inutiles (bords de rectangles qui ne sont pas bords de niveau) cee qui est nuisible pour le traitement.

II.2.4 La spécification des circuits dans COMFOR.

La méthode proposée dans le système COMFOR est une combinaison des représentations par grille et par vecteurs: le circuit est représenté par une grille compactée [JAN-79]. Dans chaque ligne on ne prend en compte que les points différents de leurs voisins, et on ne prend en compte que les lignes différentes de leurs voisines.

L'ordre des lignes définit les ordonnées des points et la position du point dans la ligne définit son abscisse. La quantité d'informations enregistrées est proportionnelle au nombre de figures de base qui constituent le dessin, et non plus à la surface du circuit.

Pour un circuit dessiné uniquement par des rectangles, soit N le nombre de rectangles et H la hauteur moyenne d'un rectangle (la hauteur est mesurée en nombre de lignes de la grille), le nombre de points enregistrés est égal à:

$$K \times N \times (H/L)$$

Dans une ligne de la grille on ne prend en compte que les points qui représentent des extrémités de rectangle. En fait on ne prend en compte que les extrémités de zône d'où $K <= 2$.

L est le coéfficient de répétition d'une ligne.
H et L dépendent de la précision du dessin. Il faut noter également que L tend vers 1 quand la taille de la figure augmente. Le tableau suivant donne des tailles de représentation de trois circuits dessinés par les membres de l'équipe d'architecture d'ordinateur.

nom du circuit	CIRC	REBON	POPY
nombre de rectangles	8 323	8 531	23 119
surface en unités de dessin	866x443= 383 638	703x954= 670 622	1187x1143 =1 356 741
taille de la représentation en rectangles	49	51	150
taille de la représentation par une grille	154	219	489
temps de transformation en seconde.	17	22	63

tableau II.2 Tailles des différentes représentations.

Les tailles de représentations sont données en Kmot mémoire. L'unité de dessin est le lambda. Le temps représente des secondes CPU sur un HB68 avec le système MULTICS.

II.2.5 Décomposition du circuit

La grille est définie par une suite de lignes où :

- chaque point pris en compte est représenté par les niveaux qu'il contient et par sa position.
- chaque ligne contient deux marqueurs:
  - début de ligne (DL) contient le numéro de la ligne,
  - fin de la ligne (FL).

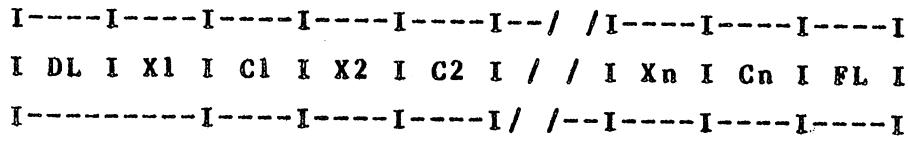


Fig II.1 :Codage d'une ligne de la grille.
Xi= abscisse du point i.
Ci= niveaux contenus dans le point i.

La génération de la grille se fait par bandes, chaque bande contient un certain nombre de lignes successives (hbande). hbande est calculé selon la taille de la mémoire de travail (M) et la taille de la figure : pour un circuit de taille (deltax, deltay), hbande est égal au plus grand entier h tel que M >= h.deltax.

Au départ le circuit est représenté par un fichier contenant la liste des figures qui constituent le dessin des masques. Il est divisé en sous-fichiers représentant chacun une bande de circuit. Vu la hauteur (deltay) des grands circuits on ne peut pas avoir un sous-fichier par bande, de hauteur hbande. Le nombre de sous-fichiers maximum (maxfich) est fixé au départ. On utilise donc une méthode dichotomique pour accélérer la division du circuit.

Algorithme

Etape 1 - Diviser le fichier principal en sous-fichiers qui contiennent des bandes de hauteurs successives deltay/2, deltay/4 ...., hbande,hbande(fig.II.2)

Etape 2 - Pour chaque sous-fichier correspondant à une bande de hauteur <= hbande, générer la partie correspondante de la grille, la compacter.

Etape 3 - S'il reste un sous-fichier qui représente une bande de

hauteur > hbande :

- la diviser avec le même procédé utilisé dans l'étape 1,
- aller à l'étape 2.

Cet algorithme permet une décomposition rapide du circuit en utilisant un nombre réduit de fichiers intermédiaires. Soit N le nombre de figures du circuit, nous obtenons :

- le nombre d'écritures de figures sur les sous-fichiers =

  C x (nombre de sous-fichiers utilisés) x N.

  C provient du fait que si une figure appartient à plusieurs bandes à la fois, elle est mise dans tous les sous_fichiers correspondants. C augmente quand hbande diminue.

- le nombre de lecture = C'x (nomre de fichier utilisé)/2+1 xN .

- la taille du circuit doit être telle que

  M >= (detay/2 maxfich-1) deltax

  <=>

  M x 2**(maxfich-1) >= deltax x deltay

Avec 16 fichiers intermédiaires et 20 kmots de mémoire, on peut donc traiter les derniers circuits d'INTEL dessinés avec une précision de 1/2 micron. Il faut noter que pour des circuits ayant une taille voisine du maximum prévu, l'algorithme devient coûteux vu le nombre d' E/S et le nombre d'initialisations de fichiers qu'il doit effectuer. Il est mieux d'augmenter la taille de la mémoire que d'augmenter le nombre de fichiers intermédiaires.

La table II.2 donne des indications sur le temps d'exécution de cet algorithme. Il faut noter que cette méthode de décomposition ne nécessite pas le tri des rectangles.

II.3 SPECIFICATION DE LA TECHNOLOGIE.

Electriquement un C.I. est un ensemble de composants interconnectés entre eux par des conducteurs. Un composant peut être un transistor, une diode etc... Spécifier une technologie consiste à décrire les formes des composants, des conducteurs, et des connexions.

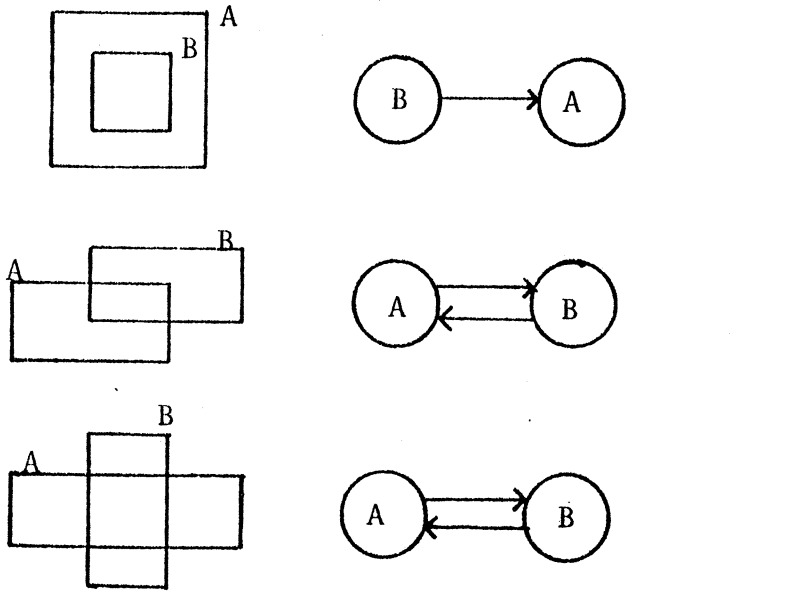
II.3.1 Pourquoi un langage de spécification de la technologie (LST)

Il permet de d'établir une correspondance entre la topologie des éléments d'un circuit et leurs fonctions électriques. Tandis que le but d'un extracteur est de retrouver les formes de ces éléments dans le dessin des masques. Pour qu'un extracteur soit général il faut qu'il travaille sur une description de la technologie. Un LST doit pouvoir décrire aussi bien les transitor MOS que les résistances en technologie bipolaire. Il peut aussi définir la sortie de l'extracteur, c.à.d définir la finesse de la description électrique demandée ce qui rend le coût d'utilisation de l'extracteur plus économique dans le sens où l'on va calculer uniquement les éléments qui seront utilisés par les outils de vérification.

Avant de décrire le langage LANDFOR, nous présentons ci-dessous deux extracteurs parmis ceux que connaissons, essentiellement à partir des publications. Ils ont chacun un langage pour la spécification de la technologie. Le premier utlise des graphes de contour pour spécifier les formes des élements de la technologie, Tandis que le deuxième utilise des arbres de décision.

II.3.2 Les graphes de contours

L'extracteur LAAS [BARK-82] décrit les éléments de la technologie par une structure de graphe orientés. Chaque noeud du graphe représente un contour. Les arcs décrivent des relations entre contours.



figures graphe correspondants

FIG II.2 Figure de bases dans LAAS.

Les relation topologique de bases qu'il peut décrire sont le recouvrement et l'intersection. Un composant est décrit par une combinaison de ces deux relations.

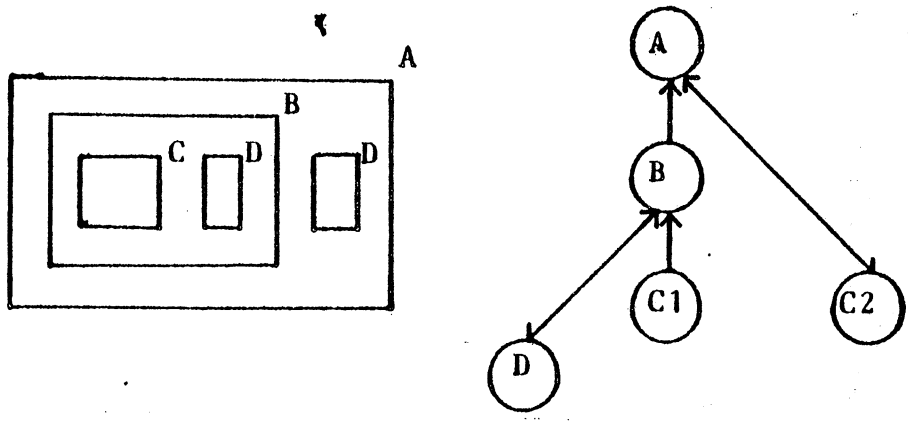


FIG II.3 Graphe de description.

Ce formalisme permet de décrire plusieurs technologies bipolaire. Mais il n'est pas bien pratique dans le cas des technologie MOS car il ne fait pas de différence entre les différents types d'intersections, par exemple, on ne peut pas spécifier un croisement. Egalement il ne peut pas spécifier un composant avec des éléments variables tel qu'un transistor multi-drain. pour celà il faut décrire toutes les occurences possibles du transistor.

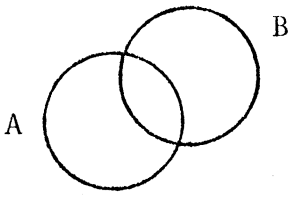
D'autre part il considère que le circuit contient deux niveaux spéciaux qui sont un niveau conducteur et un niveau contact. Ceci implique des traitements spéciaux pour les contacts et les conducteurs. Finalement ce formalisme ne tient pas compte des paramètres électriques.

Pour l'extraction du schéma électrique d'un circuit, il transforme le dessin des masques en un graphe orienté qui décrit toutes les relations qui existent entre les niveaux. Il essaie ensuite de retrouver tous les sous-graphes qui correspondent à des éléments de la technologie. Cette dernière opération est trés coûteuses car il fait les recherches dans le graphe du circuit entier. Il utilise aussi un traitement trés coûteux pour construire le graphe

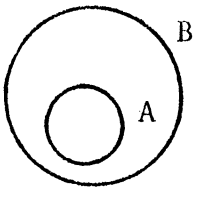
d'interconnexion.

II.3.3 Les arbres de décisions.

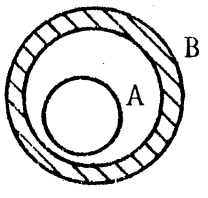
Dans le deuxième extracteur que nous décrivons [WAT-82] les éléments de la technologie sont spécifiés par des arbres de décisions. Un arbre décrit des relations entre les niveaux de masques. Il définit quatre relations pour décrire des formes de bases (FIG II.5)



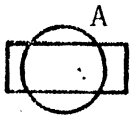
INTERSECTE/A,B



COUVRE/B,A



ENTOURE/B,A



DIVISE/A,B

FIG II.4 formes de bases.

L'utilisateur doit définir un arbre de décision pour chaque composant qu'il veut reconnaitre.

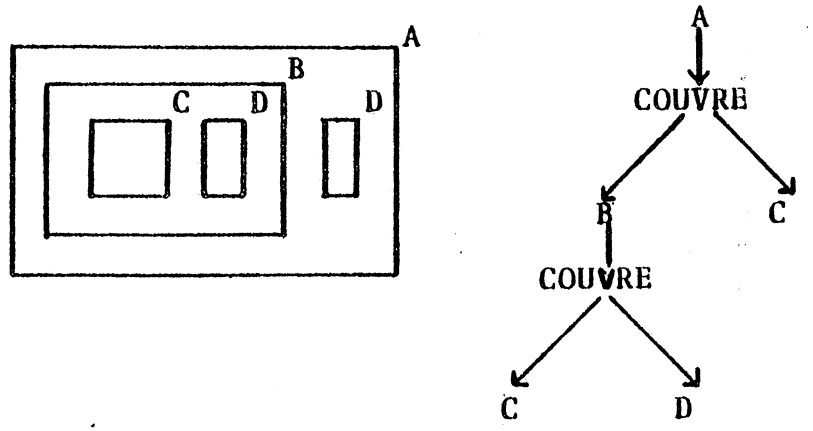


FIG II.4 Exemple d'arbre de décision.

L'algorithme de cet extracteur est semblable au précédent. Il décompose le circuit en un graphe où les sommets sont des relations et les arcs des figures, et puis il extrait toutes les les parties du graphe qui s'unifie avec l'un des arbres de décision de la description.

Ces arbres de décision permettent la description et la reconnaissance des composants sans ambiguité. Par contre il ne permettent pas de spécifier les connexions et les paramètres électriques.

L'extracteur fonctionne aussi en plusieurs étapes. Les étapes de calcul des paramètres électriques et de calcul des interconnexions sont faites par des post-processeurs, elles dépendent de la technologie. Ceci limite alors l'indépendance du système vis-à-vis de la technologie. L'algorithme est aussi très coûteux et nécessite une grande mémoire de travail vu qu'il travaille sur le graphe du circuit complet en mémoire.

II.3.4 CONCLUSION.

La structure classique d'un extracteur se compose de quatre modules:

- Le module de classification des élements du circuit: La reconnaissance des composants. Cette opération est dans certain cas précédée par d'autres étapes de prétraitements.
- Le module de calcul des connexions et formation du reseau électrique.
- Le module de calcul des paramétres électriques.
- Le module de construction d'une description électrique du circuit pour un simulateur électrique particulier.

Les extracteurs dit indépendant de la technologie n'ont que la première étape qui soit réellement indépendante de la technologie. Ceci vient du fait que leur LST ne tiennent pas compte des caractéristiques électriques des composants et des connexions. Pour adapter ces extracteurs à une nouvelles technologie, il faut alors réécrire les trois derniers modules qui sont aussi complexes que le premier.

Pour éviter ce problème il faut pouvoir décrire les paramétres qu'on veut calculer dans la spécification de la technologie. dans le cas de LANDFOR ces paramètres sont décrits par des expressions de formes comme les composants eux même. Les paramètres constituent des sous-formes dans la description d'un composant.

CHAPITRE III
LE LANGAGE DE DESCRIPTION DES FORMES : LANDFOR

III.1 INTRODUCTION.
III.2 NOTIONS TOPOLOGIQUE.
III.2.1 Les couleurs.
III.2.2 Composition de couleurs.
III.2.3 Les frontière.
III.2.5 Arbres de formes
III.3 CARACTERISTIQUES DU LANGAGE LANDFOR
III.4 DESCIPTION DES FORMES.
III.4.1 Description de topologie.
III.4.1.1 Les couleurs.
III.4.1.2 Les frontières.
III.4.1.3 Les formes complexes.
III.4.1.4 Le croisement.
III.4.1.5 Le recouvrement.
III.4.1.6 La jointure.
III.4.2 Description des connexions.
III.4.2.1 Les liaisons.
III.4.2.2 Les noeuds.
III.4.3 Description des caractéristiques des formes.
III.4.3.1 La surface
III.4.3.2 La longueur.
III.5 LES ARBRES DE FORMES.
III.6 CONCLUSION.

III.1 INTRODUCTION.

Un CI peut être interprété d'une manière topologique et d'une manière électrique. La première décrit ce qu'on voit sur son dessin des masques. La deuxième donne sa fonction électrique.

Topologiquement un CI est constitué par plusieurs plans superposés. Chaque plan contient un niveau de masque. Le nombre et la signification de ces niveaux dépendent du procédé de fabrication du CI.

Electriquement un CI est un ensemble de composants interconnectés. Un composant peut être un transistor, une diode etc... . Des chemins conducteurs lient ces composants. Une équipotentielle est constituée par un ensemble de conducteurs connectés entre eux.

Le langage de spécification de la technologie (LST) permet d'établir une correspondance entre la topologie des éléments d'un circuit et leurs fonctions électriques.

III.2 NOTIONS TOPOLOGIQUE.

Le dessin des masques d'un CI est constitué par plusieurs plans géometriques supeposés. Chaque niveau de masque définit un plan. On dit q'un niveau est présent en un point du circuit s'il existe une figure élémentaire (rectangle ou polygône) rattachée à ce niveau qui contient le point. Un point du circuit correspond à un élément de la grille qui représente le circuit.

III.2.1 Les couleurs.

Pour l'interprétation des dessins de masques il est important de pouvoir utiliser des combinaisons de niveaux. En fait un niveau de

masque peut être interprété différemment s'il est en présence d'autres niveaux. En général dans les outils d'analyses de masques (V.R.D. et extracteur) la notion de niveau est étendue à celle de couleur. On considère alors que:

- Chaque niveau de masque constitue une couleur.
- La superposition de plusieurs niveaux de masques contituent une nouvelle couleur.
- L'absence de niveau définit une couleur particulière appelée la couleur de fond du circuit.

III.2.2 Composition de couleurs.

Une couleur peut être définie par une expression de niveau à l'aide des opérateurs 'et', 'ou', et 'non'.

Exemple de couleurs:

ROUGE et VERT
ROUGE ou JAUNE
ROUGE et non (VERT ou JAUNE)

On dit qu'une couleur est présente en un point du circuit si et seulement si l'expression qui la définit est vérifiée en ce point.

III.2.3 Les frontières.

Quand deux plages de couleurs sont adjacentes elles définissent une frontière. Une frontière est définie par 2 couleurs.

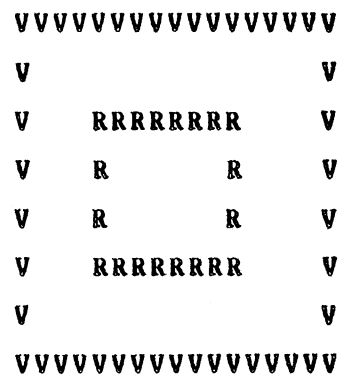


FIG III.1 Exemple de frontière.

La couleur rouge est entourée d'une frontière définie par les 2 couleurs: (rouge et vert) et (vert et non rouge).

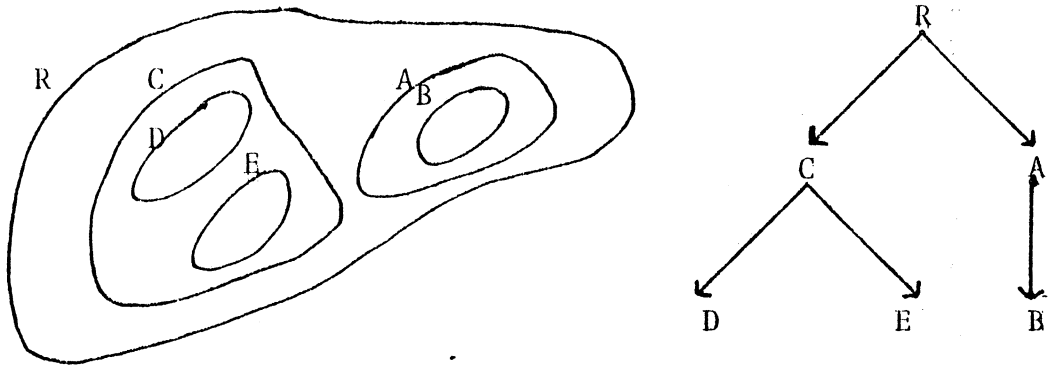
III.2.4 Dispositions de couleurs.

La plupart des extracteurs actuels se limitent aux notions de couleurs ou de frontières pour la description des composants des CI Ces notions sont insuffisantes pour décrire sans ambigùité les formes des composants d'un CI.
Une disposition de couleur decrit une région ayant une certaine couleur, délimité par un ensemble de frontière.
LANDFOR utilise des formes qui décrivent des dispositions de couleurs telque un croisement ou un recouvrement. Ces formes seront définies par des prédicats (couvre, croise joint).

III.2.5 Arbres de formes

Nous utilisons aussi la notion d'arbre pour définir des formes composées. Un arbre est un ensemble de sommets tel qu'il existe un sommet spécial appelé racine, les autres sommets sont partitionnés en m>=0 sous-ensembles disjoints. chaque sous-ensemble constitue un arbre, appelé sous-arbre de la racine.

Le nombre de sous-arbres d'un sommet est appelé le degré de ce sommet. un sommet de degré zéro est appelé feuille.
Un arbre décrit un motif composé d'une forme de fond, et d'un ensemble de sous-formes, qui peuvent elles même être décrites par des sous-formes.



(a) (b)

fig III.2 Représentation d'une forme par un arbre.

La figure ci dessus donne un exemple de représentation d'une forme par un arbre.

III.3 CARACTERISTIQUES DU LANGAGE LANDFOR

LANDFOR (LANgage de Description de FORmes) est un langage sructuré pour la description logique des formes par des prédicats. Un prédicat est une propriété topologique. Par exemple un croisement des deux couleurs 'ROUGE' et 'VERT' est une disposition de couleurs, elle est décrite en LANDFOR par le prédicat:

ROUGE croise VERT

Les couleurs ROUGE et VERT sont des valeurs constantes. 'croise' est un prédicat défini dans le langage LANDFOR.
Ce prédicat est localement vrai pour chaque région du circuit contenant un croisement des couleurs ROUGE et VERT. Dans cette

description les couleurs ROUGE et VERT peuvent être remplacées par des termes plus complexes.

Avant d'aller plus loin dans la présentation du langage LANDFOR il nous est paru utile de définir quelques termes qui serons utilises par la suite.

1) On appelle couleur toute combinaison de niveaux de masques. En particulier chaque niveau définit une couleur simple. Des couleurs composées peuvent être définies par des expressions de niveaux.

2) On apelle tache ou région une zone monochrome du circuit composée par un ensemble de points ayant la même couleur. Exemple: Une tache de couleur ROUGE désigne un ensemble de points voisins du circuit ayant la couleur ROUGE.

3) Une forme décrit une propriété topologique, elle définit un ensemble de points voisins du circuit ayant une certaine propriété. Une tache est une forme. Un croisement est une forme il décrit une tache délimitée par un ensemble de frontières.

On appelle forme simple un motif qui n'en englobe pas d'autres. Telle qu'une tache, une frontière, ou un croisement de deux couleurs. LANDFOR contient un ensemble de prédicats prédéfinis pour décrire les formes simples.

Une forme composée (qui contient des sous-formes) décrite un arbre de formes. La racine de l'arbre décrit la forme de fond (comme couleur de fond) pour ses descendants (les sous-formes). Les noeuds terminaux de l'arbre (feuilles) décrivent des formes simples. Un arbre de forme est décrit par un prédicat composé qui représente un arbre de prédicats simples où les sommets de l'arbre décrivent des sous-formes, des connexions, ou des paramètres électriques.

On considère qu'un CI contient 2 types d'éléments: les conducteurs et les composants. Un conducteur représente un fil au sens électrique tel qu'un chemin de métal par exemple. Un composant représente un élément complexe du circuit comme un transistor ou une diode.

Chaque élément (composants et conducteurs) est décrit par un prédicat composé.

Les connexions sont décrites comme étant des sous-formes de composant. Par exemple un contact entre deux niveaux est décrit par un composant qui contient une sous-forme prédefinie appelé liaison. Elle définit une connexion électrique entre les deux niveaux.

Les paramètres électriques sont aussi décrit comme des sous-formes dans les description des composant et conducteurs.

La spécification d'une technologie en LANDFOR détermine la description:

- des niveaux et des couleurs,
- des conducteurs,
- des composants.

Syntaxe :

```
<DESCRIPTION>   ::= "technologie" <NOM> <DECLARATION>
                    <ELEMENTS> "fin"
<DECLARATION>   ::= <COULEUR_SIMP> <COULEUR_COMP>
<ELEMENTS>      ::= <LIST_COND> <LIST_COMP>
<LIST_COND>     ::=
                ::= <CONDUCTEUR> <LIST_COND>
<LIST_COMP>     ::=
                ::= <COMPOSANT> <LIST_COMP>
```

Un conducteur est spécifié par un prédicat spécial appelé conducteur.

Syntaxe :

⟨CONDUCTEUR⟩ ::= "conducteur" ⟨NOM⟩
"debut" ⟨COULEUR⟩ ⟨LISTE-FONCTIONS⟩ "fin"

La forme d'un conducteur est par définition une tache. Les sous-formes contenu dans un conducteur décrivent des fonctions de calcul de paramètres électriques.

La spécification d'un composant est aussi faite par un prédicat spécial appelé composant.

Syntaxe:

⟨COMPOSANT⟩ ::= "composant" ⟨NOM⟩
"debut" ⟨FORME⟩ ⟨PARAM⟩ "fin"

Exemple:

```
technologie AA
   niveaux -------
   couleurs ------
                ------
   conducteur XX
       debut
              ------
       fin
    ------
   composant YY
       debut
              ------
       fin
    ------

fin
```

La description des niveaux et couleurs définit des constantes qui seront utilisées pour la description des autres éléments de la technologie. La description d'une technologie en LANDFOR doit au moins contenir la declaration des niveaux de masques (couleurs de bases) utilisées.

Syntaxe:

```
<NIVAUX>          ::= 'niveaux' <LISTE_DE_NOM>
<LISTE_DE_NOM>    ::= <NOM>
                  ::= <NOM> ',' <LISTE_DE_NOM>
<COULEURS>        ::= 'couleur' <NOM> '=' <COULEUR><LISTE_DE_COUL>
<LISTE_DE_COUL>:::=
                  ::= ',' <NOM> '=' <COULEUR> <LISTE_DE_COUL>
<NOM>             ::= <CARAC> <CHAINE>
<CHAINE>          ::=
                  ::= <NOM>
                  ::= <CHIFFRE> <CHAINE>
<CARAC>           ::= 'A'  'B' .....  'Z'
<CHIFFRE>         ::= '0'  '1' .....  '9'
<COULEUR>         ::= <TERME_COUL>
                  ::= <TERME_COUL> <OPER_NIV> <COULEUR>
<TERME_COUL>      ::= '(' <COULEUR> ')'
                  ::= <NOM>
                  := 'non' <TERME_COUL>
<OPER_NIV>        ::= 'et'
                  ::= 'ou
```

Nous donnons ci-dessous un exemple de description des niveaux et des couleurs pour une technologie NMOS (enrichissement déplétion), grille poly, avec précontacts.

Les niveaux sont:

- désoxydation (MD)
- polycristallin (MP)

-aluminium (MM)
-implantation ioniques (MI)
-contact (MC)
-dépassivation (MG)

La déclaration des niveaux s'ecrit:

niveaux MD, MP, MM, MI, MC, MG

Exemple de déclaration de couleurs:

couleur DIFFUSION = (MD et non MP) ou (MD et ME)
CANAL = MP et MD et non (ME ou MC)
CONTACT = (MP ou MD) et MM et MC

III.4 DESCIPTION DES FORMES.

LANDFOR contient trois types de prédicat prédéfinies pour décrire:

- la topologie des éléments de la technologie.
- des connexions entre les éléments.

et des fonctions prédéfinies pour décrire les paramètres électriques.

Les prédicats et les fonctions prédéfinis déterminent l'ensemble des formes de bases manipulées par LANDFOR. Elles constituent Les prédicats primitifs du langage, et sont utilisées pour composer des arbres de formes.

III.4.1 Description de topologie.

Les prédicats de description de la topologie spécifient des formes simples. Ils sont au nombre de cinq dans LANDFOR:

- la tache (une zone ayant une certaine couleur),
- la frontière entre deux deux couleurs,
- le croisement entre deux couleurs,
- le recouvrement pour deux couleurs,
- la jointure de deux couleurs.

III.4.1.1 Les taches.

Une tache est décrite par une couleur. Elle spécifie une zone monochrome du circuit. Ce prédicat est vrai dans toutes les parties du circuit ayant cette couleur.

III.4.1.2 Les frontières.

Le prédicat frontière décrit une forme prédéfinie. Il est défini par deux couleurs.

Syntaxe :

<FRONTIERE>::='frontiere' '(' <COULEUR> ',' <COULEUR>')'

Exemple:

frontière(ROUGE et non VERT, ROUGE et VERT)

Ce prédicat est vrai dans toutes les parties du circuit contenant une intersection de ROUGE de VERT tel que le VERT ne recouvre pas le ROUGE.

III.4.1.3 Les formes complexes

Les taches de couleurs et les frontières sont considérées comme des formes simples. Les predicats de croisement, de recouvrement, et de jointure permettent de décrire des formes composées. Ces prédicats peuvent être composés entre-eux ou avec d'autres formes simples pour décrire des arbres de formes. La description d'une forme complexe est composée de deux parties:

- la description de la topologie de la forme,
- la description des paramètres à calculer pour cette forme.

Syntaxe:

```
<FORME_COMPLEXE>::='debut'<EXP_FORME> <PARAM> 'fin'
<PARAM>          ::=
                 ::= <LISTE_PARAM>
<EXP_FORME>      ::= <FORME>
                 ::= <FORME> 'sur' <COULEUR>
<FORME>          ::= <TERME>
                 ::= <TERME> <RELATION_FORM> <TERME>
<RELATION_FORM>  ::= 'couvre'
                 ::= 'croise'
                 ::= 'joint'
<TERME>          ::= <COULEUR>
                 ::= <LISTE_FORME>
<LISTE_FORME>    ::=
                 ::= <FORME_COMPLEXE> ';' <LISTE_FORME>
```

LANDFOR contient trois prédicat pour décrire des dispositions de couleurs, ils sont croise, couvre et joint. Ces prédicat peuvent prendre comme paramètre des formes simples ou des formes composées. Dans le dernier cas ils décrivent des arbres de formes. Dans le cas ou les paramètres sont des couleurs, ils décrivent des formes prédefinies.

III.4.1.4 Le croisement.

Le croisement en tant que forme prédéfinie est défini par trois couleurs. Les 2 couleurs qui se croisent et la couleur de l'intérieur du croisement.

Exemple: ROUGE croise VERT

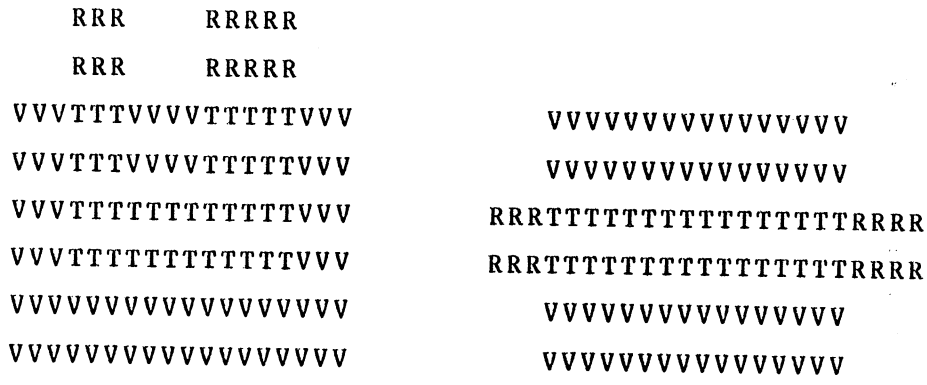


FIG III.3 Exemple de croisement.

R = ROUGE

V = VERT

T = (ROUGE et VERT)

Ce prédicat est vrai dans toutes les parties du circuit contenant une tache de couleur (ROUGE et VERT) entourée par les quatres frontières:

- deux frontiéres definies par les deux couleurs (ROUGE et non VERT, ROUGE et VERT),
- et deux frontiéres definies par les deux couleurs (VERT et non ROUGE, ROUGE et VERT),

Dans ce cas l'intérieur du croisement est une tache de couleur (VERT et ROUGE). On peut décrire des croisements tels que leur intérieur contient des couleurs supplémentaires.

Exemple: ROUGE croise VERT sur JAUNE

Le mot clef sur permet d'indiquer que l'intérieur du croisement contient du JAUNE.

En utilisant les niveaux de masques et les couleurs définis plus haut pour la technologie NMOS, nous pouvons décrire avec précision les transistors enrichis et deplétés de cette technologie.

```
composant  TRANSISTOR_ENRICHI
           début
               MP croise MD sur ( non MI et non ME)
           fin

composant  TRANSISTOR_DEPLETE
           début
               MP croise MD sur (MI et non ME)
           fin
```

III.4.1.5 Le recouvrement.

Le prédicat prédéfini recouvrement est défini par deux couleurs: L'intérieur et l'extérieur du recouvrement.

Exemple: ROUGE couvre VERT.

```
RRRRRRRRRRRRRRRRR
RRRRRRTTTTTTTTRRRR
RRRRRRTTTTTTTRRRRR
RRRRTTTTTTTTRRRRRR
RRRRRRRRRRRRRRRRR
```

fig III.4 exemple de recouvrement.

Il est évalué à la valeur 'vrai' dans chaque partie du circuit contenant une tache de couleur (ROUGE et VERT) entourée par une frontière de (ROUGE et non VERT, ROUGE et VERT). La première couleur définit l'extérieur du recouvrement, et la deuxième définit son intérieur.

III.4.1.6 La jointure.

Nous avons appelé jointure un troisième type d'intersection possible entre deux couleurs.

```
RRRRRRRRRR              VVVVVVVVVVVVVV
RRRRRRRRRR              VVVVVVVVVVVVVV
RRRRTTTTTTTVVVVVV       VVVVVTTTTTTTTTRRRRRR
RRRRTTTTTTTVVVVVV       VVVVVTTTTTTTTTRRRRRR
RRRRRRRRRR              VVVVVVVVVVVVVV
RRRRRRRRRR              VVVVVVVVVVVVVV
```

FIG III.5 Exemple de jointure.

Une jointure est définie par trois couleurs comme le croisement. Par contre l'intérieur de celle ci est entourée de deux frontières uniquement.

Exemple: ROUGE joint VERT sur JAUNE.
Ce prédicat est vérifié dans chaque partie du circuit qui contient une tache de couleur (ROUGE et VERT et JAUNE) entourée par les deux frontières:

-frontière(ROUGE et non VERT, JAUNE et ROUGE et VERT).
-frontière(VERT et non ROUGE, JAUNE et ROUGE et VERT).

La jointure permet de décrire certaines formes sans ambiguité avec le croisement. Par exemple différencier une capacité et un transistor NMOS

III.4.2 Description des connexions.

Dans un schéma électrique (S.E) les composants et les conducteurs

sont interconnectés entre eux par deux types de connexions: les liaisons et les noeuds. Une liaison connecte deux conducteurs entre eux, et un noeud connecte un composant et un conducteur. Les connexions et les paramètres électriques sont décrits comme des sous-formes de composants.

Syntaxe :

```
<LISTE_PAR>      ::=
                 ::= ';' <NB_REPETION> '*' <NOM>'='<CONNEXION> <LISTE_PA
                 ::= ';' <NB_REPETION> '*' <NOM>'='<FONCTION> <LISTE_PAR
<NB_REPETITION>::=
                 ::= '*' co indique que le nombre de répétition
                           est variable >=1 co
                 ::= '-1 *' co le nombre de répétition est <1,
                           c.à.d la sous-forme ne doit pas
                           apparaitre                    co
                 ::= <NOMBRE> '*' co nombre de répétition fixe co
<CONNEXION>      ::= <LIAISON>
                 ::= <NOEUD>
<FONCTION>       ::= <SURFACE>
                 ::= <LONGEUR>
                 ::= <LONGFRONT>
```

On peut associer un facteur de répétition avec chaque prédicats qui décrit une connexion et avec chaque fonction. Implicitement ce facteur est égal à 1. Un facteur de répétition égal à zéro indique que le nombre de de sous-formes est variable. Pour indiquer qu'une sous forme ne doit pas exister ce facteur est par convention égal à moins un (-1). Cette derniére possibilté est trés pratique pour décrire des configurations d'erreurs dans les dessins de masque.

III.4.2.1 Les liaisons.

Une liaison décrit une connnexion entre deux conducteurs, tel qu'un contact entre deux niveaux par exemple.

Le prédicat liaison est défini par une tache et deux conducteurs.
Syntaxe :

<LIAISON> ::= 'liaison' <COULEUR> ',' <NOM> ',' <NOM> ')'

Exemple: Toujours dans la technologie NMOS, Le contact diffusion métal est décrit par le composant CONTACT_ALU_DIFF ci-dessous.

```
composant CONTACT_ALU_DIFF
    début
        MM et MD et MC;
        CONT_AL_DI = liaison(MC, ALU, DIFF)
    fin
```

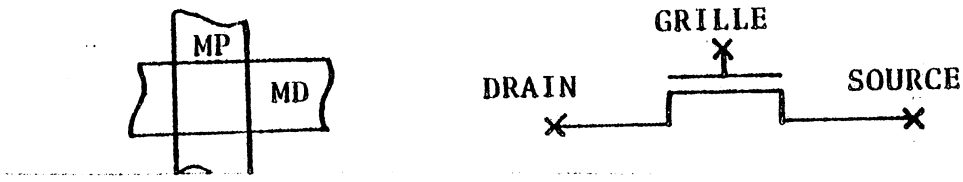
La tache de couleur (MM et MD et MC) définit la forme du composant. ce dernier contient une sous-forme CONT_AL_DI, elle décrit une liaison entre les deux conducteurs ALU et DIF. La forme de cette liaison est définie par la la tache de couleur MC.

III.4.2.2 Les noeuds.

Le prédicat noeud décrit une connexion entre un composant et un conducteur. Comme la liaison il est écrit comme une sous-forme dans un composant. Il est défini par sa forme et le nom du conducteur qu'il connecte.
Syntaxe:

<NOEUD> ::= 'noeud' '(' <COULEUR> ',' <NOM> ')'
::= 'noeud' '(' <FRONTIERE> ',' <NOM> ')'

Exemple: Le transistor NMOS est connecté à trois équipotentielles, la source, le drain, et la grille.



(a) Schéma du transistor (b) Dessin du transistor

FIG III.6 Schéma de connexion d'un transistor NMOS.

Le transistor peut être décrit avec ses connexions comme suit:

```
composant  TRANSISTOR_ENRICHI
        début
            MP croise MD sur ( non MI et non ME);
            GRILLE = noeud( CANAL, POLY ) ;
            2 * SOURCE_DRAIN = noeud
              ( frontière( CANAL,DIFFUSION), DIFF)
        fin
```

Le noeud GRILLE relie ce transistor à un conducteur POLY. Sa forme est définie par la tache de couleur CANAL. Ce prédicat est vrai dans chaque partie du transistor contenant un tache de couleur CANAL, et qui passe par un conducteur POLY.

Le source et le drain ont la même forme, il sont décrit par le prédicat SOURCE_DRAIN. Leurs forme est définie par une frontiére, il connecte le transistor à un conducteur DIFF. Ce prédicat doit être vérifié deux fois (facteur de répétition=2).

III.4.3 Description des caractéristiques des formes.

Le calcul des paramètres électriques des composants et des conducteurs peut être fait à partir des caractéristiques géometriques de ces formes. LANDFOR contient deux fonctions pour calculer la longueur et la surface des formes: longeur et surface. Elles sont aussi décrites comme des sous-formes dans les composants et le conducteurs. Chacune de ces fonctions décrit une forme. Le calcul des caractéristiques d'une forme est fait au moment de sa reconnaissance.

III.4.3.1 La surface

La fonction surface permet de calculer la surface d'une tache. Ceci est suffisant pour obtenir celle de toutes les formes. En fait pour obtenir la surface d'une autre forme il faut calculer celle de sa couleur de fond.

syntaxe

<SURFACE> ::= 'surface' '('<COULEUR>')'

Exemple:

```
composant CAPACITE
    debut
        ROUGE joint VERT
        S= surface( ROUGE et VERT)
    fin
```

La fonction S est considérée comme une sous forme du croisement. Elle calcule la surface de la tache de couleur (ROUGE et VERT).

III.4.3.2 La longueur.

La fonction longueur est aussi considérée comme une sous-forme dans un composant ou dans un conducteur. La sous-forme peut être soit une couleur soit une frontière.

Syntaxe:

```
<LONGUEUR>    ::= 'longueur' '('<COULEUR>')'
              ::= 'longueur' '('<FRONTIERE>')'
```

La longueur pour une couleur mesure une longueur électrique. La longueur d'une frontière mesure sa longueur géometrique.

Exemples

```
conducteur COND
        debut
            COUL_COND ;
            R = longueur(COUL_COND)
        fin
```

La fonction R reconnaît la tache de couleur COUL_COND et calcule sa longueur électrique c.à.d sa résistance.

```
composant TRANSISTOR
        debut
           MP croise MD;
           W = longueur( frontière( CANAL, MP ))
           L = longueur( frontière( CANAL, MD ))
        fin
```

Les fonctions W et L vont donner la longueur et la largeur du canal du transistor.

Les deux fonctions longueur et surface permettent de calculer

toutes les capacités dans un circuit. Les capacités de fond et les capacités interniveaux peuvent être décrites par la fonction surface. Les capacites de bord peuvent être décrites par la fonction longueur (de frontière).

L'exemple suivant donne une description du conducteur diffusion dans la technologie NMOS pour le calcul des capacités inter-niveaux et des capacités de bord.

```
couleur DIFFUSION = (DI non MP) ou (DI et ME)
conducteur DIFF
     début
          DIFFUSION ;
          S_DIFF = surface(DIFFUSION)
          0 * S_ALU_DIFF =surface(DIFFUSION et MM)
            co Le facteur de répétition 0 indique que la
                  forme peut être présente un nombre
                  quelconque de fois                    co
          0 * S_POLY_DIFF =surface(DIFFUSION et MP)
          CAPA_BORD = longueur(
               frontière(DIFFUSION, non DIFFUSION))
     fin
```

La fonction CAPA_BORD va retourner le périmètre de la tache de couleur DIFFUSION. Comme pour les surfaces inter-niveaux, on peut spécifier les différentes frontières qu'on peut rencontrer dans un connducteur DIFF.

III.5 LES ARBRES DE FORMES.

Un prédicat complexe décrit un arbre de formes c.à.d un motif composé d'une forme de fond qui contient des sous formes.

Les arbres de formes ont une structure de blocs. Chaque bloc décrit un sous-arbre. Un bloc est une suite de terme LANDFOR situé entre les deux termes <u>début</u> et <u>fin</u>

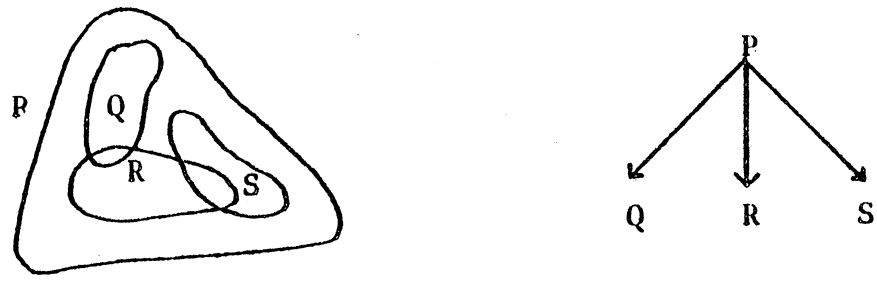
Exemple :

<u>composant</u> X

<u>debut</u> P ; Q ; R ; S

<u>fin</u>

Le composant X a une forme de fond P_0 et contient les sous-formes Q, R, et S. Il est représenté par l'arbre de la figure III.7.b.



(a) (b)

Fig III.7 Deux représentations d'une forme.

P, Q, R, et S sont des prédicats, ils décrivent des formes prédéfinies ou des formes composées.

Un prédicat complexe est constitué d'un arbre de prédicats prédéfinis. Il est vérifié si sa racine l'est, et tous les sous-arbre sont vérifiés.

Chaque élément de la technologie, composant ou conducteur est

décrit par un prédicat composé. Les connexions et les caractéristiques des formes constituent des feuilles d'arbre.

La description complète de la technologie constitue un bloc spécial qui contient la description des composants et des conducteurs.

III.6 CONCLUSION.

Bienque le langage LANDFOR soit adapté pour spécifier des technologies afin de faire une correspondance sans ambigùité entre la topologie des éléments et leurs fonctions électriques, Il faut noter qu'il est plus général.

Dans le domaine de la vérification des CI, le langage LANDFOR peut être utilisé pour décrire certaines erreurs difficiles à détecter par un VRD comme un caisson non polarisé en CMOS par exemple. Ces erreurs seront alors retrouvé par le systéme COMFOR en analysant le circuit de la mme façon que les éléments de la technologie.

Le langage LANDFOR peut être aussi appliqué dans plusieurs autres domaines classiques de reconnaissance des formes basés sur l'analyse d'images, comme la reconnaissance de cellules dans le domaine médical par exemple.

ANNEXE: EXEMPLES DE DESCRIPTIONS

A.1 Description de la technologie NMOS du CMP

Dans le cadre du CMP la technologie utilisée est la N-MOS enrichissement déplétion, grille poly, avec contact enterré.

Les niveaux de masques :

- désoxydation (MD)
- polycristallin (MP)
- aluminium (MM)
- contact (MC)
- contact enterré (ME)
- implantation ionique (MI).

```
technologie NMOS
 niveaux MD,MP,MC,MM,MI,ME
 co les niveaux désignent respectivement la désoxydation, le silicium
      polycristallain, le trou de contact, le métal, l'implantation, et
      le précontact. co
 couleurs co une couleur est définie par une expresion de niveaux co
          DI= (MD et non MP) ou (MD et ME);
          CONTACT_PA= MC et MM et MP;
          CONTACT_DA= MC et MM et MD et non MP;
          PREC = MP et MD et ME
          CANAL = MP et MD et non ME
conducteur DIFF   co DIFF désigne le nom du conducteur co
     debut
          DI ;co la forme du conducteur est constituée par une couleur
          SD = surface(MD)    co paramètres à calculer co
               co la surface servira a calculer la capacité co
     fin
```

```
conducteur POLY
     debut
          MP ;
          SP = surface(MP)
     fin
conducteur ALU
     debut
          AL ;
          SA = surface(MM)
     fin
composant CONTACT_DI    co contact DIFF-ALU co
     debut
          CONTACT_DA ; co couleur de fond du composant co;
          CDA = liaison(MC ,DIFF, ALU)
              co une liaison définit une connexion entre deux
              conducteurs, la couleur MC localise cette connection co
    fin
composant CONTACT_PO      co contact POLY-ALU co
     debut
          CONTACT_PA;
          CPA = liaison(MC ,POLY, ALU)
     fin
composant PRECONTACT
     debut
          PREC;
          PRC = liaison(MI, POLY, DIFF)
     fin
```

```
composant TRANS_E
     debut co la forme du transistor enrichi est décrite par
               le croisement de 2 couleurs, tel que, l'interieur
               du croisement contient la couleur (non MI)        co

          (MD croise (MP et non ME))sur non MI;

        co paramètres et connexions du transistor co
            GRILLE = noeud(CANAL,POLY);
              co un noeud définit une connexion entre un composant
                    et un conducteur, dans ce cas la couleur CANAL
                    définit la forme du noeud, et le noeud relie le
                    composant à un conducteur de type POLY co
           2* SOURCE = noeud(frontiere(CANAL,DI),DIFF);
co 2* signifie qu'il doit y avoir deux éléments de ce type co
            L= long(frontiere(CANAL,DI));
            W= long(frontiere(CANAL,MP et non MD));
            ST= surface(CANAL);
      fin
 composant Td co Transistor depléte, il a les même paramètres
                       que le transistor enrichi co
     debut
            (MD croise (MP et non ME)) sur  MI;
            GRILLE;
            2* SOURCE;
            L;
            W;
            ST;
      fin
```

2- Déscription d'un transistor bibolaire.

On à 6 niveaux de masques:

- couche enterrée (CE)
- mur d'isolement (MI)
- base (B)
- contact (C)
- émetteur (E)
- ALuminium (AL).

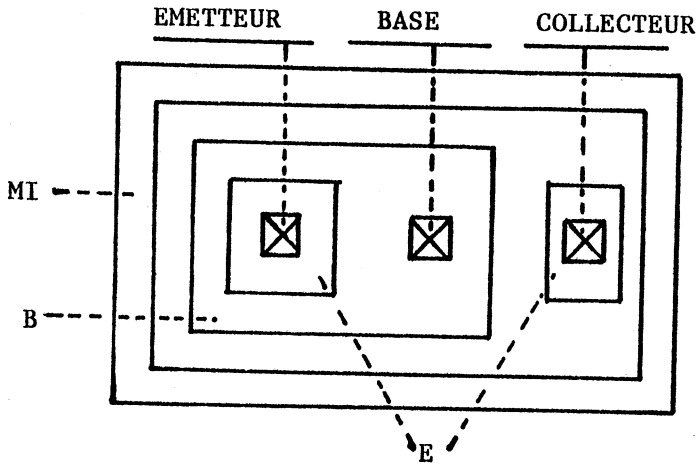


fig III.8 dessin shématique du transistor N-P-N

description du transistor

```
technologie BIPOLAIRE
    niveaux CE,MI,B,E,C,AL
    couleur CONTACT = AL et C,
        COLLECT=E et non B et CONTACT,
        EMET = E et CONTACT,
        BS = non E et CONTACT
    conducteur ALLU
       debut AL
       fin
    composant N-PN
       debut MI couvre
         debut non MI couvre
           debut  COLLECT ;
             COLLECTEUR = noeud (AL,ALLU)
           fin
           debut B couvre
             debut EMETT;
               EMETTEUR = noeud(AL,ALLU)
             fin
             debut BS ;
               BASE = noeud(AL,ALLU)
             fin
           fin
         fin
       fin
fin
```

A.3 Description de la technologie CMOS du CMP

Les niveaux de masques sont:

- Desoxydation N+ (MD)
- Désoxydation P+ (MB)
- Polycristallin N+ (MP)
- Métal (MM)
- Contact (MC)
- Précontact (ME)
- Caisson-P (MS)

```
technologie CMOS
 niveaux MD,MB,MP,MC,MM,ME,MS
 couleurs
    DN= MS et((MD et non MP) ou (MD et (MC ou ME)));
    DP= non MS et((MB et non MP) ou (MB et (MC ou ME)));
    CANALN = MP et MD et  MS
    CANAL = MP et MD et non MS
    CTN_MM_MP = MC et MM et MP;
    CTN_MM_DP = MC et MM et MB et non MS;
    CTN_MM_DN = MC et MM et MB et  MS;
   PREC = MP et MD et ME MS
conducteur DIFF_N
     debut
          DN ;
     fin
conducteur DIFF_P
     debut
          DP ;
     fin
```

```
condcteur POLY
     debut
          MP ;
     fin
conducteur ALU
     debut
          AL ;
     fin
composant CONTACT_MM_DN    co contact DIFF_N-ALU co
     debut
          CTN_MM_DN ;
          C_MM_DN = liaison(MC ,DIFF_N, ALU)
    fin
composant CONTACT_MM_DP    co contact DIFF_P-ALU co
     debut
          CTN_MM_DP ; co couleur de fond du composant co;
          C_MM_DP = liaison(MC ,DIFF_P, ALU)
    fin
composant CONTACT_MM_MP    co contact POLY-ALU co
     debut
          CTN_MM_MP ; co couleur de fond du composant co;
          C_MM_MP = liaison(MC ,POLY, ALU)
    fin
composant PRECONTACT
     debut
          PREC;
          PRC = liaison(ME, POLY, DIFF_N)
     fin
```

```
composant TRANS_P
     debut co la forme du transistor P est décrite par
             le croisement de 2 couleurs, tel que, l'interieur
             du croisement contient la couleur (non MS)        co

          (DP croise MP )sur non MS;

          GRILLE = noeud(CANAL_P,POLY);
        2* SOURCE = noeud(frontiere(CANAL_P,DP),DIFF_P);
     fin
composant TRANS_N
     debut co Pour le transistor N  l'interieur du
                croisement contient la couleur (MS) co

          (DP croise MP )sur  MS;

          GRILLE = noeud(CANAL_N,POLY);
        2* SOURCE = noeud(frontiere(CANAL_N,DN),DIFF_N);
     fin


composant ERREUR
     debut co un caisson non polarisé est consdéré
                comme erreur.                  co

          MS;
          -1 * POLAR = ( MD et MC et MM)
               co POLAR decrit une sous-forme du composant.
                  Le coéfficient -1 indique que cet élément
                  ne doit pas exister
                                                          co
     fin
```

CHAPITRE IV : SYNTAXE DE FORMES : LE LANGAGE SYNFOR

IV.1 INTRODUCTION
IV.2 RECONNAISSANCE SYNTAXIQUE DES FORMES.
IV.2.1 La méthode de balayage.
IV.2.2 Reconnaissance syntaxique de formes par la méthode de balayage.
IV.2.3 Cas d'une grille bicolore.
IV.2.3.1 Reconnaissance d'une tache rectangulaire.
IV.2.3.2 Reconnaissance d'une tache quelconque sur une grille bicolore.
IV.2.4 Cas des dessins des masques des C.I..
IV.2.4.1 La grille.
IV.2.4.2 Les formes.
IV.3 AUTOMATES POUR LA RECONNAISSANCE DES FORMES PREDEFINIES.
IV.3.1 Automates pour la reconnaissance des formes décrites par les prédicats prédéfinis.
IV.3.2 Automates pour le calcul des fonctions prédéfinies.
IV.4 LA FORME INTERMEDIAIRE SYNFOR.
IV.4.1 Processus de reconnaissance.
IV.4.2 Les primitives SYNFOR
IV.4.2.1 Primitives d'attente :
IV.4.2.2 Primitives de reconnaissance.
IV.4.2.3 Primitives de génération.
IV.4.2.4 Primitive duplique.
IV.4.2.5 Primitive recup.
IV.4.2.6 Primitive fin.
IV.5 CODAGE DE LA DESCRIPTION LANDFOR.
IV.5.1 codage des descriptions de formes
IV.5.1.1 Codage du prédicat technologie.
IV.5.1.2 Codage d'un prédicat complexe.
IV.5.1.3 Codage d'un prédicat prédéfini.
IV.5.2 Exemple de code complet pour une technologie.
IV.5.3 Le codage des couleurs :
IV.6 CONCLUSION.

IV.1 INTRODUCTION

Une description LANDFOR constitue une arbre dans lequel chaque sommet est un prédicat prédéfini. Cet arbre peut être vu comme une structure de contrôle qui définit la marche à suivre pour vérifier les prédicats. Un arbre de prédicats est dit vérifié (ou évalué "vrai") si et seulement si on a les 2 conditions suivantes:

i)le prédicat racine de l'arbre est vérifié,

ii)tous les sous-arbres sont aussi vérifiés.

La vérification d'un prédicat cconsiste à trouver la forme qu'il décrit.

Cette vision de la description LANDFOR est appelée interprétation procédurale, elle définit une syntaxe de formes. La reconnaissance des forme dans le circuit va consister alors à trouver toutes les parties du circuits qui s'unifient avec la syntaxe d'une forme. Cette syntaxe est décrite par un langage appelé SYNFOR (SYNtaxe des FORmes). La reconnaissance des formes est alors éffectuée par un interpréteur SYNFOR.

Ce chapitre présente des algorithmes de reconnaissance syntaxique de forme, basés sur la methode de balayage, à partir d'une description SYNFOR.

IV.2 RECONNAISSANCE SYNTAXIQUE DES FORMES.

Ce paragraphe s'interresse à la méthode de lecture de lecture de la spécification des circuits (présentée au deuxième chapitre) pour la reconnaissance des formes dans le système COMFOR.

IV.2.1 La méthode de balayage.

Le premier extracteur utilisant la méthode de balayage a été fait par BAKER [BAK-80]. Cet algorithme a eu un grand succés sachant qu'il est facile à mettre au point. Mais il a été vite abandonné car il est trés coûteux en temps de calcul et en place mémoire pour traiter les grands circuits.

La taille de la grille représentant le circuit (BITMAP en anglais) dépend uniquement de la surface de ce dernier. On a toujours représenté explicitement tous les points du circuit. La taille de la représentation informatique d'une grille pour un grand circuit est alors devenue excessive.
Un circuit de 50mm de surface dessiné avec une precision de 1/2 micron va être représenté par une grille de 200 millions de points. La représentation d'un tel circuit occuperait un grand disque de 200 millions d'octets si on représente chaque point par un octet. Cet exemple n'est pas exagéré quand on sait que la surface du MC68000 fait plus de 44mm carré, et que celle du HP9000 fait 40mm carré.

Le grand inconvénient de la représentation exhaustive de la grille est qu'elle ne tient pas compte de la complexité du circuit, c.à.d du remplissage de la grille. La méthode de compactage de la grille, décrite au deuxième chapitre, permet d'eliminer cet inconvénient tout en gardant l'avantage de cette methode qui est simple à programmer.

La représentation d'un C.I. par une grille élimine les informations géométriques contenues dans le dessin des masques. D'autre part elle permet d'effectuer des traitements locaux sur la grille.

L'algorithme de balayage (connu par son nom anglais RASTER SCAN ALGORITHM) consiste à balayer le circuit par une fenêtre en n'examinant à chaque instant que le contenu de la partie du circuit

qui apparait dans la fenêtre. La taille de la fenêtre dépend du traitement à effectuer sur le circuit. Si l'on veut, par exemple mesurer la dimension des éléments d'un circuit, la fenêtre doit pouvoir contenir le plus grand élément du circuit.

Pour notre cas le balayage du circuit va servir à détecter des taches et suivre leurs frontières. Donc une fenêtre de quatre points (2X2) suffit.

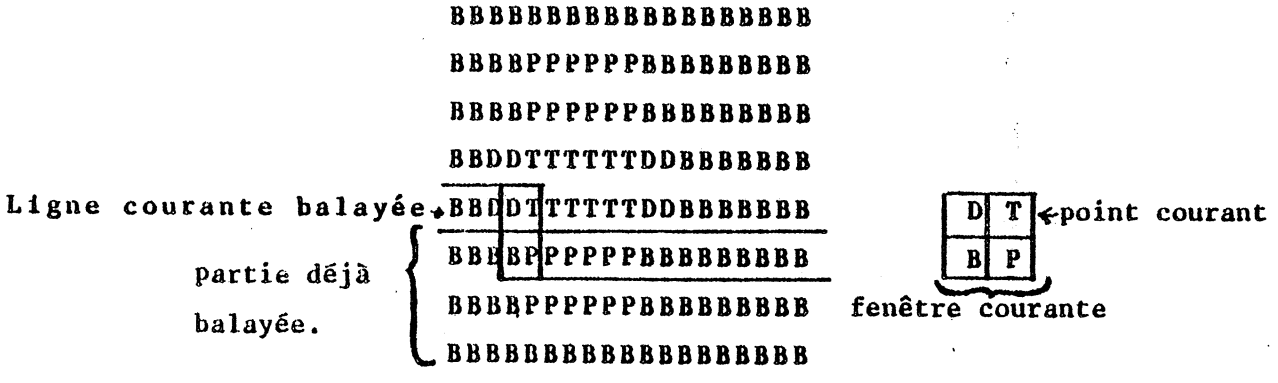


FIG IV.1 Exemple d'emplacement de la fenêtre au cours du balayage.

La fenêtre appartient à deux lignes consécutives du circuit. Elle parcourt le circuit de gauche à droite et de bas en haut.
Les autres algorithmes de balayage implémentés jusqu'ici ont toujours utilisés un balayage de haut en bas. Le sens du balayage (de bas en haut ou de haut en bas) ne modifie pas l'algorithme en lui même. Il dépend en géneral du systéme de dessin utilisé. Par exemple le systéme LUCIE [GUY-81] représente le circuit dans un plan euclidien, et prend comme point origine du circuit son point inférieur gauche. Le systéme LUCIE n'admet pas de coordonnées négatives.

Dans ce qui suit nous parlerons uniquement de balayage de gauche à droite et de bas en haut.

Au cours du balayage, on a besoin de stocker en memoire uniquement

l'équivalent d'une ligne de la grille. Dans [BAK-80] on trouve une étude plus détaillée de cette technique.

Le circuit représenté par une grille compactée peut être balayé de la même manière. La fenêtre ne s'arrêtera que sur les points différents de leurs voisins. En cas de besoin les fenêtres ignorées par le balayage peuvent être facilement reconstituées.

L'originalité de l'algorithme de balayage est dûe à deux faits:

i) Le traitement effectué à chaque étape du balayage est simple, la programmation d'un tel algorithme est donc facile à réaliser.

ii) Le temps de balayage d'un circuit est une fonction quasi linéaire de la surface vu que le temps de traitement d'une fenêtre est à peu prés constant.

Si l'on reprend l'exemple précédent, le circuit de 50 mm carré de surface, les 200 millions d'étape de balayage doivent occuper un ordinateur pendant plus q'une journée à raison de 2000 étapes par seconde (Ce qui dépasse la puissance de calcul d'un VAX/780 par exemple). Alors que la représentation du circuit par une grille compactée peut réduir le nombre d'étapes du balayage par un facteur de vingt (20). En fait ce facteur dépend de l'unité de dessin, pour des circuits déssinés avec une précision de 3 microns (dessin au lambda) ce facteur est de 5. Il dépend aussi de la densité du dessin.

IV.2.2 Reconnaissance syntaxique de formes par la méthode de balayage: Unification syntaxique.

La reconnaissance syntaxique des formes consiste à considérer la description des formes comme une grammaire, et la grille comme un langage. On dit qu'une partie du circuit contient une certaine forme si et seulement si la partie du circuit peut constituer une dérivation de la grammaire qui décrit la forme.

La grille est interprété comme une suite de fenêtres ordonnées Chacune d'elles est caractérisée par la couleur de ses quatre points et par son emplacement.

Dans ce qui suit, l'emplacement d'une fenêtre désignera celui de son point supérieur droit. Les coordonnées d'un point étant un couple (x,y) qui indique respectivement le numéro de la ligne et le numéro de la colonne qui contiennennt le point.

Les fenêtres sont ordonnées par l'ordre lexical de leur coordonnées. Cet ordre correspond à l'ordre de balayage.
Soient deux fenêtres F1 et F2 de coordonnées respectives (x1,y1) et (x2,y2). On dit que F1 est balayée avant F2 si et seulement si

$$y1 < y2 \quad \text{ou} \quad x1 < x2 \text{ et } y1 = y2$$

Il faut maintenant décrire une grammaire pour spécifier la syntaxe des formes [TOU-74]. Les éléments terminaux de cette grammaire sont des fenêtres.

IV.2.3 Cas d'une grille bicolore.

Le lecteur peut trouver des études plus complètes de ce cas dans [BAK-80].
Une grille bicolore est composée uniquement de points blancs et de points noirs (fig IV.2). On considère que les zones noires sont des taches (ou forme) sur un fond blanc.

BBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBNNNNBBBBBNBBBBB
BBBBBNNNNNNNNBBBNNNNNNBB
BBBBBNNBBBBNNBBBBNNNNNNB
BBBBBNNBBBBNNBBBBNNNNNNB
BBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBB

FIG VI.2 exemple de grille bicolore

Les points blancs et les points noirs sont représentés respectivement par "B" et "N".

Les différentes configurations de fenêtre qu'on peut rencontrer en balayant une grille bicolore sont données par la figure VI.3 [RAH-79].

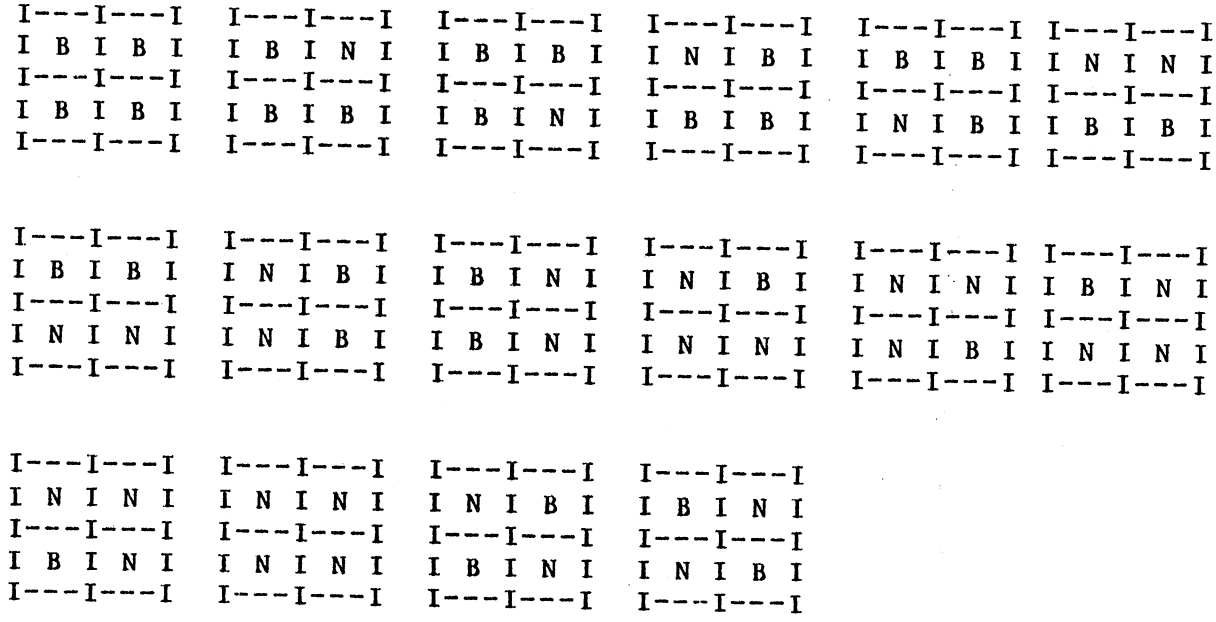


FIG IV.3 Les différentes configuration de fenêtres.

Les deux dernières fenêtres de la figure IV.3 ne seront pas traitées dans ce qui suit. Elle posent un problème de connectivité: Est ce que les deux points noirs de la fenêtre appartiennent à la

même tache?. D'autre part dans les dessins des CI ces deux configurations sont en général considérées comme des erreurs.

IV.2.3.1 Reconnaissance d'une tache rectangulaire.

La reconnaissance d'une forme (ou tache) rectangulaire sur une grille bicolore peut être faite par un automate (ou une grammaire) qui reconnait une séquence de fenêtres délivrées par le balayage du circuit.

L'alphabet de l'automate est constitué par les différentes configurations de fenêtres que peut contenir la tache.

Au cours du balayage d'une ligne de la grille, l'automate à besoin des trois informations suivantes:

i)La position de la fenêtre courante, appellée XFEN

ii)La position de la tache, sur la ligne courante balayée, appelé XDEB.

iii)La configuration de la fenêtre c.à.d la couleur de ses quatres points.

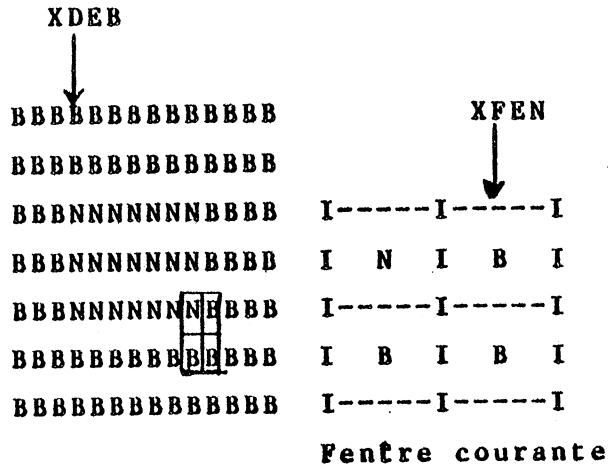


fig IV.4 position de la fenêtre et de la tache.

La position de la fenêtre (XFEN) est mise à jour au cours du balayage. Elle varie de 1 à N, N étant le nombre de colonnes de la

grille. La position de la tache sur la ligne balayée (XDEB) est fixe dans le cas d'un rectangle.

Dans le cas d'une forme quelconque, cette position est mise à jour par l'automate de reconnaissance. Elle est utilisée pour détecter l'apparition de la forme dans la fenêtre au cours du balayage. Elle est initialisée à la première apparition de la tache dans la fenêtre.

Les transitions de l'automate peuvent contenir des actions tel qu'initialiser une valeur par exemple. Une transition de l'état E1 à l'état E2 est représenté par la figure IV.5

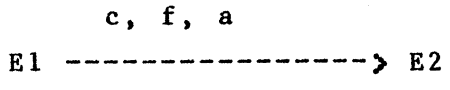


FIG IV.5 Transition de l'automate de reconnaissance.

"f" désigne une configuration de fenêtre.
"c" est une condition nécessaire pour que la transition ait lieu.
"a" est une action effectuée au moment de la transition.
La condidion et l'action peuvent être absentes dans certaines transitions.

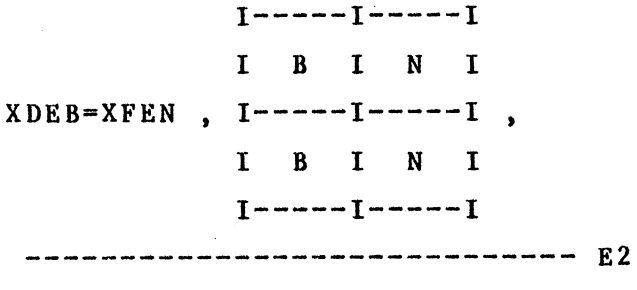


FIG IV.6 Exemple de transition.

dans la figure IV.6 la transition se fait si la condition (XDEB=XFEN) est vérifiée, et si la fenêtre courante à la configuration désignée.

Dans le cas de la grille bicolore les conditions ne sont utilisées que pour se positionner sur le début de la forme au cours du balayage.

La figure IV.7 décrit un automate pour reconnaitre une tache rectangulaire noire sur une grille bicolore.

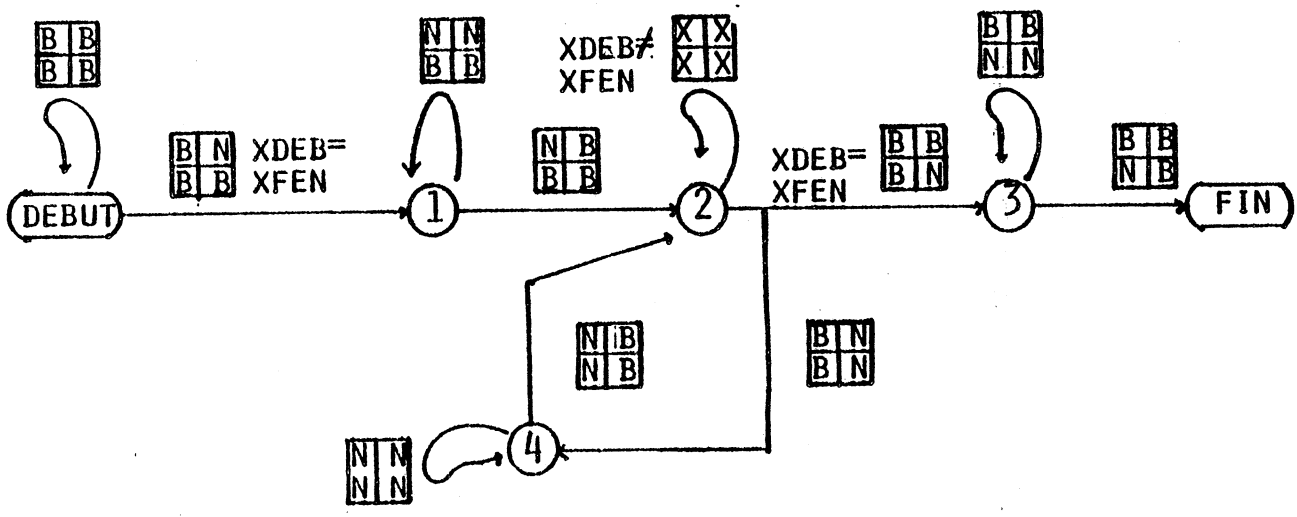


FIG IV.7 Automate pour la reconnaissance d'une tache rectangulaire.

L'automate est initialisé à l'état "debut". A chaque étape du balayage l'état est mis à jour selon la fenêtre courante et la condition. Si l'automate atteint l'état "fin" c'est que la partie balayé de la grille contient un rectangle. Sinon le processus de reconnaissance échoue.

Dans cet exemple l'action dans les transitions de l'automate ne sert qu'a initialiser la position de la forme (XDEB) dans le circuit. Dans d'autres cas ces actions peuvent servir à collecter des informations sur la forme reconnue, tels que la surface ou le périmètre des formes.

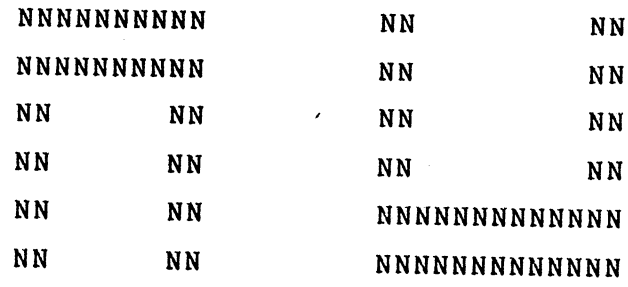
IV.2.3.2 Reconnaissance d'une tache quelconque sur une grille bicolore.

La tache rectangulaire est un cas particulier des formes qu'on peut rencontrer sur une grille bicolore. Ce paragraphe introduit un automate plus complet que le précédent pour la reconnaissance de formes quelconques, et décrit un mécanisme pour retrouver toutes les formes contenues dans une grille en un seul balayage.

REMARQUES.

i) A un moment donnée du balayage, une ligne peut couper plusieurs taches à la fois. Pour reconnaitre toutes les formes d'un circuit en un seul balayage il faut prévoir l'exécution de plusieurs automates en parallèle. Les taches seront alors distinguées par leurs emplacement.

ii) Une forme peut être constituée par plusieurs branches (fig IV.9).



a) forme détectée par une fusion de deux branches.

b) forme détectée après une division de la forme en branches.

FIG IV.9 formes et branches.

Pour reconnaitre de telle forme il faut completer l'automate précédent pour détecter les fusions et divisions des taches. Les différentes formes vont être reconnues par des processus différents appellés processus suiveurs. Un processus suiveur exécute un

automate de reconnaissance. Les fusions et divisions de formes correspondent à des divisions et des fusions de processus suiveurs.

iii)L'emplacement des formes sur la ligne balayée (XDEB) sera initialisé par un processus spécial qu'on appellera le détecteur. Ce processus détecte la première apparition de chaque tache au cours du balayage. Une nouvelle forme est signalée chaque fois que le detecteur rencontre la fenêtre de la figure IV.9.

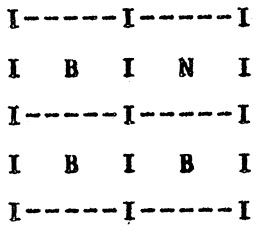


FIG IV.9 fenêtre de détection des nouvelles formes.

Chaque fois que le détecteur trouve une nouvelle tache il crée un suiveur pour la reconnaitre. Ce dernier va exécuter l'automate de reconnaissance (fig IV.10)

Cet automate fonctionne comme le premier, il commence à l'état "début".
La transition de l'état "1" à l'état "3" contient l'action fusion. La fenêtre de transition permet de détecter une fusion de deux branches. Le processus courant va alors chercher celui qui reconnait l'autre branche, le deuxième processus doit être aussi actif (il à la même fenêtre de transition et passe de l'état "2" à l'état "3"), ces deux processus sont alors fusionnées en un seul qui continuera l'exécution de l'automate à l'état "3".

REMARQUE

Dans le cas du systéme COMFOR les processus sont activés successivement selon leurs ordre d'apparition dans la fenêtre au cours du balayage. La transition de l'état "2" à l'état "3" avec la fentre de fusion n'est pas nécessaire puisque les processus ne sont pas activés simultanément.

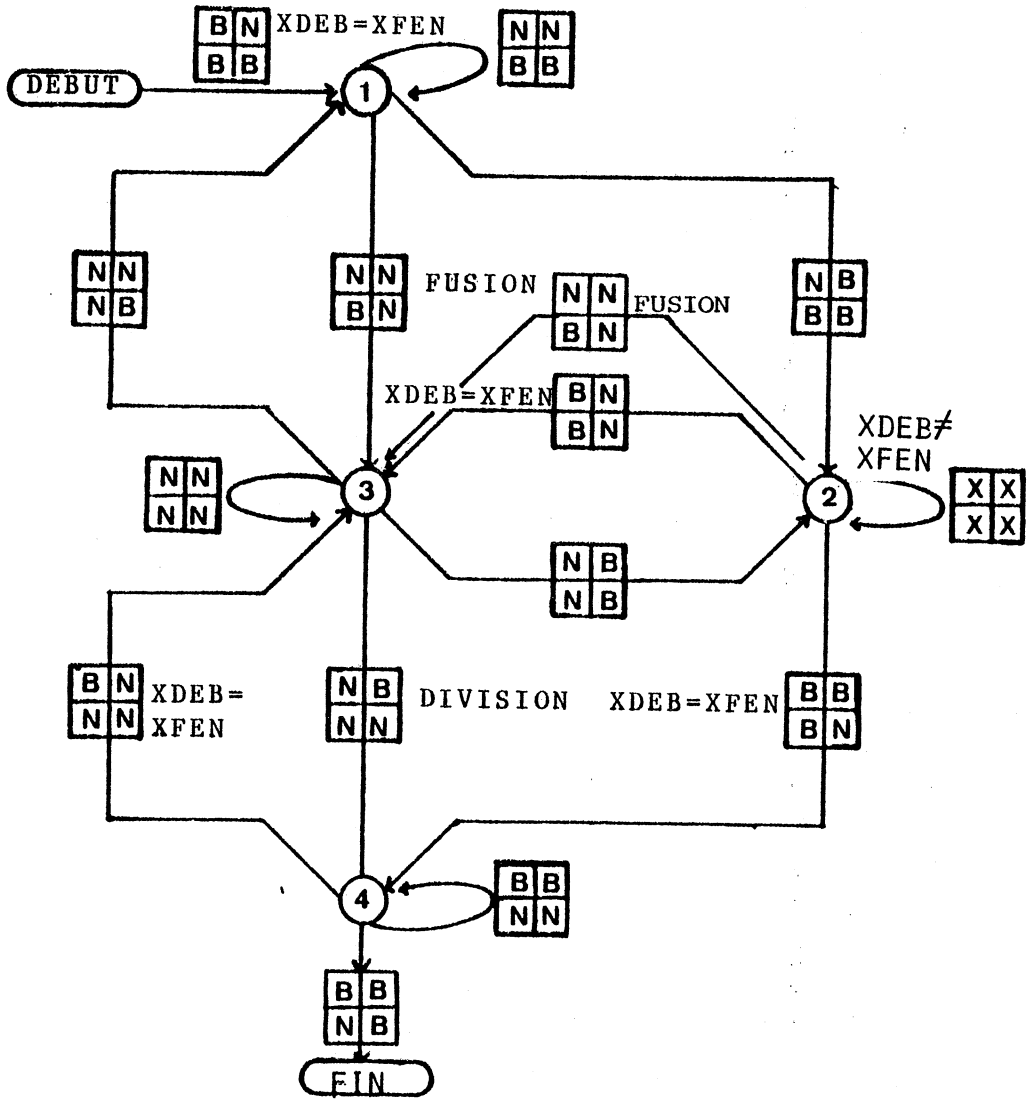


FIG IV.10 automate complet pour la reconnaissance des formes sur une grille bicolore.

La transition de l'état "3" à l'état "4" contient l'action division. La fenêtre de transition détecte que la tache courante se divise en deux branches. L'action de division consiste à créer un nouveau processus suiveur pour reconnaitre l'une des deux branches.

Le processus courant continuera la reconnaissance de l'autre branche. Le nouveau processus est appelé frère. On dit que les deux processus collaborent pour la reconnaissance de la forme.

La fin d'un processus (l'état "fin") correspond à la fin de reconnaissance d'une branche. Si le processus n'a plus de frères, cet état correspond à la fin de la reconnaissance de la tache.

IV.2.4 Cas des dessins des masques des C.I..

Ce paragraphe décrit l'extension du système décrit dans le cas d'une grille bicolore pour reconnaitre les composant d'un C.I. sur un dessin de masques.

IV.2.4.1 La grille.

La grille qui représente un C.I. contient plus que deux couleurs. La couleur d'un point de la grille est définie par l'ensemble des niveaux de masques présent en ce point (c.f. chapitre IV). Le nombre des configurations possibles de la fenêtre est très grand vu le nombre de couleurs possibles pour chacun des quatres points.

Si on considère un circuit dessiné avec six (6) niveaux de masques. Le nombre de couleurs possibles pour un point est égal à 2**6 (soit 64) Le nombre des configurations possibles pour une fenêtre de quatre points est égal à 64**4 (soit 16.777.216). Il est donc impensable de construire des automates qui énumèrent toutes ces fenêtres. En fait juste un petit nombre de configurations est nécessaire pour l'identification des formes dans un C.I.

IV.2.4.2 Les formes.

Les formes à reconnaitre dans un C.I. sont celles décrites par les

prédicats LANDFOR. Elles sont classsées en deux ctégories:

-les formes prédéfinies (ou simples)

-les arbres de formes (ou formes composées).

La reconnaissance de toutes les formes prédéfinies peut être faite par des automates du type de celui qui identifie les taches noires dans une grille bicolore.

La vérification d'un prédicat complexe (ou composé) est faite par plusieurs processus: Chacun vérifie un prédicat prédéfini. Ces processus sont des suiveurs ils exécutent des automates et sont organisés dans un arbre qui a la même structure que celui qui décrit le prédicat composé en LANDFOR. Un processus non terminal de l'arbre vérifie aussi un prédicat composé. Il identifie la forme de fond et en même temps il contrôle les processus qui font la reconnaisance des sous-formes.

Chaque forme prédéfinies est détectée par un processus particulier (il y a autant de processus détecteurs que de prédicats prédéfinis). Chaque suiveur d'une forme composée contrôle aussi les processus chargé de la détection de ses sous-formes.

La reconnaissance est lancée via un prédicat global appelé "technologie". Il contrôle les détecteurs et les suiveurs des composants et des conducteurs.

Dans le cas des C.I. il faut aussi calculer certaines caractéristiques des formes, tel que la surface ou la longueur d'un composant. Ces calculs sont inclus dans les actions des automates de reconnaissances.

IV.3 AUTOMATE POUR LA RECONNAISANCE DES FORMES PREDEFINIE.

Le langage LANDFOR contient actuellement huit (8) prédicats prédéfinis pour la déscription des formes simples et des connexions, et trois (3) fonctions pour le calcul des paramètres

électriques.
Les prédicats sont:

-La couleur,
-la frontière entre deux coleurs,
-le croisement entre deux couleurs,
-le recouvrement entre deux couleur,
-la jonction de deux couleurs,
-la liaison,
-le noeud en forme de frontière,
-le noeud en forme de couleur.

Les fonctions sont:

-la surface d'une couleur,
-la longueur d'une frontière,
-la longueur éléctrique,

A chacun de ces prédicats et fonctions correspond un automate de reconnaissance. Il faut noter que cette liste n'est pas limitative. Le langage peut être étendu par d'autres prédicats et fonctions simples qui peuvent être utilisés pour ce genre de reconnaissance.

Les onzes automates se ressemblent, ils fonctionnent tous comme celui de la figure V.10.

IV.3.1 Automates pour la reconnaissance des formes décrites par les prédicats prédéfinis.

La reconnaissance d'une tache ayant une couleur donnée peut se ramener au cas de la reconnaissance des formes sur une grille bicolore. On considère que tous les point qui contiennent la couleur recherchée sont noirs, les autres étant blancs.

Le prédicat recouvrement est défini par deux couleurs: L'intérieure et l'extérieure. Sa reconnaissance peut être ramenée a celle d'une tache. Il faut alors considérer que l'intérieur constitue des points noir et l'extérieur des points blancs. Le

processus qui le vérifie echoue si l'automate de reconnaissance rencontre un point qui n'a pas l'une de ces deux couleurs.

La frontière est une forme prédéfinie en LANDFOR, elle est parametrée par deux couleurs.
La figure IV.11 décrit un automate pour la reconnaissance d'une frontière. Le fonctionnement de cet automate est le même que celui de l'automate qui reconnait une tache. Il est initialisée à l'état "début", et une frontière est trouvée à chaque fois que l'automate atteint l'état "fin".

La figure IV.12 donne l'automate de reconnaissance d'un croisement.

Les automates de reconnaissance des connexions fonctionnent ausssi de la même manière.

IV.3.2 Automates pour le calcul des fonctions prédéfinies.

Le calcul d'une surface est fait par un automate qui reconnait une tache et qui, en même temps calcule sa surface. Cet automate (Fig IV.13) dispose d'un compteur incrémenté à chaque fois qu'il rencontre un nouveau point qui appartient à cette tache.

Le calcul de la longueur d'une frontière est aussi fait par un automate qui reconnait la frontière et en même temps il calcule sa longueur.

L'automate qui calcule la longueur électrique d'une tache fournit:

- une "squelette" de la tache formée par un ensemble de points qui décrit la structure du fil électrique correspondant. Les points peuvent être des emplacements de connexions ou des "coins" de la tache.
- la surface de la tache.

Le squelettte et la surface d'une tache permettent d'approximer sa résistance.
La figure IV.14 décrit l'automate qui calcule la longueur électrique.

ETAT	CONDITION	FENETRE DE TRANSITION	ACTION	NOUVEL ETAT
DEBUT			XDEB=XFEN	1
DEBUT			XDEB=XFEN	2
DEBUT			XDEB=XFEN	3
DEBUT			XDEB=XFEN	4
DEBUT			DIVISION	X
1				FIN
1			FUSION	FIN
1			XDEB=XFEN	3
1				1

Fig IV.11 Automate pour la reconnaissance d'une frontière les deux couleurs qui définissent la frontière sont désignées par "1" et "2".

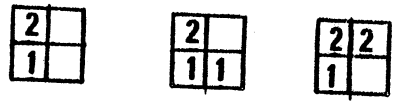
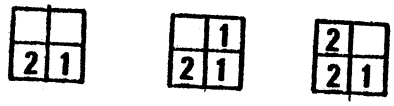
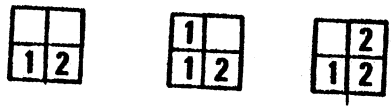
ETAT	CONDITION	FENETRE DE TRANSITION	ACTION	NOUVEL ETAT
2				FIN
2			FUSION	FIN
2			XDEB=XFEN	4
2				2
3	XDEB=XFEN			FIN
3	XDEB=XFEN			2
3	XDEB=XFEN			3
4	XDEB=XFEN			FIN
4	XDEB=XFEN			1
4	XDEB=XFEN			4

Fig IV.11 Automate pour la reconnaissance d'une frontière les deux couleurs qui définissent la frontière sont désignées par "1" et "2" (suite).

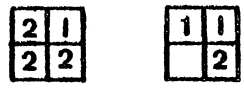
ETAT	CONDITION	FENETRE DE TRANSITION	ACTION	NOUVEL ETAT
DEBUT			XDEB=XFEN	2
DEBUT			XDEB XFEN	1
1				1
1				3
1			FUSION	4
2			FUSION	4
2				3
2				2
3	XDEB=XFEN			4
3	XDEB=XFEN			6
3	XDEB=XFEN			5

Fig IV.12 Automate pour la reconnaissance d'un croisement les deux couleurs qui se croisent sont désignées par "1" et "2", "I" désigne l'intérieur.

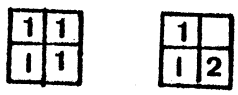
ETAT	CONDITION	FENETRE DE TRANSITION	ACTION	NOUVEL ETAT
4				4
4				1
4				3
4			DIVISION	X
4				2
6			XDEB=XFEN	4
6				6
6				FIN
5				5
5			XDEB=XFEN	4
5				FIN

Fig IV.12 Automate pour la reconnaissance d'un croisement les deux couleurs qui se croisent sont désignées par "1" et "2", "I" désigne l'intérieur (suite).

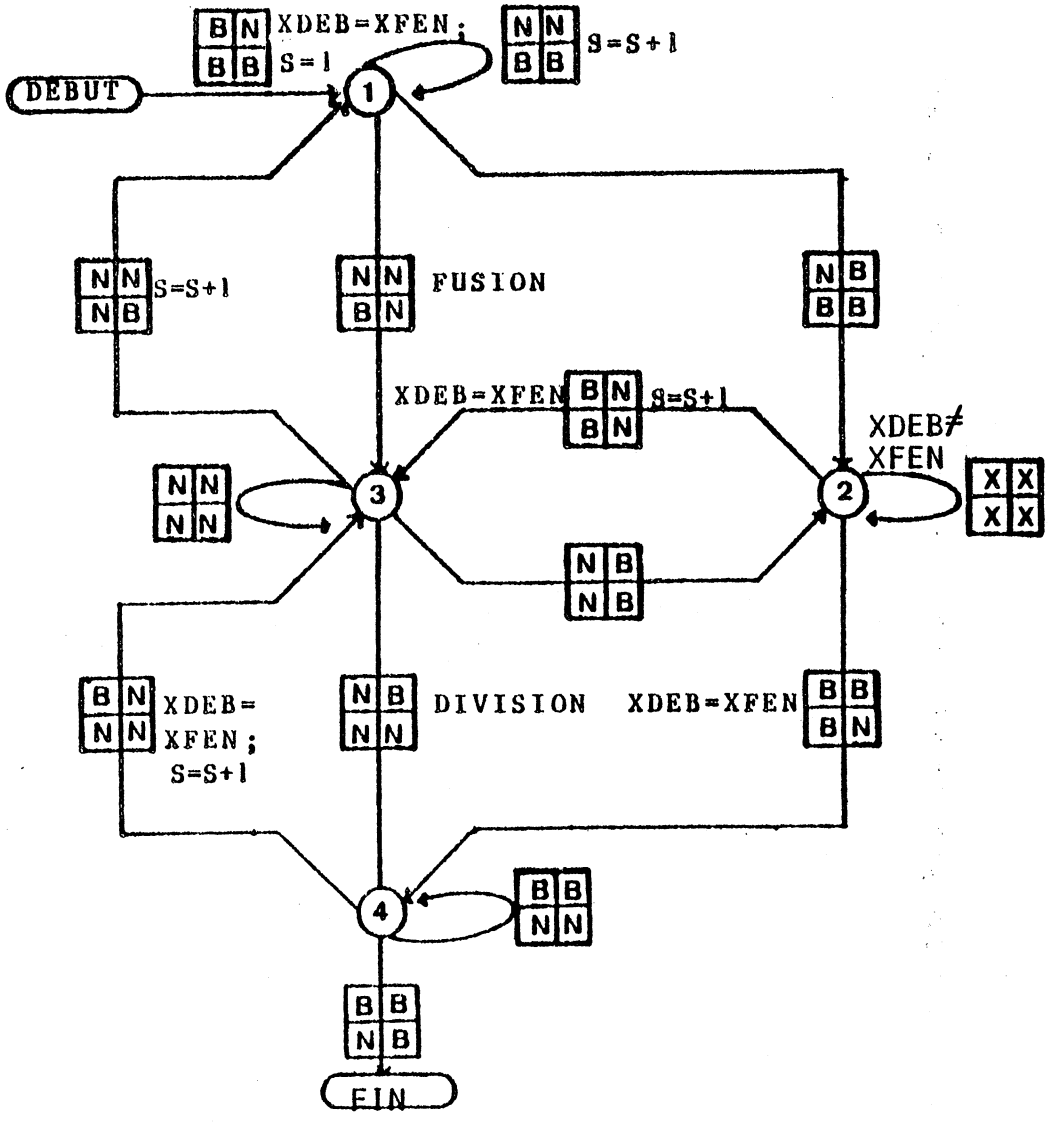


Fig IV.13 Automate pour le calcul de la surface d'une tache de couleur "N", "B" désignée les point ne contenant pas la couleur.

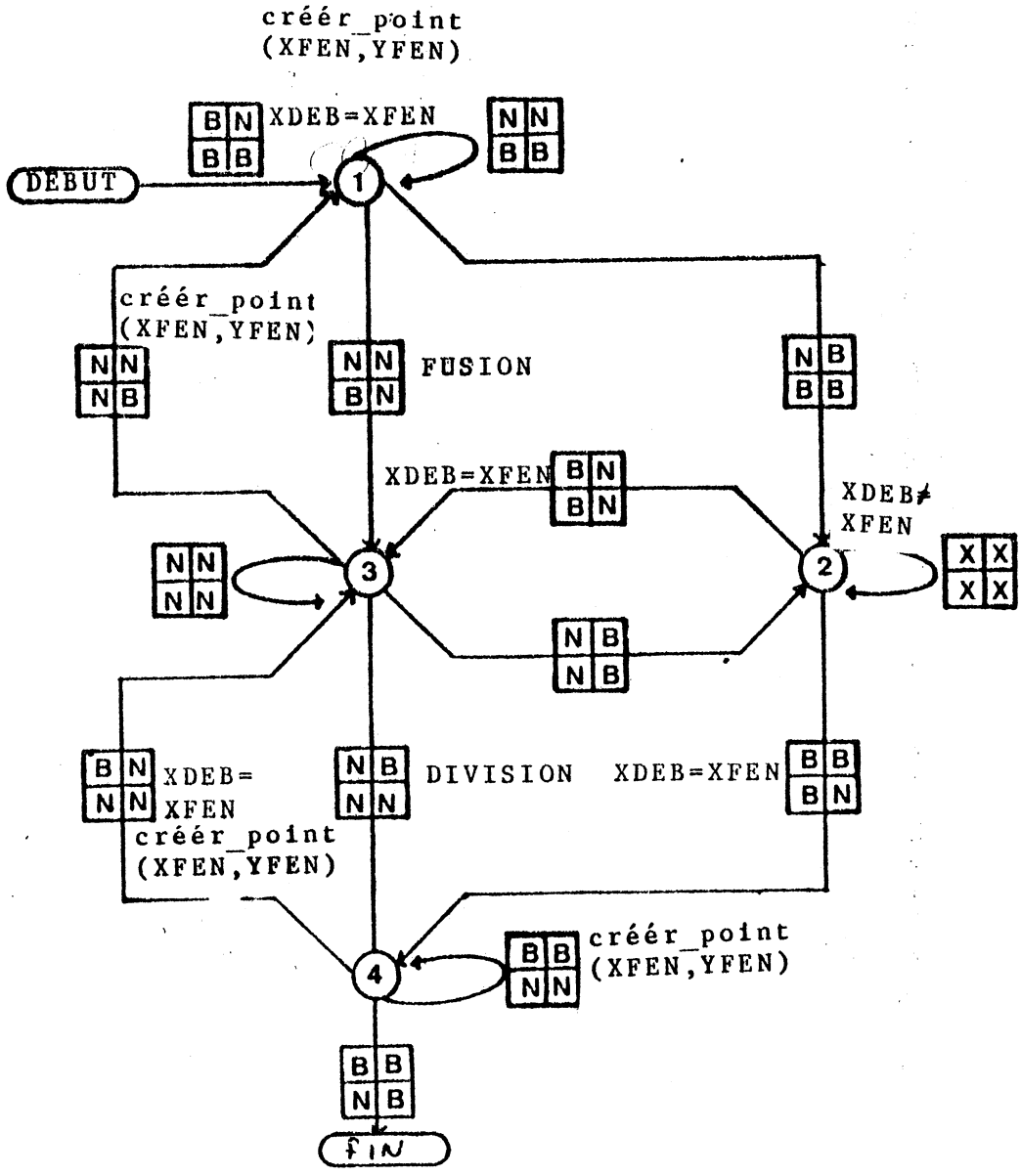


Fig IV.13 Automate pour le calcul de la longueur d'une tache de couleur "N", "B" désignée les point ne contenant pas la couleur.
L'action créér_point ajoute un point dans une liste qui servira à reconstituer une squelette de la tache.

IV.4 LA FORME INTERMEDIAIRE SYNFOR.

Les automates décrits plus haut permettent la reconnaissance des formes prédéfinies. Celle des formes composées est aussi ramenée à une suite d'opérations de reconnaissance de formes prédéfinies.

L'interprétation procédurale d'une description LANDFOR définit un processus de reconnaissance. Ce dernier décrit un mécanisme d'unification entre les formes rencontrées au cours du balayage et les syntaxes des formes. Il est codé en une forme intermédiaire appelé SYNFOR (SYNtaxe de FORme).

Une description SYNFOR est composée par un ensemble de primitives. Chacune décrit une opération élémentaire du processus de reconnaissance. La description SYNFOR est organisée en procédures. Pour un arbre de forme, chaque procédure décrit le processus de reconnaissance d'un sous-arbre.

IV.4.1 Processus de reconnaissance.

Un processus de reconnaissance [WAN-82] détermine les étapes à suivre pour retrouver une forme.

Exemple : On se propose de reconnaitre une forme dessinée par trois couleurs: BLEU, ROUGE, et VERT telle qu'on ait une tache de VERT et une tache de ROUGE recouverte par du BLEU.
La figure IV.15 donne trois configurations possibles de cette forme.

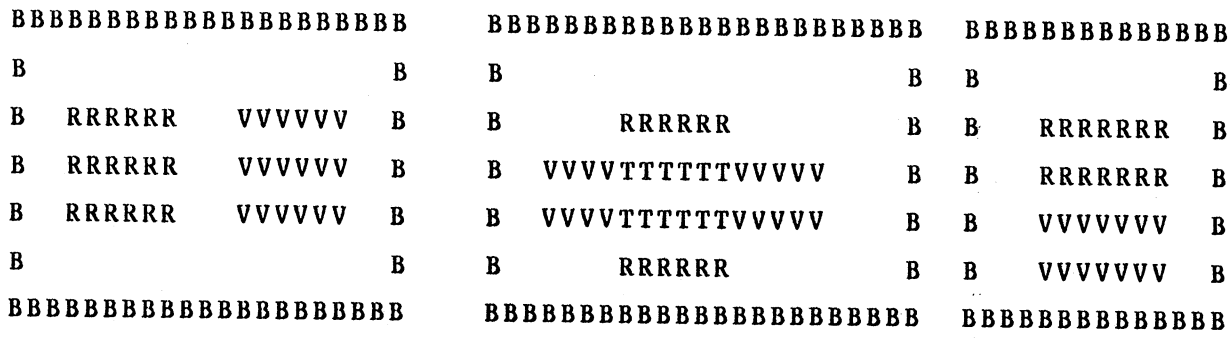


fig IV.15 Eemples d'occurence de la forme BLEU_ROUGE_VERT

Cette forme peut être décrite en LANDFOR par le prédicat suivant:

```
composant BLEU_ROUGE_VERT
     debut  BLEU couvre
        debut  ROUGE
        fin
        debut VERT
        fin
     fin
  fin
```

Intuitivement le processus de reconnaissance de cette forme peut être decomposé en six étapes:

i)Attendre l'apparition d'une tache bleue dans la fenêtre.

ii)Lancer deux processus pour la reconnaissance du vert et du rouge. Ces deux processus seront des processus fils.

iii) Faire la reconnaissance du bleu.

iv) Vérifier qu'on a trouvé une tache de rouge et une tache de vert . Si les deux processus lancés à l'étape (ii) se sont terminés sans erreur alors recupérer le résultat de leur

travail, sinon le processus en cours (qui reconnait la forme BLEU_ROUGE_VERT)est abandonné.

v)Générer des informations sur la forme trouvée.

vi) Fin de la reconnaissance.

La reconnaissance des arbres de formes peut être décomposée de cette manière (voir plus loin), les étapes (ii) et (iv) ne sont pas nécessaires pour la reconnaissance des formes simples.

Les processus de reconnaissance sont décrits en SYNFOR.

IV.4.2 Les primitives SYNFOR

Chaque primitive SYNFOR décrit une opération élémentaire d'un processus de reconnaissance. Il existe 5 types d'opération élémentaires:

- Attente d'une forme("attendre").
- Reconnaissance d'une forme ("reconnait").
- Générer les résultats de reconnaissance d'une forme ("génère")
- Démarrer des processus de reconnaissance de sous-formes ("duplique")
- Récupérer les résultats générés par les processus de reconnaisssance de sous-formes ("recup")
- Terminer un processus de reconnaissance ("fin")

IV.4.2.1 Primitives d'attente :

Les primitives d'attente détectent la première apparition d'une forme prédéfinie dans la fenêtre au cours du balayage.

Il existe autant de primitives d'attente que de formes simples.
Notation :

attendre_XX (C1,C2,...)

XX = nom de la forme,

C1,C2,.. = liste de couleurs paramètres de la primitive, le nombre de couleur dépend de la forme XX :

Pour l'attente d'une forme constituée par une tache de couleur C on écrit

attendre_couleur (C)

et pour l'attente de la forme (C1 croise C2 sur C3) on écrit

attendre_croisement (C1,C2,C1 et C2 et C3)

La couleur (C1 et C2 et C3) définit l'intérieur du croisement.

Les primitives d'attente sont exécutées par les processus détecteurs. Chaque fois qu'une nouvelle forme est repérée, le détecteur va créer un suiveur pour la rreconnaitre.

IV.4.2.2 Primitives de reconnaissance.

Chacune de ces primitives correspond à l'exécution d'un automate.

Notation :

reconnait_XX (C1,C2,...)

Les paramètres ont la même signification que pour les "attendre". Chaque processus suiveur exécute une primitive de reconnaissance.

IV.4.2.3 Primitives de génération.

Ces primitives mettent en forme les informations collectées au cours de l'exécution de l'automate. A chaque forme simple est associée une primitive de génération .

Notation :

génère_XX

Cette primitive est exécuté par les suiveurs aprés l'étape de reconnaissance, si cette dernère s'est bien terminée.
Les processus suiveurs de sous-formes passent les informations qu'ils ont calculées aux processus qui les contrôlent.

Les suiveurs des composants et des conducteurs mettent ces informations dans les fichiers de sortie.

Le schéma électrique extrait est constitué par l'ensemble des informations générées par les processus qui ont réussit à trouver un composant ou un conducteur.

IV.4.2.4 Primitive "duplique".

Un suiveur de forme composée commence par exécuter une primitive "duplique". Elle à pour effet de créer une liste de processus pour détecter les sous-formes.
Elle prend comme paramétres la liste des procédures qui décrivent les sous-formes.

Notation :

duplique (liste de sous-formes)

IV.4.2.5 Primitive recup.

Récupère le résultat de l'exécution des processus qui ont trouvé des sous-formes. Elle vérifie que chaque sous-forme a été trouvée au moins une fois.

Notation :

recup (n1,f1,n2,f2,.....)

f1,f2,... : la liste des sous-formes à récuperer

n1,n2,... indiquent le nombre minimum d'exemplaires de la sous-forme qu'on doit rencontrer.

Cette primitive exploite la liste des sous-formes reconnue (c.f. V.2.1.3). Cette dernière est remplie par les processus fils. Si la liste ne contient pas les sous-formes demandées par la description, alors le processus de reconnaissannce échoue. Si au contraire, la liste contient plus de sous-formes qu'il n'est demandé, la primitive récup dénombre toutes les combinaisons différentes des sous-formes conforme à la description LANDFOR de la forme. La primitive "génère" va alors considérer qu'il y a autant de formes reconnues que de combinaisons (c.f. V.3.3).

IV.4.2.6 Primitive fin.

L'exécution de cette primitive indique la terminaison normale d'un processus.

IV.5 CODAGE DE LA DESCRIPTION LANDFOR.

Le code SYNFOR est organisé en procédures, chaque prédicat est codé par une procédure. Les couleurs sont traitées au moment du codage de la description.

IV.5.1 codage des descriptions de formes

Au cours du balayage chaque processus suiveur correspond à l'exécution d'une procédure SYNFOR. Il existe trois formes différentes de procédures SYNFOR qui correspondent aux différents types de prédicats:

- La description globale d'une technologie constitue un prédicat particulier qui contient la déclaration des niveaux et couleurs, et les blocs de description des conducteurs et composants.
- Les prédicats complexes décrivent des arbres de formes.
- Les prédicats et les fonctions prèdéfinies décrivent des formes simples.

IV.5.1.1 Codage du prédicat technologie.

La procédure technologie référence tous les conducteurs et composants.

Code:

```
<NOM>       duplique (liste des composants et conducteurs)
            balayage_circuit
            fin
```

Cette procédure est le noyau de l'extracteur. En fait l'extraction d'un circuit consiste à exécuter cette procédure. La primitive balayage-circuit effectue le balayage du circuit et à chaque nouvelle fenêtre tous les fils sont activés, i.e. les détecteurs et les suiveurs des composants et des conducteurs de la description.

IV.5.1.2 Codage d'un prédicat complexe.

Un prédicat complexe est consstitué par un arbre de prédicats. La procédure SYNFOR correspondante référence les procédures qui décrivent les sous-formes.

Code :

```
<NOM>       duplique ( liste des sous-formes)
            reconnait_XX (liste de couleur )
            recup (n1,f1,n2,f2...)
            génère_XX
            fin
```

IV.5.1.3 Codage d'un prédicat prédéfini.

Un prédicat prédéfini décrit une forme simple.

Code:

```
<NOM>       reconnait_XX (liste de couleur )
            génère_XX
            fin
```

A chaque procédure SYNFOR (autre que la procédure technologie) est associé une primitive attendre qui a les même paramètres que la primitive "reconnait" de la procédure, et qui sera exécutée par le détecteur de la forme correspondante.

IV.5.2 Exemple de code complet pour une technologie.

Ce paragraphe décrit le programme SYNFOR géneré pour une technologie simple qui contient trois niveaux, un conducteur, un composant et le calcul des surfaces du conducteur et d'une sous-forme du composant.

Soit la description en LANDFOR de cette technologie exemple :

```
technologie EXEMPLE
     niveaux R,B,V
     couleur R_N_B = R et non B
     conducteur ROUGE
          debut  R ;
            S_ROUGE = surface( R )
          fin
     composant B_R_V
          debut  B couvre
             debut  V ;
               S_VERT = surface ( V )
             fin
             debut R
             fin
          fin
     fin.
```

Le codage de cette description en SYNFOR va produire un ensemble de procédures qui décrivent la syntaxe de ses formes. La description SYNFOR résultat constitue un compilateur de circuit (grille) pour reconnaître des formes. La description SYNFOR de la technologie EXEMPLE va être:

```
CO  procédure technologie EXEMPLE CO
EXEMPLE  duplique ( ROUGE, B_R_V )
         balayage_circuit
         fin
CO procédure conducteur ROUGE CO
ROUGE    duplique (S_ROUGE )
         reconnait_couleur( R )
         recup ( 1, S_ROUGE )
         génère_conducteur
         fin
 CO procédure pour calculer la surface de rouge CO
S_ROUGE  reconnait_surface( R)
         génère_surface
         fin
 CO procédure composant B_R_V CO
B_R_V    duplique ( f1, f2 )
  CO f1 et f2 sont les noms des 2 sous-formes non nommées
              explicitement  CO
         reconnait_couleur ( B )
         recup ( 1, f1, 1, f2 )
         génère_composant
         fin
f1       duplique (S_VERT)
         reconnait_couvre( B, V, B et V )
         recup ( 1, S_VERT)
         génère_couvre
         fin
S_VERT   reconnait_surface( V)
         génère_surface
```

```
        fin
f2      reconnait_couvre( B, R, B et R )
        génère_couvre
        fin
```

IV.5.3 Le codage des couleurs:

Les couleurs ne sont pas gardées en tant qu'expressions de niveaux pour la phase de balayage. Elles sont évaluées et codées dans une table. Cette dernière contient toutes les couleurs utilisées dans la description, i.e les couleur déclarées explicitement dans la description, et les couleurs calculées dans les arbres de formes nécessaires à la description SYNFOR.

Au moment du balayage les couleurs ainsi codées peuvent être utilisées pour faire des comparaisons avec les points du circuit.

La couleur d'un point indique la présence ou l'absence des différents niveaux. Une couleur de la description indique la présence ou l'absence d'un certain nombre de niveaux et ignore les autres, on à donc trois possibilités par niveau:

- le niveau n'est pas pris en compte,
- le niveau est obligatoirement absent,
- le niveau est obligatoirement présent.

les couleurs formées par une expression contenant l'opérateur "ou" indiquent une présence conditionnelle de niveaux.
Exemple : La couleur (MP ou MD) peut être localisée sur le circuit dans deux cas :

- les points qui contiennent MP (MD est ignoré),
- les points qui contiennent MD (MP est ignoré).

Cette couleur peut être représentée par ces deux alternatives de couleurs. Pour qu'elle soit présente en un point du circuit, il faut donc que l'une de ces deux alternatives soit vérifiée. D'autre

part, en un point du circuit il ne faut s'intéresser qu'aux niveaux cités par la description.

Les couleurs formées par une expression contenant l'opérateur "ou" peuvent être transformées en une liste d'alternatives de couleurs sans l'opérateur "ou".
Une combinaison de couleur est représentée par une chaîne de bits de longueur, 2 x nombre de niveaux, il en est de même pour la couleur des points du circuit.

Le tableau ci dessous donne la représentation des niveaux dans une alternative de couleur.

	REPRESENTATION DESCRIPTION (A)	REPRESENTATION DESSIN (B)	A et B
NIVEAU NON PRIS EN COMPTE	0 0	X X	0 0
NIVEAU PRESENT	0 1	1 0	0 0
NIVEAU ABSENT	1 0	0 1	0 0

Fig 4.16 codage d'un niveau

Au cours du balayage pour voir si une couleur est présente ou absente en un point du circuit il suffit d'une seule opération "et" logique sur les deux chaines de bits qui représentent la couleur du point et la couleur recherchée, si le résultat est différent de zéro on considère que la couleur est absente en ce point.

IV.6 CONCLUSION.

Avec le langage SYNFOR, nous avons aussi essayé de garder l'indépendance du système vis-à-vis de la technologie.

Le langage permet de décrire des mécanismes de reconnaissances pour des formes spécifiées en LANDFOR sans poser de contrainte sur la nature des formes décrites. Ces mécanismes peuvent fonctionner pour décrire d'autres formes que les composants des circuits integrés.

Le dernier chapitre présentera la réalisation logicielle du sytème COMFOR et en particulier une machine qui interprète le code SYNFOR.

CHAPITRE V : IMPLANTATION LOGICIELLE DU SYSTEME COMFOR

V.1 INTRODUCTION
- V.1.1 Le compilateur LANDFOR
- V.1.2 L'interpreteur SYNFOR
- V.1.3 Sortie.

V.2 ORGANISATION DES PROCESSUS.
- V.2.1 Evolution des processus
- V.2.2 Représentation des processus
  - V.2.2.1 Contexte du détecteur
  - V.2.2.2 contexte du suiveur.
  - V.2.2.3 Les sous-formes reconnues.
  - V.2.2.4 Les Références.
- V.2.3 Chainage des processus

V.3 GESTION DES PROCESSUS.
- V.3.1 création de processus.
  - V.3.1.1 création de détecteur
  - V.3.1.2 création de suiveur
- V.3.2 activation de processus.
  - V.3.2.1 activation de détecteur
  - V.3.2.2 activation de suiveur
- V.3.3 La récupération des sous-formes
- V.3.4 Désactivation de processus
- V.3.5 Division de processus
- V.3.6 Fusion de processus

V.4 SORTIE DE L'EXTRACTEUR
- V.4.1 formation de la sortie
- V.4.2 Format de la sortie

V.5 APPLICATION
- V.5.1 La simulation électrique.
- V.5.2 Dessin du S.E.

V.6 EVALUATION

V.1 INTRODUCTION

Ce Chapitre décrit la réalisation logicielle du système COMFOR, et plus particulièrement, l'interpréteur SYNFOR.

Le but du système est de retrouver des formes, décrites en LANDFOR, dans un circuit représenté par une grille.
Le schéma général de fonctionnement du système est donnée par la figure V.1.

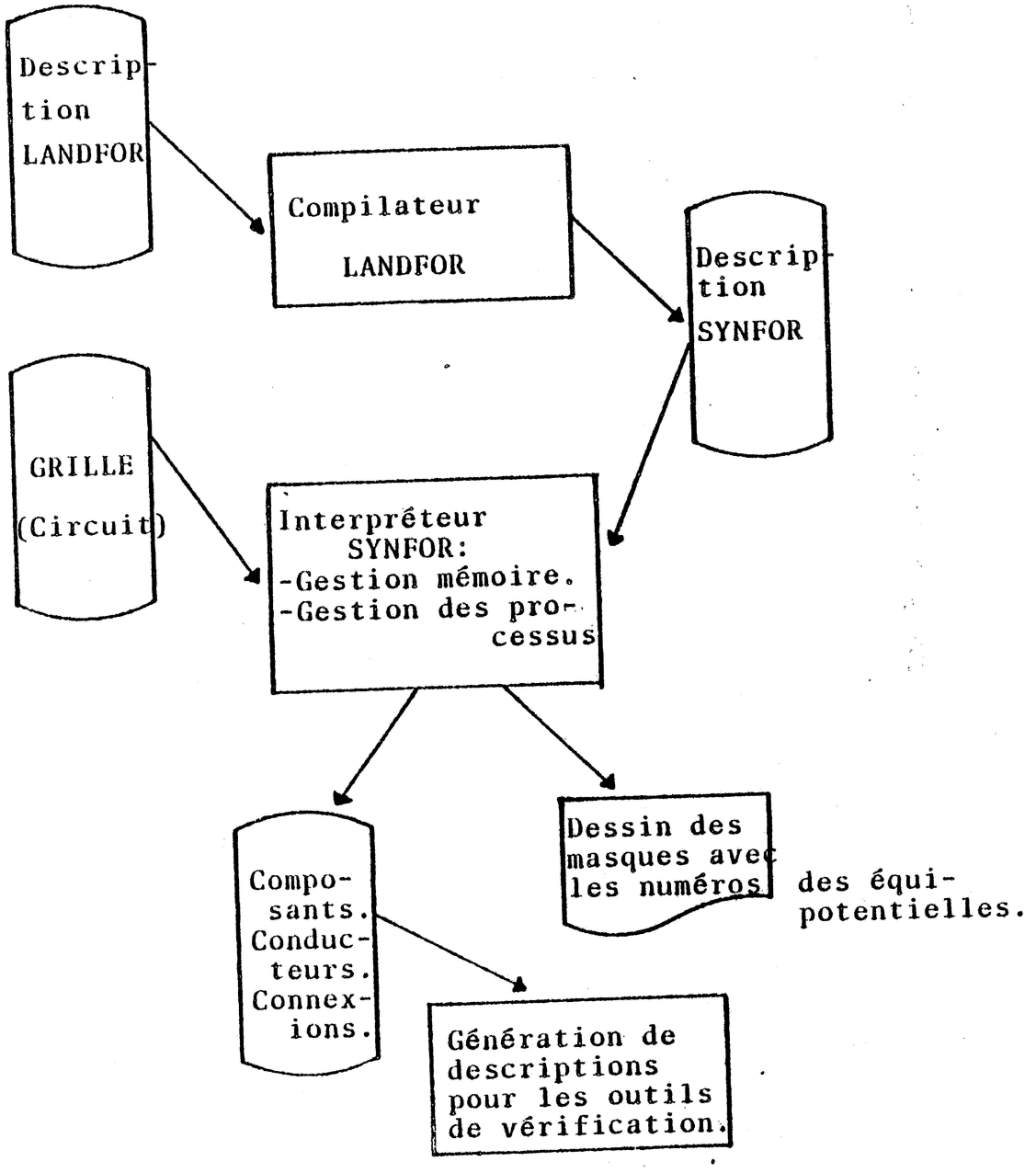


Fig V.1 Schéma général du système COMFOR.

La description LANDFOR est transformé en une forme intermédiaire (SYNFOR). Cette dernière décrit les mécanismes de reconnaissance de formes. Elle sera interprètée par une machine SYNFOR pour extraire les formes de circuits.

V.1.1 Le compilateur LANDFOR

Plusieurs aspects de ce compilateur ont déjà été évoqué dans le chapitre IV. Il génère un extracteur adapté à la technologie décrite en LANDFOR. Il travaille en trois passes. La première construit un arbre qui décrit la structure des formes, et en même temps elle analyse la syntaxe et détermine toutes les couleurs qui seront utilisées pour la reconnaissance des formes. La seconde et la troisième fonctionnent comme deux passes d'un assembleur pour générer une description SYNFOR.

V.1.2 L'interpreteur SYNFOR

Une description SYNFOR constitue un extracteur, elle décrit:

- La nature des formes à reconnaitre composants, conducteurs, et connexions.
- Les mécanismes de reconnaissance de ces éléments.
- Et, implicitement, le format de sortie.

L'interprèteur SYNFOR est la réalisation logicielle du modèle décrit dans le chapitre IV.

L'exécution d'un programme SYNFOR met en oeuvre deux types de processus: les détecteurs et les suiveurs. Les premiers repèrent les nouvelles formes, et les deuxièmes se chargent de continuer la reconnaissance de ces formes. Avant de se terminer, chaque suiveur produit une description de la forme qu'il a reconnu.

L'action de l'interpréteur consiste à créer un processus qui

exécute la procédure technologie, et puis, à l'activer jusqu'à la fin du balayage.

Algorithme:

```
action interpréteur
        fin_de_grille := faux
        créér_suiveur ( technologie )
        tant que non fin_de_grille
        activer_suiveur( technologie )
fin action interpréteur
```

Les actions activer_suiveur et créér_suiveur font partie d'un ensemble de primitives de gestion de processus qui seront exposées en détail dans ce qui suit.

Durant le balayage, le processus technologie va contrôler un arbre de processus où chacun fait la reconnaissance ou la détection d'une forme.
Quand un suiveur est activé, il traite la fenêtre courante, et active tous ses fils dans l'arbre. le processus technologie fait avancer la fenêtre de balayage au point suivant de la grille et active les détecteurs et suiveurs des composants et des conducteurs, qui vont à leur tour provoquer l'activation en chaine de tous les processus de l'arbre.

En fait le parcours de l'arbre est optimisé, seul les processus qui s'interressent à la fenêtre seront activés.

V.1.3 Sortie.

Le résultat de l'exécution d'un programme SYNFOR est constitué par la liste des composants et des conducteurs trouvés dans le circuit balayés. Chaque élément est décrit par une expression parenthésée. Cette dernière à la même structure que l'arbre de forme LANDFOR qui spécifie l'élément. Cette liste sera utilisée pour dessiner le schéma électrique, ou afin de produire des descriptions destinées aux outils de vérifications.

V.2 ORGANISATION DES PROCESSUS.

Vu que le système COMFOR est écrit dans le langage FORTRAN Il a fallu définir un module pour la gestion de la mémoire afin de pouvoir créer dynamiquement:

- des processus possédant chacun un contexte (descripteur) particulier qui peuvent être de tailles variables.
- des zones de travail de tailles variables pour ces processus.

Une politique d'allocation et de libération dynamique de blocs de tailles variables de la mémoire a été réalisée [AND-81]. En fait deux tailles de blocs sont disponibles et on peut constituer une grande mémoire de travail par une liste de blocs chainés.

V.2.1 Evolution des processus

Chaque forme est reconnue en trois étapes. Elle est découverte par un détecteur. Ce dernier crée un suiveur pour continuer sa reconnaissance. Si la deuxième étape se termine bien, le suiveur génère une description de la forme reconnue.

L'exécution d'un "duplique" provoque la création d'une liste de détecteurs. Chacun découvre toutes les premières apparitions d'une sous-forme donnée. Quand un détecteur rencontre un début de sous-forme il crée un suiveur pour finir sa reconnaissance. Les deux processus seront contrôlés par le même père, c.à.d celui qui reconnait la forme englobante.

Un suiveur exécute une procédure SYNFOR. Quand il est réveillé, il active tous ses fils qui s'interressent à la fenêtre courante. Donc, un processus ne peut être activé que lorsque son père, lui aussi, s'interresse à la fenêtre. Avec ce mécanisme les sous-formes sont toujours recherchées dans la forme englobante correspondante. A la fin de l'étape de reconnaissance (exécution de l'automate), la

primitive génère place la description de la forme trouvée dans la liste des sous-formes rattachées au père.

V.2.2 Représentation des processus

Les processus sont organisés dans une structure d'arbre. Le suiveur d'une forme complexe contrôle deux listes de processus: les suiveurs et les détecteurs des sous-formes. Le contexte de chaque processus est représenté par un descripteur.

V.2.2.1 Contexte du détecteur

Le descripteur d'un détecteur contient:

- L'adresse de la procédure SYNFOR qui décrit la forme attendue.
- L'adresse de l'instruction "attendre" qu'il exécute.
- Un chainage dans la liste des détecteurs du père.

Les fils détecteurs d'un suiveur donné sont organisés dans une liste contrôlé par ce dernier.

V.2.2.2 contexte du suiveur.

Un suiveur exécute une procédure SYNFOR, il est le sommet d'un sous-arbre de processus si cette procédure décrit une forme composée. Son descripteur contient:

- L'adrese de la procédure SYNFOR qui décrit la forme.
- L'adresse de l'instruction SYNFOR qu'il exécute.
- L'état de l'automate de reconnaissance, il peut être:
  - Début de reconnaissance.
  - Fin de reconnaissance (sortie de l'automate).
  - Erreur, cet état causera une fin prématurée du processus.
  - Fin normale de la procédure SYNFOR.

- Fin normale d'un processus qui à des frères. La fin du processus correspond à la fin de reconnaissance d'une branche de la forme (c.f. IV.2.3.2).

- Le chainage dans la liste des suiveurs.
- L'emplacement de la forme sur la ligne courante, il est constitué par les coordonnées des deux points extrémités de la forme sur la ligne.
- L'adresse de la zone de travail du processus, elle contient les informations collectées au cours de la reconnaissance.
- Le descripteur d'une forme complexe contient aussi l'adresse des listes de descendants:
  - La liste des fils détecteurs.
  - La liste des fils suiveur
  - La liste des sous-formes déjà reconnues par les fils.
- Le chainage de frères: si la forme contient plusieurs branches, tous les processus qui coopèrent pour sa reconnaissance se connaissent par un chainage spécial.
- Pour les conducteurs et les connexions, l'adresse de la liste des processus qui le référencent (c.f. V.2.1.4).

Tous les suiveur fils d'un processus qui reconnait une forme complexe sont organisés dans une liste. Ils sont ordonnés par leur emplacement sur la ligne balayée. Ceci permet de faciliter le repérage des processus qui s'interressent à la fenêtre du balayage.

V.2.2.3 Les sous-formes reconnues.

Les résultats générés par le suiveur d'une sous-forme sont classés dans une liste particulière rattachée à son père dans l'arbre des processus. Chaque élément de cette liste décrit une forme reconnue, et contient:

- L'adresse de la procédure SYNFOR qui décrit la sous-forme.
- Un Chainage dans la liste.
- L'adresse de la zone mémoire qui décrit la sous-forme reconnue.

V.2.2.4 Les Références.

Les connexions (noeuds et liaisons) décrivent des sous-formes de composants, et adressent des conducteurs.
Le problème:

- Au moment de la reconnaissance chaque conducteur doit connaitre les connexions qui le référencent, et ceci pour pouvoir les éliminer s'il se trouve en état d'erreur.
- Chaque suiveur de connexion, doit connaitre les conducteurs qu'il référence pour pouvoir les informer de l'état de son évolution.

D'autre part, dans la sortie de l'extracteur, chaque conducteur contient, en plus de sa propre description, la liste des connexions par lesquelles il passe.

Les suiveurs des connexions et des conducteurs tiennent chacun une liste des références, où chaque élément décrit un processus par:

- Son adresse s'il est encore vivant.
- L'adresse de la procédure SYNFOR qu'il exécute.
- Le numéro de la forme qu'il à reconnu s'il s'est terminé.
- L'adresse de l'élément suivant dans la liste des références.

V.2.3 Chainage des processus

Les descripteurs forment une arborescence de listes ayant pour racine le descripteur du processus technologie. Ce dernier existe tout le long du balayage

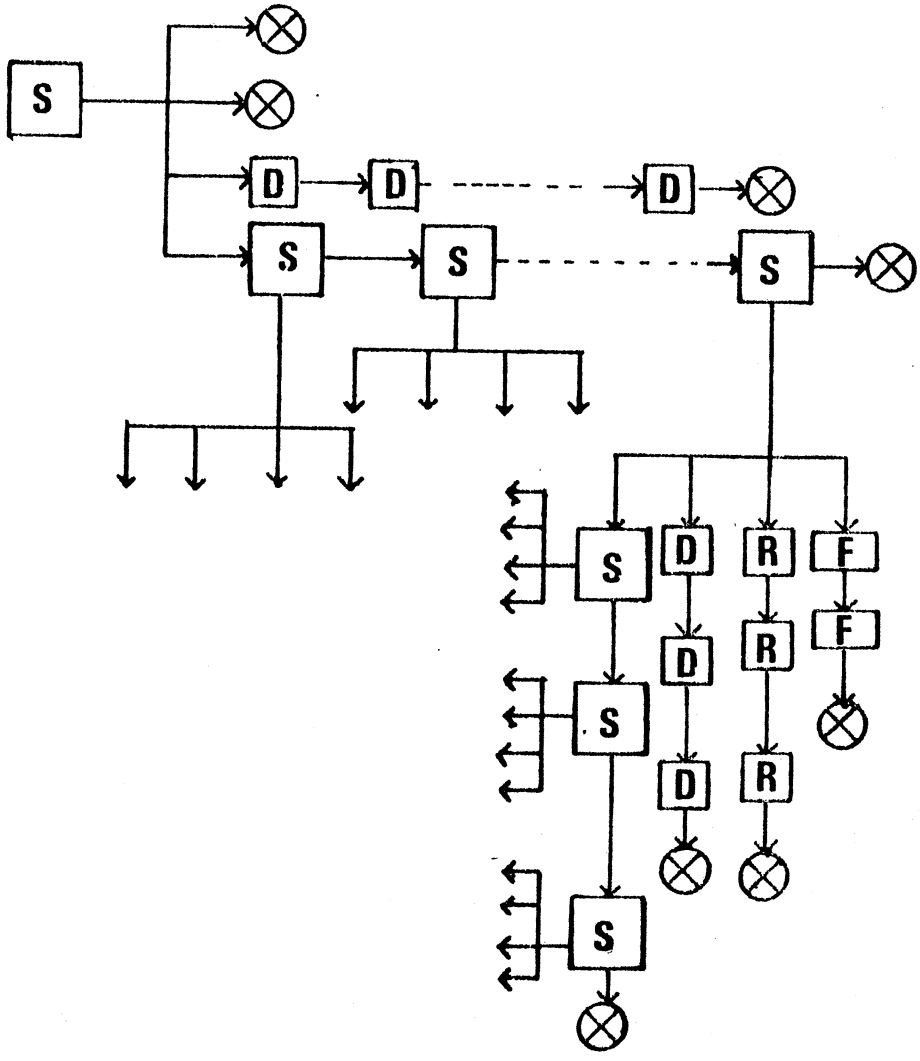


fig V.3 Organisation des listes de descripteurs.
S= processus suiveur.
D= processus détecteur.
F= sous-forme déjà reconnue.
R= référence.

V.3 GESTION DES PROCESSUS:

L'interprèteur est constitué par un ensemble de primitives pour:
- La reconnaissance des formes simples, c.à.d l'exécution des automates (c.f.IV.3)

- La gestion des processus.

Ces primitives exécutent des opérations, telles que création, activation, fusion etc..., sur les processus.

Ce paragraphe décrit les dernières, Les primitives de reconnaissance ont déjà été traitées en détail dans le chapitre IV.

V.3.1 création de processus.

V.3.1.1 création de détecteur

La primitive crée détecteur crée un exemplaire de processus détecteurs selon la procédure SYNFOR donnée en paramètre, et l'insère dans la liste des détecteurs, elle est appelée lors de l'execution d'un "duplique".

V.3.1.2 création de suiveur

La primitive crée suiveur crée un exemplaire de processus suiveur selon la procédure SYNFOR donnée en paramètre, et l'insère dans la liste des suiveurs correspondante. Elle est appelée lors de l'apparition d'une nouvelle forme, ou à la suite d'une division de forme. Dans le cas d'une nouvelle forme composée elle initialise les détecteurs des sous-formes (primitive "duplique")

V.3.2 activation de processus:

Quand un processus est activé il effectue tout le traitement concernant la fenêtre courante, et rend le contrôle au processus qui le contrôle, il sera eventuellement réactivé à la prochaine fenêtre.

V.3.2.1 activation de détecteur

Cette primitive provoque l'activation d'un processus détecteur qui va exécuter un "attendre".
Algorithme:

```
action active_détecteur(descripteur détecteur)
      -exécuter le "attendre"
      -si une nouvelle forme apparait alors
            créér_suiveur(procédure SYNFOR)
fin action active_détecteur
```

V.3.2.2 activation de suiveur

Quand un suiveur est activé il effectue son traitement de reconnaissance avec la fenêtre courante, et puis active successivement tous ses fils détecteurs, et ses fils suiveurs qui s'interessent à la fenêtre courante. Dans le cas ou il a terminé la reconnaissance de la forme demandée (son état d'évolution = fin de reconnaissance) le processus exécute le reste de son code, ce qui n'exige pas d'autres fenêtres. Ce dernier traitement consiste à:

- Récuperer les sous-formes trouvées par ses descendants (récup).
- Générer le code correspondant à la forme trouvée (génère).
- Se mettre à l'etat fin.

Algorithme:

```
action active_suiveur(descripteur suiveur)
     -exécuter une étape de l'automate de reconnaissance:
           reconnait_XX (descripteur suiveur)
     -activer tous les fils détecteurs:
       pour chaque élément de la liste des détecteus faire
            activer_détecteur ( élément )
       fin pour
```

```
-activer les fils suiveur qui s'interresse à la fenêtre:
  pour chaque élément de la liste des suiveurs faire
      si la sous-forme touche la fenêtre alors
                active_suiveur( élément )
      si ( le noveau état du fils = erreur ou fin ) alors
                désactive_suiveur ( élément )
   fin pour
-cas état de
  erreur: désactiver tous les fils
  fin normale:
     début
        si ( le suiveur à des frères)
          alors fusionner ( processus suiveur,un frère)
          sinon
               début
                  récup( descripteur suiveur)
                  génère( descripteur suiveur )
                  désactiver tous les fils
               fin
        finsi
          état = fin de procédure
     fin
  autre état:
 fin cas
fin action active_suiveur
```

V.3.3 La récupération des sous-formes

Elle consiste à vérifier que toutes les sous-formes demandées ont été reconnues. Si le nombre de sous-formes est supérieur au nombre demandé, on génère autant de formes que de combinaisons de sous-formes qui correspondent à la description.

L'exemple du transistor en "T" est un problème classique des extracteurs: En technologie N-MOS on peut avoir la forme présenté par la figure V.3.

fig V.3 dessin du transistor en "T"

Le schéma électrique correspondant à cette forme peut être dessiné par trois transistors connectés à la même grille (fig V.4)

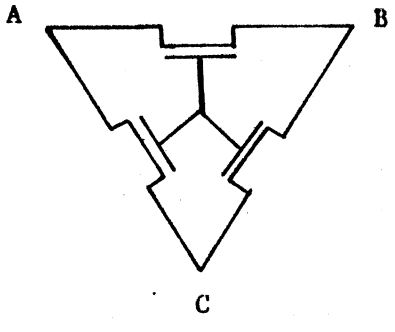


fig V.4 Schéma électrique correspondant au transistor en "T"

Si on décrit le transistor (en LANDFOR) comme étant un croisement qui contient trois noeuds:

- un noeud grille
- deux noeuds source-drain (voir chapitre III description N-MOS)

L'analyse de cette forme va donner un croisement qui contient un noeud grille et trois noeuds source-drain. En prenant ces derniers deux à deux on va obtenir l'équivalent de trois occurences du transistor, soit trois croisements chacun contenant une grille et deux source-drain.

La récupération des sous-formes est effectuée par la primitive récup. Cette dernière unifie l'arbre de description LANDFOR et l'arbre des sous-formes effectivement trouvées.

V.3.4 Désactivation de processus

La primitive désactive retire le sous-arbre de processus contrôlé par un suiveur: Elle désactive ses descendants, et le retire de la liste des suiveurs où il se trouve . Elle libère aussi tout les descripteurs occupés par ces processus. Si le processus à des frères il les informe de sa désactivation, en fait ceci peut arriver en cas de terminaison anormale d'un processus qui coopère avec d'autres pour reconnaitre la même forme, dans ce cas les fréres sont aussi considérés comme des processus invalides et sont forcés à l'état erreur.

Algorithme:

```
action désactive (processus suiveur)
      si (le suiveur a des frères) alors
                les forcer à l'état erreur.
      - libérer les fils détecteurs et les listes de sous-formes
      - pour un conducteur forcer toutes les suiveurs des
            connexions référencées à l'état erreur.
      - pour une conexion retirer les reférences qui la
            concerne chez les conducteurs.
      pour chaque fils suiveur faire
            désactive (fils)
      fin pour
      retirer le suiveur
fin action désactive.
```

V.3.5 Division de processus

Crée un processus frère au processus courant lors de la division d'une forme en deux branches. Les deux processus vont gérer en

commun les listes de fils et les listes des références et des sous-formes reconnues.

Algorithme:

action division(suiveur)
- créér un nouveau processus frère en dupliquant le descripteur du suiveur.
- établir les chainages du nouveau processus.
- créér une zone de travail pour le frère.

fin action division.

V.3.6 Fusion de processus

cette primitive ne concerne que les processus suiveurs. Au cours du balayage deux processus sont fusionnés dans deux cas:

- s'il sont en cours de reconnaissance de deux formes qui ont la même description (ils exécutent la même procédure SYNFOR), et si ces deux formes se rejoignent (c.f. IV.2.3.2),
- si les deux processus sont des frères (ils collaborent à la reconnaissance de la même forme) et l'un des deux arrive à la fin de la branche qu'il reconnait.

La jonction de deux formes est detectée par les automates de reconnaissance, le premier des deux suiveurs qui la détecte exécute la primitive fusion.

Algorithme.

action fusion(suiveur)
- trouver le frère
- fusionner les zones de travail.
- si (les deux processus n'ont pas les mêmes descendant)
  alors - fusionner le chainage des fr]res.
  - fusionner les listes de sous-formes.
  - fusionner les listes de descendants.

fin si

- si (la fusion résulte de la rencontre de deux branches)
  alors forcer le frère à l'état fin
  sinon forcer le processus courant à l'état fin
  fin si

fin action fusion

L'arbre des processus est contrôlé par l'interpréteur qui crée le processus technologie au debut du balayage, et l'active à chaque nouvelle fenêtre, ce dernier active ses descendants qui vont activer les leurs etc....

V.4 SORTIE DE L'EXTRACTEUR

A la fin du balayage on a la liste des formes reconnues. Elle est le résultat des primitives de génération. Chaque composant et chaque conducteur est decrit par une structure de liste déduite de sa description. Cette sortie constitue un schéma brut du circuit qui peut servir à produire une description pour un simulateur, ou à dessiner le schéma électrique. Le système produit aussi la liste des connexions trouvées dans le circuit.

V.4.1 formation de la sortie

La primitive génére produit la description des éléments trouvés dans le circuit, chaque élément est décrit par une forme de fond et une liste de sous-formes, cette dernière est vide pour les formes simples.
Le suiveur d'une forme composée produit lui même la description de la forme de fonds, tandis que celle des sous-formes est trouvée par ses fils. En exécutant la primitive génére, il va obtenir la description de la totalité de la forme composée dans une expression parenthésée:

(forme de fond (sous-formel) (sous-forme2) .....)

La sortie est produite sous deux formes : L'une textuelle et l'autre codée. Cette dernière est plus facile à traiter par programme car elle contient des informations plus compactées. Par contre la sortie textuelle de l'extracteur est plus lisible.

V.4.2 Format de la sortie

Il dépend de la description LANDFOR. Pour chaque élément reconnu dans le circuit on trouve son nom, son numéro, son emplacemnt, et la liste des sous-formes qu'il contient (fig V.5). La sortie correspondante à un conducteur contient aussi la liste des connexions qui le lient à d'autres éléments. Les composants qui ne décrivent que des connexions (les contacts par exemple) ne figurent pas dans cette liste.
Chaque forme prédéfinie génére un code particulier:

- les formes de description de topologie (croisement, couleur recouvrement, ...) génèrent leur nom et emplacement.
- Celles qui decrivent des connexion génèrent leur nom et numéro.
- les primitives de description des caractéristiques produisent leurs nom et la valeur trouvée (longueur ou surfaces).

Exemple: La première ligne de la figure V.5 à le format suivant:

cd (ALU 1 ,34 ,37 ,3 ,6 ,CPA 1 (SA 16))

Cette ligne décrit le conducteur ALU numéro 1, sa position dans le circuit est donnée par le quadruplet (34, 37, 3, 6) (xmin, xmax, ymin, ymax). Le conducteur passe par la connexion CPA 1 (contact POLY ALU numéro 1, voir la description NMOS en annexe du chapitre III). SA est est la surface du conducteur, elle a une valeur de 16.

Le système génére aussi la liste des connexions avec les numéros des équipotentielles. Pour chaque connexion on trouve son nom, son numéro, son empalcement, l'identification des deux éléments reliés par cette connexion, et le numero de l'équipotentielle qui contient

la connexion (fig V.6).

La première ligne de la figure V.6

CPA 1, 36, 5, ALU 1, POLY 3, 1

décrit la connexion CPA 1, cette dernière est placée au point de coordonnées (36,5), elle lie les deux conducteurs ALU 1 et POLY 3.

La liste des connexions est produite à la fin du balayage pour ne pas avoir à gérer des chaines de reprises trop longues necessaires à la formation des équipotentielles.

Une équipotentielle est constituée par un ensemble de conducteurs connectés entre eux. Nous utilisons une table de connexions pour former et numéroter les équipotentielles. Chaque entrée de cette table servira à produire la description d'une connexion (Fig V.6). Chaque suiveur de connexion qui se termine ajoute une entrée dans la table avec son identification (nom, numéro, et emplacement). Quand un composant se termine il met son nom et son nummméro dans les entrées correspondantes aux connexions qu'il contient. Et quand un conducteur se termine il utilise sa liste les références pour compléter les entrées de la tables contenant des connexions qui l'interressent. Il fait aussi l'union de toutes les équipotentielles rattachée à ces connexions.

```
cd (ALU 1 ,34 ,37 ,3 ,6 ,CPA 1 (SA 16))
cd (ALU 2 ,14 ,51 ,9 ,12 ,CPA 3 ,CPA 4 (SA 122))
cp (TRANS_E 1 ,62 ,63 ,7 ,18 ,4 (GRILLE 9)(SOURCE 10)(SOURCE 8)
      (W 24)(L 4)(ST 24))
cd (ALU 3 ,25 ,68 ,16 ,19 ,CDA 5 ,CDA 6 ,CDA 7 ,CDA 11 (SA 176))
cd (DIFF 1 ,64 ,68 ,7 ,19 ,SOURCE 10 ,CDA 11 (SD 37))
cd (POLY 1 ,62 ,73 ,4 ,20 ,GRILLE 9 (SP 54))
cd (DIFF 2 ,56 ,61 ,3 ,26 ,PRC 2 ,SOURCE 8 ,PRC 15 ,SOURCE 14
      (SD 82))
cd (ALU 4 ,23 ,67 ,23 ,26 ,CPA 12 ,CPA 13 (SA 143))
cd (DIFF 3 ,25 ,29 ,16 ,28 ,SOURCE 16 ,CDA 5 (SD 38))
cp (TRANS_E 2 ,30 ,31 ,16 ,28 ,4 (GRILLE 17)(SOURCE 18)(SOURCE 16)
      (W 26)(L 4)(ST 26))
cp (TRANS_E 3 ,36 ,37 ,16 ,28 ,4 (GRILLE 20)(SOURCE 21)(SOURCE 19)
      (W 26)(L 4)(ST 26))
cp (TRANS_E 4 ,44 ,45 ,16 ,28 ,4 (GRILLE 23)(SOURCE 24)(SOURCE 22)
      (W 26)(L 4)(ST 26))
cp (TRANS_E 5 ,50 ,51 ,16 ,28 ,4 (GRILLE 26)(SOURCE 27)(SOURCE 25)
      (W 26)(L 4)(ST 26))
cd (DIFF 4 ,52 ,56 ,16 ,28 ,SOURCE 27 ,CDA 7 (SD 38))
cp (Td 1 ,57 ,58 ,27 ,28 ,4 (GRILLE 28)(SOURCE 29)(SOURCE 14)
      (W 4)(L 4)(ST 4))
cd (POLY 2 ,55 ,60 ,25 ,28 ,PRC 15 ,GRILLE 28 (SP 24))
cd (POLY 3 ,34 ,37 ,3 ,30 ,CPA 1 ,GRILLE 20 (SP 64))
cd (POLY 4 ,44 ,57 ,5 ,30 ,PRC 2 ,GRILLE 23 (SP 76))
cd (POLY 5 ,48 ,51 ,9 ,30 ,CPA 4 ,GRILLE 26 (SP 52))
cd (POLY 6 ,23 ,31 ,14 ,31 ,CPA 12 ,GRILLE 17 (SP 68))
cd (DIFF 5 ,17 ,18 ,29 ,32 ,PRC 31 ,SOURCE 30 (SD 8))
cd (DIFF 6 ,63 ,64 ,29 ,32 ,PRC 33 ,SOURCE 32 (SD 8))
cp (Td 2 ,17 ,18 ,33 ,34 ,4 (GRILLE 34)(SOURCE 35)(SOURCE 30)
      (W 4)(L 4)(ST 4))
cp (Td 3 ,63 ,64 ,33 ,34 ,4 (GRILLE 36)(SOURCE 37)(SOURCE 32)
      (W 4)(L 4)(ST 4))
cp (Td 4 ,28 ,29 ,36 ,37 ,4 (GRILLE 43)(SOURCE 44)(SOURCE 42)
      (W 4)(L 4)(ST 4))
         -----
         -----
         -----
cd (DIFF 24 ,31 ,50 ,16 ,109 ,SOURCE 108 ,SOURCE 99 ,SOURCE 84 ,
      SOURCE 56 ,SOURCE 21 ,SOURCE 22 ,CDA 6 ,SOURCE 57 ,SOURCE 85 ,
      SOURCE 100 ,SOURCE 109 ,CDA 121 ,CDA 122 ,CDA 123 (SD 860))
cd (DIFF 25 ,57 ,74 ,66 ,109 ,SOURCE 117 ,CDA 124 ,CDA 125 ,CDA 126
      (SD 184))
cd (POLY 20 ,72 ,77 ,58 ,109 ,PRC 75 (SP 120))
cd (ALU 8 ,1 ,80 ,106 ,109 ,CDA 118 ,CDA 119 ,CDA 120 ,CDA 121 ,
      CDA 122 ,CDA 123 ,CDA 124 ,CDA 125 ,CDA 126 (SA 320))
```

Fig V.5 Liste des éléments générés par COMFOR
(sortie optionnelle)

IDENTIFICATION		POSITION X	Y	LES 2 ELEMENTS CONNECTES		EQUIPOTENTIELLE
CPA	1	36	5	ALU 1	POLY 3	1
PRC	2	57	6	DIFF 2	POLY 4	2
CPA	3	16	11	ALU 2	POLY 15	3
CPA	4	50	11	ALU 2	POLY 5	3
CDA	5	27	18	ALU 3	DIFF 3	4
CDA	6	41	18	ALU 3	DIFF 24	4
CDA	7	55	18	ALU 3	DIFF 4	4
SOURCE	8	62	7	DIFF 2	TRANS_E 1	2
GRILLE	9	63	18	POLY 1	TRANS_E 1	5
SOURCE	10	64	7	DIFF 1	TRANS_E 1	4
CDA	11	67	18	ALU 3	DIFF 1	4
CPA	12	25	25	ALU 4	POLY 6	6
CPA	13	66	25	ALU 4	POLY 16	6
SOURCE	14	57	27	DIFF 2	Td 1	2
PRC	15	58	26	DIFF 2	POLY 2	2
SOURCE	16	30	16	DIFF 3	TRANS_E 2	4
GRILLE	17	31	28	POLY 6	TRANS_E 2	6
SOURCE	18	32	16	DIFF 7	TRANS_E 2	7
SOURCE	19	36	16	DIFF 7	TRANS_E 3	7
GRILLE	20	37	28	POLY 3	TRANS_E 3	1
SOURCE	21	38	16	DIFF 24	TRANS_E 3	4
SOURCE	22	44	16	DIFF 24	TRANS_E 4	4
GRILLE	23	45	28	POLY 4	TRANS_E 4	2
SOURCE	24	46	16	DIFF 8	TRANS_E 4	8
SOURCE	25	50	16	DIFF 8	TRANS_E 5	8
GRILLE	26	51	28	POLY 5	TRANS_E 5	3
SOURCE	27	52	16	DIFF 4	TRANS_E 5	4
GRILLE	28	58	28	POLY 2	Td 1	2
SOURCE	29	57	29	DIFF 16	Td 1	9
SOURCE	30	17	33	DIFF 5	Td 2	3
PRC	31	18	32	DIFF 5	POLY 15	3
SOURCE	32	63	33	DIFF 6	Td 3	6
PRC	33	64	32	DIFF 6	POLY 16	6
GRILLE	34	18	34	POLY 15	Td 2	3

CDA	119	16	108	DIFF 23	ALU 8	4
CDA	120	21	108	DIFF 23	ALU 8	4
CDA	121	36	108	DIFF 24	ALU 8	4
CDA	122	41	108	DIFF 24	ALU 8	4
CDA	123	46	108	DIFF 24	ALU 8	4
CDA	124	61	108	DIFF 25	ALU 8	4
CDA	125	66	108	DIFF 25	ALU 8	4
CDA	126	71	108	DIFF 25	ALU 8	4

Fig V.6 Liste des connexions du circuit générées par COMFOR.

V.5 APPLICATION

La sortie du système COMFOR n'est pas toujours bien adapté aux outils susceptible de l'utiliser. Par contre elle peut contenir toute l'information nécessaire (selon la description LANDFOR) pour produire des spécifications de circuit pour les outils de vérification. Un programme de transformation de cette sortie est alors nécessaire. Ce dernier doit tenir compte de deux facteurs: la technologie et l'outil de vérification cible. Le premier détermine l'entrée à transformer, et le dernier définit le format de la description à générer.

Le système COMFOR utilise la description LUCIE pour produire un dessin du circuit dans lequel les équipotentielles sont étiquetées (fig V.7). Ce dessin peut servir pour vérifier certaines erreurs de connexion tel qu'un court-circuit entre deux signaux externes.

V.5.1 La simulation électrique.

La figure V.8 donne une description électrique (du circuit de la figure V.7) générée par COMFOR pour le simulateur MSINC. Il est aussi possible de produire des descriptions pour SPICE, un autre simulateur électrique.

La description de la figure V.8 est composée de trois parties: Les connexions externes, les transistors, et les capacités.

Les connexions externes sont spécifiées par l'utilisateur en tant que tel. Elles désignent les points d'alimentations et d'entrées/sorties du circuit. Cette partie doit être complété pour décrire le comportement de ces équipotentielles pour la simulation (fig V.9).

Chaque transistor est décrit par ses connexion, ses dimensions (W et L), et les sufaces du drain et de la source.

Les capacitées sont calculée pour chaque équipotentielles en additionnant les capacités des fils qui la composent. Elles sont calculées avec précision à partir des surfaces des conducteurs. Le calcul peut aussi inclure les capacités inter-niveaux si ces dernieres ont été décrites dans la description LANDFOR. A chaque équipotentielle correspond une capacité rattachée à la masse.

Le calcul des résistances avec précision sera étudié lors de l'implantation des primitives relatives au calcul de la longueur électrique. Actuellement un calcul approché des résistances peut être fait à partir de la surface et du périmètre. Il est évident que ce calcul ne peut être appliqué qu'aux conducteurs qui ont des formes quasi régulières. En fait la surface et le périmètre permettent d'evaluer avec précision la résistance d'un élément de forme rectangulaire.

L'introduction des résistances dans le S.E généré pose le problème de la répartition des capacités [TAR-83].

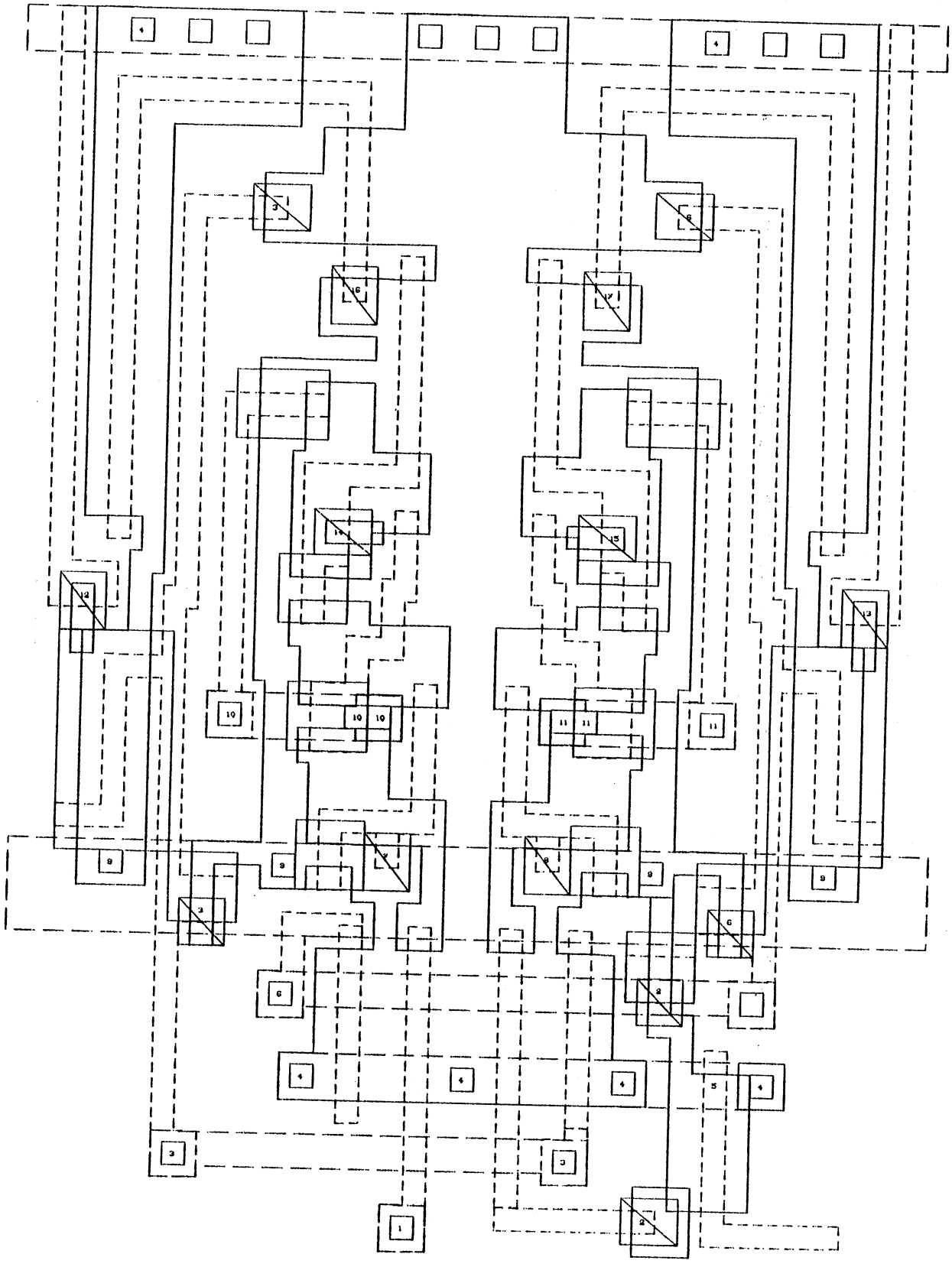


FIG V.7 Dessin du circuit avec les numeros des quipotentielles.

```
vdd 1 0
vin 2 0
h 3 0
hb 4 0
M1 2 6 0 0 ENR 36 6 0 0 738 12069
M2 8 7 0 0 ENR 39 6 0 0 828 0
M3 8 2 0 0 ENR 39 6 0 0 0 0
M4 9 2 0 0 ENR 39 6 0 0 828 0
M5 9 5 0 0 ENR 39 6 0 0 0 0
M6 2 2 1 0 DEP 6 6 0 0 0 3978
M7 5 5 1 0 DEP 6 6 0 0 522 0
M8 7 7 1 0 DEP 6 6 0 0 522 0
M9 8 8 1 0 DEP 6 6 0 0 0 0
M10 9 9 1 0 DEP 6 6 0 0 0 0
M11 10 8 0 0 ENR 24 6 0 0 225 0
M12 11 9 0 0 ENR 24 6 0 0 225 0
M13 10 10 1 0 DEP 9 18 0 0 0 0
M14 11 11 1 0 DEP 9 18 0 0 0 0
M15 3 5 1 0 DEP 57 6 0 0 1800 0
M16 4 7 1 0 DEP 57 6 0 0 1800 0
M17 12 12 1 0 DEP 6 6 0 0 261 0
M18 13 13 1 0 DEP 6 6 0 0 261 0
M19 12 10 0 0 ENR 27 6 0 0 0 0
M20 13 11 0 0 ENR 27 6 0 0 0 0
M21 14 10 1 0 DEP 12 6 0 0 711 0
M22 15 11 1 0 DEP 12 6 0 0 711 0
M23 14 12 0 0 ENR 45 6 0 0 0 0
M24 15 13 0 0 ENR 45 6 0 0 0 0
M25 5 14 0 0 ENR 30 6 0 0 0 0
M26 7 15 0 0 ENR 30 6 0 0 0 0
M27 3 14 0 0 ENR 156 6 0 0 0 0
M28 4 15 0 0 ENR 156 6 0 0 0 0
CP1 1 0 23040E-5
CP2 2 0 6480E-5
CP3 3 0 4320E-5
CP4 4 0 4320E-5
CP5 5 0 15552E-5
CP6 6 0 1944E-5
CP7 7 0 15876E-5
CP8 8 0 2088E-5
CP9 9 0 2088E-5
CP10 10 0 7848E-5
CP11 11 0 7848E-5
CP12 12 0 3384E-5
CP13 13 0 3384E-5
CP14 14 0 5544E-5
CP15 15 0 5544E-5
END
```

Fig V.8 du circuit générée par COMFOR.

```
plot plh

VDD 1 0 5
VIN 2 0 PULSE 0 5 150N 10N 10N 150N 320N
TIME 10N 700N
PLOT 3 4
OPT NP
MODEL ENR NMO VTO=.6 UB=800,1,.03,,3.5E4 TOX=630
+      DNB=.94E15 CJUN=1E-4 XW=.5 XJ=.5 XN=.5 GDS=2,.2,3
MODEL DEP NMO VTO=-1.4 UB=780,1,.03,,3.5E4 TOX=630
+      DNB=.94E15 CJUN=1E-4 XW=.5 XJ=.5 XN=.5 GDS=2,.2,3
M1  2  6  0  0 ENR 36 6 0 0 738 12069
M2  8  7  0  0 ENR 39 6 0 0 828 0
M3  8  2  0  0 ENR 39 6 0 0 0 0
M4  9  2  0  0 ENR 39 6 0 0 828 0
M5  9  5  0  0 ENR 39 6 0 0 0 0
M6  2  2  1  0 DEP 6 6 0 0 0 3978
M7  5  5  1  0 DEP 6 6 0 0 522 0
M8  7  7  1  0 DEP 6 6 0 0 522 0
M9  8  8  1  0 DEP 6 6 0 0 0 0
M10 9  9  1  0 DEP 6 6 0 0 0 0
M11 10 8  0  0 ENR 24 6 0 0 225 0
M12 11 9  0  0 ENR 24 6 0 0 225 0
M13 10 10 1  0 DEP 9 18 0 0 0 0
M14 11 11 1  0 DEP 9 18 0 0 0 0
M15 3  5  1  0 DEP 57 6 0 0 1800 0
M16 4  7  1  0 DEP 57 6 0 0 1800 0
M17 12 12 1  0 DEP 6 6 0 0 261 0
M18 13 13 1  0 DEP 6 6 0 0 261 0
M19 12 10 0  0 ENR 27 6 0 0 0 0
M20 13 11 0  0 ENR 27 6 0 0 0 0
M21 14 10 1  0 DEP 12 6 0 0 711 0
M22 15 11 1  0 DEP 12 6 0 0 711 0
M23 14 12 0  0 ENR 45 6 0 0 0 0
M24 15 13 0  0 ENR 45 6 0 0 0 0
M25 5  14 0  0 ENR 30 6 0 0 0 0
M26 7  15 0  0 ENR 30 6 0 0 0 0
M27 3  14 0  0 ENR 156 6 0 0 0 0
M28 4  15 0  0 ENR 156 6 0 0 0 0
CP1  1  0 23040E-5
CP2  2  0 6480E-5
CP3  3  0 4320E-5
CP4  4  0 4320E-5
CP5  5  0 15552E-5
CP6  6  0 1944E-5
CP7  7  0 15876E-5
CP8  8  0 2088E-5
CP9  9  0 2088E-5
CP10 10 0 7848E-5
CP11 11 0 7848E-5
CP12 12 0 3384E-5
CP13 13 0 3384E-5
CP14 14 0 5544E-5
CP15 15 0 5544E-5
END
```

Fig V.9 Description MSINC du circuit complétée par les cartes de contrôles.

V.5.2 Dessin du S.E.

Le concepteur n'a pas besoin de revoir le S.E de son circuit s'il dispose des outils de vérifications nécessaires, par contre il peut lui servir comme moyen de vérification.

Le S.E. est dessiné pour décrire l'implantation éffective des composants dans le circuit [FAJ-82]. Il est fait à partir :

- d'une bibliothèque de composants qui décrit leurs formes et leurs points de connexion,
- de la description fournie par l'extracteur.

Le dessin est alors fait par un parcours de la description :

- les composants sont dessinés aux emplacements et avec la taille indiqué par la description.
- les conducteurs sont dessinés par des lignes déduites de leur forme approximative donnée par la description (liste des connexions).
- les noeuds servent de repères dans le dessin.

Cette méthode est décrite avec plus de détail dans [FAJ-82], elle peut être utilisé pour des petites cellules. Mais pour les grands circuits elle se heurte à la quantité de fils à dessiner. En fait le dessin de tous les fils rend le dessin illisible.

Cet algorithme peut être amélioré par l'introduction de certaines procédure de simplification des fils, tel que l'élimination des équipotentielles les plus complexes et les remplacer par des noms. D'autre part la liste des connexion d'un conducteur est insuffisante pour pour déduire son chemin. Nous pensons ajouter une nouvelle primitive SYNFOR qui calcule le chemin d'un conducteur.

V.6 EVALUATION

Le tableau de la figure V.10 donne des statistiques sur deux

circuits dessinés en technologie NMOS (dans le cadre du CMP 83). Il montre que le traitement d'un circuit contenant 934 transistors, coûte moins que 30 minutes en temps CPU avec le système MULTICS du HB68. Le traitement consiste à générer une description MSINC (ou SPICE) complète (avec le calcul des capacités) à partir du dessin des masques du circuit décrit en LUCIE [GUY-81].

	NOM DU CIRCUIT		
	CIRC	REBON	POPY
SURFACE en LAMBDA	383 638	610 662	1356 741
ETAPES DE BALAYAGES	87 861	122 286	273 592
NOMBRE DE TRANSISTORS	639	934	2 346
TEMPS DE CALCUL (1)	5	6	15
TEMPS DE CALCUL (2)	22	29	64

Fig V.10 Performances du système COMFOR.
-Les temps sont donés en minutes CPU sur un HB68/MULTICS
-(1)Extraction sans calculs des paramètres électriques.
-(2)extraction fine du circuit.

Ce tableau montre aussi que le temps de traitement dépend aussi bien de la taille du circuit que de la description LANDFOR. En fait l'extraction d'un circuit sans calculer les dimensions des transistors et les capacités coûte à peu pres quatre fois moins cher que son extraction avec une description complète de la technologie.

C O N C L U S I O N

La réalisation du système COMFOR cherche à répondre à trois questions:

- Comment représenter les masques d'un circuit intégrés pour pouvoir les manipuler sans avoir à poser de contraintes sur la taille du circuit.

- Comment paramétrer le système par la spécification des technologies afin d'obtenir un extracteur général.

- Comment reconnaître les éléments d'un circuit en utilisant sa spécification donnée en paramètre.

Le chapitre II présente une méthode qui consiste à transformer la description du circuit en une grille compactée. Cette technique permet au système de se libérer des aspects géometriques du dessin (grâce à la décomposition), tout en garantissant la possibilité de spécifier de grands circuits (grâce au compactage). D'autre part, cette technique facilite le parcours de la totalité du circuit à l'aide d'une petite mémoire de travail.

La deuxième question à conduit à l'étude d'un langage général pour la description des formes. Pour celà nous nous sommes inspiré du langage PROLOG pour définir LANDFOR. Ce dernier langage (présenté au chapitre III) permet de spécifier les éléments de la technologie par des arbres de formes. Les paramètres électriques sont eux aussi décrits par des formes.

La plus grande partie de ce travail a consisté à répondre à la troisième question, soit, la réalisation d'un systéme pour reconnaitre les formes décrites en LANDFOR. Là-aussi nous nous sommes inspiré des méthodes de reconnaissance syntaxique des formes et de l'interpréteur PROLOG pour définir celui de LANDFOR.
Le modéle d'interprétation décrit au chapitre IV fonctionne en deux étapes:

- La première déduit la syntaxe des formes à partir de

descriptions spécifiées en LANDFOR. Cette syntaxe est décrite par le langage SYNFOR. Elle constitue un extracteur proprement dit.

- La deuxième interprète la description SYNFOR pour retrouver les parties du circuit qui obéissent à la syntaxe de l'un des éléments de la technologie.

Le chapitre V décrit la réalisation de cet interpréteur. Actuellement, COMFOR est utilisé pour extraire des circuits NMOS et CMOS. Il n'a pas encore été utilisé pour des technologies bipolaire. Nous avons bien entendu compilé certains motifs de ces technologies afin de tester les fonctions du système qui ne sont pas utilisées par les technologies MOS.

Bien que les solutions choisies au départ (la méthode de balayage et les techniques de reconnaissance syntaxique des formes) semblent être coûteuses en temps de calcul, nous pouvons constater que le système COMFOR est aussi rapide que la plupart des extracteurs actuels. De plus un grand intérêt du système COMFOR est son universalité puisqu'il est paramétré par la technologie. D'autre part ce système peut être utilisé pour d'autres applications que l'extraction de schémas électriques, et en particulier pour la reconnaissance de toutes formes pouvant être décrites en LANDFOR.

Enfin, il faut noter que les arbres de formes sont limités à la description des motifs constitués d'une forme de fond contenant des sous-formes. LANDFOR ne permet donc pas de spécifier des éléments qui ne sont pas connexes ou des éléments définis par des dimensions particulières. Ce genre de motifs est nécessaire à la description de certains composants tels que les transistors en technologie I^2L. Ce problème peut être résolu par l'introduction des notions de distance entre les formes et de dimension de formes.

Bibliographie

[ANC-83] ANCEAU F.
CAPRI: a design methodology and a silicon compiler for VLSI circuits specified by algorithms.
RR. No 346 IMAG 1983.

[ANC-82] ANCEAU F. SCHOELLKOPF J.P.
CAPRI: a silicon compiler for VLSI circuits specified by algorithms.
EEC CREST summer school BRISTOL juillet 1982.

[AND-81] ANDRE F.
Contribution à l'étude des systèmes distribués à partir d'une démarche expérimentale. Conception et réalisation d'un système de compilation réparti.
Thèse d'état université de RENNES 81.

[BAI-76] BAIRD H.S .
Design of family of algorithms for large scale IC mask artwork analysis
RCA Laboratories, Princeton, New Jersey, 1976.

[BAK-80] BAKER C.M.
Artwork analysis tools for VLSI circuits
MIT 80.

[BAK-82] BAKER C.M., TERMANN C.
Tools for verifying integrated circuit designs.
LAMBDA Fourth quarter 80

[BARK-82] BARKE E.
A Technology independant approach for device recognition from IC mask artwork data
Journal of digital systems vol. IV n 4, 1982

[BAS-83] BASTIAN J.D ELLEMENT M. FOWLER P.J. HUANG C.E. McNAMEE L.P.
Symbbolic parasitic extractor for circuit simulation (SPECS)
ACM IEEE 20TH DAC PROCEEDINGS MIAMI BEATCH, FLORIDA 83.

[BLO-81] BLOCH M.
Génération de taches bicolores, application aux caractères d'imprimerie, problème de nature ordinale.
Thèse de docteur ingénieur 1981 INPG.

[BON-81] BONNEFOY J.P. , JOUNET P. LORETTE G. , GAUDAIRE M.
Reconnaissance automatique, en temps reél, de signature manuscrite.
3ème congrés reconnaissance des formes et intelligence artificielle. NANCY 1981.

[BRI-78] BRIAN L. C. & C.A SAMMUT
Pattern recognition and learning with a structural description langage.
Fourth intenational joint conference on PATTERN RECOGNITION KYOTO-JAPON 1978.

[CLA-82]; K.L. CLARK, S.A. TARNLUND
Logic Programming.
ACADEMIC PRESS 1982.

[COL-79] C. COLMERAUER, H. KANOUI, M. VAN CANEGHEN.
Etude et réalisation d'un système PROLOG.
Rapport IRIA/SESORI No 77030

[DEM-80] DE MAN H.
Computer aided design techniques for VLSI.
proc. of the NATO advanced study institute on computer design aids for VLSI circuits. BELGIUM, Juillet 1980

[FAJ-82] FAJER P. DARDAINE J.M.
Dessin de schéma electrique de C.I.
projet 3ème année spéciale ENSIMAG 1982

[FU-78] FU K.S.
Recent advance in syntactic pattern rcognition.
Fourth intenational joint conference on PATTERN RECOGNITION KYOTO-JAPON 1978.

[GIM-81] GIMBERT D.
VERDICT: Vérificateur des règles de dessins.
rapport de 3ème année ENSIMAG 1981.

[GUP-83] GUPTA A.
ACE: A Circuit Extractor ACM IEEE 20TH DAC PROCEEDINGS MIAMI BEACH, FLORIDA 83.

[GUY-81] GUYOT A., JERRAYA AA et RAYMOND J.
LUCIE langage et programmes
1981. IMAG GRENOBLE

[GUY-83] GUYOT A. REIS R. SUPRIANA I.
FLOPE: un éditeur graphique de plan de masse de C.I.
R.R No 333, IMAG 1983.

[HEI-82] HEINTZ Christian.
Un extracteur de circuits integrés.
Thése de 3ème cycle ORSAY 1982.

[HEN-81] HENDERSON T.C. , DAVIS L.
Compilateurs de grammaires de formes.
3ème congrés reconnaissance des formes et intelligence artificielle. NANCY 1981.

[HIR-80] HIR ABAYASHI K. and KAWAMURA M. .
MACLOS mask checking logic simulator.
IEEE Journal solid state circuits, june 1980 (pp. 368/370).

[HOF-83] HOFMANN M. ULRICH L.
HEX: An instruction driven approach to feature extraction
ACM IEEE 20TH DAC PROCEEDINGS MIAMI BEACH, FLORIDA 83.

[JAN-79] JANKOWSKI M.T.
Compression of memory mapped pictures in microcomputer graphic systems.
EUROMICRO 1979 (pp.43/47).

[JER-83] JERRAYA A.A.
COMFOR: Une nouvelle approche pour la vérification des C.I. VLSI
RR No 373, IMAG 1983

[JER-81] JERRAYA A.A.
Extraction automatique de schéma électrique
rapport de DEA ENSIMAG 1981.

[LOS-79] LOSLEBEN P. and THOMPSON K.
Topological analysis for VLSI circuits.
16th DAC (pp.461/473).

[MAL-83] MALEK A.
Analyse de description de schéma électrique (Programme PROLOG)
rapport de fin d'étude d'ingénieur faculté des sciences de TUNIS --IMAG

[MAR-81] MARTHON P. , BRUEL A.
Une méthode de description d'une image numérique.
3éme congrés sur la reconnaissance des formes et intelligence artificielle. NANCY 1981.

[McC-79] McCAW CR.
USC: a checking tool for VLSI. 16th DAC (pp.81/87).

[McC-79] McCAW E.J.
Design integrity and imunity checking.
California Institute of Technology 1979.

[MEA-80] MEAD C. and CONWAY L.
Introduction to VLSI systems, 1980, Addison-Wesley.

[19] MUNCH P.H, MUNCH K.H.
A general solution for design verification.
ICCC 82.

[MOR-82] MORI S. DOH M.
A Sequential tracking extraction of shape features and its constructive description.
computer grafics and image processing 19th, p 349-366 1982.

[NOB-78] NOBUYUKI G., TAKASHI K., KIMIHISA I., MITSUO K.
An automatic inspection system for mask patterns.
Fourth intenational joint conference on PATTERN RECOGNITION KYOTO-JAPON 1978.

[PED-81] PEDERSON D.O
Computer aids in I.C design
proc. of the NATO advanced study institute on computer design aids for VLSI circuits. ITALY, aout 1981.

[RAH-79] RAHIER M.
Microélectronique personnalisée pour le lecteur de caractères OCRB.
TH. docteur en sc. appliquée Université catholique de louvain BELGIQUE 81.

[RIC-81] RICHARD F.
Simplified design rules for VLSI design.
LAMBDA, first quarter 1981

[SCH-83] SCHOELLKOPF J.P.
LUBRICK: A Silicon assembler and its application to data path design for FISC.
proc. of VLSI83, Trondheim, 1983

[SZY-83] SZYMINSKI T.G VAN WYK C.J
Space efficient algorithm for VLSI artwork Analysis.
ACM IEEE 20TH DAC PROCEEDINGS MIAMI BEACH, FLORIDA 83.

[TAR-83] TAROLLY G.M HERMAN W.J
Hierarchical circuit extraction with detailed parasitic capacitance
ACM IEEE 20TH DAC PROCEEDINGS MIAMI BEACH, FLORIDA 83.

[TOU-74] TOU J.T. GONZALEZ R.C.
Pattern recognition principles
ed. Addison-wesley publishing company 1974.

[VEI-77] VEILLON F.
Une méthode de calcul en un passage de plusieurs propriétés topologiques et géometriques d'objets dans les images digitalisées.
RR No 67 Janvier 77 IMAG.

[WANG-81] WANG P.S
Parallel context free array grammar normal forms
computer grafics and image processing n 15 p 296-300 1981.

[WAN-81] WANTANABE Y., SHUTOU T., KAWAI R., KIKUCHI Y.
A fundamental experiment on automatic LSI mask pattern drawing reading
Fourth intenational joint conference on PATTERN RECOGNITION KYOTO-JAPON 1978.

[WAN-82] WANTANABE T., YAMADA H., MIYAKAWA T., IWASAKI H.
Precise device recognition technique for bipolar IC/LSI
ICCC 82 (pp.304/307).

[WAR-81] DAVID H. D. WARREN
Logic Programming and compiler writing
Software-practice and experience vol 10 1981.

[WER-82] WERNER J.
CAD and DA in JAPAN
VLSI mai/juin 82.

[WIL-81] WILLIAMS L.
Automatic VLSI Layout Verification
18th DAC (pp.726/732)

[WILM-81] WILMORE J.A.
Efficient Boolean Operation on IC Masks.
18th DAC (pp. 571/578)

[YOS-81] YOSHIDA J. , OZAKI T. , GOTO Y.
PNAMAP-B: A Mask Verification System For Bipolar IC.
18th DAC P. 690-695

AUTORISATION DE SOUTENANCE

VU les dispositions de l'article 3 de l'arrêté du 16 avril 1974,

VU les rapports de présentation de Messieurs

. F. ANCEAU, Professeur

. JP. MOREAU

Monsieur DJARRAIA (JERRAYA) Ahmed Amine

est autorisé à présenter une thèse en soutenance pour l'obtention du diplôme de DOCTEUR-INGENIEUR, spécialité "Informatique".

Fait à Grenoble, le 9 novembre

Le Président de l'INP-G

D. BLOCH
Président
de l'Institut National Polytechnique
de Grenoble

P.O. le Vice-Président,

TITRE: UNE NOUVELLE APPROCHE POUR LA VERIFICATION DES MASQUES DES CIRCCUITS INTEGRES.

COMFOR: UN EXTRACTEUR DE SCHEMA ELECTRIQUE PARAMETRABLE PAR LA TECHNOLOGIE.

RESUME.

Ce travail présente une nouvelle approche pour la réalisation d'outils de vérification des masques de circuit integrés (CI). Le système COMFOR est un extracteur de schéma électrique paramétrable par la technologie. Ce système analyse des images de CI pour reconnaitre les composants électriques et calculer leurs caractéristiques.

Le système COMFOR est basé à la fois sur des notions de programmation logique et des techniques de reconnaissance syntaxique de formes.

Le systéme COMFOR est opérationnel à l'IMAG, il permet de générer une description électrique de CI pour des simulateurs électrique (MSINC et SPICE) à partir de leurs dessins de masques décrits en LUCIE.

MOTS-CLES:

Vérification des masques de C.I.
Programmation logique.
Reconnaisssance syntaxique des formes.