Test fonctionnel des circuits intégrés digitaux

Eric Archambeau

THESE

Présentée par

Eric ARCHAMBEAU

pour obtenir le titre de DOCTEUR

de l'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Spécialité : Microélectronique

TEST FONCTIONNEL DES CIRCUITS INTEGRES DIGITAUX

Soutenue le 21 octobre 1985 devant la Commission d'Examen :

Président :

Ch. LANDRAULT

Examinateurs :

C. BELLON

J.-C. GEFFROY

C. GRILLOT

G. SAUCIER

Thèse préparée au sein du Laboratoire Circuits et Systèmes

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Président: Daniel BLOCH

Vice-Présidents: B. BAUDELET
H. CHERADAME
R. CARRE
J.M PIERRARD

Année universitaire 1984-1985

Professeur des Universités

ANCEAU	François	E.N.S.I.M.A.G	JOUBERT	Jean-Claude	E.N.S.I.E.G
BARIBAUD	Michel	E.N.S.E.R.G	JOURDAIN	Geneviève	E.N.S.I.E.G
BARRAUD	Alain	E.N.S.I.E.G	LACOUME	Jean-Louis	E.N.S.I.E.G
BAUDELET	Bernard	E.N.S.I.E.G	LATOMBE	Jean-Claude	E.N.S.I.M.A.G
BESSON	Jean	E.N.S.E.E.G	LESIEUR	Marcel	E.N.S.H.G
BLIMAN	Samuel	E.N.S.E.R.G	LESPINARD	Georges	E.N.S.H.G
BLOCH	Daniel	E.N.S.I.E.G	LONGEQUEUE	Jean-Pierre	E.N.S.I.E.G
BOIS	Philippe	E.N.S.H.G	LOUCHET	François	E.N.S.E.E.G
BONNETAIN	Lucien	E.N.S.E.E.G	MASSELOT	Christian	E.N.S.I.E.G
BONNIER	Etienne	E.N.S.E.E.G	MAZARE	Guy	E.N.S.I.M.A.G
BOUVARD	Maurice	E.N.S.H.G	MOREAU	René	E.N.S.H.G
BRISSONNEAU	Pierre	E.N.S.I.E.G	MORET	Roger	E.N.S.I.E.G
BUYLE BODIN	Maurice	E.N.S.E.R.G	MOSSIERE	Jacques	E.N.S.I.M.A.G
CAVAIGNAC	Jean-François	E.N.S.I.E.G	PARIAUD	Jean-Charles	E.N.S.E.E.G
CHARTIER	Germain	E.N.S.I.E.G	PAUTHENET	René	E.N.S.I.E.G
CHENEVIER	Pierre	E.N.S.E.R.G	PERRET	René	E.N.S.I.E.G
CHERADAME	Hervé	U.E.R.M.C.P.P	PERRET	Robert	E.N.S.I.E.G
CHERUY	Arlette	E.N.S.I.E.G	PIAU	Jean-Michel	E.N.S.H.G
CHIAVERINA	Jean	U.E.R.M.C.P.P	POLOUJADOFF	Michel	E.N.S.I.E.G
COHEN	Joseph	E.N.S.E.R.G	POUPOT	Christian	E.N.S.E.R.G
COUMES	André	E.N.S.E.R.G	RAMEAU	Jean-Jacques	E.N.S.E.E.G
DURAND	Francis	E.N.S.E.E.G	RENAUD	Maurice	U.E.R.M.C.P.P
DURAND	Jean-louis	E.N.S.I.E.G	ROBERT	André	U.E.R.M.C.P.P
FELICI	Noël	E.N.S.I.E.G	ROBERT	François	E.N.S.I.M.A.G
FONLUPT	Jean	E.N.S.I.M.A.G	SABONNADIERE	Jean-Claude	E.N.S.I.E.G
FOULARD	Claude	E.N.S.I.E.G	SAUCIER	Gabrielle	E.N.S.I.M.A.G
GANDINI	Alessandro	U.E.R.M.C.P.P	SCHLENKER	Claire	E.N.S.I.E.G
GAUBERT	Claude	E.N.S.I.E.G	SCHLENKER	Michel	E.N.S.I.E.G
GENTIL	Pierre	E.N.S.E.R.G	SERMET	Pierre	E.N.S.E.R.G
GUERIN	Bernard	E.N.S.E.R.G	SILVY	Jacques	U.E.R.M.C.P.P
GUYOT	Pierre	E.N.S.E.E.G	SOHM	Jean-Claude	E.N.S.E.E.G
IVANES	Marcel	E.N.S.I.E.G	SOUQUET	Jean-Louis	E.N.S.E.E.G
JALINIER	Jean-Michel	E.N.S.I.E.G	VEILLON	Gérard	E.N.S.I.M.A.G
JAUSSAUD	Pierre	E.N.S.I.E.G	ZADWORNY	François	E.N.S.E.R.G

Professeurs Associés

BLACKWELDER	Ronald	E.N.S.H.G	PURDY	Gary	E.N.S.E.E.G
HAYASHI	Hirashi	E.N.S.I.E.G			

Professeurs Université des Sciences Sociales (Grenoble II)

BOLLIET	Louis	CHATELIN	Françoise

Chercheurs du C.N.R.S

CARRE	René	Directeur de recherche			
FRUCHART	Robert	Directeur de recherche	GUELIN	Pierre	Maître de recherche
JORRAND	Philippe	Directeur de recherche	HOPFINGER	Emil	Maître de recherche
VACHAUD	Georges	Directeur de recherche	JOUD	Jean-Charles	Maître de recherche
			KAMARINOS	Georges	Maître de recherche
ALLIBERT	Michel	Maître de recherche	KLEITZ	Michel	Maître de recherche
ANSARA	Ibrahim	Maître de recherche	LANDAU	Ioan-Dore	Maître de recherche
ARMAND	Michel	Maître de recherche	LASJAUNIAS	Jean-Claude	Maître de recherche
BINDER	Gilbert	Maître de recherche	MERMET	Jean	Maître de recherche
BORNARD	Guy	Maître de recherche	MUNIER	Jacques	Maître de recherche
			PIAU	Monique	Maître de recherche
DAVID	René	Maître de recherche	PORTESEIL	Jean-Louis	Maître de recherche
DEPORTES	Jacques	Maître de recherche	THOLENCE	Jean-Louis	Maître de recherche
DRIOLE	Jean	Maître de recherche	VERDILLON	André	Maître de recherche
GIGNOUX	Damien	Maître de recherche	SUERY	Michel	Maître de recherche
GIVORD	Dominique	Maître de recherche			

Personnalités habilitées à diriger des travaux de recherche
(Decision du Conseil Scientifique)

E.N.S.E.E.G.

ALLIBERT	Colette	DIARD	Jean Paul	NGUYEN TRUONG	Bernadette
BERNARD	Claude	EUSTATHOPOULOS	Nicolas	RAVAINE	Denis
BONNET	Roland	FOSTER	Panayotis	SAINFORT	(CENG)
CAILLET	Marcel	GALERIE	Alain	SARRAZIN	Pierre
CHATILLON	Catherine	HAMMOU	Abdelkader	SIMON	Jean Paul
CHATILLON	Christian	MALMEJAC	Yves (CENG)	TOUZAIN	Philippe
COULON	Michel	MARTIN GARIN	Régina	URBAIN	Georges (Laboratoire des ultra-réfractaire ODEILLO).

E.N.S.E.R.G.

BARIBAUD	Michel	CHEHIKIAN	Alain	HERAULT	Jeanny
BOREL	Joseph	DOLMAZON	Jean Marc	MONLLOR	Christian
CHOVET	Alain				

E.N.S.I.E.G.

BORNARD	Guy	KOFMAN	Walter	MAZUER	Jean
DESCHIZEAUX	Pierre	LEJEUNE	Gérard	PERARD	Jacques
GLANGEAUD	François			REINISCH	Raymond

E.N.S.H.G.

ALEMANY	Antoine	MICHEL	Jean Marie	ROWE	Alain
BOIS	Daniel	OBLED	Charles	VAUCLIN	Michel
DARVE	Félix			WACK	Bernard

E.N.S.I.M.A.G.

BERT	Didier	COURTOIS	Bernard	FONLUPT	Jean
CALMET	Jacques	DELLA DORA	Jean	SIFAKIS	Joseph
COURTIN	Jacques				

U.E.R.M.C.P.P.

		CHARUEL	Robert		

C.E.N.G.

CADET	Jean	JOUVE	Hubert (LETI)	PERROUD	Paul
COEURE	Philippe (LETI)	NICOLAU	Yvan (LETI)	PEUZIN	Jean Claude (LETI)
DELHAYE	Jean Marc (STT)	NIFENECKER	Hervé	TAIEB	Maurice
DUPUY	Michel (LETI)			VINCENDON	Marc

Laboratoires extérieurs :

C.N.E.T.

DEMOULIN	Eric	GERBER	Roland	MERCKEL	Gérard
DEVINE	R.A.B.			PAULEAU	Yves

I.N.S.A. Lyon

		GAUBERT	C.		

****

ECOLE NATIONALE SUPERIEURE DES MINES DE SAINT-ETIENNE

Directeur : Monsieur M. MERMET
Directeur des Etudes et de la formation : Monsieur J. LEVASSEUR
Directeur des recherches : Monsieur J. LEVY
Secrétaire Général : Mademoiselle M. CLERGUE

Professeurs de 1ère Catégorie

COINDE	Alexandre	Gestion
GOUX	Claude	Métallurgie
LEVY	Jacques	Métallurgie
LOWYS	Jean-Pierre	Physique
MATHON	Albert	Gestion
RIEU	Jean	Mécanique - Résistance des matériaux
SOUSTELLE	Michel	Chimie
FORMERY	Philippe	Mathématiques Appliquées

Professeurs de 2ème catégorie

HABIB	Michel	Informatique
PERRIN	Michel	Géologie
VERCHERY	Georges	Matériaux
TOUCHARD	Bernard	Physique Industrielle

Directeur de recherche

LESBATS	Pierre	Métallurgie

Maîtres de recherche

BISCONDI	Michel	Métallurgie
DAVOINE	Philippe	Géologie
FOURDEUX	Angeline	Métallurgie
KOBYLANSKI	André	Métallurgie
LALAUZE	René	Chimie
LANCELOT	Francis	Chimie
LE COZE	Jean	Métallurgie
THEVENOT	François	Chimie
TRAN MINH	Canh	Chimie

Personnalités habilitées à diriger des travaux de recherche

DRIVER	Julian	Métallurgie
GUILHOT	Bernard	Chimie
THOMAS	Gérard	Chimie

Professeur à l'UER de Sciences de Saint-Etienne

VERGNAUD	Jean-Maurice	Chimie des Matériaux & chimie industrielle

****

"Felix qui potuit rerum cognoscere causas."

Heureux celui qui a pu pénétrer les causes secrètes des choses.

Virgile, "Georgiques".

"Today I saw a red-and-yellow sunset and thought, How insignificant I am !

Of course, I thought that yesterday, too, and it rained."

Woody Allen, "Feathers".

Je tiens à exprimer toute ma reconnaissance à Madame Gabrièle SAUCIER, Professeur à l'ENSIMAG, d'avoir bien voulu m'accueillir dans son laboratoire et de m'avoir guidé durant la réalisation de cette thèse.

Je tiens à remercier :

- Monsieur C. LANDRAULT, Professeur à l'Université des Sciences et Techniques du Languedoc à Montpellier d'avoir accepté d'être rapporteur de cette thèse et de m'avoir fait l'honneur de présider le jury de cette thèse,

- Monsieur J.C. GEFFROY, Professeur à l'Institut des Sciences Appliquées de Toulouse d'avoir accepté d'être rapporteur de cette thèse et d'avoir bien voulu me faire profiter de ses remarques constructives,

- Madame C. BELLON, Maitre Assistant à l'ENSIMAG, de m'avoir fait profiter de son expérience dans le domaine du test,

- Monsieur F. GRILLOT, Chef de service Test Automatique et CAO chez Electronique Serge Dassault, d'avoir accepté de faire partie du jury de cette thèse.

Je tiens également à remercier :

- Les membres de l'équipe CAO de Matra Design Systems qui par leur soutien ont contribué à ce travail. En particulier Jacques-Olivier PIEDNOIR qui m'a fait profiter de son expérience dans le domaine du bi-partitionnement et Michel CAMPMAS qui a bien voulu relire et corriger le texte de cette thèse.

TABLE DES MATIERES

RESUME

L'objet de cette thèse est l'étude de deux méthodes de génération automatique de vecteurs de test pour les circuits intégrés digitaux. Après un rappel des problèmes actuels posés par le test des circuits VLSI (partie I), deux méthodes de génération automatique de vecteurs de test adressant deux types différents d'hypothèses de pannes sont présentées: une méthode heuristique de génération de vecteurs (partie II) et une méthode de test pseudo-exhaustif (partie III).

Dans la première partie, nous rappelons les intérêts économiques liés au test des circuits et les difficultés de la génération de tests pour les circuits VLSI. Puis nous précisons les modèles de pannes et les règles de conception de circuits testables utilisés dans le reste de la thèse.

Dans la deuxième partie, nous rappelons et critiquons les méthodes d'analyse de testabilité basées sur les notions de controlabilité et d'observabilité. Nous proposons une méthode globale d'analyse de testabilité. Puis nous introduisons des règles heuristiques qui permettent d'appliquer efficacement cette méthode à la génération de vecteurs de test, faisant l'hypothèse de panne unique de type collage, pour les circuits combinatoires, ou pouvant être testés comme tels.

Dans la troisième partie, nous définissons la notion de test pseudo-exhaustif puis nous étudions le taux de couverture des tests pseudo-exhaustifs. Nous évaluons la difficulté du problème du partitionnement optimal des circuits pour le test pseudo-exhaustif et nous proposons une méthode efficace de partitionnement.

MOTS CLES:
Controlabilité, Observabilité, Génération de tests, Taux de couverture, Test, Testabilité, Test Pseudo-Exhaustif.

AVANT-PROPOS

Ces quelques dernières années ont vu la naissance de puissants outils de CAO (conception assistée par ordinateur) sans lesquels le développement spectaculaire des circuits à très haute intégration, ou VLSI (Very Large Scale Integration), n'eût été possible. En particulier, les circuits utilisant des blocs précaractérisés ou des réseaux prédiffusés ne présenteraient que peu d'intérêt sans des outils automatiques ou semi-automatiques de conception topologique [Saucier 84] , [Hild 85] et de simulation logique et fonctionnelle [Archambeau 83a]. Pour ces familles de circuits, la longueur du cycle de conception a été ramenée à quelques semaines seulement, grâce à des systèmes de CAO intégrés sur des stations de travail (Daisy, Mentor, Silvar-Lisco et autres) disponibles dans le commerce (voir [VLSI 85a] et [VLSI 85b] pour une liste détaillée de ces systèmes) et bientôt des compilateurs de silicium [Buric 85].

Le test des circuits VLSI reste le dernier domaine qui résiste à l'automatisation du cycle complet de conception, et par conséquent, le coût de la génération de test devient un facteur de plus en plus important dans le coût global de conception des circuits intégrés (CIs).

Dans cette thèse, après avoir rappelé les problèmes liés au test et à la génération de test des CIs, nous présentons les modèles de pannes et les règles de conception de circuits testables qui nous serviront d'hypothèses (partie I).

Puis nous présentons deux méthodes de génération automatique de vecteurs de tests concernant deux types différents d'hypothèses de pannes: une méthode heuristique de génération automatique de vecteurs pour le test des pannes uniques de type collage (partie II) et une méthode de test pseudo-exhaustif qui n'est pas liée à un modèle de panne spécifique (partie III).

PARTIE I

Chapitre 1: Rappel des conditions économiques et techniques du test des CIs.

Chapitre 2: Rappel des problèmes liés à la génération de test et définition des hypothèses de pannes et des règles de conception utilisées dans la suite.

PARTIE II

Chapitre 1: Rappel et critique des méthodes d'analyse de testabilité des CIs.

Chapitre 2: Présentation d'une méthode d'analyse globale de testabilité des CIs.

Chapitre 3: Présentation d'une extension de cette méthode de génération automatique de vecteurs pour le test des pannes de type collage unique

Chapitre 4: Présentation des résultats obtenus avec un programme intégrant cette procédure.

PARTIE III

Chapitre 1: Justification et définition du test pseudo-exhaustif

Chapitre 2: Etude du taux de couverture obtenu avec les tests pseudo-exhaustifs.

Chapitre 3: Présentation d'une procédure de partitionnement des CIs pour la génération de tests pseudo-exhaustifs.

Chapitre 4: Présentation des résultats obtenus avec un programme intégrant cette procédure de partitionnement.

P A R T I E I

P R E S E N T A T I O N

1 LE TEST DES CIRCUITS INTEGRES

Les progrès récents réalisés dans le domaine de la conception assistée par ordinateur, permettent aux concepteurs de CIs digitaux, grâce à des outils de conception topologique et de vérification logique, électrique et topologique, de réaliser des circuits corrects par construction dès la première itération. Mais la fabrication et l'assemblage des circuits VLSI sont des opérations technologiquement très complexes. Il en résulte une nécessité absolue de s'assurer que les produits finis correspondent aux spécifications initiales, même pour les circuits dont la conception est correcte.

Le test en production des CIs a pour but de vérifier que les processus de fabrication et d'assemblage se sont déroulés correctement, et garantir ainsi le fonctionnement attendu des circuits finis.

Avec la prolifération de l'utilisation des CIs dans les domaines les plus variés, la défaillance des circuits eux-mêmes devient un facteur de plus en plus important dans le coût final du produit les utilisant. L'impact des CIs sur la marge bénéficiaire des systèmes les intégrant est basé, non seulement sur le prix d'achat du circuit, mais aussi sur sa fiabilité. Si une carte contient 20 circuits, chacun avec une probabilité d'être sans panne de 95%, le taux de cartes défectueuses est de 10%. Si un système contient lui-même 5 de ces cartes, la probabilité que ce système soit défaillant passe à 40%. On voit

donc l'enjeu économique du problème du test des CIs.

Le test complet des CIs est constitué d'un ensemble de tests statiques, dynamiques et fonctionnels. Ces tests vérifient respectivement:

- les paramètres physiques du circuit, assurant ainsi que le processus de fabrication a été réalisé suivant les normes établies. Notons que ce test est une vérification a posteriori des divers paramètres utilisés dans les modèles des logiciels de conception et de vérification.

- les caractéristiques de commutation liés à certains temps de propagation critiques à l'intérieur du CI.

- l'exactitude de la fonction logique realisée par le circuit. La présence de défauts physiques au niveau du silicium peut modifier complètement ou partiellement cette fonction, même pour un circuit ayant subi avec succès les tests précédents.

Nous nous intéressons dans la suite à cette dernière catégorie de tests: le test fonctionnel des CIs digitaux.

2 LE TEST FONCTIONNEL

2.1 Définitions Et Rappels

Le test fonctionnel d'un circuit est le processus de détection et d'identification des causes du fonctionnement incorrect de la logique d'un circuit. Nous nous bornerons ici à

la détection de ce fonctionnement incorrect. Un bilan exhaustif des differents aspects du test fonctionnel des circuits et des systèmes est donnée dans [Muehldorf 81].

Lorsqu'un disfonctionnement d'un CI se produit après son intégration dans un système, l'évènement peut être percu à quatre niveaux distincts, et une syntaxe différente est utilisée pour chaque niveau [Avizienis 82]:

- défaut au niveau physique,
- panne au niveau logique,
- erreur au niveau des données,
- non fonctionnement global au niveau de l'utilisateur

Durant le test des CIs, un fonctionnement incorrect aux deux premiers niveaux ne peut être détecté qu'au troisième niveau. Pour les circuits de grande et moyenne complexité, il est extrèmement difficile d'étudier chacune des nombreuses possibilités de défauts physiques et ses différents effets sur les paramètres physiques du circuit (courant, tension, etc..). C'est pourquoi cette étude se fait en général au niveau logique, afin de pouvoir travailler avec des algèbres simples (dont les variables sont souvent binaires ou ternaires, parfois à 4,5 ou 9 valeurs). De plus, l'effet au niveau logique des défauts physiques est souvent modélisé par des modèles de panne assez simples.

2.2 Modélisation Des Défauts

On peut distinguer entre deux grandes familles de pannes:

- Les pannes "statiques" qui modélisent un phénomène physique constant dans le temps (par exemple, deux lignes court-circuitées).

- Les pannes "dynamiques" qui modélisent un phénomène physique évoluant dans le temps (par exemple, une perte par courant de fuite).

En addition, on pourrait citer les pannes dues aux effets de proximité dans les mémoires extrèmement denses: ces problèmes sont résolus par des algorithmes ou des heuristiques très spécialisés, dont la description sort du cadre de cette étude.

Les pannes dynamiques constituent un problème important dont la détection peut être réalisée par un test fonctionnel si le phénomène est assez rapide pour être détectable dans un temps de l'ordre du cycle de test. Sinon, la détection de ces défauts nécessite l'application de tests spécifiques, à mi-chemin entre test fonctionnels et tests dynamiques.

Ces pannes sont particulièrement importantes dans le domaine des mémoires et des circuits MOS dynamiques. Notre étude couvrant plus spécifiquement les circuits TTL et MOS statiques, nous nous bornerons à considérer dans la suite uniquement des pannes statiques.

Le modèle de panne statique le plus répandu est celui de la panne de type collage. Pour un circuit décrit au niveau de la porte logique, l'hypothèse d'une panne de type collage unique suppose le fonctionnement logique normal des portes, avec le "collage" à 0 ou à 1 d'une connexion unique. Rappelons que cette hypothèse est faite au niveau logique, non physique, et qu'elle représente le comportement d'une connexion en présence d'un défaut physique, même s'il est geographiquement éloigné de celle-ci. La validité de ce modèle dépend de la technologie envisagée.

2.2.1 Circuits TTL -

En technologie TTL, la coupure d'une connexion se traduit par un collage à 1 vis-à-vis des opérateurs dont elle est une entrée. Les défauts internes des opérateurs se traduisent par un collage à 0 ou 1 de leur sortie.

Les défauts affectant plusieurs portes, ou l'occurence simultanée de plusieurs défauts peuvent être modélisés par une panne de type collage multiple, pour laquelle plusieurs connexions sont "collées".

Un autre modèle de panne à considérer est la panne de court-circuit. Une panne de court-circuit entre deux connexions ajoute, suivant la technologie (TTL ou nMOS), une porte ET ou OU supplémentaire entre les connexions. En TTL, certaines pannes de court-circuit peuvent conduire à la création d'un point mémoire dans le circuit [Breuer 76].

Néanmoins, le modèle de panne de type collage unique a été presque le seul utilisé par l'industrie depuis de nombreuses années. Il s'est avéré suffisant pour les circuits TTL de moyenne et grande complexité. Des études [Carter 82], [Hughes 84] ont prouvé que les tests assurant la détection des pannes de type collage unique permettaient aussi de détecter la plupart des pannes de type collage multiple et une partie non négligeable des pannes de type court-circuit. Nous adopterons donc ce modèle de panne dans la Partie II, et la méthode proposée permettra de générer des tests pour les pannes de type collage unique.

Pour les circuits de très haute complexité, la probabilité d'occurence des pannes complexes (pannes de court-circuit multiple, pannes de collage multiple) devient non négligeable, et il peut être intéressant de s'assurer de la détection de ces pannes, au prix d'une longueur de test accrue. Ces modèles devenant extrêmement complexes, la meilleure solution à ce problème semble être de s'affranchir de la nécessité de modéliser ces pannes pour la simulation. La méthode présentée dans la Partie III ne requiert pas de simulation de fautes, tout en garantissant la détection de ces pannes complexes.

2.2.2 Circuits CMOS -

En technologie CMOS, les collages à 0 ou à 1 sont seulement partiellement représentatifs des défauts internes des portes complexes. Depuis les travaux de Wadzack [Wadzack 78], il a généralement été admis que le test complet des circuits CMOS nécessite de considérer les pannes de "collage ouvert" des transistors (et ne pas se limiter aux pannes au niveau de la porte logique). En effet, en la présence d'un transistor "collé ouvert", les portes CMOS combinatoires peuvent devenir des points mémoires et la détection d'une telle panne requiert l'application d'une séquence de test plutôt que d'un vecteur de test.

La considération d'autres pannes dont la détection est encore plus difficile, comme le "collage fermé" de transistors ou le court circuit entre bornes de transistors, transforme le test complet des circuits CMOS en un tour de force pour le concepteur de tests [Bashiera 85].

Néanmoins, depuis 1978 un certain nombre d'algorithmes ont été publiés pour la détection des pannes de "collage ouvert" des circuits CMOS [El Ziq 81], [Chiang 83]. D'autres algorithmes permettent d'assurer la détection simultanée des pannes de "collage fermé" [Crouzet 78]. Enfin, des techniques ont été présentées pour détecter les courts-circuits en CMOS [Malaiya 82], [Acken 83].

L'insuffisance théorique d'un test couvrant les pannes de collage à 0 ou 1 pour la détection de ces pannes CMOS complexes est évidente [El Ziq 83]. Néanmoins, en pratique, pour les circuits CMOS pré-caractérisés actuels, l'analyse des défauts ayant provoqué le rejet des circuits en production fait peu état de cette famille de défauts. Nous ne considérerons donc pas explicitement ces familles de pannes dans le présent travail, et nous nous limiterons à la modélisation des pannes au niveau de la porte logique.

2.3 Complexité Du Test Fonctionnel

La difficulté majeure du test des CIs vient du fait que l'intérieur d'un CI n'est accessible, durant le test en production, que par ses broches d'entrée et de sortie. Cette difficulté va en s'accroissant car, si le nombre d'éléments que l'on peut intégrer dans un seul circuit VLSI grandit chaque année, le nombre de broches du circuit n'augmente pas en proportion.

L'objectif du test est de stimuler l'intérieur du circuit à partir des broches d'entrée et d'observer la réponse sur les broches de sortie. Un test se présente sous la forme d'une séquence logique de 0 et 1 qui, appliquée aux entrées du circuit, entraine au moins une valeur logique de sortie à être différente suivant qu'une panne est présente ou absente. L'ensemble des 0 et 1 logiques appliqués simultanément sur les entrées du circuit constitue un vecteur de test. Une séquence de test est la réunion ordonnée de ces vecteurs.

Une mesure largement utilisée de l'efficacité d'une séquence de test est le taux de couverture de cette séquence. Le taux de couverture est le pourcentage des pannes détectées par la séquence, par rapport au nombre total de pannes possibles dans le circuit. Le taux de couverture est donc lié au modèle de panne utilisé. Néanmoins pour n'importe quel modèle de panne utilisé, il peut exister dans un circuit des pannes indétectables [Smith 78] (par exemple, si le circuit est

redondant): le taux de couverture maximum du test est alors inférieur à 100%. C'est pourquoi l'identification des pannes indétectables est une information importante dans l'estimation du taux de couverture réel d'un test.

Les tests fonctionnels considérés dans le présent mémoire sont obtenus à partir d'une description logique du circuit, comme le sont aujourd'hui la plupart des tests de circuit de moyenne et grande complexité.

Néanmoins, pour les circuits très complexes (microprocesseurs, circuits d' entrées/sorties, co-processeurs) il existe d'autres méthodes de génération de tests fonctionnels qui ne sont pas basés sur la description détaillée du circuit, mais seulement sa fonctionnalité [Bellon 82], [Bellon 84]. Ces méthodes permettent de traiter globalement le problème du test des circuits complexes dans un temps raisonnable, sans avoir à contraindre les concepteurs à des règles très strictes de conception pour le test. Nous n'aborderons pas ici la description détaillée de ces approches, qui sont souvent complémentaires de celles proposées dans les parties II et III. Par la suite, toute mention du terme "méthode de génération de test fonctionnel" fera implicitement référence à une méthode utilisant une description du circuit au niveau logique.

Le problème de la génération automatique de test pour les CIs, à partir de la description logique du circuit, est un problème dont la complexité croît exponentiellement avec la taille du circuit et le nombre d'états possibles du circuit [Goel 81]. Même restreint aux circuits combinatoires, le problème est encore NP-complet [Ibarra 75] (voir Partie III

Chapitre 2 pour une définition plus détaillée des problèmes NP-complets). En fait, aucun programme n'est disponible pour réaliser efficacement la génération automatique de test de circuits complexes [Goel 81] [Jensen 82].

La solution la plus fréquemment employée jusqu'à présent dans l'industrie pour la determination de tests, consiste à réutiliser les vecteurs de tests utilisés pour la simulation logique du circuit [Yamada 77]. Lorsque le circuit n'est pas trop complexe, le taux de couverture du circuit est évalué avec un simulateur de fautes [Ulrich 73] [Archambeau 83b]. Mais fréquemment ce taux de couverture n'est même pas évalué, ce qui résulte en des taux de fiabilité des tests trop bas.

La seule solution viable actuellement proposée est d'imposer des règles très strictes de conception structurée qui permettent de limiter la complexité de la génération de tests [Williams 83], [McCluskey 85].

2.4 Conception Orientée Vers Le Test

Une solution, pour éliminer les problèmes liés au caractère séquentiel des circuits, consiste à concevoir les circuits de telle manière que chacun des états de la machine séquentielle soit facilement accessible, pour le contrôle et l'observation. Le problème du test des circuits séquentiels est alors ramené au problème (soluble) du test des circuits combinatoires. Parmi plusieurs méthodes proposées, [Williams 73] [Funatsu 75] [Eichelberger 77] [Koenemann 79] [Ando 80] [Mercer 81] (l'ensemble de ces méthodes étant décrit dans [McCluskey 84]), la méthode dite du "LSSD" (Level Sensitive Scan Design), développée par IBM [Eichelberger 77], [Botorff 77], [Godoy 77], [Eichelberger 77], [Botorff 79], [Lowden 79], [Arzoumanian 81], [Botorff 81], [DasGupta 82], est celle qui a recu le plus d'attention et a inspiré de nombreux fabricants de circuits et systèmes (STC, Amdhal, Fujitsu, UK5000). Cette méthode a également conduit au développement de circuits autotestables [Koenemann 80].

La diffusion rapide de ce type de règles de conception orientée vers le test des circuits VLSI, nous ont amenés à prendre comme hypothèse dans la suite de cette thèse l'application de règles de conception similaires au LSSD, pour ainsi ne considérer que la génération automatique de test de circuits combinatoires.

2.5 Complexité Du Test Des Circuits Combinatoires

Même pour les circuits VLSI de type LSSD pouvant être testés comme des circuits combinatoires, le coût de la génération automatique ou manuelle de tests est prohibitif [Goel 82]. Si le coût de la génération de tests est en général réduit par l'utilisation de génération pseudo-aléatoire, il faut encore évaluer le taux de couverture du test, et le coût de ce calcul peut devenir comparable au coût de la génération elle-même [Shedletsky 77]. Ne serait-ce que le calcul d'un intervalle de confiance pour ce taux de couverture peut devenir un problème non trivial [Savir 83].

Les deux méthodes de génération de test proposées dans cette thèse entendent résoudre une partie des problèmes soulevés dans la Partie I. La méthode décrite dans la Partie II [Archambeau 85b] a pour but de générer des tests pour les pannes de type collage unique des circuits combinatoires en un temps croissant lentement et proportionnellement avec la complexité du circuit. Cette rapidité d'exécution est obtenue grâce à l'utilisation de méthodes heuristiques. La technique de test décrite dans la Partie III [Archambeau 84a] permet de s'affranchir de la nécessité d'utiliser un simulateur de fautes et garantit par construction un taux de détection des défauts complexes très élevé.

PARTIE II

PROCEDURE HEURISTIQUE DE GENERATION DE TESTS

1 CONTROLABILITE ET OBSERVABILITE

1.1 Mesures De Testabilité

Un certain nombre de méthodes heuristiques très rapides ont été développées pour surmonter les difficultés potentielles liées à la génération de tests pour les circuits intégrés (CIs) digitaux [Dussault 78], [Breuer 79], [Goldstein 79], [Grason 79], [Bennets 80], [Kovijanic 81], [Susskind 81], [Agrawal 82], [Berg 82]. SCOAP [Goldstein 80] est probablement le plus connu et le plus utilisé parmi les programmes d'analyse de testabilité. Différentes versions [Ratiu 81], [Archambeau 83c], [Wang 84] ont été développées à partir du programme original et proposent un certain nombre d'extensions tout en gardant l'algorithme de base. SCOAP est basé sur les concepts de controlabilité (contrôle) et observabilité (visibilité) au niveau de la porte logique.

1.1.1 Programme SCOAP -

Considérons une simple porte ET, illustrée Fig. 2-1, et supposons qu'un test doive être généré pour la panne "A collée à 1". En forçant A à 0 et B à 1, C peut être observé pour vérifier si la panne "A collée à 1" est présente (C=1) ou non (C=0). Dans l'exemple ci-dessus, vérifier la présence d'une panne de collage sur cette porte ET est un processus très simple. Mais si les noeuds A et B doivent être controlés et le

noeud C observé à travers de nombreuses couches de logique, le processus de génération de tests pour les pannes de collage devient beaucoup plus compliqué. SCOAP génère, pour chaque noeud du circuit, une mesure (nous critiquerons sa validité au chapitre 1.2) de la difficulté pour contrôler ce noeud à 0 ou 1 (respectivement appelées mesures CC0 et CC1) et pour observer ce noeud sur les broches de sortie du CI (appelée mesure CO).

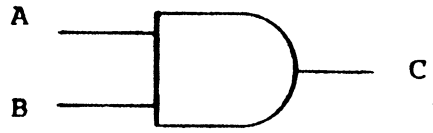


Figure 2-1: Controle et observation des connexions pour une simple porte ET.

Pour les circuits séquentiels, en plus des mesures combinatoires citées plus haut, SCOAP génère, pour chaque noeud du circuit, trois mesures supplémentaires, appelées mesures séquentielles, de controlabilité à 0 (SC0) ou à 1 (SC1) et d'observabilité (SCO). Ces mesures séquentielles représentent le nombre de séquences temporelles (coups d'horloge dans le cas des circuits synchrones) nécessaires pour atteindre une certaine condition sur un noeud. Les mesures combinatoires représentent alors le nombre de noeuds dont la valeur logique a été forcée à 1 ou à 0 dans chaque séquence temporelle. Des modèles simplifiés sont utilisés pour les éléments séquentiels, et les différents ensembles de noeuds contraints pour chaque séquence

sont regroupés en un seul ensemble simplifié.

L'algorithme de SCOAP traite les rebouclages asynchrones et a été implementé en une procédure dont le temps d' éxécution croit, au plus, de facon quadratique avec la taille du circuit - ARCOP [Archambeau 83c] - qui peut être décrite comme suit:

- Les mesures CC0 et CC1 sont calculées pour chaque noeud durant la première passe. Les valeurs associées aux entrées primaires, le minimum possible pour CC0 et CC1, sont initialisées à 1. Pour tous les autres noeuds les CC0 et CC1 sont initialisées à une valeur maximale représentant l'infini. On a aussi CC0(VDD) = infini et CC1(VSS) = infini.

  Puis une mesure de controlabilité est calculée pour chaque noeud, en progressant des entrées primaires vers les sorties primaires. Les tableaux 2-1 et 2-2 illustrent les règles utilisées pour le calcul des valeurs CC0 et CC1 des noeuds de sortie des portes logiques en fonction des valeurs des noeuds d'entrées.

Tableau 2-1: Calcul des mesures CC0 pour les portes logiques AND2, NAND2, OR2, NOR2, inverseur (INV) et tampon direct (BUF).

AND2	CC0 (OUT) = min (CC0 (IN1), CC0 (IN2)) + 1
NAND2	CC0 (OUT) = CC1 (IN1) + CC1 (IN2) + 1
OR2	CC0 (OUT) = CC0 (IN1) + CC0 (IN2) + 1
NOR2	CC0 (OUT) = min(CC1(IN2), CC1(IN2)) + 1
INV	CC0 (OUT) = CC1 (IN) + 1
BUF	CC0 (OUT) = CC0 (IN) + 1

Tableau 2-2: Calcul des mesures CCi pour les portes logiques AND2, NAND2, OR2, NOR2, inverseur (INV) et tampon direct (BUF).

AND2	CC1 (OUT) = CCi (INi) + CCi (IN2) + 1
NAND2	CC1 (OUT) = min (CC0 (IN1), CC0 (IN2)) + 1
OR2	CCi (OUT) = min(CCi(IN2), CCi(IN2)) + 1
NOR2	CC1 (OUT) = CC0 (INi) + CC0 (IN2) + 1
INV	CC1 (OUT) = CC0 (IN) + 1
BUF	CC1 (OUT) = CCi (IN) + 1

Une fois le calcul effectué, le résultat n'est propagé que si la nouvelle valeur est strictement inférieure à la valeur précédente (la valeur initiale est infinie). L'algorithme converge donc toujours, même en cas de rebouclage. Figure 2-2 représente un simple circuit çombinatoire, utilisé pour illustrer le calcul de la mesure de controlabilité.

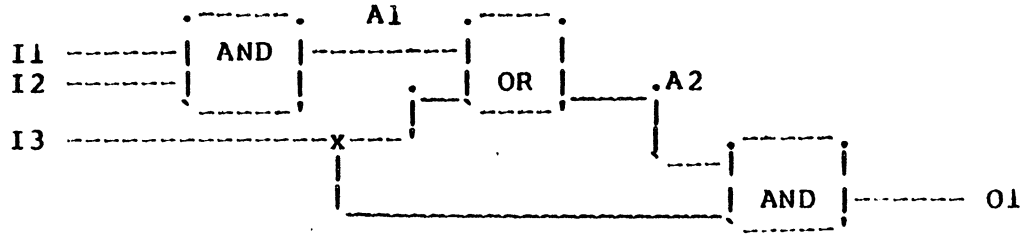


Figure 2-2: Exemple du calcul de la mesure de controlabilité

Dans l'exemple ci-dessus, les valeurs CC0 et CC1 sont d'abord initialisées:

CC0(I1) = 1 CC0(A1) = CC0(A2) = infini

CC1(I1) = 1 CC1(A1) = CC1(A2) = infini

CC0(I2) = 1

CC1(I2) = 1

CC0(I3) = 1

CC1(I3) = 1

Puis le calcul est effectué en progressant des entrées primaires vers les sorties:

CC0(A1) = min(CC0(I1),CC0(I2)) + 1 = 2

CC1(A1) = CC1(I1) + CC1(I2) + 1 = 3

CC0(A2) = CC0(I3) + CC0(A1) + 1 = 4

CC1(A2) = min(CC1(I3),CC1(A1)) + 1 = 2

CC0(O1) = min(CC0(A2),CC0(I3)) + 1 = 2

CC1(O1) = CC1(A2) + CC1(I3) + 1 = 4

- La mesure d'observabilité (CO) pour chaque noeud est ensuite calculée dans une second passe. Tous les CO sont initialisés: à 0 pour les sorties primaires, à l'infini pour les autres noeuds. L'algorithme progresse des sorties vers les entrées primaires, et les COs de chaque entrée de porte sont calculés à partir des CC0 et CC1 des autres

entrées de la porte et du CO de la sortie de cette porte, comme illustré par le tableau 2-3.

Tableau 2-3: Calcul de la mesure d'observabilite pour les portes logiques AND2, NAND2, OR2, NOR2, inverseur (INV) et tampon direct (BUF).

AND2	CO (INi) = CC1 (IN2) + CO (OUT) + 1
NAND2	CO (INi) = CC1 (IN2) + CO (OUT) + 1
OR2	CO (IN1) = CC0 (IN2) + CO (OUT) + 1
NOR2	CO (IN1) = CC0 (IN2) + CO (OUT) + 1
INV	CO (IN) = CO (OUT) + 1
BUF	CO (IN) = CO (OUT) + 1

Pour les noeuds à sortance multiple, l'observation de ce noeud peut être plus ou moins difficile suivant le chemin suivi jusqu'aux sorties primaires. L'algorithme calcule une valeur de CO (un CO de branche) pour chaque connexion sortante. La valeur finale du CO de ce noeud est le minimum des CO de ces branches.

Reprenons l'exemple de la Fig. 2-2 pour illustrer le calcul de la mesure d'observabilité. Les valeurs de CO pour chaque noeud sont initialisées de la facon suivante:

$CO(Oi) = 0$

$CO(A1) = infini$

$CO(A2) = infini$

Puis l'algorithme progresse des sorties vers les entrées, en considérant les noeuds à sortance multiple:

$CO(A2) = CO(O1) + CC1(I3) = 1$

$CO(I3-O1) = CO(O1) + CC1(A2) = 2$

$CO(I3-A2) = CO(A2) + CC0(A1) = 3$

$CO(I3) = \min (CO(I3-A2), CO(I3,O1)) = 2$

$CO(A1) = CO(A2) + CC0(I3) = 2$

$CO(I1) = CO(A1) + CC1(I2) = 3$

$CO(I2) = CO(A1) + CC1(I1) = 3$

- Enfin ces mesures de controlabilité et d'observabilité sont utilisées pour générer des mesures de testabilité des pannes de collage, suivant le raisonnement suivant: la détection de la panne "noeud p collé-a-zero" nécessite de forcer p à 1 (une mesure de cette difficulté est justement CC1) et d'observer cette valeur de p sur l'une des sorties primaires (une mesure de cette difficulté est justement CO). En supposant que les opérations de contrôle et d'observation d'un même noeud soient indépendants (voir chapitre 1.2 pour la validité d'une telle hypothèse) une mesure de la testabilité de cette panne est:

TESTABILITE(p collé à 0) = CC1(p) + CO(p)

Et symétriquement,

TESTABILITE(p collé à 1) = CC0(p) + CO(p)

Dans le circuit de la Fig. 2-2, nous pouvons ainsi calculer une valeur de testabilité pour chacun des noeuds. Par example, pour le noeud A1:

testabilité(A1-collé-à-0) = 2 + 3 = 5

testabilité(A1-collé-à-1) = 2 + 2 = 4

Un exemple plus complexe des résultats fournis par ARCOP est donné en ANNEXE 1.

1.1.2 Programmes Similaires À SCOAP -

Une version modifiée de SCOAP, COMET [Berg 82] a été introduite pour répondre aux besoins des circuits pré-caractérisés. COMET inclut une mesure de testabilité supplémentaire, appelée "predictabilité", qui est une mesure de la facilité avec laquelle le circuit est initialisé dans un etat connu.

La version originale de SCOAP ne donne pas directement de données spécifiques au test du circuit. Des extensions de SCOAP ont donc été necessaires pour rendre le programme plus proche des problèmes de génération de tests. Des programmes comme ARCOP à Matra Design Systems [Archambeau 83c] et DTA à Daisy Systems [Wang 84] sont de telles extensions de SCOAP. Le calcul des estimations de la testabilité des pannes du circuit en fonction des mesures CC0, CC1 et CO a été donné au chapitre

précédent.

TESTSCREEN [Kovijanic 79] a été développé indepedemment de SCOAP, mais est basé sur le meme travail fondamental de Breuer: TEST/80 [Breuer 78]. TESTSCREEN est très semblable à SCOAP, mais son algorithme n'augmente pas necessairement la valeur de controlabilité lorsqu'un composant est traverse. Il tient aussi compte du nombre total de portes et d'entrées primaires. En fait, on peut redéfinir les mesures de controlabilité et d'observabilité de TESTSCREEN comme le nombre minimum nécessaire d'entrées primaires à contraindre (plutôt que de portes logiques à contraindre) pour controler un noeud à 0 ou 1 ou observer la valeur logique d'un noeud sur une sortie primaire. Grace à ce changement, les résultats de TESTSCREEN sont souvent plus facile à interpréter par un concepteur de circuit que les résultats de SCOAP. Par exemple, le dernier élément d'une chaîne de n inverseurs reliée à une entrée primaire, a une mesure de controlabilité de 1 avec TESTSCREEN, n-1 avec SCOAP. Il est plus difficile d'interpréter le résultat de SCOAP car, intuitivement, la sortie du dernier inverseur est aussi facile à controler que l'entrée du premier. Notons qu'un programme similaire à TESTSCREEN, COP, est utilisé à Texas Instrument.

I.1.3 Autres Programmes

CAMELOT [Bennetts 81] et TMEAS [Grason 79] sont deux programmes très voisins l'un de l'autre, qui font aussi une analyse de testabilité des circuits digitaux. Ces programmes utilisent des modèles plus simples que SCOAP, ce qui tend à diminuer encore plus la précision des résultats. Par exemple, avec une seule mesure de controlabilité (notée CY) au lieu de deux (CC0 et CC1) pour SCOAP, CAMELOT détecte difficilement des anomalies potentielles crées par une source de tension directement liée à une porte. Par contre les simplifications des modèles conduisent à une analyse plus rapide qu'avec SCOAP. Une très bonne description détaillée de CAMELOT est donnée dans [Bennetts 84]. A noter qu'une version évoluée de CAMELOT est commercialisée par GenRad sous le nom de HITAP. En plus de ces programmes basés sur les concepts de controlabilité et d'observabilité, des approches différentes pour l'analyse de testabilité ont été également abordées. Dussault [Dussault 78] utilise la notion d'entropie de l'information dans un circuit pour définir une mesure de testabilité. Savir [Savir 83] se sert de la notion de retard à la détection d'une erreur ("error latency") dans un circuit combinatoire pour étudier sa testabilité. Dans l'analyse critique qui suit, seules seront considérées les méthodes basées sur les notions de controlabilité et d'observabilité.

1.2 Analyse Critique

La simplicité et la rapidité de ces méthodes d'analyse de testabilité les rend particulièrement attirantes lorsqu'elles sont comparées aux lourdes méthodes algorithmiques de génération automatique de test. Malheureusement les résultats obtenus possèdent des limitations inhérentes aux simplifications de la procédure: nous examinons dans cette section les possibilités d'utilisation et les limitations des résultats d'analyse de testabilité du type SCOAP. Une analyse de la signification de ces résultats a par ailleurs été publiée dans [Agrawal 82].

Ces résultats ont une application directe pour laquelle ils sont particulièrement bien adaptés: l'insertion de points de test (contrôle, observation ou les deux) dans le circuit. Les meilleurs candidats pour l'insertion de points de contrôle ou d'observation sont les noeuds avec les plus hautes valeurs de mesure de controlabilité ou d'observabilité. Des programmes comme ARCOP (Matra Design Systems) ou DTA (Daisy Systems) vont jusqu'à évaluer l'impact de l'insertion d'un ou plusieurs points de test dans le circuit sur la testabilité de celui-ci, en calculant les nouvelles valeurs de testabilité des pannes de collage pour le circuit modifié et en les comparant aux anciennes valeurs. Le problème majeur lié à cette méthode d'insertions de points de test est qu'elle est difficile à mettre en oeuvre dans un environnement industriel. Dans la plupart des cas, le circuit utilise déjà toutes les broches disponibles sur le support, et le coût pour changer ce boitier

serait plus important que les bénéfices éventuels apportés par l'insertion de points de tests.

Les résultats de l'analyse de testabilité peuvent être utilisés de manière plus directe dans la génération de test en remarquant qu'il existe une corrélation entre la testabilité d'une panne et la probabilité de détection de cette panne: les pannes qui ont une faible valeur de testabilité ont une probabilité plus grande d'être détectées par un vecteur de test selectionné au hasard. Une strategie de génération de test consiste donc à commencer par générer des tests pour les pannes qui ont une forte valeur de testabilité. Statistiquement, les pannes avec une plus faible valeur de testabilité seront aussi détectées par ces tests.

Un problème demeure pour l'utilisation de ces mesures dans n'importe laquelle des strategies précitées: quelle confiance peut-on accorder à ces mesures de testabilité ?

Nous avons déjà mentionné les simplifications du modèle utilisé dans les calculs de mesure de controlabilité et d'observabilité. Plusieurs méthodes ont été étudiées pour prendre en compte des modèles de calcul plus complexes: malheureusement, toute complication du calcul résulte en la perte du bénéfice de la vitesse et le retour vers des problèmes NP-complets. Les limitations de ces mesures de testabilité résultent essentiellement de l'analyse qui est faite du circuit, une analyse basée uniquement sur la structure logique locale du circuit. De ce fait, certaines des causes de base des problèmes de génération de test ne sont pas detectées par cette analyse.

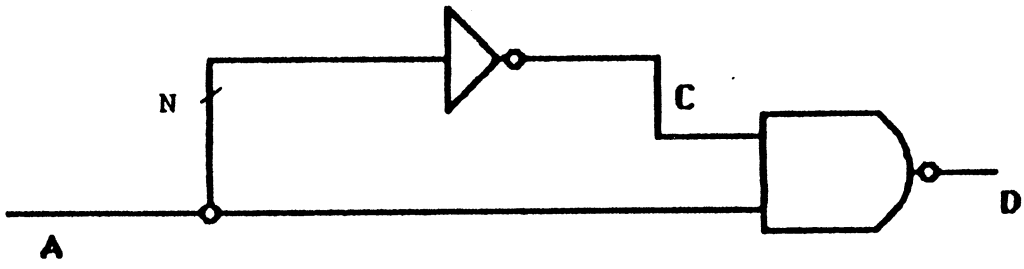


Figure 2-3: Problème de testabilité non detecté par SCOAP.

Dans la simple structure de la Fig. 2-3, la panne "N-collé-à-0" est typiquement indetectable. Mais SCOAP génère pour cette panne une valeur de testabilité (ici 2) qui ne retiendra pas l'attention du concepteur du circuit. Une procédure d'analyse de testabilité produisant des résultats utilisables pour la génération de test doit donc faire une analyse globale (et non locale) de la testabilité du circuit.

2 ANALYSE GLOBALE DE TESTABILITÉ

2.1 Definitions

Nous présentons ici les modèles et notations utilisés dans la partie II de la thèse.

2.1.1 Modèle De Circuit Logique -

Un circuit logique est un réseau de cellules interconnectées par des liens unidirectionnels, appelés noeuds, en communication avec l'univers extérieur à travers ses entrées primaires (EPs) et ses sorties primaires (SPs). Chaque cellule du réseau est un bloc fonctionnel qui remplit une certaine fonction logique prédéfinie et a ses propres entrées/sorties.

Les fonctions des différentes cellules, combinées par la topologie des connexions, réalisent la fonction globale du circuit. Chaque lien entre cellules représente un lieu possible pour une panne de type collage. Chaque noeud reliant exactement deux cellules correspond à deux pannes de collage equivalentes, une panne sur la sortie de la cellule attaquante et une panne sur l'entrée de la cellule attaquée.

2.1.2 Modélisation Des Pannes De Type Collage -

Un noeud qui relie une cellule à plusieurs autres est appelé un noeud à sortance multiple (ou point de sortance) et comporte une racine et des branches associées. La sortance d'un noeud en n directions signifie la reproduction de ce noeud en n positions physiques distinctes. Bien que la racine et les branches d'un point de sortance représentent le même noeud du circuit logique, les pannes situées sur la racine et sur les branches ne sont pas équivalentes. Un test qui détecte une panne sur la racine détecte nécessairement une panne sur au moins une des branches (mais une analyse purement locale ne peut prédire quelle branche). Un test qui détecte une panne sur

l'une des branches doit nécessairement détecter la panne sur la racine, mais détecte rarement les pannes sur les autres branches. Une analyse correcte des pannes du circuit nécessite donc la distinction entre les pannes sur les racines et les pannes sur les branches des points de sortance.

En général, le langage de description des circuits logiques fait correspondre un nom unique à chaque noeud. Il en résulte que le même nom désigne à la fois la racine et les branches d'un point de sortance. Pour distinguer entre ces lieux de pannes différentes, la panne sur la racine est repérée par le nom du noeud lui-même et les pannes sur les branches sont repérées par un nom composite de branche, de la forme: "racine-sortie". Dans ce nom composite, "racine" est le nom original du noeud dans le circuit, "sortie" est le nom du noeud, sortie de la première cellule connectée à la branche considérée. La figure 2-4 illustre cette notation.

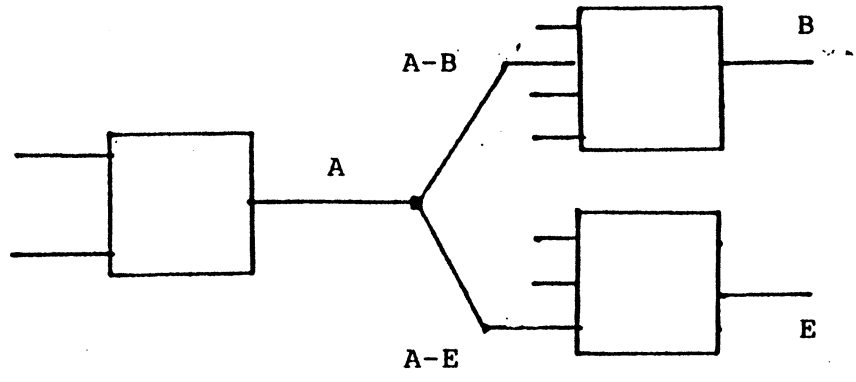


Figure 2-4: Notations pour les racines et les branches des noeuds du circuit.

Enfin pour un noeud X (X pouvant être un nom composite pour les branches) les pannes de type collage à 0 et 1 seront notées respectivement X/0 et X/1.

2.1.3 Classification Des Circuits Combinatoires -

Nous proposons ici une classification des circuits combinatoires basée sur la topologie des points de sortance. Plutôt que d'utiliser la notation "sortance reconvergente ou non-reconvergente" [Armstrong 66], nous utilisons la notation "convergente ou divergente" pour notre classification. Nous distinguons 3 classes de circuits: circuits de sortance unité, circuits à points de sortance divergents et circuits à points de sortance convergents.

Les circuits de sortance unité sont des circuits ne contenant aucun point de sortance multiple, c'est à dire dont chaque noeud relie exactement deux cellules. Cette structure est la plus simple, mais est en général limitée à des circuits de très faibles dimensions. La figure 2-5 est un exemple d'une telle structure.

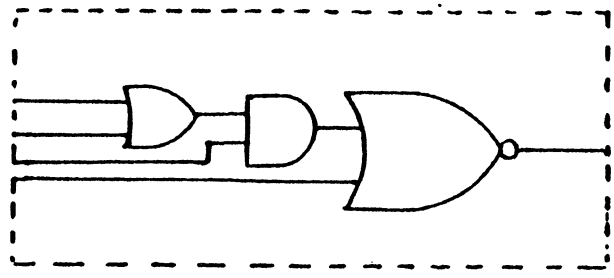


Figure 2-5: Cellule CG8 (Matra-Harris Gate Arrays Library)

Les circuits à points de sortance divergents contiennent des points de sortance dont toutes les branches divergent, c'est à dire que, pour tous les points de sortance, deux signaux issus de deux branches differentes ne sont jamais des entrées pour la même cellule. Ces circuits sont donc des arbres et des forêts et leur utilisation est généralement limitée à des applications specifiques, tels que certains circuits de contrôle combinatoire de parité.

Les circuits à points de sortance convergents sont des circuits pour lesquels au moins un noeud du circuit a deux branches convergentes, c'est à dire deux branches dont au moins deux signaux issus de ce noeud servent d'entrées à une même cellule. La plupart des circuits digitaux et virtuellement tous les circuits VLSI appartiennent à cette dernière catégorie.

2.1.4 Classification Des Pannes Indetectables -

Par définition, nous appelerons panne indetectable, toute panne pour laquelle il n'existe aucun test qui la detecte. Un noeud N est totalement redondant si les pannes N/0 et N/1 sont simultanément indetectables. Si seulement l'une des deux pannes est indetectable, N est dit partiellement indetectable. Trois classes de pannes indetectables peuvent être distinguées: les pannes incontrolables, les pannes inobservables et les pannes "intestables-controlables-observables".

Les pannes incontrolables ne peuvent être activées depuis les EPs. Pour ces pannes, il n'existe pas de vecteur d'entrée qui permet de forcer le noeud correspondant au niveau logique correct. Ainsi, les pannes non forcables de collage à 1 correspondent à des noeuds non forcables à 0, et les pannes incontrolables de collage à 0 correspondent à des noeuds incontrolables à 1. Par exemple, le noeud K de la Fig. 2-6 est toujours au niveau logique 0. puisque la branche A-K et le noeud J sont toujours à des valeurs logiques opposées. La panne K/0 ne peut être activée et donc la panne K/0 est incontrolable à 1.

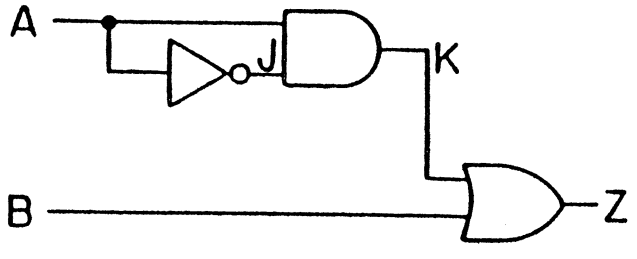


Figure 2-6: Circuit contenant une panne incontrolable (K/0)

Les pannes inobservables ne peuvent être propagées jusqu'aux SPs, c'est à dire qu'il n'existe pas de vecteur d'entrée qui permette d'observer le niveau logique du noeud correspondant sur une sortie du circuit. Dans la Fig. 2-7, une panne sur le noeud A est propagée jusqu'à la sortie Z si et seulement si B-J = 1 et B-Z = 0 . Mais cette condition est impossible à réaliser, puisque la racine B ne peut être simultanément à 0 et à 1. Donc A/0 et A/1 sont toutes deux totalement indetectables.

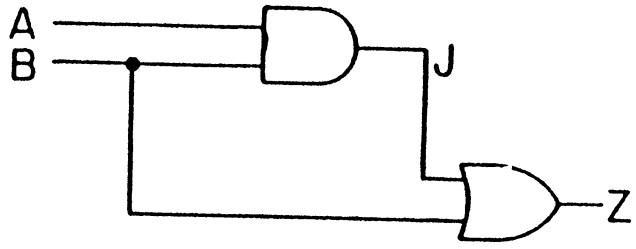


Figure 2-7: Circuit contenant des pannes inobservables

Les pannes "intestables-controlables-observables" (notées par la suite plus simplement "intestables") peuvent être activées depuis les EPs. Elles peuvent être propagées jusqu'aux SPs, mais il n'existe pas de vecteur d'entrée qui, simultanément, active et propage la panne. Lorsque les deux conditions sont imposées au même moment un conflit apparait sur les EPs, pour toutes les combinaisons possibles. L'exemple donné dans [Schneider 67], illustré par la Fig. 2-8, contient deux pannes, B-K/0 et C-K/0 qui sont à la fois controlables et observables, mais intestables. Le vecteur 0000 est le seul qui propage les pannes jusqu'aux SPs, mais il n'active aucune des deux pannes, bien que celles-ci soient facilement controlables.

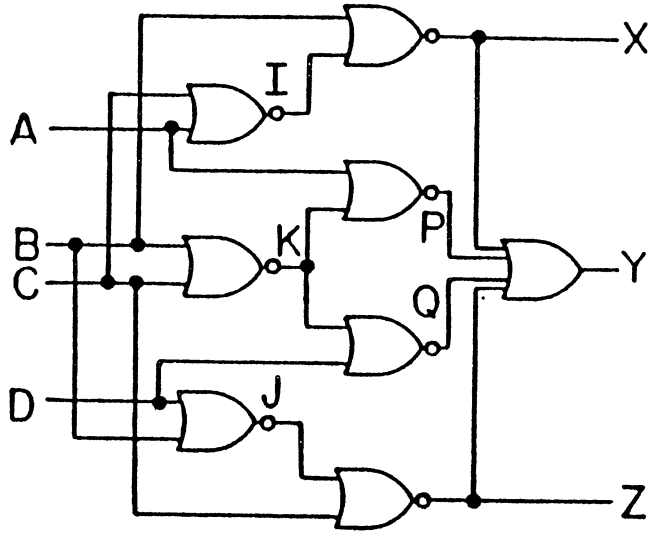


Figure 2-8: Circuit contenant des pannes intestables-controlables-observables (B--K/0 et C-K/0)

Une panne est indetectable si et seulement si elle est incontrolable, inobservable ou intestable. Les pannes incontrolables sont les plus faciles à mettre en évidence, puisque ce sont celles qui mettent en jeu le moins d'EPs. Il est plus difficile d'identifier les pannes inobservables car pour établir un chemin de propagation jusqu'aux SPs demande une serie de conditions de contrôle sur les noeuds adjacents au chemin. Une analyse globale du circuit est absolument indispensable pour identifier les pannes intestables. Une méthode qui effectue seulement une analyse partielle ne peut garantir l'identification de ce dernier type de panne.

2.2 Théorèmes De Base

Dans cette section, nous établissons deux résultats de

base:

- l'interdépendance entre les points de sortance convergents et la dépendance des signaux.
- l'existence de pannes indetectables est liée à l'existence de points de sortance convergents.

2.2.1 Sortances Convergentes Et Dépendence Des Signaux -

A l'intérieur du circuit, les points de sortance jouent le même rôle de distribution pour les signaux internes que les EPs pour les signaux externes. Les points de sortance et les EPs sont parfois appelés les " points de contrôle" (checkpoints) du circuit [Breuer 76]: tout test qui détecte les pannes de collage unique (resp. multiple) sur tous les points de contrôle détecte aussi toutes les pannes de collage unique (resp. multiple) dans tout le circuit.

Théorème 2-1: Dans un circuit combinatoire, les entrées d'une même cellule dépendent les unes des autres si et seulement si elles appartiennent à un même arbre convergent (elles sont issues d'un même point de sortance).

Demonstration: Si deux branches du circuit dépendent l'une de l'autre, elles doivent nécessairement appartenir à un même arbre, sinon elles sont indépendantes. Puisqu'elles sont entrées d'une même cellule, par définition, l'arbre est convergent. La réciproque se déduit directement de la définition d'une sortance convergente et des propriétés d'un

arbre.

Si deux entrées d'une cellule dépendent l'une de l'autre, une condition imposée sur l'une peut affecter l'autre aussi. Le théorème 2-1 spécifie qu'il existe une relation entre la fonction et la structure dans un circuit combinatoire: les analyses structurelles et fonctionnelles du circuit sont équivalentes si elles sont faites globalement et non localement. De plus, ce resultat garantit qu'une analyse structurelle (resp. fonctionnelle) complète du circuit est effectivement complète même si une partie de cette analyse utilise la description fonctionnelle (resp. structurelle) du circuit, à condition que la globalité de l'analyse soit respectée.

2.2.2 Sortances Convergentes Et Redondances -

La relation entre sortances convergentes et pannes indetectables a été pour la première fois analysée dans [Armstrong 66], lorsque la méthode de chemin unique de sensibilisation (single path sensitization) a été introduite. Armstrong distingue deux cas:

- Cas 1: Tous les chemins reconvergents entre un point de sortance donné et un point de reconvergence donné ont la parité d'inversion (la parité du nombre d'inversions sur chaque chemin).

-- Cas 2: Les chemins n'ont pas tous la même parité.

Armstrong montre que le premier cas entraine toujours des pannes détectables, alors que le deuxième cas peut conduire à des pannes non détecables. Par exemple, dans le circuit de la Fig. 2-6, les chemins <A,A-K,K> et <A,A-J,J,K> ont des parités d'inversion différentes, et le noeud K est totalement redondant.

Mais la parité d'inversion n'est une condition ni nécessaire ni suffisante pour obtenir une redondance. Par exemple, le circuit de la Fig. 2-7 contient un noeud totalement redondant, A, même si les deux chemins convergents, <B,B-J,J,Z> et <B,B-Z,Z>, ont même parité d'inversion. Réciproquement, le circuit de la Fig. 2-9 contient deux paires de chemins convergents, <A,A-M,M,Z>,<A,A-K,K,N,Z> et <B,B-J,J,M,Z>,<B,B-N,N,Z>, chacun avec des parités d'inversion différentes, mais aucune panne n'est indetectable.

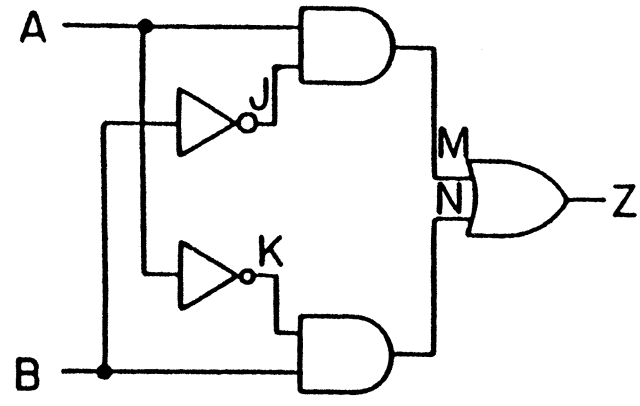


Figure 2-9: Circuit contenant deux paires de chemins convergents

Puisque la parité d'inversion n'est une condition ni nécessaire ni suffisante pour l'existence de redondances, son utilisation pour leur identification est limitée. Par contre, il existe un lien fondamental entre l'indetectabilité des pannes de collage et les sortances reconvergentes.

Théorème 2-2: Une panne de type collage unique est indetectable si et seulement si tous les chemins reliant les EPs et les SPs qui passent par l'endroit de la panne traversent un même arbre convergent.

Démonstration: Un chemin reliant les EPs aux SPs traverse un même arbre convergent si et seulement si tous les chemins de contrôle sont convergents, tous les chemins qui permettent l'observation sont convergents, ou encore si les chemins de contrôle et d'observation sont séparément divergents mais sont convergents lorsque ils sont considérés comme un seul chemin. Dans chacun de ces cas, les signaux sont dépendants (grâce au théorème 2-1). Supposons maintenant que les signaux soient tous indépendants, donc que le chemin ne traverse pas de circuit convergent. Le changement de tout signal affecte de manière unique la fonction du circuit et, par définition, il n'existe pas de panne indetectable. Donc, le chemin doit traverser un arbre convergent pour que la panne soit indetectable.

Le théorème 2-2 prouve que la redondance nécessite l'existence de convergence: aucune panne ne peut être indetectable s'il existe un chemin divergent à travers lequel la panne peut être controlée et observée. La réciproque de ce

théorème n'est pas vraie: l'existence de convergence n'implique pas l'existence de pannes indetectables. L'exemple de la Fig. 2-9, comme expliqué précédemment, en est la preuve.

3 PROCEDURE REPTIL

Les mesures de testabilité définies précédemment (controlabilité et observabilité) peuvent être affinées de la manière suivante: au lieu de calculer pour chaque noeud du circuit un triplet d'entiers représentant une mesure abstraite et approximative de la difficulté de controler ou observer chaque noeud, on calculera un triplet de vecteurs d'entrée qui permettent de controler ou observer chaque noeud. En fait, pour chaque noeud du circuit, il existe de nombreux vecteurs d'entrée qui permettent le contrôle ou l'observation de ce noeud et il conviendra de définir des règles heuristiques pour selectionner un sous-ensemble de ces vecteurs de dimensions acceptables. L'indépendance des signaux n'est plus supposée, comme elle l'est dans SCOAP, puisque la notion de dépendance est conservée dans les calculs de vecteurs. L'analyse du circuit passe du niveau local (SCOAP) à un niveau global, permettant ainsi une analyse correcte de la testabilité. Pour chacune des pannes du circuit, au lieu de calculer une valeur de testabilité, difficile à interpréter, un vecteur de test est calculé. Les pannes du circuit sont alors classées en deux groupes: celles pour lesquelles un vecteur de test a été généré et celles pour lesquelles les règles heuristiques n'ont pas permis la génération d'un vecteur de test, soit qu'elles soient indétectables, soit qu'elles soient "difficiles à tester". Le but de la méthode exposée ici est de limiter le groupe des "pannes difficiles", ou fausses redondances, tout en conservant

la vitesse d'exécution qui permette la génération automatique de vecteurs de tests en un temps qui soit une fonction polynomiale de la complexité du circuit combinatoire considéré.

3.1 Définitions:

- Pour un circuit combinatoire à n entrées primaires, un vecteur d'entrée est un n-uplet représentant les niveaux logiques qui doivent être appliqués aux entrées primaires pendant un pas élémentaire de la procédure de test.

- Dans un circuit combinatoire à n entrées primaires, on definit pour chaque noeud 3 ensembles de ces vecteurs d'entrée:

  * les vecteurs de remise à 1, appelés vecteurs **"set"**, qui représentent des conditions suffisantes à appliquer sur les entrées primaires pour forcer le noeud à 1.

  * les vecteurs de remise à 0, appelés vecteurs **"reset"**, qui représentent des conditions suffisantes à appliquer sur les entrées primaires pour forcer le noeud à 0.

  * les vecteurs de contrôle, appelés vecteurs "moniteur" , qui représentent des conditions suffisantes à appliquer sur les entrées primaires pour créer un chemin de sensibilisation et observer la valeur du noeud sur une sortie primaire (ou une pseudo-sortie dans la cas d'un circuit LSSD).

- Chaque bit des vecteurs de contrôle ou d'observation est l'un des 4 éléments suivants:

  . 0: niveau logique 0

  . 1: niveau logique 1

  . x: indéfini (n'importe quelle condition)

  . #: conflit (conditions incompatibles ou impossibles sur les EPs)

  Pour un circuit CMOS, on utilise un élément supplémentaire:

  . Z: haute impédance (aucun niveau logique n'est forcé)

La figure 2-10 illustre les vecteurs "set", "reset" et "moniteur" pour chaque noeud d'un circuit simple. Chacun de ces vecteurs représente une condition suffisante pour remettre à 1, remettre à 0, ou observer (sur la sortie primaire) le noeud correspondant.

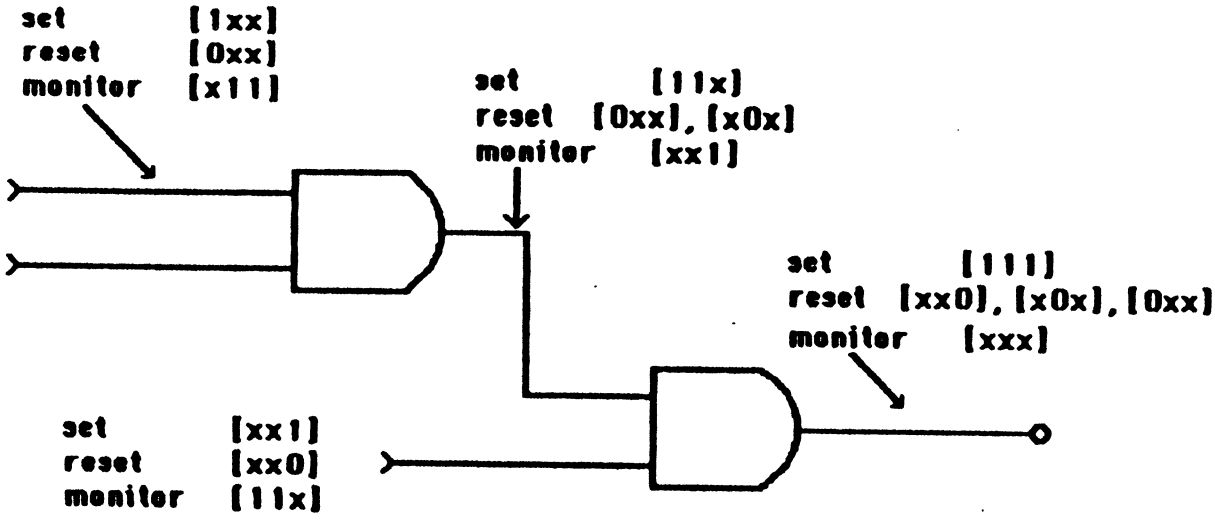


Figure 2-10: Vecteurs set, reset et moniteur pour un circuit simple

- Un vecteur de test pour une panne de type collage sur un noeud N est défini comme étant la combinaison d'un vecteur de contrôle et d'un vecteur d'observation pour ce noeud N. Par exemple, un vecteur de test pour une panne de type collage à 0 sera obtenu par l'application simultanée de la condition "remettre ce noeud à 1" (provocation de la panne) et de la condition "sensibiliser un chemin entre ce noeud et la sortie" (observation de la panne). C'est à dire appliquer simultanément des vecteurs set et moniteur. Si ces conditions ne sont pas compatibles, une condition de conflit est générée pour le vecteur résultant, et une autre

paire de vecteurs contrôle-observation est alors considérée.

Si au moins deux de ces conditions sont compatibles, un vecteur de test est généré. L'opération qui permet de combiner les deux vecteurs contrôle et moniteur est appelée "intersection", et est définie par le tableau 2-4a pour le cas TTL et par le tableau 2-4b pour le cas CMOS.

Tableau 2-4a: Operation Intersection (TTL)

	0	1	x	#
0	0	#	0	#
1	#	1	1	#
x	0	1	x	#
#	#	#	#	#

Tableau 2-4b: Operation Intersection (CMOS)

	0	1	Z	x	#
0	0	#	#	0	#
1	#	1	#	1	#
Z	#	#	Z	Z	#
x	0	1	Z	x	#
#	#	#	#	#	#

Le tableau 2-5 illustre la génération d'un vecteur de test à partir de vecteurs de contrôle (set et reset) et de vecteurs moniteur, pour un circuit combinatoire à 5 entrées. (tous les vecteurs de contrôle, moniteur ou tests sont donc des vecteurs de 5 bits).

Tableau 2-5: Génération des vecteurs de test avec les vecteurs de contrôle et observation du noeud N

Type de panne (au noeud N)	Vecteur Set	Vecteur Reset	Vecteur Moniteur	Vecteur Test	Problème de Testabilité
c.a.0	[00xx1]	-	[xxx01]	[00x01]	Non
c.a.1	-	[11xx1]	[xxx01]	[11x01]	Non
c.a.1	-	[11xx0]	[xxx01]	[11x0#]	Potentiel
c.a.1	-	[xxxZ0]	[xxx01]	[xxx##]	Potentiel

3.2 Génération Des Vecteurs

L'algorithme qui calcule les vecteurs de contrôle et d'observation pour chaque noeud du circuit est très similaire à l'algorithme de calcul des valeurs de contrôlabilité et d'observabilité de SCOAP décrit dans le chapitre précédent. Les bases du présent algorithme ont été introduites dans [Ratiu 82]. La procédure présentée ici a été améliorée pour supporter les différents niveaux des circuits CMOS et pour permettre une génération efficace de vecteurs de tests (plutôt que de se borner à une simple analyse de testabilité).

L'algorithme de base de la procédure REPTIL peut être décrit comme suit:

- Premièrement le circuit est structuré par couches logiques. La couche logique d'un noeud (ou porte logique, d'après notre convention) est définie itérativement comme étant:
  - Zéro si le noeud est une entrée primaire (ou pseudo-entrée)
  - La valeur maximum de la couche logique des entrées augmentée de un dans le cas d'une porte logique booléenne.
  - La valeur maximum de la couche logique de ses connections dans le cas d'une porte cablée (cas du CMOS). En CMOS, les notions de portes et de couches logiques sont parfois ambigues lorsqu'on emploie des portes de transert bi-directionnelles et de la logique cablée. Pour lever cette ambiguité, nous définirons comme une porte complexe unique (étant donc sur une seule couche logique) la réunion de toutes les portes de transfert et de portes cablées, qui sont (même si cela dépend d'un signal de contrôle) reliées directement entre elles.

Tableau 2-6: Regles de calcul pour les vecteurs set/reset pour les portes logiques AND2, NAND2, OR2, NOR2, inverseur (INV) et tampon direct (BUF).

& indique une intersection
OUT est le noeud de sortie de la porte
IN, IN1 et IN2 sont les entrees

AND2	set (OUT) = set (IN1) & set (IN2) reset (OUT) = reset (IN1) ou reset (IN2)
NAND2	reset (OUT) = set (IN1) & set (IN2) set (OUT) = reset (IN1) ou reset (IN2)
OR2	set (OUT) = set (IN1) ou set (IN2) reset (OUT) = reset (IN1) & reset (IN2)
NOR2	reset (OUT) = set (IN1) ou set (IN2) set (OUT) = reset (IN1) & reset (IN2)
INV	set (OUT) = reset (IN) reset (OUT) = set (IN)
BUF	reset (OUT) = reset (IN) set (OUT) = set (IN)

o Puis, calculer les vecteurs set et reset pour chaque noeud.

Les vecteurs de contrôle pour chaque entrée primaire sont tout d'abord initialisés: les vecteurs de contrôle pour l'entrée j a tous ses bits à x (indéfini) sauf le bit j qui est à 1 pour le vecteur set et à 0 pour le vecteur reset. Puis, en commencant à la couche logique 1 et en allant par couche logique croissante, toutes les portes d'une même couche logique sont examinées et des conditions suffisantes de contrôle sont déterminées en fonction des conditions déjà déterminées pour leurs entrées.

Les règles de calcul des vecteurs de contrôle pour les portes logiques booléennes sont données dans le tableau 2-6. Ces règles sont très similaires aux règles de calcul des CC0 et CCi de SCOAP données au chapitre précédent. A noter que pour les portes spécifiques au CMOS un certain nombre de règles spéciales ont été établies. Pour les portes cablées, le contrôle de la sortie est obtenu en contrôlant toutes les entrées à haute impédance, sauf une, qui est controlée à la valeur qui doit être reproduite sur la sortie (voir fig. 2-11). Si toutes les entrées sont identiques, le contrôle de la sortie est obtenu par n'importe quel vecteur de contrôle d'un noeud d'entrée (voir fig. 2-12). Enfin, dans le cas de VDD ou VSS, qui sont des entrées primaires particulières, un symbole conflit marque l'impossibilité de contrôler VDD à 0 ou VSS à 1.

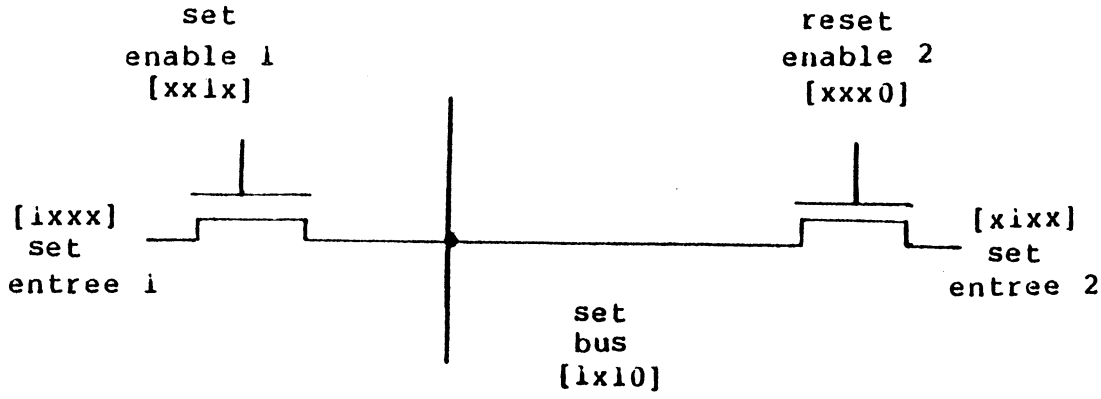


Figure 2-11: Vecteurs set pour un bus en CMOS

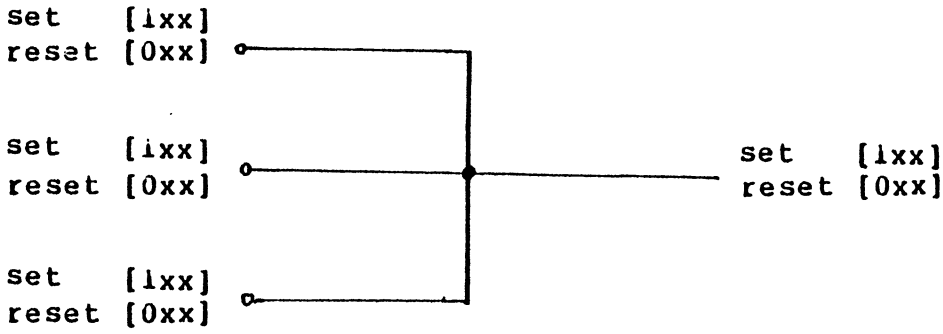


Figure 2-12: Vecteurs set/reset pour un cablage de trois portes CMOS identiques

o Les vecteurs moniteurs sont alors calculés. En partant des sorties primaires, et en remontant par couche logique décroissante jusqu'aux entrées primaires, ces vecteurs d'observation sont calculés d'une manière similaire au calcul des valeurs CO de SCOAP, présenté au chapitre précédent. Les règles de calcul pour les portes logiques booléennes sont données dans le tableau 2-7. Le cas de la logique cablée est illsutré par la figure 2-13.

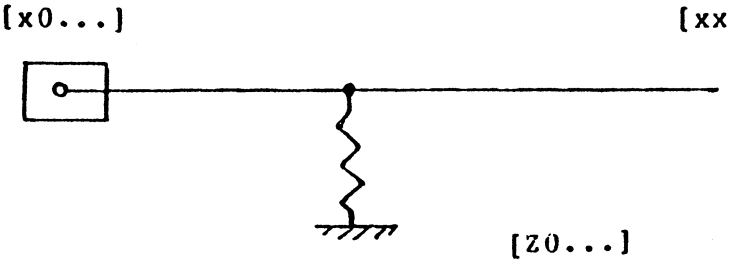


Figure 2-13: vecteur moniteur pour un tampon trois-états avec résistance de tirage à zero

Tableau 2-7: Calcul de vecteurs moniteurs pour les portes logiques AND2, NAND2, OR2, NOR2, inverseur (INV) et tampon direct (BUF).

AND2	moniteur (IN1) = set (IN2) & moniteur (OUT)
NAND2	moniteur (IN1) = set (IN2) & moniteur (OUT)
OR2	moniteur (IN1) = reset (IN2) & moniteur (OUT)
NOR2	moniteur (IN1) = reset (IN2) & moniteur (OUT)
INV	moniteur (IN) = moniteur (OUT)
BUF	moniteur (IN) = moniteur (OUT)

- Enfin, des vecteurs de test sont calculés pour chaque panne de collage du circuit, et un problème de testabilité est identifié lorsque cette procédure échoue pour une panne donnée. Le calcul d'un vecteur de test se réduit à une opération intersection entre deux représentants des vecteurs de contrôle et d'observation pour un noeud. Pour une panne

de collage à 0: intersection entre un vecteur set et un vecteur moniteur. Pour une panne de collage à 1: intersection entre un vecteur reset et un vecteur moniteur. Un problème de testabilité est identifié lorsque toutes les combinaisons possibles entre tous les vecteurs de contrôle et tous les vecteurs d'observation ont donné des resultats incompatibles (conditions de conflit): la panne est alors indétectable. Lorsque seulement un sous-ensemble de l'ensemble des nombreux vecteurs possibles de contrôle et d'observation est gardé pour chaque noeud du circuit, l'existence d'un conflit entre chacun des vecteurs de test n'indique plus forcément une panne indétectable. Toutes les possibilités de vecteurs de test n'ayant pas été examinées, le conflit indique seulement un un problème potentiel. On identifie ainsi une panne difficile à détecter, de la même manière que SCOAP identifiait une panne difficile à détecter par une valeur de testabilité élevée.

3.3 Procédure REPTIL

Le nombre de vecteurs de contrôle et d'observation possibles croit presque exponentiellement avec le nombre d'entrées primaires et de points de sortance dans le circuit. Garder tous les vecteurs possibles pour chaque noeud du circuit resulterait en des tailles de mémoire et des temps de calcul rapidement inacceptables. Pour illustration, nous avons calculé le nombre total de vecteurs set, reset et moniteur pour chaque

noeud d'un circuit de taille moyenne: l'UAL/générateur 74i8i.

Pour ce circuit, dont la profondeur logique ne dépasse pas 6 niveaux, le nombre de vecteurs pour chaque noeud atteint déja plusieurs centaines pour les vecteurs de contrôle (set et reset) et plusieurs dizaines pour les vecteurs moniteur des noeuds des dernières couches logiques. A noter qu'il s'agit là de vecteurs tous différents et compactés: les vecteurs qui ne diffèrent que d'un bit sont regroupés deux à deux (par exemple, [i0x1] et [i0x0] sont remplacé par le vecteur [i0xx]). Il en résulte plusieurs milliers de combinaisons à examiner pour les vecteurs de test de certains noeuds (rappelons que 16384 vecteurs sont possibles pour ce circuit). Les portes de type OU-EXCLUSIF ont un effet particulièrement aggravant pour la multiplication des vecteurs: pour une porte OU-EXCLUSIF à 2 entrées, dont chaque entrée a 10 vecteurs set et 10 vecteurs reset, la sortie a jusqu'à 200 vecteurs set et 200 vecteurs reset (10x10 + 10x10 combinaisons pour chaque).

Pour éviter une explosion exponentielle du temps de génération avec la complexité du circuit, seulement un petit nombre k de vecteurs de chaque ensemble, set, reset et moniteur, doit être gardé pour chaque noeud du circuit. Lorsque les vecteurs de contrôle et d'observation de chaque noeud sont générés, il est donc nécessaire d'effectuer une sélection pour garder un maximum de k vecteurs par noeud.

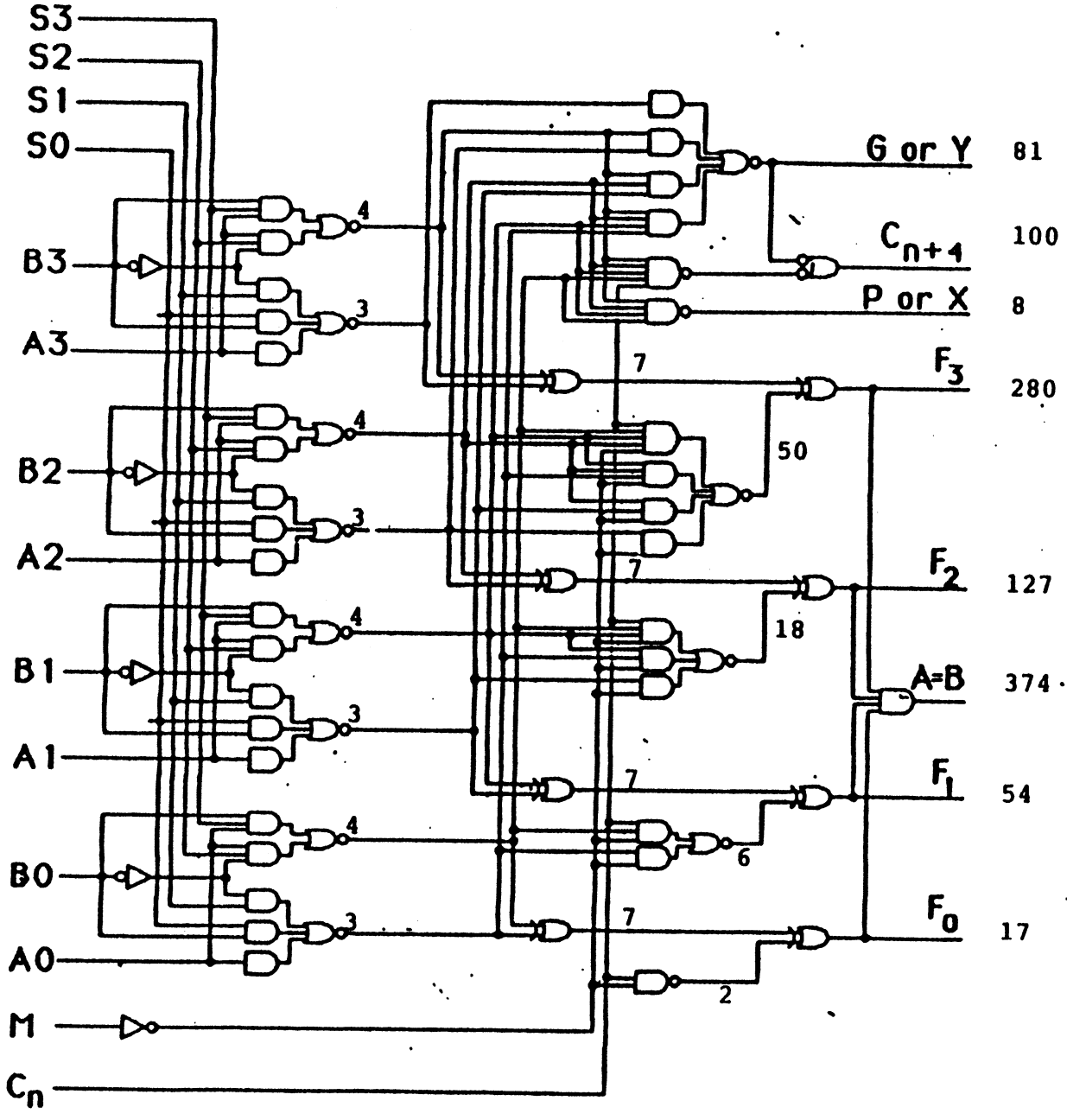


Figure 2-14: Nombre de vecteurs set pour l'UAL 74181

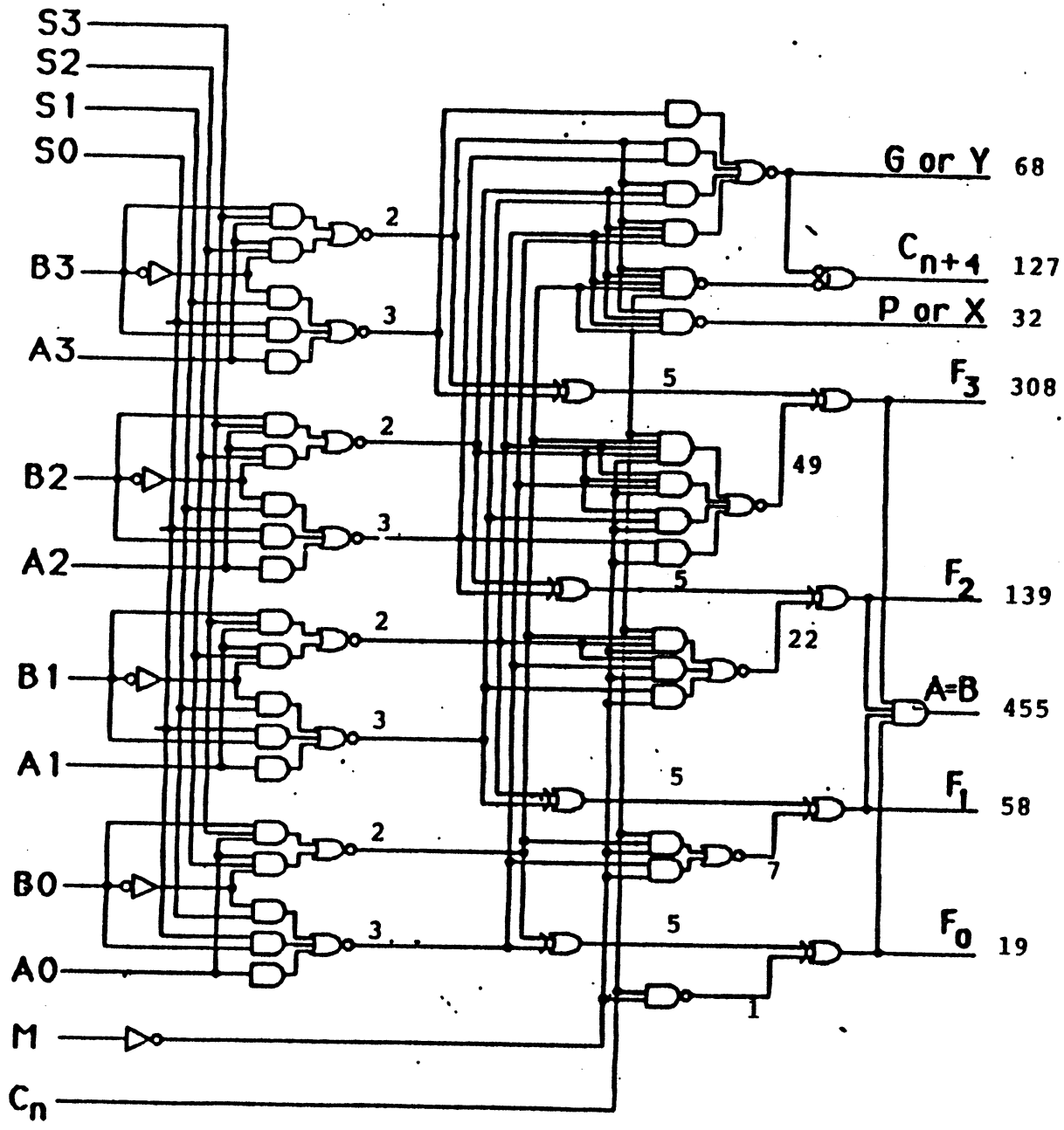


Figure 2-15: Nombre de vecteurs reset pour l'UAL 74181

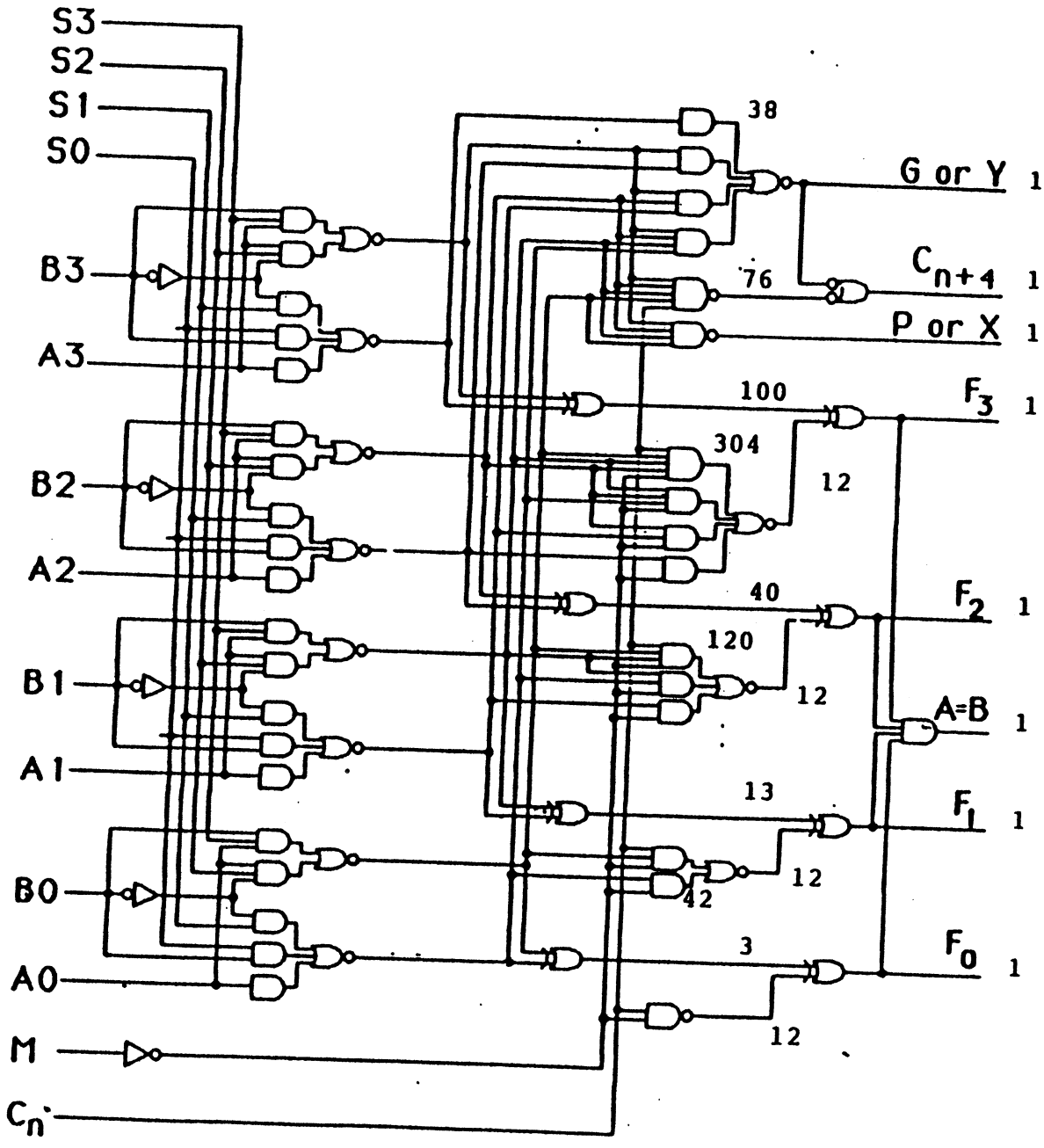


Figure 2-16: Nombre de vecteurs moniteur pour l'UAL 74181

L'opération **sélection** choisit k vecteurs parmi m , et est définie comme suit:

- Trier les m vecteurs suivant une règle heuristique préétablie, puis
- Selectionner les m premiers vecteurs de la liste triée.

Cette opération sélection est l'élement le plus important de la procédure Reptil, puisqu' une "heureuse" sélection des vecteurs peut résulter en une analyse exacte de la testabilité du circuit, alors qu'une mauvaise sélection des vecteurs peut résulter en une faible proportion de pannes détectables détectées. A noter que la procédure est sûre, en ce sens que, dans le pire cas, le circuit est déclaré intestable alors qu'il est testable. Mais un problème de testabilité réel n'échappe pas au programme.

Différentes règles heuristiques peuvent être choisies pour l'opération sélection. La (ou les) règles choisies devront avant tout résoudre au mieux le problème suivant: minimiser le nombre de faux problème de testabilité identifiés tout en respectant l'avantage de rapidité de la procédure globale. La génération de faux problèmes de testabilité provient uniquement de la perte d'information durant l'opération de sélection. L'exemple de la Figure 2-17 illustre ce problème sur un circuit à 2 entrées. Supposons que la règle de sélection soit purement aléatoire, et choisisse un seul vecteur, le premier généré qui n'ait pas de condition de conflit. Lorsqu'un vecteur de

contrôle à 0 (reset) est généré pour le noeud N, l'intersection des vecteurs set de C [x0] et set de D [xi] résulte en un vecteur [x#] contenant un conflit. La noeud N est donc identifiée comme potentiellement incontrollable à 0, alors que le vecteur [0i] contrôle N à 0.

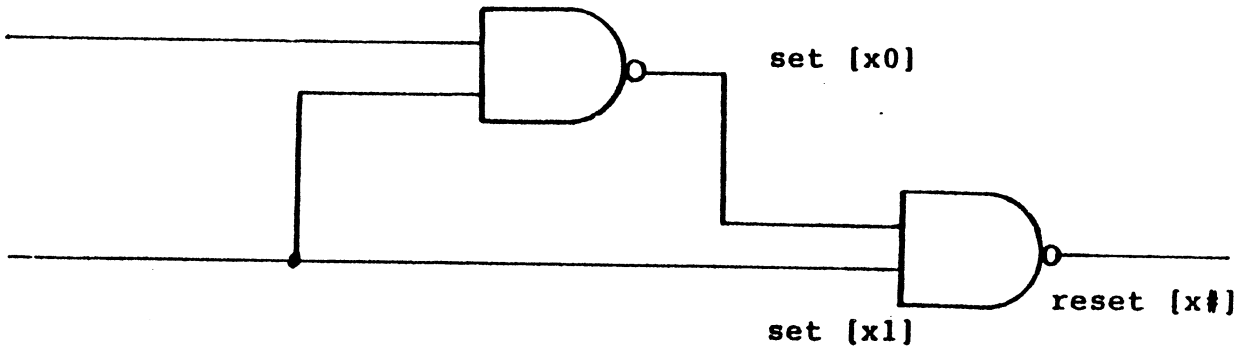


Figure 2-17: La sélection d'un seul vecteur par noeud ne permet pas de garantir de trouver le vecteur reset [0i] pour le noeud N.

- Une première solution pour selectionner les vecteurs consiste à les trier suivant leur nombre de bits non définis (bits à x). Cette méthode repose sur le fait que plus un vecteur a de bits à x, plus la probabilité de création de conflit (#) lors de l'intersection avec un autre vecteur selectionné de manière aléatoire est faible. Pour prouver ce résultat, il suffit de compter le nombre d'occurence de conflits dans chaque colonne du tableau 2-4. La colonne correspondant à x est la colonne contenant le moins de conflits: en fait si les vecteurs opérandes ne contiennent pas de conflit, un vecteur ne contenant que des x ne peut générer de #. Cette règle heuristique de selection a l'avantage d'être facile à implementer dans un programme et d'avoir un faible coût de calcul. Par contre cette méthode, de par sa simplicité, ne prend absolument pas en compte la structure du circuit analysé: la selection de vecteurs pour la première couche logique sé fait indépendamment de la structure du circuit pour les couches suivantes, et en particulier indépendamment de la probabilité pour le chemin selectionné de traverser des circuits convergents.

- Une technique de selection plus puissante, mais plus difficile à implémenter, consiste à évaluer, pour chaque vecteur, le risque que le chemin sensible qu'il crée soit convergent. Cette technique est donc une tentative heuristique d'identification des redondances, ou tout au moins des pannes difficiles à tester.

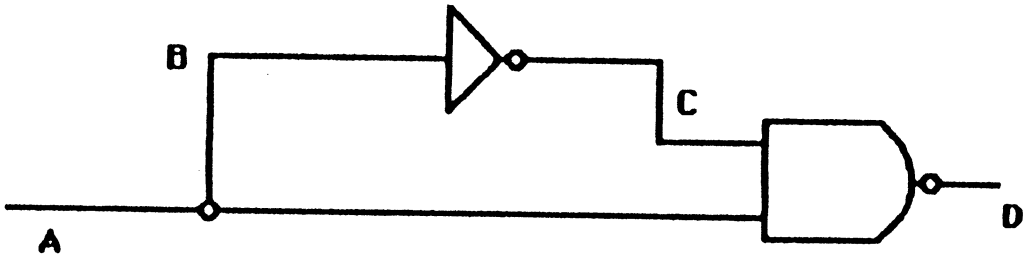


Figure 2-18: Exemple de panne indetectable.
B/0 est indetectable.

Le théorème 2-2 du chapitre précédent a montré que l'existence de chemins convergents entraine l'existence de conflits lors du calcul de vecteurs de tests pour les pannes indetectables, mais aussi pour certaines pannes détectables. Les pannes pour lesquelles un conflit apparait durant la génération des vecteurs de tests sont appelées pannes potentiellement indetectables. Rappelons que, puisque seulement un sous-ensemble de l'ensemble des vecteurs de tests possibles a été examiné, cet ensemble de pannes potentiellement indetectables contient de manière indiscernable, l'ensemble des pannes indétectables du circuit (voir Figure 2-i8) plus un certain nombre (qui peut être zero) de pannes detectables, difficiles à tester, que nous appelons "fausses redondances".

Si les vecteurs correspondants au risque de convergence le plus faible sont sélectionnés, d'après le théorème 2-2, le risque de conflit lors de la génération est aussi

minimisé. Ainsi une sélection basée sur le plus faible risque de convergence diminue le nombre de fausses redondances durant la génération et augmente le nombre de vecteurs générés, donc le taux de couverture obtenu pour la séquence finale. En appliquant cette règle successivement à chaque cellule du circuit pour finalement produire un vecteur de test sans conflit (sans #) pour chaque panne, cette optimisation locale permet de s'approcher de l'optimum global, une technique classique en CAO.

Cette méthode est basée sur la mesure de risque. L'implémentation de l'estimation de cette mesure pour chaque vecteur du circuit est un problème non trivial qu'il convient d'approximer pour garder la methode globale avec un temps de calcul polynomial (presque lineaire) en fonction de la taille du circuit.

Chacune des deux techniques exposées repose sur une théorie probabiliste simplifiée de l'occurence de conflits lors de la génération de tests par la méthode exposée au chapitre précédent. Aucune ne permet donc de prédire un resultat optimal en terme de taux de couverture de la séquence produite: une procédure avec un temps de calcul polynomial ne peut prétendre résoudre un problème NP-complet. Néanmoins, la rapidité d'exécution de la procédure nous permet d'envisager l'implémentation des deux techniques de manière complémentaire et ainsi d'optimiser au maximum les résultats de la procédure REPTIL.

4 RESULTATS PRATIQUES

4.1 Implémentation De La Procédure Reptil

Nous avons implémenté la procédure Reptil en langage C sur un mini-ordinateur VAX/750 et sur un micro-ordinateur IBM PC-AT.

La procédure décrite précédemment a été implémentée avec la possibilité de choisir à chaque éxécution du programme les paramètres suivants:

- Choix du paramètre k, le maximum de vecteurs de chaque type (set, reset et moniteur) qui sont conservés pour chaque noeud. Une valeur de k élevée peut augmenter la résolution du programme, mais ralentir son éxécution.

- Choix entre deux règles différentes de sélection, qui peuvent être successivement utilisées pour générer les vecteurs de contrôle et d'observation:

  1. sélectionner les vecteurs correspondant à un risque minimum dans un premier passage du programme, puis

  2. si nécessaire, générer un nouvel ensemble de vecteurs en utilisant la règle de selection "nombre maximum de x" par vecteur, et comparer les deux résultats.

Les règles de calcul pour la sélection des vecteurs sont encodées dans le programme. Le choix du paramètre k est effectué par le concepteur.

4.1.1 Calcul Du Risque De Reconvergence -

Comme remarqué précédemment, le calcul d'une valeur exacte pour le risque de reconvergence des chemins établis par un vecteur donné est une tâche fastidieuse. Nous avons donc défini une procédure heuristique d'évaluation de ce risque.

Dans ce chapitre, nous appellons "risque" l'évaluation (non nécessairement exacte) du risque de reconvergence d'un vecteur. Ce risque est défini pour chaque vecteur associé à chaque noeud du circuit. Les règles de calcul sont les suivantes:

- Le risque des vecteurs de contrôle (set et reset) des entrées primaires sont initialisées au produit du nombre de chemins pour lesquels ce signal diverge par la profondeur maximum du circuit (en nombre de couches logiques). Cette initialisation est illustrée par le schéma de la Figure 2-19.

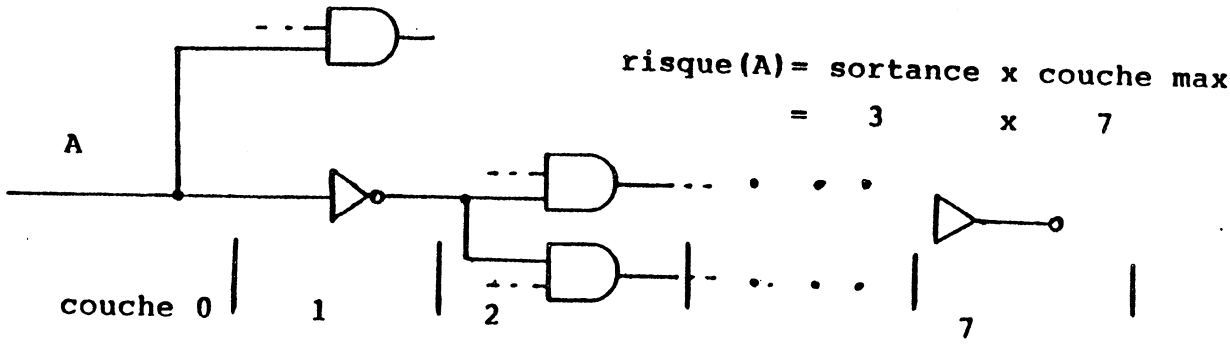


Figure 2-19: Initialisation de la mesure de risque

- Lorsqu'un vecteur de contrôle est calculé, son risque est calculé simultanément. Un vecteur résultant d'une intersection entre deux vecteurs a un risque égal à la somme des risques des deux vecteurs opérandes.

- De manière analogue, les vecteurs moniteurs obtenus par l'opération intersection ont un risque égal à la somme des risques des vecteurs intersectés.

Il est intéressant de noter que les analyses obtenues en employant la sélection par le "risque" sont souvent différents de ceux obtenus par la sélection par les "x", tout au moins pour k petit. En effet, les "fausses redondances" identifiées par une analyse sont différentes des fausses redondances obtenues par l'autre, et la combinaison des deux séquences de tests permet parfois de les éliminer complètement. Si la somme des temps CPU nécessaire pour obtenir ces deux séquences pour un k petit est inférieure au temps CPU d'une seule execution pour un k grand (qui génère un test pour toute les pannes), utiliser la combinaison des 2 règles de sélection peut-être une technique intéressante.

Enfin, les résultats comparés du tableau 2-8 suggèrent que la sélection par le risque converge plus vite (avec k croissant) vers la solution exacte du problème de la génération de tests pour chaque panne, que la selection par les x. Ce qui justifie l'existence d'une méthode de sélection (calcul du risque) plus élaborée que le seul décompte des x dans chaque vecteur.

4.1.2 Choix Du Paramètre K -

Une remarque importante concerne le paramètre k, le nombre de vecteurs représentant chaque type (set, reset et moniteurs) pour chaque noeud. K doit être assez petit pour permettre au programme de tenir dans un espace mémoire raisonnable (la taille du programme est proportionelle à 3k) et en un temps raisonnable (le programme doit effectuer des recherches de listes de dimension k^2), mais suffisamment grand pour permettre un choix de vecteurs important lors des opérations intersection (le choix est de 1 parmi 1 si k=1, 2 parmi 4 si k=2, 3 parmi 9 si k=3 ...) afin de réduire le nombre de fausses redondances. Il semble que la valeur optimale de k varie avec la taille du circuit, plus exactement avec la dimension des sous-circuits entre deux points de reconvergence. Pour la taille actuelle des circuits personnalisés Matra Design Systems, soit des réseaux pré-diffusés de 400 à 1500 cellules élémentaires, les résultats du tableau 2-8, qui sont représentatifs des "blocs" de logique vus par Reptil au moment de la génération de tests, suggèrent une valeur de k égale à 5 ou 7 pour optimiser le rapport temps CPU et résolution du programme.

Tableau 2-8 : Influence de k et du choix de la technique de selection sur les performances du programme Reptil.

Circuit 1: 192 pannes

Selection: Risque				Selection: Nombre de x			
k	CPU IBM-AT	Potentiel Indetect.	Couvert. (Simul.)	k	CPU IBM-AT	Potentiel Indetect.	Couvert. (Simul.)
1	18"	94	78%	1	18"	94	78%
2	30"	57	92%	2	30"	54	89%
3	1'04"	36	95%	3	1'02"	40	92%
4	1'41"	20	98%	4	1'34"	31	93%
5	2'09"	18	99,5%	5	2'01"	30	93%
6	2'41"	16	99.5%	6	2'32"	30	93%
7	3'31"	13	100%	7	3'20"	28	93%
8	4'37"	7	100%	8	4'06"	28	93%
9	5'09"	7	100%	9	4'42"	27	93%
10	6'28"	0	100%	10	6'07"	24	94%
				11	6'38"	23	94%
				12	8'04"	20	95%
				13	9'11"	9	100%

Circuit 2: 68 pannes

Selection: Risque				Selection: Nombre de x			
k	CPU IBM-AT	Potentiel Indetect.	Couvert. (Simul.)	k	CPU IBM-AT	Potentiel Indetect.	Couvert. (Simul.)
1	7"	0	100%	1	7"	0	100%

Circuit 3: 140 pannes
(dont: 4 indetectables)

Selection: Risque				Selection: Nombre de x			
k	CPU IBM-AT	Potentiel Indetect.	Couvert. (Simul.)	k	CPU IBM-AT	Potentiel Indetect.	Couvert. (Simul.)
1	16"	19	90%	1	15"	28	88%
2	18"	19	90%	2	18"	20	88%
3	21"	17	92%	3	20"	18	92%
4	27"	11	96%	4	24"	12	96%
5	31"	10	96%	5	29"	10	97%
6	37"	4	100%	6	33"	10	97%
				7	37"	10	97%
				8	42"	6	99%
				9	46"	5	100%

4.1.3 Compactage Des Vecteurs De Contrôle -

Afin de minimiser l'espace mémoire nécessaire pour garder les vecteurs set, reset et moniteur, les vecteurs de contrôle de chaque noeud sont compactés au maximum. En particulier si deux vecteurs représentent des conditions complémentaires sur un bit et équivalentes par ailleurs, (par exemple [0x01] et [0x00] pour le bit 4), ils sont compactés en seul vecteur, en utilisant un bit à x ([0x0x] dans notre exemple).

4.1.4 Compactage Des Vecteurs De Test -

A la fin de la génération des vecteurs de test, une procédure de tri rapide (en n log n) est utilisée pour comparer les vecteurs et éliminer les occurences multiples. Puis les vecteurs équivalents, c'est à dire ceux dont toutes les colonnes sont compatibles grâce aux x (valeurs indifférentes), sont regroupés entre eux. Le Tableau 2-9 montre le resultat du regroupement des vecteurs compatibles.

Tableau 2-9: Regroupement des vecteurs compatibles

$$\left.\begin{array}{l}[\,0\;1\;x\;x\;1\,]\\ [\,0\;x\;1\;x\;1\,]\\ [\,x\;x\;1\;Z\;x\,]\end{array}\right\} \;=\; [\,0\;1\;1\;Z\;1\,]$$

Enfin si la séquence compactée contient encore des valeurs indifférentes, celles-ci sont remplacées par des 0,1 ou Z. Ce remplacement est fait en suivant les règles générales d'optimisation du taux de couverture des tests pseudo-aléatoires [Savir 83]. Nous employons ici une procédure simplifiée qui consiste à évaluer, pour chaque entrée primaire, le nombre de portes OR/NOR ,AND/NAND ou trois-états qui sont directement attaquées par ces entrées. Si une entrée est principalement reliée à des portes OR, l'occurence d'un 0 à la sortie de cette porte étant de probabilité la plus faible, un 0 remplacera le x dans le vecteur. L'assignement des 0 et Z est faite de manière analogue avec les portes AND et trois-états.

4.2 Description Du Programme Reptil

La description détaillée du programme C implémentant la procédure REPTIL est donnée en Annexe 2.

4.3 Résultats Et Performances.

Le programme Reptil a été utilisé pour générer automatiquement des vecteurs de test pour un certain nombre de circuits ou de parties de circuits concus par Matra Design Systems. Nous avons choisi de donner le détail de l'analyse de trois de ces circuits.

Le premier exemple est un circuit client qui a été obtenu en combinant sur un même réseau prédiffusé plusieurs PALs (circuits programmables de Monolithic Memories) dans un but de plus grande intégration. A noter que la transformation a été faite automatiquement à l'aide d'un logiciel de synthèse automatique des PALs en réseaux prédiffusés [Hild 84]. Le circuit, avec 12 entrées primaires et 6 sorties primaires, est constitué de 77 cellules pré-caractérisées CMOS MDS (plus 25 tampons d'E/S) et comprend environ 120 cellules élémentaires comprenant chacune 2 paires de transistors MOS. Le circuit ne contient pas de redondances et toutes les pannes (256) sont

testables. Le programme réussit à générer, pour n'importe quelle valeur choisie du paramètre k (nombre de vecteurs set, reset et moniteurs gardés pour chaque noeud), un test pour chaque panne en 5 s de CPU sur l'IBM AT. La séquence de test résultante est donnée dans le fichier PAL.sm0:

```
$
$ Liste des sorties primaires: FR FW VO TB MO C
$ Longeur du test: 28.
$PATTERN A76 A4 A3 A2 A1 A0 DX RW B02 REF DS1 DS0
        0  0iiiii0iiiii
      100  0iiiii100iii
      200  0iiiiii0iiii
      300  iiiiii100iii
      400  00iiii001i00
      500  010iii001110
      600  0ii0ii001iii
      700  0iii0i0iiiii
      800  0iiii0011111
      900  01111i010iii
     1000  0iiiii10ii011
     1100  0iii0i00i101
     1200  0iiii0001111
     1300  0iiii1000111
     1400  0iiii1001011
     1500  i0001i0iiii0
     1600  i0001i011111
     1700  i001i1011110
     1800  i010i1011i01
     1900  i001111111i0
     2000  i001110i0110
     2100  i001i10ii010
     2200  i01iiiiiii01
     2300  i011i1011001
     2400  i0iii1001111
     2500  i001i1011101
     2600  i0i0ii010101
     2700  iii0iiiiiii0
$END
```

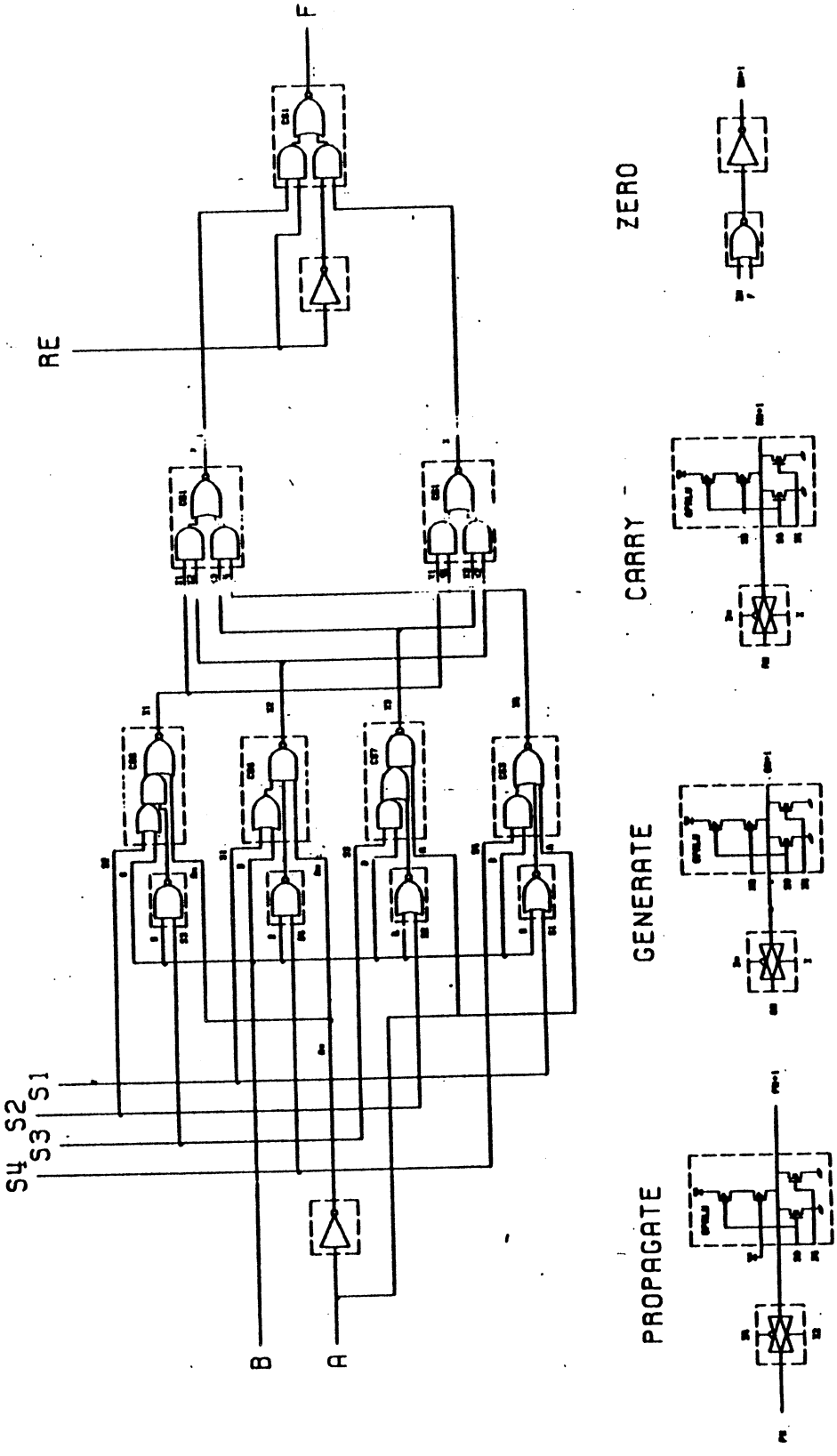


Figure 2-20: 1-bit UAL

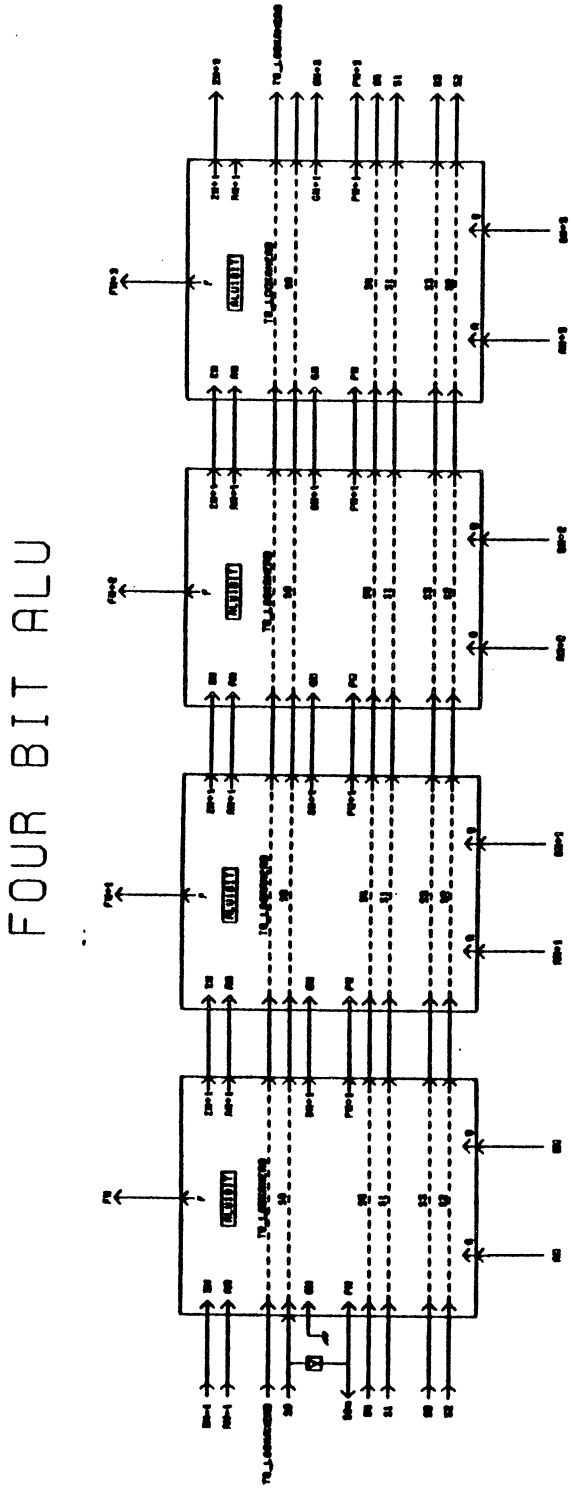


Figure 2-21: 4-bit UAL

Le deuxième exemple est une cellule de la bibliothèque de cellules des circuits à cellules pre-caractérisées de Matra Harris Semiconducteurs. Cette cellule est une UAL en trancne de 1 bit utilisée pour la réalisation d'UALs de taille arbitraire. La Figure 2-20 montre la réalisation de cette UAL et la figure 2-21 son utilisation dans une UAL 4-bit. Un certain nombre de redondances ont été utilisées lors de la conception de cette UAL: elles permettent l'utilisation simultanée des signaux X, X* , X3 et X4 pour les cellules Propagate, Generate et Carry, sans avoir besoin d'une couche logique additionelle. Il en résulte un certain nombre de pannes de collage indetectables. Reptil identifie ces pannes indetectables. En choisissant le paramètre k supérieur ou égal à 2, l'analyse du circuit reste constante et indique 7 pannes potentiellement indetectables (les pannes pour lesquelles aucun test n'a été detecté sont qualifiées de potentiellement indetectables et sont décrites au concepteur dans le fichier ALU1.tst). Le fichier ALU1.tst, qui contient les informations sur l'analyse de testabilité et le déroulement du programme est donné ci-après:

Resume de l'analyse

de pannes detectables : 117
de pannes potentiellement indetectables : 7

Test Patterns Compactes pour les pannes
de collage uniques detectables:

(Le nombre de droite indique le nombre de pannes
differentes detectees par chaque vecteur)

ABSSSSR 001234E	
1111000	13
1011001	12
011xx10	3
11x00x1	1
10x11x0	1
11x00x0	1
001xx10	1
010xx00	2
10x10x0	1
011xx00	1
010xx01	1
10x11x1	1
001xx01	1
001xx11	1
10x00x1	8
11x1111	5
000xx01	4
11x11x0	7
11x10x1	2
000xx00	8
10x00x0	4
011xx11	6
011xx01	12
001xx00	17
1000xx1	2
1000xx0	1
1x11001	1

Total detectees: 117

Total vecteurs
apres compactage: 27

Liste des pannes potentiellement indetectables:

Panne	Type	Vecteur
A0-A-X3	s.a. 1	#000dd0
A-A*-A-X1	s.a. 0	#1dd111
A-X2-CG1*1-1	s.a. 1	#11d1#1
A-X4-CG1*2-1	s.a. 0	#11d100
A-X1-CG1*2-1	s.a. 0	#010d00
A-X3-CG1*1-2	s.a. 1	#11d#01
CG1*2-1-A-X*	s.a. 0	##10100

Total indetectables: 7

Une analyse manuelle de la detectabilité de ces 7 pannes et une simulation de faute du circuit montrent que ces résultats s'avèrent exacts. A noter que ces résultats n'etaient pas connus des concepteurs du circuit avant l'analyse de Reptil. Le programme a donc ainsi prouvé son utilité comme analyseur de testabilité. De plus, une séquence de test a été générée pour cette cellule:

```
$  Liste des sorties primaires: F0
$  Sequence de test: 27 vecteurs
$PATTERN A0 B0 S1 S2 S3 S4 RE
          0           1111000
        100           1011001
        200           0110110
        300           1100011
        400           1001110
        500           1100010
        600           0010110
        700           0100100
        800           1001010
        900           0110100
       1000           0100101
       1100           1001111
       1200           0010101
       1300           0010111
       1400           1000011
       1500           1101111
       1600           0000101
       1700           1101110
       1800           1101011
       1900           0000100
       2000           1000010
       2100           0110111
       2200           0110101
       2300           0010100
       2400           1111001
       2500           1000111
       2600           1000110
$END
```

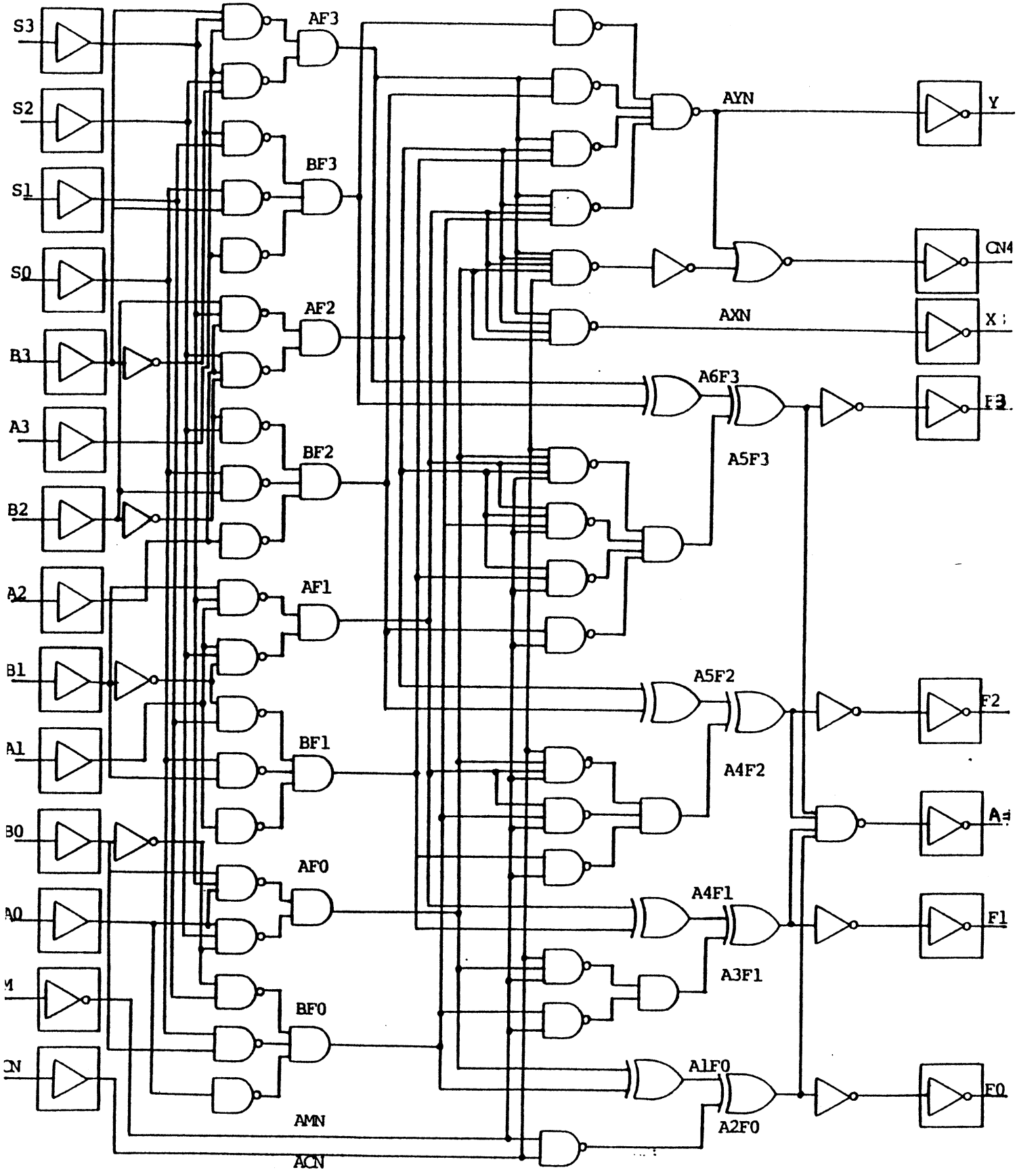


Figure 2-22: UAL 74181 en Gate-Arrays Matra Design Systems

Enfin le troisième exemple envisagé est celui de l'UAL/générateur de fonction 74i81. Cet exemple classique est fréquemment utilisé pour évaluer les procédures de génération de vecteurs de test pour circuits combinatoires. Nous avons utilisé ici une description réseau prediffusé de cette UAL, qui est réalisée avec 78 cellules pré-caractérisées et comprend 114 cellules élémentaires de 4 transistors MOS. La figure 2-22 montre l'implémentation du circuit 7418i. Le programme a considéré 330 pannes de collage uniques, qui ont été ramenées à 192 classes d'équivalence. En choisissant le paramètre k=5, REPTIL génère un test pour 174 pannes, identifiant 18 pannes potentiellement indetectables. Le test obtenu par Reptil comporte 50 vecteurs (il a été rapporté un test optimal de 14 vecteurs, trouvé manuellement, pour ce circuit) et est donné ci-après:

```
$ Liste des sorties  primaires: Y CN4 X F3 F2 F1 F0 AB
$ Sequence de test. Longueur: 50
$PATTERN A0 A1 A2 A3 B0 B1 B2 B3 S0 S1 S2 S3 M CN
       0  0111110000010
     100  0111101001010
     200  0101110001101
     300  0110110001101
     400  0110101001101
     500  1011111000010
     600  1011110101001
     700  1011110100000
     800  1111111011011
     900  1111110111011
    1000  1111101111011
    1100  1111011111011
    1200  1011110101011
    1300  1010111001101
    1400  1101110110001
    1500  1101110110000
    1600  1111011111001
    1700  1111010111010
    1800  0111101101001
    1900  0111001100000
    2000  1111111111011
    2100  1011101110001
    2200  1111111111010
    2300  1111000011010
    2400  1111110100000
    2500  1101010110011
    2600  1101110011011
    2700  1101101001010
    2800  0111011110001
    2900  1110101001011
    3000  1110111101001
    3100  1110110010011
    3200  1110111110101
    3300  1101111101111
    3400  1101111110111
    3500  0101110110000
    3600  1111111010000
    3700  1011101110000
    3800  1110011010000
    3900  0000110110111
    4000  0000111011111
    4100  1100101000000
    4200  1101110111010
    4300  1011111110111
    4400  1000111101111
    4500  1111111111001
    4600  0001011110111
    4700  0010111101111
    4800  0100011111111
    4900  1011101111101
```

```
$ Pannes potentiellement indetectables: 182 189 192 194 196
$ + 198 200 212 214 216 218 220 222 228 230 241 243 275
$ Nombre total: 18
```

Ce test a été utilisé pour simuler le 74181 en utilisant le simulateur de fautes COFIS (développé par E. Archambeau et présentement utilisé à Matra Design Systems). Le taux de couverture s'est avéré être de 99,5%: une seule panne reste indetectée. Il est intéressant de noter que la valeur de testabilité de cette panne est presque maximum (cette valeur de testabilité est la deuxième en valeur décroissante) dans l'analyse faite par SCOAP du circuit: l'analyse de SCOAP,aussi simplifiée qu'elle soit, peut toutefois apporter un certain nombre d'éléments d'information sur la testabilité d'un circuit. Enfin, en prenant comme paramètre k=7, le taux de couverture passe à 100%, mais le temps CPU (sur l'IBM AT) passe de 2 mn à 3 mn 30 s.

Le tableau 2-16 résume les resultats obtenus pour les trois exemples étudiés. Pour chaque circuit, le tableau donne le nombre total de pannes de collage unique (entre parenthèses le nombre de classes d'équivalence après réduction), le nombre de pannes indetectables dans le circuit, le paramètre k utilisé pour l'analyse, le temps CPU de l'analyse sur un IBM-AT, le taux de couverture obtenu avec le simulateur de fautes (simulation des pannes détectables seulement), la longueur du test obtenu par Reptil et la longueur du test donné par le constructeur pour

un taux de couverture de 100%. Ces performances permettent d'espérer des resultats tout à fait acceptables, en vitesse d'execution et taux de couverture obtenus pour les circuits personnalisés des années à venir (circuits de 2500 à 10000 portes logiques, avec, pour le test, des blocs indépendants de complexité allant de 400 à 2000 portes logiques).

Tableau 2-16: Résumé des résultats précédents

Circuit	Pannes Detect.	Pannes Indet.	k	CPU IBM-AT	Taux de Couvert.	Long. Test	Construc Test
PAL	256 (109)	0	1	4"	100 %	28	75
UAL_1B	117	7	2	7"	100 %	27	-
74181	330 (192)	0	5	2'30"	99,5%	50	14

4.4 Conclusions Et Futurs Développements.

Dans cette deuxième partie du mémoire, nous avons décrit une procédure très rapide de génération automatique de vecteurs de tests pour circuits combinatoires ou qui peuvent être testés comme tels. Cette procédure, qui peut être considérée comme une extension des méthodes d'analyse de controlabilité et d'observabilité, génère une séquence de test pour la détection des pannes de type collage unique d'un circuit. Les performances prometteuses obtenues par cette première version du programme Reptil, qui ont été présentées dans le paragraphe précédent, nous ont conduit à envisager de poursuivre son développement et de l'utiliser dans une procédure globale de génération de programmes de test.

Une nouvelle version de Reptil, également écrite en langage C, est en cours d'implémentation. De nouvelles règles heuristiques sont étudiées pour tenter de prendre en compte à la fois la structure locale et la structure globale des circuits. Ce programme sera au coeur d'une procédure de génération automatique de séquences de test pour les circuits personnalisés (circuits à réseaux prédiffusés et circuits à cellules précaractérisées) de la compagnie Matra Design Systems. Cette procédure sera employée en conjonction avec des méthodes structurées de conception pour le test, utilisant des bascules MD (bascules D dont l'entrée D est multiplexée) et basée sur la méthode de conception développée à l'université de Stanford (Stanford scan-path design technique [Williams 73]).

Le temps de génération des vecteurs sera de l'ordre de quelques minutes de temps CPU (sur VAX 785) pour l'ordre de grandeur des applications envisagées (250 à 5000 cellules). Encore une fois, ces performances sont possibles grâce à la procécure heuristique utilisée, qui n'effectue pas de "backtracking" et à la non-utilisation d'un simulateur de fautes.

La description du circuit et l'ordre des bascules dans la chaine sont retrouvés dans une base de donnée préalablement établie. La séquence des bascules MD est determinée automatiquement par la procédure de placement et de routage automatique des cellules et des connexions, qui, le placement étant déjà effectué, minimise la longueur des connexions de la chaine de bascules. Cette procédure a été determinée comme la plus efficace pour limiter le coût supplementaire dû au scan-path [Agrawal 84].

En complément de Reptil, un simulateur logique calcule la réponse combinatoire du circuit pour chaque pas de test et un programme de mise en forme effectue le lien avec le testeur. Ce dernier programme traduira automatiquement la séquence de test générée par Reptil et la réponse simulée du circuit dans le langage du testeur, et effectuera la verification de la compatibilité de ce programme de test avec les contraintes du testeur (par exemple en "masquant" les broches en haute-impédance ou en transition pendant les intervalles d'échantillonnage). Enfin un dernier programme utilise ce test fonctionnel pour générer, de manière interactive avec l'ingénieur de test, un programme complet de test pour le circuit intégré (tests paramétriques et test fonctionnel).

P A R T I E III

T E S T

P S E U D O - E X H A U S T I F

1 PRESENTATION

Le test exhaustif d'un circuit combinatoire comportant n entrées primaires ou pseudo-primaires (les entrées pseudo-primaires sont les points de contrôle de la chaine de test dans un circuit de type LSSD) consiste en l'application au circuit des 2^n vecteurs d'entrées possibles. Le test exhaustif permet d'obtenir un taux de couverture extrèmement élevé. Si l'on considère l'ensemble des pannes détectables, seules les pannes qui augmentent le nombre d'états du circuit ne sont pas nécessairement détectées par le test exhaustif. Parmi ces pannes on trouve en particulier certaines pannes de court-circuit en TTL [Mei 74], [David 75] et certaines pannes de type collage ouvert en MOS.

Le test exhaustif d'un circuit combinatoire détecte toutes les pannes de type collage à zéro et toutes les pannes de type collage à un, aussi bien les pannes de type collage unique que les pannes de collage multiple. L'évaluation du taux de couverture du test par une simulation de faute est donc inutile puisque la performance du test est pré-déterminée. De plus, l'effort de génération d'un test exhaustif est minimal: la génération consiste simplement à produire les 2^n vecteurs possibles. Cette génération peut être réalisée in-situ (à l'intérieur du circuit), en utilisant un LFSR (Linear Feedback Shift Register) modifié de telle manière qu'il passe par tous les états possibles (voir figure 3-1). Sur cette figure, l'addition dans l'arbre des OU-exclusifs d'une fonction NON-OU de tous les étages de 1 à n-1 permet d'obtenir l'état zéro. Cela permet de faire passer par tous les 2^n états un LFSR maximum d'ordre n, normalement limité à $(2^n - 1)$ états.

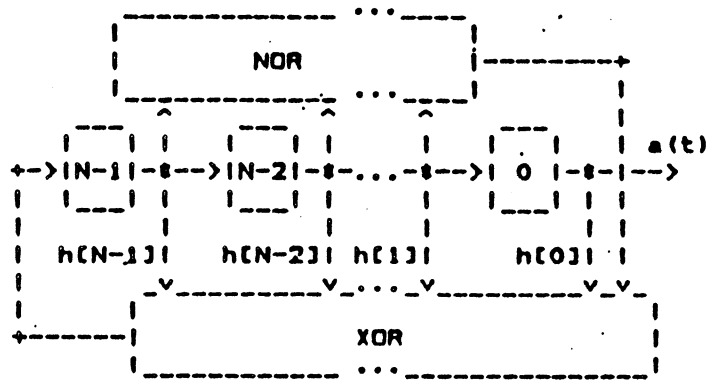


Figure 3-1: LFSR pour le test exhaustif

L'analyse des réponses peut aussi être réalisée in-situ en utilisant un autre LFSR, permettant ainsi un autotest complet [Hassan 83], [Hassan 84].

Si les avantages du test exhaustif sont importants, il présente un inconvénient majeur: la longueur du test exhaustif devient prohibitive lorsque le nombre d'entrées du circuit est élevé. Le test exhaustif devient pratiquement impossible à réaliser dès que le nombre d'entrées primaires et pseudo-primaires est supérieur à 30 ou 40. En effet avec un testeur de fréquence maximum de 10MHz, il faudrait approximativement 1 an pour tester exhaustivement un circuit combinatoire à 48 entrées. La fréquence d'application d'un LFSR n'étant, au mieux que du même ordre de grandeur que celle d'un testeur, le problème reste le même dans le cas de l'autotest.

Conserver la plupart des avantages liés au test exhaustif, tout en réduisant de plusieurs ordres de grandeur la longueur du test, tels sont les objectifs des méthodes de test pseudo-exhaustif présentées dans les deux chapitres suivants.

1.1 Test De Vérification

Pratiquement tous les circuits combinatoires ou de type LSSD ont plus d'une sortie primaire ou pseudo-primaire (ou point d'observation de la chaine de test). Dans la plupart des cas, chacune de ces sorties n'est fonction que d'un nombre réduit de l'ensemble des entrées primaires et pseudo-primaires.

Par exemple, le générateur de parité du circuit TI SN54/74LS630 (extrait du catalogue des produits standards Texas Instrument), représenté Figure 3-2, a 23 entrées primaires et 6 sorties primaires, mais chaque sortie n'est fonction que de 10 des entrées. Le test exhaustif du circuit consistant à appliquer toutes les combinaisons possibles sur les entrées, 2^{23} dans ce cas, est extrèmement lourd d'utilisation pour un circuit de cette taille. Il est beaucoup plus rapide de tester exhaustivement la fonction de chacune des sorties du circuit, en appliquant toutes les combinaisons binaires possibles à chaque sous-ensemble des entrées primaires associés à une sortie donnée.

Dans l'exemple de la Figure 3-2, chaque sortie peut être testée exhaustivement avec 2^{10} =1024 vecteurs. En fait, pour ce circuit, il est possible, par un choix judicieux des vecteurs de test, d'appliquer la séquence exhaustive pour chaque sortie en parallèle plutôt qu'en série. Le nombre total de vecteurs de test est ainsi ramené de (6)x(1024) à 1024. Cette technique pseudo-exhaustive de test est appelée "test de vérification" (ainsi dénommé car le test vérifie chacune des fonctions de sortie exhaustivement). Plusieurs variantes d'obtention des vecteurs des tests de vérification ont été rapportées [Barzilai 81], [McCluskey 82a].

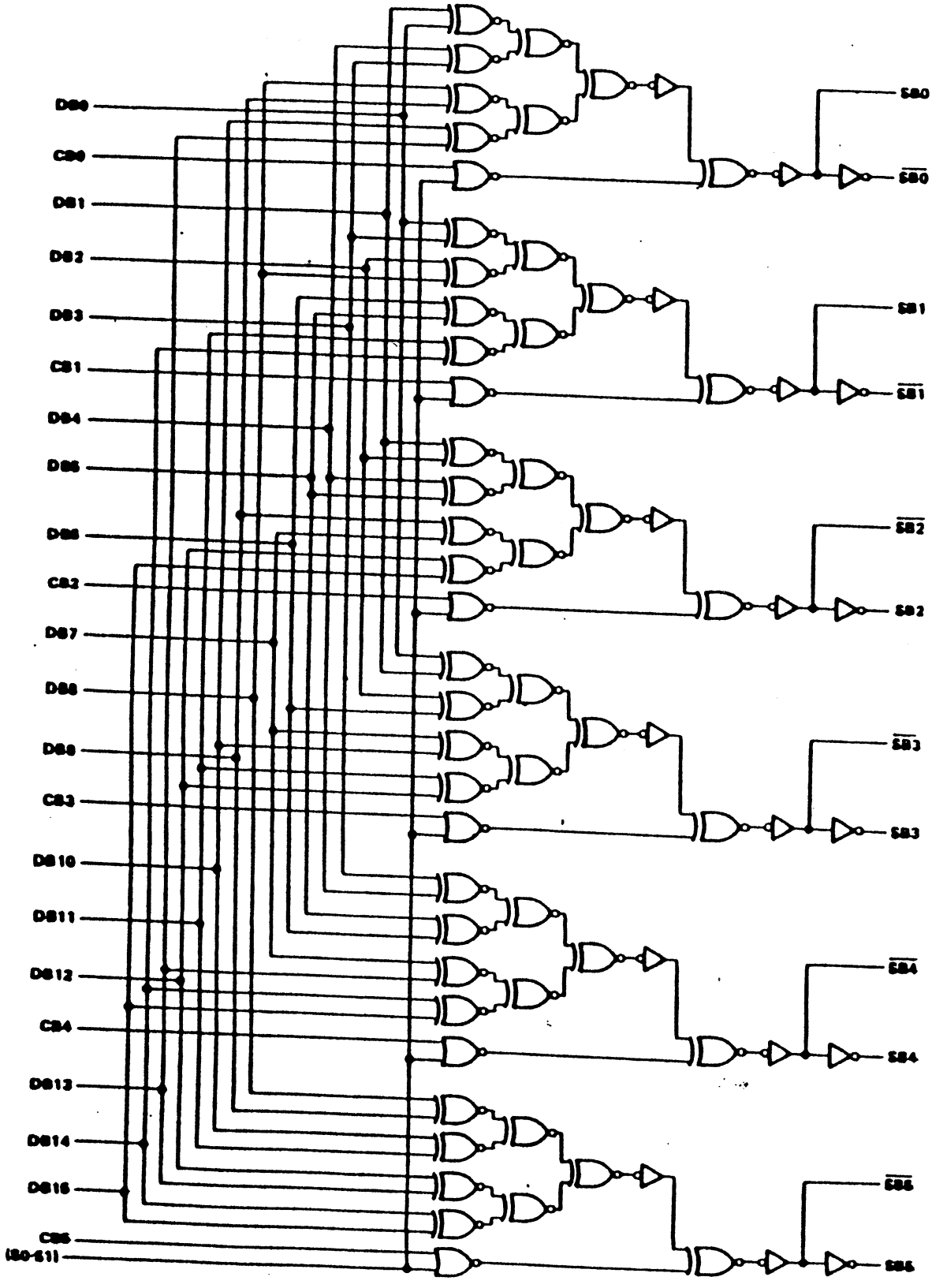


Figure 3-2: Circuit TI SN54/74LS630

Tout circuit dont aucune sortie ne dépend de toutes ses n entrées peut-être testé pseudo-exhaustivement avec moins de 2 vecteurs: deux méthodes voisines d'obtention de ces vecteurs de test de vérification ont été décrites dans [McCluskey 82b] et [Tang 83]. Les vecteurs de test obtenus sont des vecteurs de poids constants, et forment des séquences pseudo-exhaustives de vérification de taille minimale pour un grand nombre de circuits [Tang 83]. Le caractère "poids constant" de ces séquences les rend particulièrement intéressantes pour la génération concurrente de vecteurs de test, puisqu'elles sont facilement générées in-situ par des compteurs à poids constant.

1.2 Test Par Segments

1.2.1 Présentation

Pour certaines catégories de circuits, les séquences de test obtenues par les méthodes de vérification pseudo-exhaustives sont encore trop longues pour être raisonnablement applicables.

En effet, dès qu'une ou plusieurs sorties du circuit dépendent de toutes les entrées, la longueur du test de vérification devient égale à celle du test exhaustif. Si cela ne se produit pas, le circuit peut néanmoins comporter un nombre très élevé de sorties primaires et pseudo-primaires. En effet, le nombre de sorties pseudo-primaires est égal au nombre, parfois élevé, de bascules dans le circuit. La réunion des tests exhaustifs des fonctions de chacune des sorties est alors, dans son ensemble, un test de longueur trop importante pour être applicable.

L'objectif du test pseudo-exhaustif par segments est de réduire la longueur du test de vérification en réalisant un partitionnnement de chacune des fonctions des sorties du circuit [McCluskey 81].

1.2.2 Définitions

1. Un circuit est défini comme un ensemble de noeuds reliant soit une porte à une porte, soit une entrée à une porte, soit une porte à une sortie.

2. La segmentation d'un circuit est le découpage de ce circuit en sous-circuits non nécessairement disjoints. Une partition est donc un cas particulier de segmentation.

3. Le cone de dépendance d'un noeud N est l'ensemble des noeuds appartenant à tous les chemins simples reliant les entrées primaires au noeud N. A partir de cette définition, le test pseudo-exhaustif de vérification peut être redéfini comme le test exhaustif des cones de dépendance de chacune des sorties primaires et pseudo-primaires du circuit.

4. Le cone de dépendance C d'un noeud N peut être décomposé (ou segmenté) en 2 segments (ici des sous-cones) C1 et C2 (voir figure 3-3): C1 a pour sommet le point de segmentation S et pour entrées les entrées du cone originel dont S dépend; C2 a pour sommet le sommet N du cone originel et pour entrées les entrées du cone originel dont N dépend (par les chemins simples ne comprenant pas S) plus le sommet S (qui devient une pseudo-entrée pour le sous-cone de sommet N).

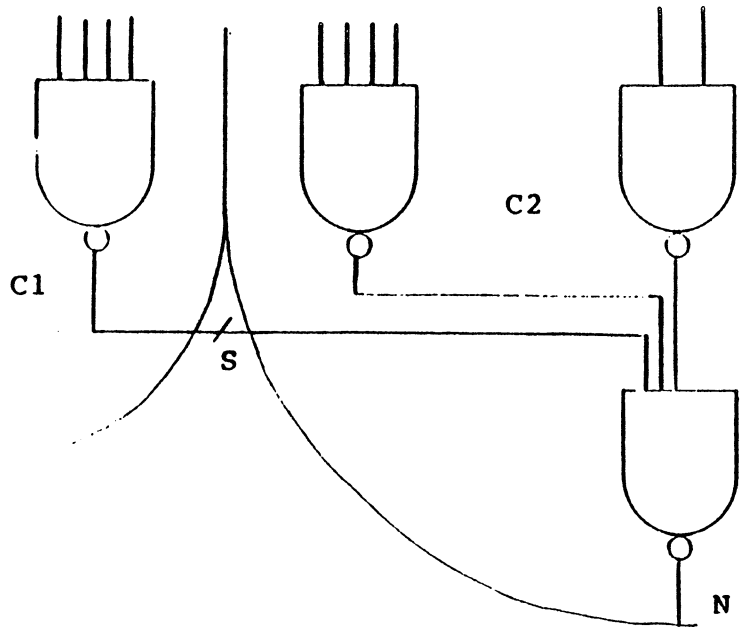


Figure 3-3: Segmentation pour le test pseudo-exhaustif

5. Cette procédure de "bi-segmentation" des cones peut être appliquée itérativement à chacun des sous-cones produits: on obtient alors un recouvrement du cone originel de k segments (non nécessairement disjoints, mais dont la réunion englobe le cone originel).

6. Le test pseudo-exhaustif par segments, du cone de sommet N, associé à un recouvrement Rk de k segments, est la réunion des tests exhaustifs de chacun des k segments. Si le segment j a Ij entrées, la longueur du test pseudo-exhaustif par segment est:

$$2^{I1} + 2^{I2} + ... + 2^{Ik}$$

A noter que si cette segmentation est une partition, la longueur du test exhaustif correspondant est:

$$2^{I1 + I2 + ... + Ik}$$

La longueur du test pseudo-exhaustif par segments est donc généralement de plusieurs ordres de grandeur plus petite que celle du test exhaustif du même circuit. Par exemple, dans le cas de la figure 3-4 la longueur du test exhaustif est 1024. Si l'on fait une segmentation en 3 sous-cones (ici disjoints) la longueur du test pseudo-exhaustif par segments devient:

$$2^4 + 2^4 + 2^{2+2} = 48.$$

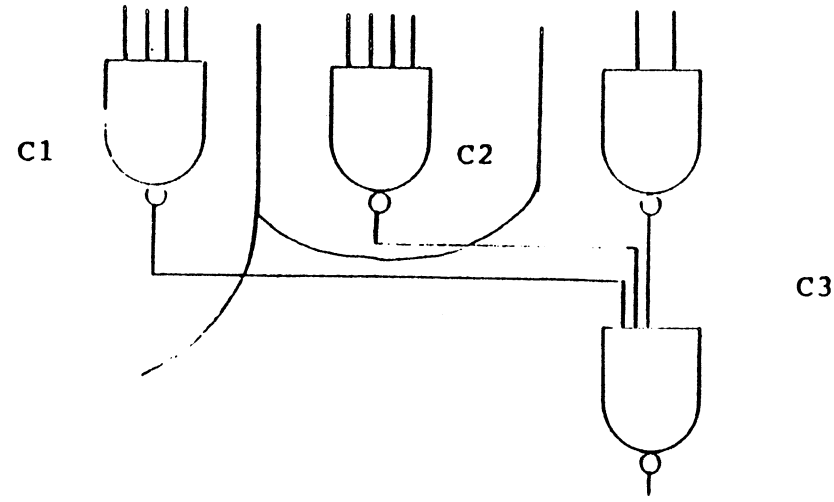


Figure 3-4: Exemple de segmentation pour le test.

Il est important de remarquer que le test pseudo-exhaustif par segments soulève un problème qui n'existait pas pour le test de vérification: il est nécessaire de s'assurer que les différents combinaisons d'entrées sont bien appliquées aux segments dont les entrées ne sont plus nécessairement des points de contrôle direct du circuit (entrées primaires et pseudo-primaires), et que les sorties des segments soient bien observables à la sortie du cone initial. Une méthode de sensibilisation de chemin doit donc être appliquée pour chaque vecteur de test afin d'éviter l'addition de multiplexeurs à l'intérieur du cone originel.

1.2.3 Procédure De Génération De Test

Nous pouvons maintenant proposer une procédure originale de génération de tests pseudo-exhaustifs par segment:

1. Pour chaque sortie primaire ou pseudo-primaire du circuit, évaluer la longueur du test exhaustif de son cone de dépendance.

2. Si la longueur de ce test est supérieure à une valeur limite (calculé en fonction du nombre d'entrées/sorties du circuit et du temps maximum du test fonctionnel), générer une segmentation (comme définie précédemment) du cone en sous-cones, qui diminue la longueur du test en dessous de la valeur limite.

3. Identifier d'éventuels sous-cones identiques appartenant à plusieurs cones afin d'éviter la duplication de tests.

4. Générer pour chaque segment du circuit un test exhaustif, en s'assurant que toutes les combinaisons d'entrées soient bien appliquées aux entrées du segment et que la sortie soit bien observée sur les sorties primaires ou pseudo-primaires du circuit.

5. La réunion, après compactage, des tests exhaustifs de chacun des segments constitue un test pseudo-exhaustif par segment du circuit.

Les Figures 3-5-a à 3-5-e illustrent le concept du test pseudo-exhaustif par segments sur l'ALU 74S181 (unité arithmétique et logique extraite du catalogue de produits standards Texas Instruments).

La figure 3-5-a montre le circuit 74S181. Les sorties primaires F3 et "A=B" du circuit dépendent de toutes les 14 entrées primaires. Le circuit n'est donc pas un bon candidat pour le test pseudo-exhaustif de vérification: test exhaustif et test pseudo-exhaustif de vérification ont la même longueur: 16384 vecteurs.

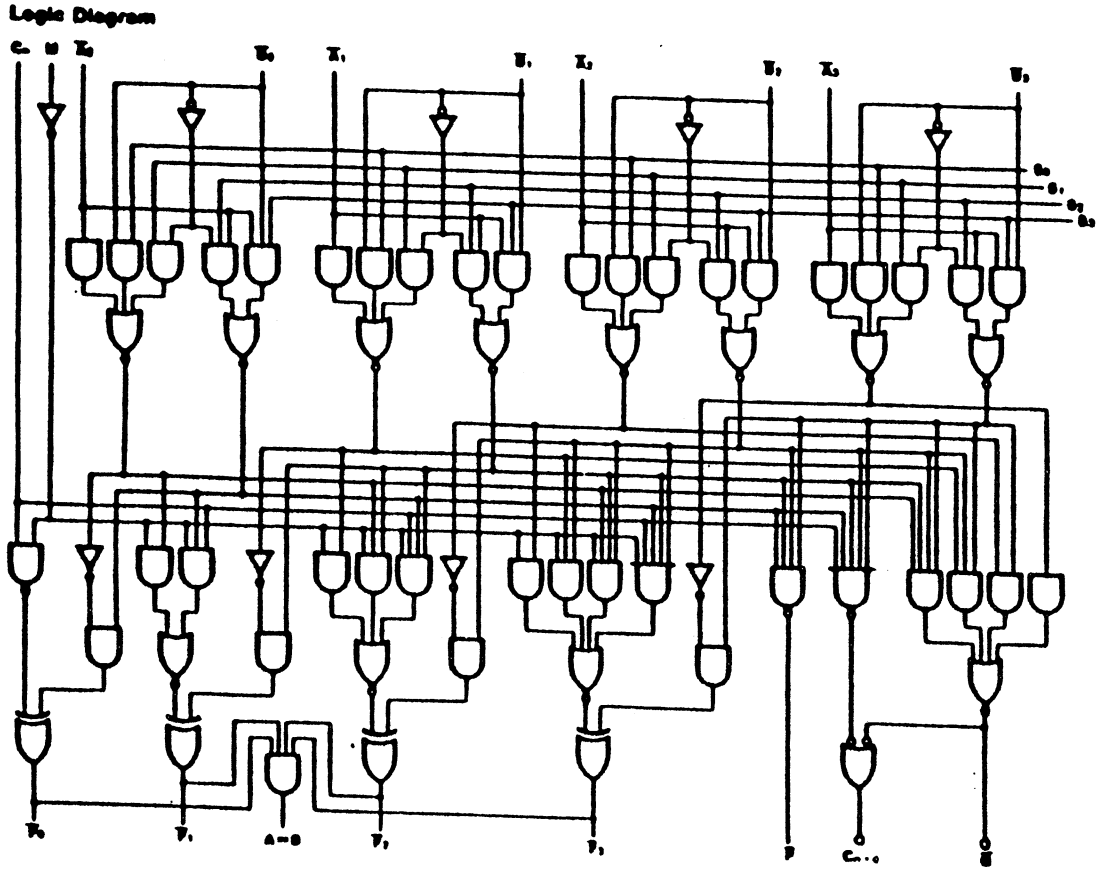


Figure 3-5a: Circuit ALU TI 74S181
Certaines sorties dépendent des 14 entrées.
Test exhaustif et test pseudo-exhaustif de vérification ont la même longueur (16384).

Tableau 3-1: Dépendance des E/S pour le circuit de la figure 3-5-a.

		A0	B0	Ai	Bi	A2	B2	A3	B3	S0	Si	S2	S3
Blocs de niveau 1	H0	x	x									x	x
	L0	x	x							x	x		
	Hi			x	x							x	x
	Li			x	x					x	x		
	H2					x	x					x	x
	L2					x	x			x	x		
	H3							x	x			x	x
	L3							x	x	x	x		
		H0	L0	Hi	Li	H2	L2	H3	L3	M	Cn		
Blocs de niveau 2	F0	x	x							x	x		
	Fi	x	x	x	x					x	x		
	F2	x	x	x	x	x	x			x	x		
	F3	x	x	x	x	x	x	x	x	x	x		
	A=B	x	x	x	x	x	x	x	x	x	x		
	P	x		x		x		x					
	Cn+4	x	x	x	x	x	x	x	x		x		
	G		x	x	x	x	x	x	x				

La figure 3-5-b montre une segmentation (dans ce cas une partition) du circuit pour le test pseudo-exhaustif. La figure montre 5 segments. En fait ces segments contiennent chacun plusieurs segments élémentaires (les segments à une seule sortie définis précédemment) qui ont été regroupés parce qu'ils peuvent être testés en parallèle. Ainsi le segment N2 est en fait constitué de 8 segments (un pour chaque sortie primaire) qui peuvent être testés exhaustivement en parallèle. Le tableau 3-1 montre les relations de dépendance entre les entrées et les sorties des blocs de niveau 1 et de niveau 2.

Le test exhaustif des fonctions Li (i=0,3) peut être effectué en parallèle. L'assignation des broches de contrôle est montrée Figure 3-5-c: toutes les combinaisons de A,B,S0 et S1 sont appliquées pendant que S2, S3, M et Cn sont gardées constantes. La longueur du test est de 16 vecteurs.

Le test exhaustif des fonctions Hi (i=0,3) peut aussi être effectué en parallèle. L'assignation des broches de contrôle est montrée Figure 3-5-d: toutes les combinaisons de A,B,S2 et S3 sont appliquées pendant que S0, Si, M et Cn sont gardées constantes. La longueur du test est de 16 vecteurs.

Finalement toutes les fonctions du bloc N2 sont testées exhaustivement en parallèle (figure 3-4-e). Puisque les combinaisons HiLi=01 (i=0,3) sont impossibles, le "test exhaustif" (nous appelerons "test exhaustif" d'un bloc de couche 2 ou plus, le sous-ensemble maximal de vecteurs différents qui peuvent être appliqués à ce bloc compte tenu des restrictions apportées par les blocs en amont) du bloc N2 est effectué avec seulement 324 vecteurs (81 combinaisons pour les 4 groupes HiLi fois 4 combinaisons pour M, Cn). Le test pseudo-exhaustif associé à cette segmentation nécessite donc seulement 356 vecteurs (16384 pour le test exhaustif).

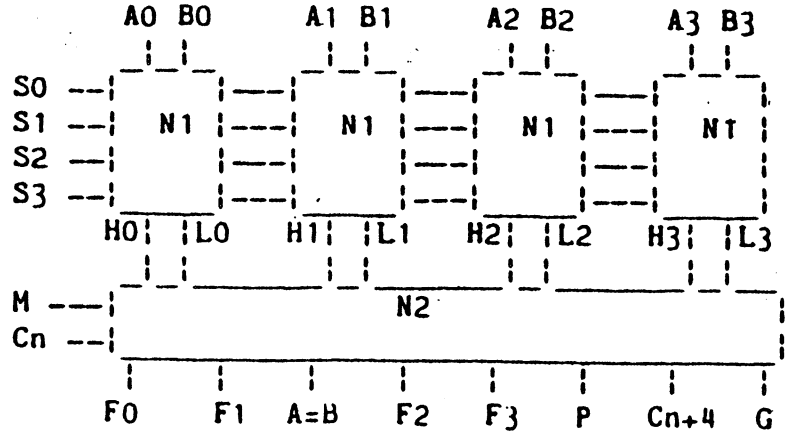


Figure 3-5b: Segmentation de l'ALU 181 pour le test pseudo-exhaustif par segments.

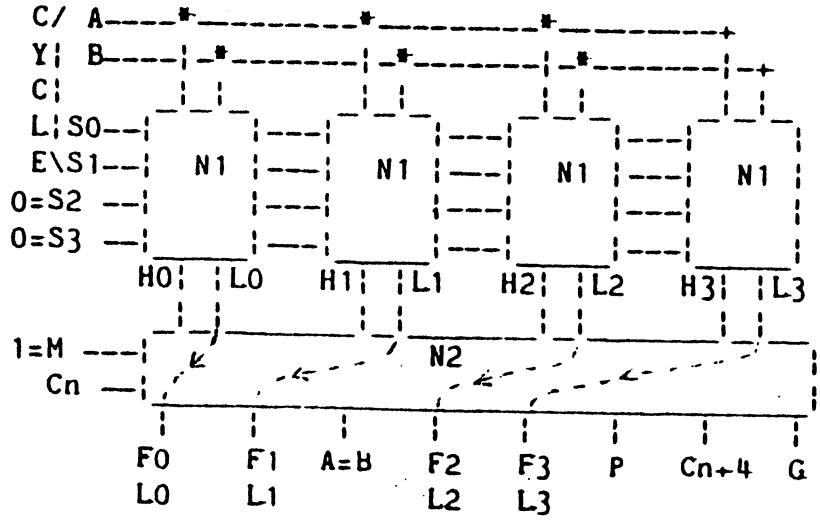


Figure 3-5c: Le test exhaustif des fonctions Li (i=0,3) peut être fait en parallèle; longueur du test: 16 vecteurs.

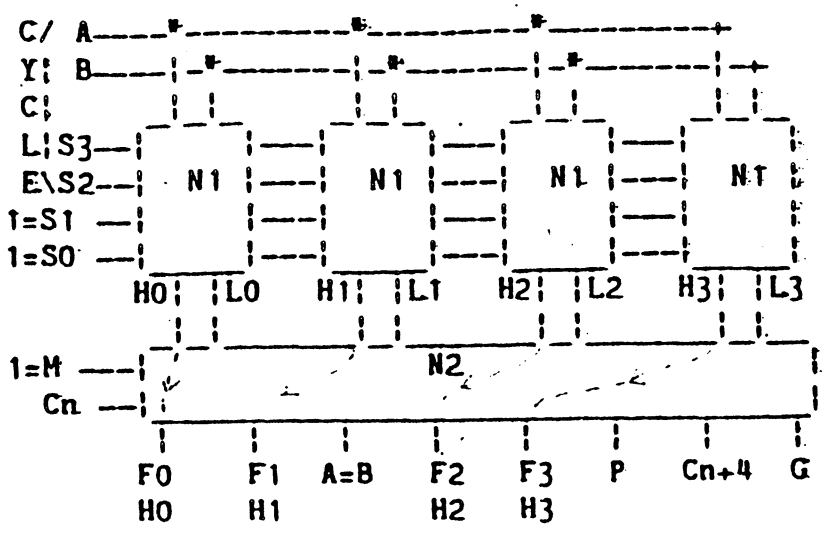


Figure 3-5d: Le test des functions Hi peut aussi etre fait en parallèle avec 16 vecteurs. Comme dans la figure précédente la sensitisation des fonctions Hi est assurée en assignant M à 1.

$Hi = (Ai\ Bi)'$, $Li = Ai'$ (Pas HiLi= 01)

3^4 AB tests 2^2 MCn tests, = 324 tests

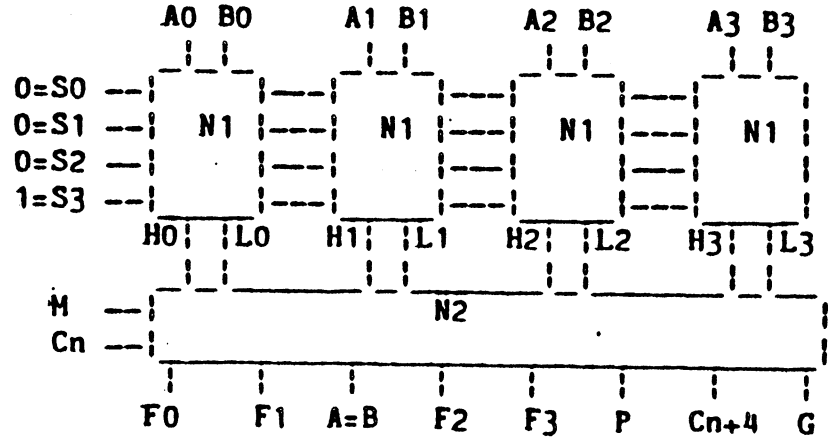


Figure 3-5e: Finalement toutes les fonctions du bloc N2 peuvent être testées exhaustivement en parallèle.

1.2.4 Optimum Et Compromis

On notera que, pour chaque cone de dépendance des sorties primaires, la segmentation en sous-cones ne constitue pas nécessairement une partition exacte du cone initial: les seules contraintes imposées pour la segmentation sont que la réunion des segments recouvre le cone initial et que la longueur du test pseudo-exhaustif résultant soit inférieure à la longueur du test exhaustif du cone initial. Toutes les segmentations de cone ne vérifient pas cette deuxième propriété, comme le montre l'exemple de la Figure 3-6. Considérons le cone de dépendance de la deuxième sortie de ce circuit. Les éléments de ce cone apparaissent en foncé sur la figure. Il n'existe pas de segmentation (ou partitionnement) de ce cone dont la longueur du test pseudo-exhaustif associé soit inférieure à la longueur du test exhaustif du cone. Mais cet exemple est un cas extrême, pour lequel le faible nombre d'entrées ne permet pas de gain important dans la diminution de la longueur du test exhaustif.

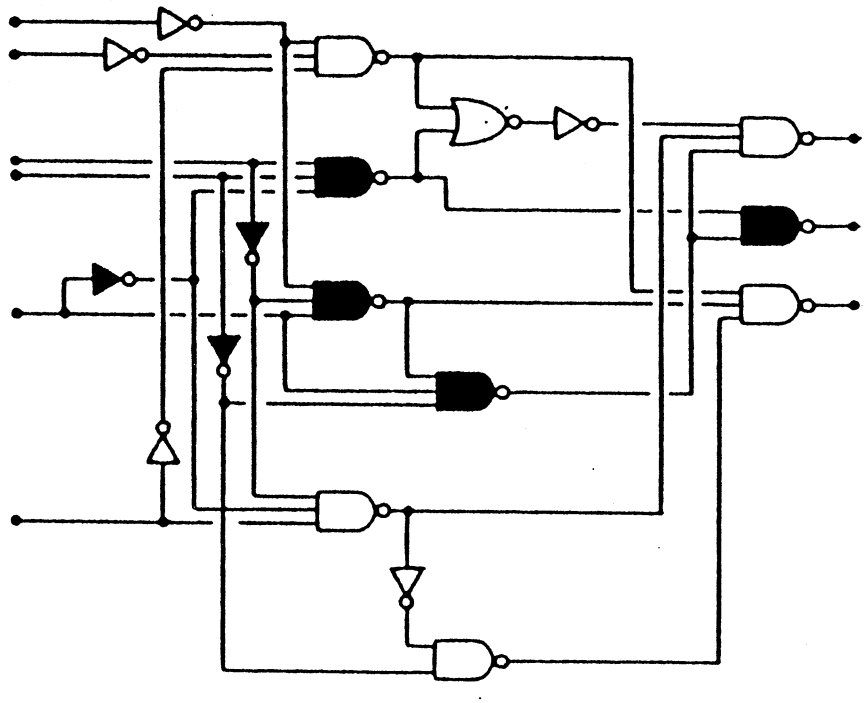


Figure 3-6: Segmentation d'un cone.

La génération de tests pseudo-exhaustifs par segments nécessite (on l'a vu) l'emploi d'une technique de sensibilisation pour créer un chemin pour les entrées et les sorties non-primaires des sous-cones du circuit. Une technique semblable au D-algorithme peut être employée. Dans ce cas, l'algorithme n'est employé que pour un très petit nombre de noeuds du circuit (les points d'entrées/sorties des sous-cones) au lieu de chaque noeud du circuit dans le cas des techniques de génération de vecteurs de test classiques (si une technique de "fault dropping" est utilisée, ce nombre diminue d'un ordre de grandeur). Dans le cas de la génération de tests pseudo-exhaustifs par segments, plus le nombre de segments est faible, plus l'effort de génération nécessité par la création de chemins de sensibilisation sera faible.

On concoit intuitivement que le nombre de segments utilisés influe sur la performance du test pseudo-exhaustif par segments: des pannes situées dans des segments différents peuvent peut être se masquer les unes les autres lorsque ces segments sont testés indépendemment, alors que les mêmes pannes seraient détectées par le test exhaustif du circuit complet. La génération du test pseudo-exhaustif "optimal" par segments consiste donc à trouver la segmentation qui permette d'obtenir le meilleur compromis entre une longueur de test minimale et un taux de couverture maximal, tout en tenant compte de la difficulté de génération des vecteurs de tests apportée par la segmentation. Pour arriver à cet optimum, il convient tout d'abord d'étudier le taux de couverture du test pseudo-exhaustif en fonction de la segmentation réalisée.

2 TAUX DE COUVERTURE DES TESTS PSEUDO-EXHAUSTIFS

2.1 Hypothèses De Pannes

Un taux de couverture étant calculé par rapport à un modèle de panne, nous devons définir une hypothèse de panne pour le calcul du taux de couverture des tests pseudo-exhaustifs. Notre but étant de comparer l'efficacité du test pseudo-exhaustif à celle du test exhaustif, nous comparerons les pannes détectées par le test pseudo-exhaustif aux pannes couvertes par le test exhaustif.

Rappelons que le test exhaustif d'un circuit combinatoire non redondant détecte toutes les pannes qui laissent au circuit son caractère combinatoire. L'hypothèse de panne adoptée dans la suite consistera donc à envisager toutes les pannes non séquentielles détectables du circuit. Cela inclut en particulier toutes les pannes détectables de type collage unique et multiple, et la plupart des pannes détectables de court-circuit.

2.2 Borne Inférieure Du Taux De Couverture

Il existe $NT=2^{2^n}$ manières de remplir le tableau de Karnaugh d'une fonction binaire à n entrées et une sortie. Pour une fonction logique donnée, il existe donc NT-1 transformations différentes possibles de cette fonction en une autre fonction combinatoire à n entrées. Nous appelerons ces NT-1 fonctions possibles, fonctions erronnées.

Le test exhaustif applique les 2^n vecteurs possibles et teste exactement les NT-1 fonctions erronnées. Le test pseudo-exhaustif, qui par construction a moins de vecteurs que le test exhaustif, ne couvre pas toutes les fonctions erronnées possibles. Si NP est le nombre de vecteurs dans le test pseudo-exhaustif, le nombre de fonctions erronnées qui n'ont pas été testées est:

$$2^{(2^n - NP)} - 1$$

Cette valeur représente la borne inférieure du nombre de fonctions non testées par le test pseudo-exhaustif. En effet, le nombre de fonctions erronnées qui peuvent physiquement exister est beaucoup plus faible que NT. La réalisation d'une fonction logique avec des portes logiques intégrées sur une pastille de silicium réduit considérablement le nombre de fonctions erronnées qui peuvent survenir à la suite d'un défaut physique (et non pas d'une erreur de conception). Pour illustrer cette remarque, considérons la réalisation en CMOS d'une porte NON-OU et d'une porte NON-ET, toutes deux à deux entrées (Figure 3-7). Il est facile de voir qu'un nombre élevé (et, du coup, peu réaliste) de pannes doivent se produire simultanément pour transformer la porte NON-ET en porte NON-OU et vice-versa.

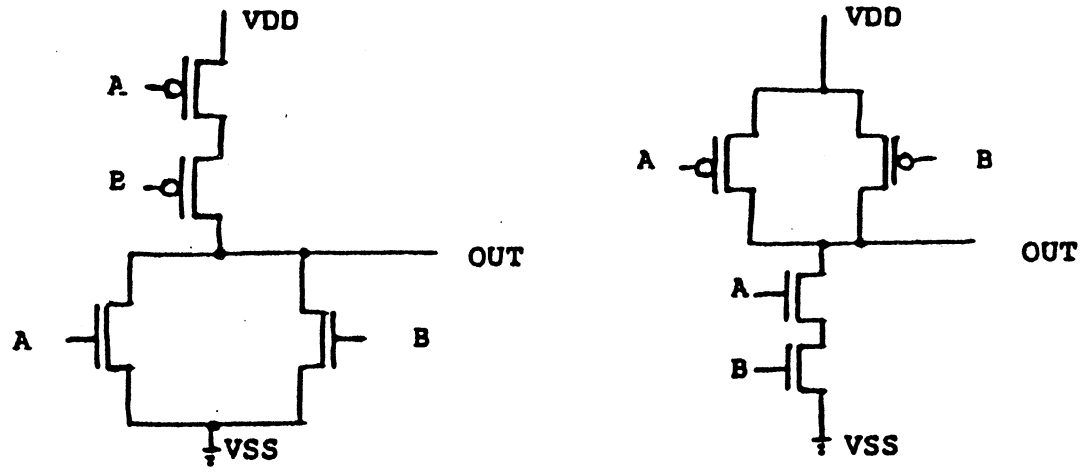


Figure 3-7: Realisation en CMOS de portes NON-ET et NON-OU

Le concept de modèle de panne a été introduit pour réduire le nombre de fonctions erronnées envisagées et pour limiter l'analyse à un petit nombre de fonctions dont la probabilité d'existence soit non négligeable. Ainsi, des 63 différentes fonctions à 3 entrées qui peuvent être obtenues à partir d'un porte ET à 3 entrées, le modèle de panne de type collage unique réduit le nombre de fonctions envisagées à 5: une des 3 entrées ou la sortie collée à 1 (4 pannes) et n'importe quel entrée/sortie collée à 0 (1 panne équivalente).

Nous nous bornerons donc par la suite à évaluer les taux de couverture en fonction des modèles de panne classique: collage simple et multiple à 0 ou 1 et court-circuit.

2.3 Evaluation Du Test Pseudo-exhaustif De Vérification

Avec un test pseudo-exhaustif de vérification, toutes les fonctions réalisées par les sorties primaires du circuit sont testées exhaustivement. Il en résulte que toutes les pannes situées à l'intérieur des cones de dépendance des sorties primaires sont détectées par un test pseudo-exhaustif de vérification. En effet, puisque le reste du circuit n'a d'influence ni sur le cone ni sur la panne, le test pseudo-exhaustif est absolument identique au test exhaustif du cone considéré comme un circuit isolé. On en déduit que toutes les pannes de type collage unique ou multiple sont détectées par les tests pseudo-exhaustifs de vérification.

Par contre les pannes qui rendent les fonctions réalisées par les cones des sorties primaires dépendantes les unes des autres (par exemple à cause d'une panne de court-circuit entre deux connexions de deux cones normalement disjoints) ne sont plus nécessairement détectées par n'importe quel test pseudo-exhaustif de vérification.

Dans l'exemple de la figure 3-8, considérons la panne de court-circuit de type OU entre les connexions x1 et x6. Un test pseudo-exhaustif de vérification du circuit consiste à tester en parallèle chacune des fonctions z1 et z2 avec 16 vecteurs. Si l'on choisit, pour la durée du test, x1=x6 et x2=x5, la panne de court-circuit entre x1 et x6 (ainsi que celle entre x2 et x5) n'est pas détectée.

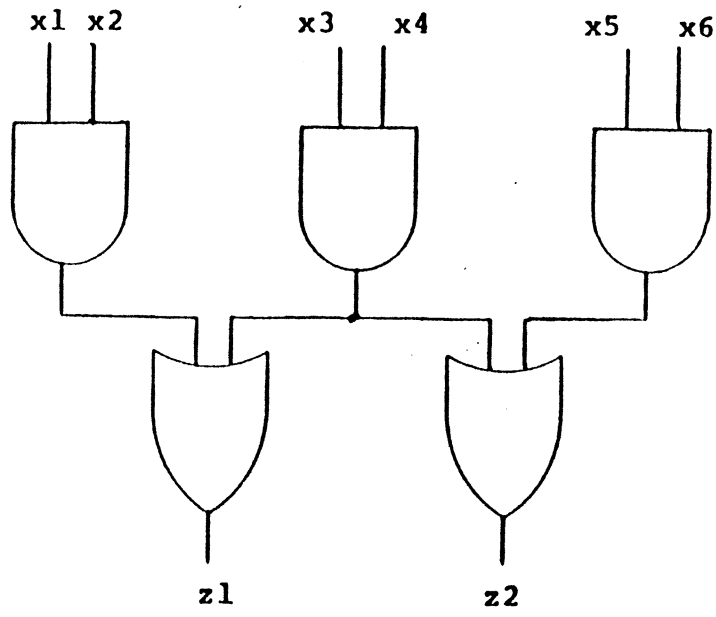


Figure 3-8: Panne de court-circuit non detectee par un test pseudo-exhaustif

2.4 Evaluation Du Test Pseudo-exhaustif Par Segments

2.4.1 Rappel

Rappelons la définition du test pseudo-exhaustif par segments: chacun des cones de dépendance des sorties du circuit est divisé en sous-cones ayant pour entrées des entrées primaires du circuit ou des sorties d'autres sous-cones, et chacun de ces segments est testé exhaustivement. Même pour un nombre réduit de segments, la longueur du test pseudo-exhaustif par segments peut être de plusieurs ordres de grandeur plus faible que la longueur du test pseudo-exhaustif de vérification. Cette réduction de la longueur de test entraine nécessairement une diminution du taux de couverture. L'objet de cette étude est d'évaluer si cette diminution est relative à des pannes physiquement probables et comment la segmentation agit sur le taux de couverture.

2.4.2 Pannes Confinées À Un Segment

Lemme 3-1: Toute panne qui affecte un segment S et laisse le complément de ce segment dans le circuit inaffecté est détectée par n'importe quel test pseudo-exhaustif par segments du circuit, pour lequel au moins un élément de la segmentation du circuit choisi recouvre entièrement le segment S.

Démonstration:

Soient Xj $(j = 1, 2^n)$ les vecteurs du test exhaustif de S, tels qu'ils sont appliqués aux entrées de S. Soit s la fonction réalisée par S. On notera s(Xj,f) la réponse de S au vecteur Xj en présence d'une panne f à l'intérieur de S, et s(Xj,0) la réponse de S en l'absence de panne. Si une panne f dans S est détectable, c'est qu'il existe nécessairement au moins un vecteur Xi qui peut être appliqué à S, tel que: $s(Xi,f) \neq s(Xi,0)$ et tel que la valeur de s peut-être observée sur la sortie primaire. En fait, ce vecteur Xi peut être obtenu par simple observation des entrées de S lorsque un vecteur qui détecte la panne (et qui existe par définition d'une panne détectable) est appliqué sur les entrées primaires.

Lorsque tous les vecteurs Xj possibles sont appliqués aux entrées de S et s(Xj,f) sensibilisé jusqu'à la sortie primaire du circuit, alors la panne est détectée dès que l'un des vecteurs Xi défini ci-dessus appliqué: en effet, $s(Xi,f)$ $(\neq s(Xi,0))$ est sensibilisé à travers le reste du circuit qui, n'ayant pas de panne, n'altère pas la réponse s jusqu'à la sortie. Et comme il existe au moins un Xi qui peut être appliqué à S alors que s est sensibilisé jusqu'à la sortie, la panne est toujours détectée. C.Q.F.D.

Remarque: Cette propriété fondamentale peut être illustrée de la manière suivante. Tant que les pannes considérées restent internes à un segment, le test pseudo-exhaustif a le même pouvoir de détection que le test exhaustif: la provocation et la sensibilisation des pannes à l'extérieur du segment reste inchangée (puisque l'extérieur du segment est sans panne) et par conséquent la détection des pannes est inchangée. Le test pseudo exhaustif de segmentation, avec une segmentation au niveau de la macro-cellule, de la porte logique ou du transistor permet donc d'obtenir un taux de couverture identique à celui du test exhaustif si l'on ne considère que les pannes internes à ces différents segments.

Corollaire 3-1: Toutes les pannes détectables de type collage unique sur les entrées des portes logiques d'un circuit sont détectées par n'importe quel test pseudo-exhaustif par segments de ce circuit.

Corollaire 3-2: Toute panne de type collage multiple détectable, située sur des noeuds entièrement contenus dans un segment S est détectée par n'importe quel test pseudo-exhaustif par segments pour lequel au moins un élément de la segmentation du circuit choisie recouvre entièrement le segment S.

Corollaire 3-3: Toute panne de court circuit, entre noeuds d'un même segment et détectable par un test exhaustif, est détectable par n'importe quel test pseudo-exhaustif par segments pour lequel au moins un élément de la segmentation du circuit choisie recouvre entièrement le segment S.

Ces corollaires sont des conséquences directes du lemme 3-i.

2.4.3 Pannes De Type Collage Affectant Plusieurs Segments

Propriété i: Pour un test réalisant le test pseudo-exhaustif d'un circuit, il peut exister des pannes de type collage multiple qui ne soient pas détectées par ce test.

Démonstration: Pour prouver cette propriété, il suffit de donner un exemple de panne de type collage multiple qui ne soit pas détectée par un test pseudo-exhaustif.

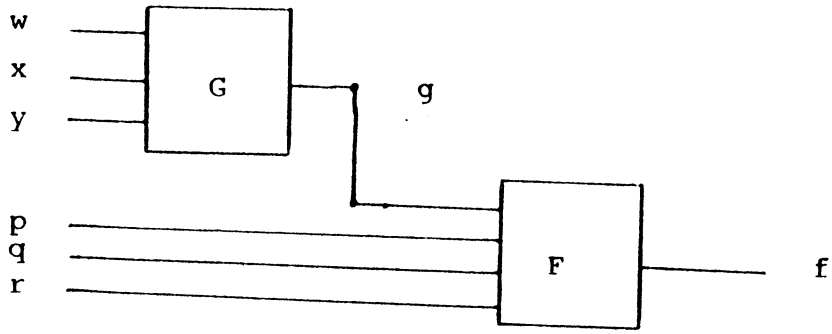


Figure 3-9 : Circuit H

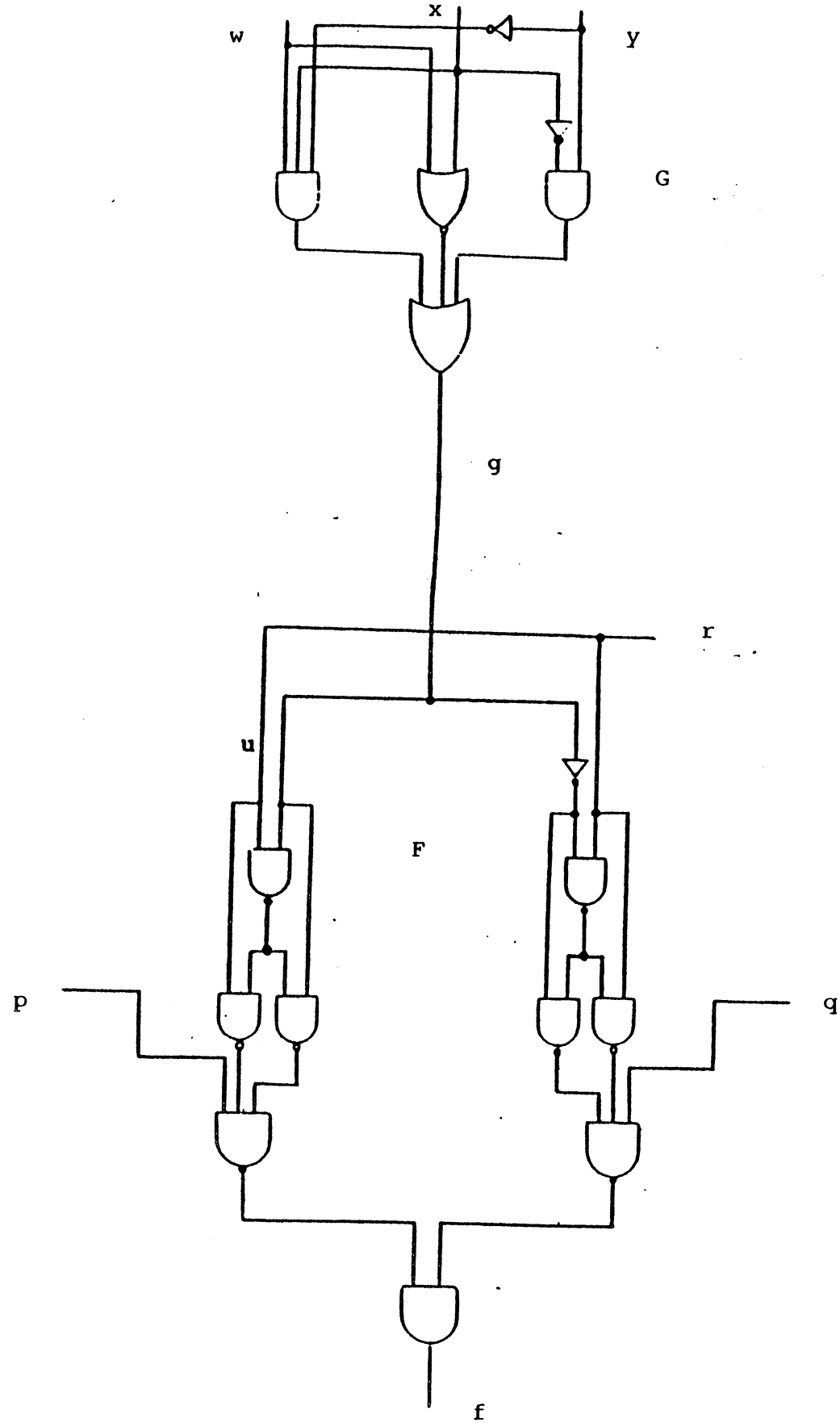
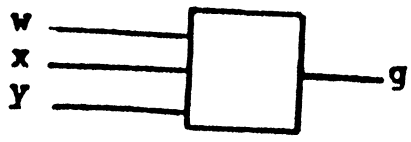


Figure 3-10: Realisation du circuit H

Considérons le circuit H, illustré par les figures 3-9 et 3-10. H est partitionné en deux segments, F et G. G a pour entrées les signaux w, x et y, et réalise la fonction g(w,x,y). F a pour entrées les signaux g, p, q et r, et réalise la fonction f(g,p,q,r). Considérons les deux pannes de type collage: u collé à 1 (panne interne à F, notée u/1) et w collé à 1 (panne interne à G, notée w/1) qui transforment la fonction f en fonction fu et g en fonction gw, comme indiquées sur les figures 3-11 et 3-12.



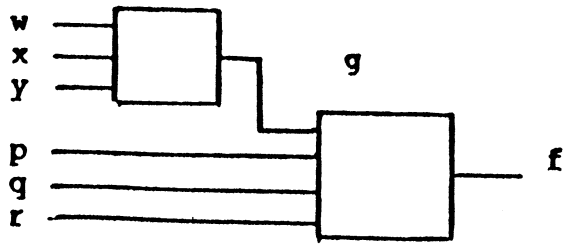
g

y \ wx	00	01	11	10
0	1	1	1	0
1	0	0	0	1

y \ wx	00	01	11	10
0	1	1	1	0
1	0	0	0	1

$g_{w/1}$

Figure 3-11: Segment G



r \ pq	00	01	11	10
0	1	g'	0	g
1	1	g	0	g'

f

r \ pq	00	01	11	10
0	1	g'	g'	g'
1	1	g	0	g'

$f_{u/1}$.

Figure 3-12: Segment F

Tout test pseudo-exhaustif de H correspondant à la segmentation (F,G) détecte chacune de ces pannes lorsqu'elles surviennent séparément. Par contre, nous allons montrer qu'il existe des tests pseudo-exhaustifs de H qui (avec cette segmentation) ne détectent pas l'ocurrence simultanée de u/1 et w/1.

Un test pseudo-exhaustif par segments de H, pour la segmentation choisie, consiste en la réunion d'un test exhaustif du segment F et d'un test exhaustif du segment G, chacun de ces tests devant être contrôlé et observé à partir des noeuds d'entrée/sortie de H.

Tableau 3-2: Test exhaustif pour le segment F

	p	q	r	g	gw	w	x	y
0	0	0	0	0				
1	0	0	0	1				
2	0	0	1	0				
3	0	0	1	1				
4	0	1	0	0		0	1	1
5	0	1	0	1		0	1	0
6	0	1	1	0		1	0	0
7	0	1	1	1		1	0	1
8	1	0	0	0	1			
9	1	0	0	1	0			
10	1	0	1	0		1	1	1
11	1	0	1	1		1	1	0
12	1	1	0	0	1			
13	1	1	0	1	0			
14	1	1	1	0				
15	1	1	1	1				

Pour construire un test pseudo-exhaustif pour H, construisons tout d'abord un test exhaustif pour F, en générant les 16 combinaisons d'entrées possibles pour F. Pour les combinaisons de (p,q,r) résultant en une valeur de f dépendante de g, mais indépendante de la présence de la panne u/1, nous choisirons les valeurs de (w,x,y) qui résultent en une valeur de g correcte en présence de la panne w/1, comme l'indique le

tableau 3-3. De plus, on s'efforcera de couvrir le maximum de combinaisons de (w,x,y) pour permettre de tester G en parallèle, d'où le choix des vecteurs 4,5,6,7,10 et 11. Mais pour les vecteurs 8,9 et 12,13, pour lesquels la présence de u/1 change la valeur de g, nous choisirons (w,x,y) tel que la présence de w/1 masque la présence de u/1: ces choix sont indiqués dans le tableau 3-3.

Tableau 3-3: Assignation des entrées pour les vecteurs qui provoquent la panne

	w	x	y	f	fu
8	0	0	1	0	0
9	0	0	0	1	1
12	0	0	1	0	0
13	1	0	1	0	0

A ce point, tous les vecteurs pour le test de G ont aussi été générés. Le tableau 3-4, obtenu en complétant le reste des combinaisons (w,x,y) dans le tableau 3-1, représente donc un test pseudo-exhaustif par segments de H qui, par construction, ne détecte pas la double panne u/1 et w/1. Pourtant cette panne est clairement détectable par le vecteur:

$$(p,q,r,w,x,y) = (0,1,0,0,0,0)$$

Tableau 3-3: Test pseudo-exhaustif

pour le circuit H

	p	q	r	w	x	y
0	0	0	0	1	1	0
1	0	0	0	1	1	1
2	0	0	1	1	1	0
3	0	0	1	1	1	1
4	0	1	0	0	1	0
5	0	1	0	0	1	1
6	0	1	1	1	0	0
7	0	1	1	1	0	1
8	1	0	0	0	0	1
9	1	0	0	0	0	0
10	1	0	1	1	1	0
11	1	0	1	1	1	1
12	1	1	0	0	0	1
13	1	1	0	1	0	1
14	1	1	1	1	1	0
15	1	1	1	1	1	1

Nous avons donc prouvé l'existence de tests pseudo-exhaustifs par segments de circuit combinatoires qui ne détectent pas toutes les pannes de type collage multiple. C.Q.F.D.

Remarque: Il existe de nombreuses manières de construire un test pseudo-exhaustif pour l'exemple précédent: autant que de chemins de sensibilisations possibles pour la sortie f à travers G et de choix des combinaisons (w,x,y) pour générer l'entrée f. Pour chaque vecteur appliqué à G, il existe (d'après le tableau de Karnaugh de f) 4 choix de (w,x,y) pour produire f, et le test de G nécessite 16 vecteurs: il existe donc 4^{16} tests differents, dont $3^{8} \times 4^{4}$ laissent la double panne indétectée. La probabilité de non détection de la panne, si le

choix du test pseudo-exhaustif est fait de manière aléatoire est de 10^{-4}.

Cette panne de type collage multiple est donc détectée par la plupart des tests pseudo-exhaustifs du circuit.

La panne de type collage multiple considérée n'est pas détectée que si la sensibilisation de f à travers G est effectuée par des chemins avec des parités d'inversion différentes durant le test: si le chemin de sensibilisation de f est gardé constant pendant le test de F, la panne est détectée. De manière générale, si le chemin de sensibilisation de la sortie de chaque segment est constant pendant le test de ce segment, toutes les pannes de type collage unique ou multiple du circuit sont détectées [Archambeau 84b]. Nous allons montrer ici un résultat plus général.

Notations: Un noeud e dépend d'une entrée primaire p si et seulement si il existe un chemin simple reliant p à e. Le nombre total d'entrées primaires dont e dépend est noté D(e). Pour un circuit à n entrées primaires, pour tout noeud e: $1 \leq D(e) \leq n$.

Le test exhaustif du segment E1, de sommet e1, est un ensemble de $2^{D(e1)}$ vecteurs Y de longueur n. Chaque vecteur Y peut être considéré comme un triplet (Xd,Xs,X) où Xd est un vecteur de longueur D(e1) correspondant aux variables prises de manière exhaustive, Xs représente les variables utilisées pour sensibiliser les entrées et la sortie du cone, et X représente les entrées primaires non utilisées.

Définition: Si Xs est constant durant le test, c'est à dire lorsque Xd applique toutes les combinaisons possibles à E1, le test est dit à "chemin de sensibilisation fixe".

Si z=f(Xd,Xs,X) est la sortie primaire du circuit et e1=g(Xd), dans le cas d'un test à "chemin de sensibilisation fixe": z=h(g(Xd),Xs,X).

Définition: Xs(1),....,Xs(p) réalise un chemin de sensibilisation uniforme si et seulement si: z=h(g(Xd),Xs(i),X) est égal à g pour tout i, ou g pour tout i.

La sensibilisation uniforme signifie que sur ces chemins, qui peuvent être différents (contrairement au cas du chemin fixe), la parité d'inversion est constante, soit paire, soit impaire. Le test à chemin de sensibilisation fixe est donc un cas particulier de test à chemin de sensibilisation uniforme.

Nous appelerons conditions de contrôle, les conditions sur les entrées primaires qui permettent d'obtenir les valeurs 0/1 sur les pseudo-entrées des segments internes.

Théorème 3-1: Pour un cône combinatoire, toutes les pannes de type collage unique ou multiple détectables, le sont par un test pseudo-exhaustif à chemin de sensibilisation uniforme et conditions de contrôle fixes.

Démonstration:

Nous emploierons ici un raisonnement itératif sur le nombre de segments.

Supposons que le circuit C soit segmenté en 2 segments: le cone E1, de sommet A et E2, le complément de E1 dans C (voir figure 3-13). Un test pseudo-exhaustif T de C consiste en la réunion d'un test exhaustif T1 de E1 et d'un test exhaustif T2 de E2.

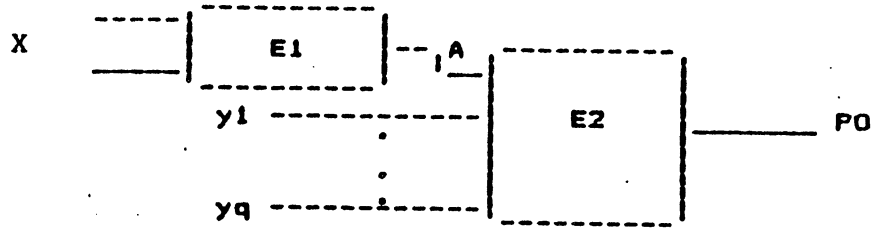


Figure 3-13: Circuit C

Pour E1: PO = f(x1,...,xp,y1(i),....,yq(i)) où x1,...xp sont les variables prises de manière exhaustive, et y1(i),...yq(i), (i=1,s) permettent de réaliser les différents chemins de sensibilisation uniforme.

Les pannes de type collage peuvent être distribuées dans les différents segments. Les cas suivants peuvent être envisagés:

1. Les pannes sont toutes dans Ei, et sont alors détectées lors du test exhaustif de E1 (Lemme 3-1).

2. Les pannes sont toutes dans E2, et sont alors détectées lors du test exhaustif de E2 (Lemme 3-1).

3. Les pannes sont distribuées entre E1 et E2.

Nous supposerons maintenant que nous sommes dans le troisième cas. Si une panne p détectable est contenue dans E1, il existe nécessairement un vecteur d'entrée de E1, X, tel que: $A(X,p) \neq A(X,0)$. C'est à dire $A(X,p)=d$, où d = 0 lorsque la panne est visible et 1 lorsque elle est invisible ou absente.

Nous emploierons les notations suivantes:

PO = f(x1,...,xp,y1(i),...yq(i)), ou

PO = g(A,y1(i),...yq(i)) = g(A,Y(i)).

Pendant le test pseudo-exhaustif de E2, on utilisera deux vecteurs X0 et X1 tels que A(X0)=0 et A(X1)=1 (lorsque E1 est sans panne) afin d'appliquer toutes les valeurs possibles sur la pseudo-entrée A de E2.

Supposons que tous les chemins de sensibilisation uniforme, c'est à dire tous les vecteurs Y(i), i=1,s, ont été choisis pour que g(A,Y(i)) = A lorsque E2 est sans panne (la démonstration pourrait être faite de manière équivalente avec g(A,Y(i)) = A*). On peut alors distinguer les cas suivants pour E2:

1. Pour tous les vecteurs Y(i) utilisés, g(A,Y(i)) = A, c'est à dire la (ou les) panne(s) de E2 ne masquent jamais la panne de E1 pendant le test de E1: alors la panne de E1 sera détectée pendant le test de E1, et la panne multiple sera ainsi détectée.

2. Pour certains chemins de sensibilisation, la panne de E2 conduit à:

   g(A,Y(i)) = 0

   Dans ce cas, si la configuration f(X,Y(i))=g(1,Y(i)) n'est pas appliquée pendant le test de E1, la panne p de E1 conduisant à A(X,p)=A(X,0) sera masquée. Par contre, la panne de E2 conduisant à f(A(X1),Y(i))=0 sera détectée pendant le test de E2 lorsque X1 sera appliqué.

3. Pour certains chemins de sensibilisation, la panne de E2 conduit à:

   g(A,Y(i)) = 1

   Ce cas est similaire au cas précédent (avec X0 à la place X1).

4. Pour certains chemins de sensibilisation, la panne de E2 conduit à:

$$g(A,Y(i)) = A^*$$

Dans ce qui suit, nous supposerons que nous sommes dans ce dernier cas. Nous pouvons distinguer les sous-cas suivants:

(a) Il existe un chemin sensibilisé j tel que:

$$g(A,Y(j)) = A$$

Soient X0 et X1, les deux vecteurs tels que A(X0,0)=0 et A(X1,0)=1, utilisés pour le test exhaustif de E2 sous des conditions de contrôle fixes. Si A(X0,p)=1 ou A(X1,p)=0, lorsque E2 est testé, f(X0,Y(j))=d ou f(X1,Y(j))=d, et la panne est détectée.

(b) Pour tous les chemins de sensibilisation uniforme:

$$g(A,Y(i)) = A^* \ , \ i = 1,s$$

dans le segment E2, et il existe un vecteur X tel que: A(X) = d ou $\bar{d}$ pour le segment E1. Alors la panne est détectée pendant le test de E1.

(c) Pour tous les chemins de sensibilisation uniforme:

$$g(A,Y(i)) = A^* \ , \ i = 1,s$$

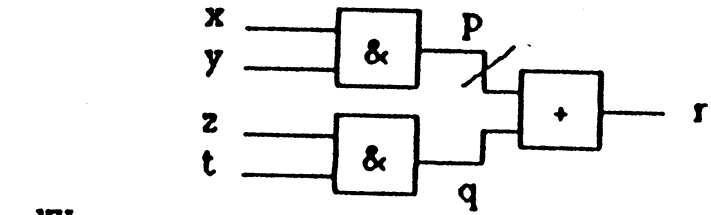
dans le segment E2, et pour tous les vecteurs X, A(X) = d ou $\bar{d}$ pour le segment E1. Mais puisque la panne est détectable, elle sera détectée pendant le test exhaustif de E2.

Donc, pour une segmentation en 2 sous-circuits, le résultat est vrai. Pour une segmentation en segments multiples, le même raisonnement peut être repris en divisant à chaque fois les

segments en 2 et en effectuant le même raisonnement sur chacun des segments résultants. Ce qui achève la démonstration du Théorème 3-i.

2.4.4 Pannes De Court-circuit Entre Segments

La figure 3-14 illustre une panne de court-circuit qui résulte en un OU logique des deux connexions y et z. Cette panne est détectable par un test exhaustif du circuit (de sommet r). Si le circuit est partitionné en deux segments, comme indiqué sur la figure, la panne réalise un court-circuit entre les deux segments. Le tableau 3-5 donne un test pseudo-exhaustif qui, pour cette segmentation, ne détecte pas la panne. Pourtant, le vecteur (x,y,z,t)=(0,1,0,1) détecte la panne. De manière générale, tout test pseudo-exhaustif qui contient l'un des vecteurs (0,1,0,1) ou (1,0,1,0) détecte la panne. Si la sélection des vecteurs (lorsqu'un choix est possible) lors de la construction du test pseudo-exhautif est faite aléatoirement, la probabilité de détection de la panne de court-circuit par le test est élevée.



zt \ xy	00	01	11	10
00	0	0	1	0
01	0	0	1	0
11	1	1	1	1
10	0	0	1	0

Sans Panne

zt \ xy	00	01	11	10
00	0	0	1	0
01	0	1	1	0
11	1	1	1	1
10	0	0	1	1

y-z "wired-or"

Figure 3-14: Exemple de court-circuit entre segments

Tableau 3-5: Test pseudo-exhaustif pour le circuit

x	y	z	t
0	1	0	0
1	0	0	0
1	1	0	0
0	0	0	0
0	0	0	1
1	1	0	1
0	0	1	1
1	1	1	1
0	0	1	0
1	1	1	0

D'après les résultats précédents, on remarque que plus le nombre de segments est élevé, plus le nombre de pannes de court-circuit qui peuvent rester indétectées par un test pseudo-exhaustif est grand. En fait, seul un nombre réduit de pannes de court-circuit entre segments peuvent physiquement se manifester. Un moyen de minimiser leur probabilité d'occurrence est d'imposer dans le programme de génération automatique de la topologie, une règle qui maintienne une distance plus grande que normal entre les connexions appartenant à des segments logiques distincts, si ces lignes sont physiquement adjacentes.

2.4.5 Conclusion

En conclusion, les pannes de type collage multiple affectant plusieurs segments et les pannes de court-circuit entre segments sont détectables par les tests pseudo-exhaustifs par segments, à condition de suivre, dans l'algorithme de génération des vecteurs de tests, des conditions strictes de création de chemins de contrôle et d'observation pour les entrées et les sorties des segments. Ces contraintes sont d'autant plus difficiles à observer que le nombre de segments est élevé. Le taux de couverture des tests pseudo-exhaustifs par segments sera d'autant plus élevé que le nombre de segments sera faible, et la difficulté de génération des vecteurs de test d'autant plus élevée que le nombre de segments sera grand. Ces résultats devront être pris en compte lors de l'établissement d'une stratégie de segmentation des circuits en vue du test pseudo-exhaustif.

3 SEGMENTATION DES CIRCUITS POUR LE TEST PSEUDO-EXHAUSTIF

3.1 Notations

Etant donné un circuit combinatoire, définissons le graphe acyclique orienté G=(V,A) comme suit. (Remarque: dans tout ce qui suit, les graphes, orientés ou non, n'ont ni rebouclages ni arcs ou sommets parallèles.) L'ensemble des sommets V est la reunion disjointe de Ve, Vp et Vs, correspondants respectivement aux entrées primaires, portes logiques et sorties primaires du circuit. Nous appelerons donc les sommets de Ve (resp. Vp et Vs) entrées (resp. portes et sorties.) L'ensemble des arcs A contient tous les couples de sommets (a,b) tels qu'il existe une connexion (sans point de reconvergence) qui ne passe à travers aucune autre porte et qui relie a à b. Dans la cas où une entrée primaire est directement reliée à une sortie primaire, une porte "transparente" t (tampon direct, par exemple) est ajoutée à Vp et les deux arcs reliant respectivement l'entrée à la porte t et la porte t à la sortie sont ajoutés à A. Ainsi un arc reliera soit une entrée à une porte, une porte à une porte ou une porte à une sortie.

Un recouvrement R = {R1,R2,...,Rj} de Vp est un ensemble de parties non-vides de Vp, dont la réunion contient Vp. Pour toute partie Ri de Vp, on définira E(Ri) comme le nombre de sommets extérieurs à Ri dont les arcs entrant Ri sont issus (c'est a dire le nombre d'entrées du sous-circuit correspondant à Ri, qui est inferieur ou egal au nombre d'arcs entrant dans

Ri).

La valeur $2^{E(Ri)}$ est donc le nombre de vecteurs nécessaires pour le test exhaustif du segment associe à Ri. La taille de R est son cardinal j. Le coût de R, C(R), est défini comme le coût du test pseudo-exhaustif associé à cette segmentation, c'est à dire:

$$C(R) = \sum_{i=1}^{j} 2^{E(Ri)}$$

3.2 Tests Pseudo-exhaustifs Par Segments

Lorsque le nombre d'entrées primaires d'un circuit est trop élevé pour permettre le test exhaustif de ce circuit, le test exhaustif d'un ensemble de sous-circuits le recouvrant (appelé test pseudo-exhaustif par segments [Bozorgui-Nesbat 81]), constitue une vérification du circuit presque aussi efficace que le test exhaustif lui-même. Ce résultat est la conclusion du chapitre précédent. L'objet du présent chapitre est de résoudre le problème de la segmentation du circuit en vue d'un test pseudo-exhaustif.

Intuitivement, la longueur du test pseudo-exhaustif décroit à peu près exponentiellement (en base 2) avec le cardinal du recouvrement: plus un circuit est segmenté et plus la longueur du test pseudo-exhaustif décroît. Ce résultat est exact pour les circuits dont la sortance maximum est 1, comme le prouve le Théorème 3-1. Mais ne l'est pas nécessairement pour les circuits comportants de nombreux points de sortance reconvergents.

Notations: Un noeud p dépend d'une entrée primaire e si et seulement si il existe un chemin direct de e vers p, dans le graphe orienté G associé au circuit combinatoire considéré.

Le nombre total d'entrées primaires dont dépend le noeud p est noté D(p). Pour un circuit à n entrées primaires, un noeud p est appelé "intérieur" si et seulement si: $1 < D(p) < n$.

Théorème 3-1: Pour un circuit à n entrées ($n > 3$) et une sortie unique, dont le graphe G associé est un arbre (c'est à dire où tous les noeuds ont une sortance au plus égale à 1), pour tout bi-partitionnement en un noeud p (tel que $D(p)>1$), la longueur du test pseudo-exhaustif correspondant est strictement inférieure au nombre de vecteurs nécessaires au test exhaustif.

Demonstration: Puisque les noeuds du circuit ont une sortance au plus égale à 1, un bi-partitionnement effectué en un noeud intérieur quelconque p, divise le circuit en deux sous-circuits disjoints, notés A et B, comme l'illustre la Figure 3-17. Le sous-circuit A a D(p) entrées primaires et une sortie: p. Le sous-circuit B a n-D(p) entrées primaires et une entrée pseudo-primaire p, ainsi que la sortie primaire o comme sortie. Le test exhaustif du sous-circuit A nécessite l'application de:

$$2^{D(p)} \text{ vecteurs.}$$

Le test exhaustif du sous-circuit B nécessite l'application de :

$$2^{n-D(p)+1} \text{ vecteurs.}$$

Le nombre de vecteurs pour le test pseudo-exhaustif associé à cette partition est donc:

$$2^{D(p)} + 2^{n-D(p)+1}$$

Puisque p est un noeud intérieur, $D(p) \leq n-1$.

D'autre part, $2 \leq D(p)$ implique que $n-D(p)+1 \leq n-1$. Mais pour tout $n > 3$ ces deux relations ne peuvent être simultanément des égalités. Il en résulte que:

$$2^{D(p)} + 2^{n-D(p)+1} < 2^{n-1} + 2^{n-1} = 2^{n} \quad \text{CQFD.}$$

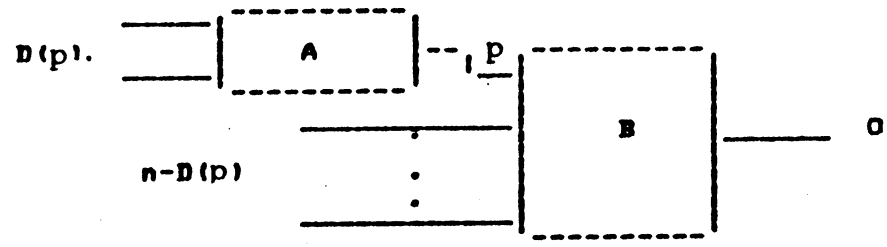


Figure 3-17: Segmentation en 2 sous-circuits

En extrapolant ce résultat, on trouve qu'un test pseudo-exhaustif de longueur minimale sera obtenu avec un partitionnement de cardinalité maximale. Encore une fois ce résultat n'est démontré que pour les circuits dont le graphe associé est un arbre ou une forêt. Pour les circuits contenant des points de reconvergence multiples, ce résultat n'est plus forcément vrai, mais l'expérience montre qu'un test de longueur minimal est le plus souvent obtenu avec une segmentation (plus nécessairement une stricte partition) de cardinal élevé,

Théorème 3-2: Pour un circuit combinatoire dont le graphe G (comme défini précédemment) est une forêt F, le test pseudo-exhaustif de longueur minimal est obtenu pour un partitionnement P consistant en la réunion des |Sp| singletons contenant chacun une porte du circuit.

Démonstration: Nous démontrons ce résultat par l'absurde.

Supposons qu'il existe une partition différente P' telle que: $C(P')<C(P)$. Il existe dans P' au moins une partie Pi' de Sp telle que Pi' contienne au moins deux portes (sinon, P' ne contiendrait que des singletons, mais comme P' recouvre Sp, P' contiendrait P, ce qui contradirait $C(P')<C(P)$.) Soit p une porte de Pi' telle que p ait toutes ses entrées provenant de l'extérieur de Pi' (une telle porte existe nécessairement car F est acyclique.) Définissons Pi" comme le complément de p dans Pi' et remplacons Pi' dans le recouvrement P' par la reunion du singleton {p} et de Pi". Nous obtenons alors un nouveau recouvrement noté P". Notons $E(\{p\})=m$ et $E(Pi')=n$. Par définition, Pi' a au moins 2 portes, donc: $2=<m<n$. Mais $E(Pi")=n-m+1$ car exactement m arcs ont été retirés de Q et 1 seul ajouté, et ces m+1 arcs proviennent de m+1 sommets différents, aucun dans Pi" (par définition de p).

On a $2^{E(\{p\})} + 2^{E(P")} =< 2^{E(P')}$ puisque $2^{m} + 2^{n-m+1} =< 2^{n}$

pour $1 < m < n$ (l'égalité n'est vraie que pour m=2 et n=3 simultanément.) Donc: $C(P")=<C(P')$. Ce processus peut être répété sur P" pour donner un nouveau recouvrement P''' qui vérifie: $C(P''')=<C(P")$. Une séquence P',P",P'''.. est ainsi formée jusqu'à l'ordre r pour lequel le recouvrement Pr n'est formé que de singletons. Par transitivité, $C(Pr)=<C(P')$. Donc

C(Pr)<C(P), et Pr ne contient que des singletons et recouvre Sp.

Donc Pr contient P, et alors C(Pr)>=C(P), ce qui contredit le résultat précédent. Ainsi l'hypothèse initiale: "il existe une partition differente P' telle que: C(P')<C(P)", conduit à une contradiction. C'est donc que le test pseudo-exhaustif de longueur minimale est obtenu pour un partitionnement P consistant en la réunion des |Sp| singletons contenant chacun une porte du circuit. CQFD.

Les résultats précédents suggèrent que:

- La longueur du test pseudo-exhaustif décroit (strictement pour les arbres, au sens large pour les autres circuits) lorsque le nombre de segments augmente.

- Un test pseudo-exhaustif de longueur minimal (ou presque minimal) sera généralement obtenu pour une segmentation de cardinalité maximale.

Pour utiliser ces conclusions, il convient de définir l'optimalité d'un test pseudo-exhaustif et de la segmentation correspondante. Rappelons que le but du test pseudo-exhaustif est de générer un test qui ait un taux de couverture aussi proche que possible de celui du test exhaustif mais dont la longueur soit largement inférieure à celle du test exhaustif. Cette longueur ne doit pas nécessairement être minimale: il suffit qu'elle soit inférieure à un maximum acceptable B.

D'autre part les résultats du chapitre précèdent ont établi que:

- Le test pseudo-exhaustif par segment a un taux de couverture d'autant plus élevé que le nombre des segments formant le recouvrement est faible.

- La complexité de la génération de chemins sensibilisés permettant l'accès aux entrées (non-primaires) et aux sorties (non-primaires) des segments , lors du test exhaustif de ces derniers, augmente avec le nombre de segments. Cette complexité, nulle pour le test exhaustif, devient égale à la complexité du D-algorithme lorsque la taille des segments est de l'ordre de la porte logique.

Le problème de la segmentation pour le test pseudo-exhaustif peut donc etre défini comme suit:

Segmenter le circuit de telle manière que le test pseudo-exhaustif associé soit inférieur à une longueur maximale acceptable (borne B), tout en maintenant le nombre de segments suffisamment faible pour garantir un taux de couverture élevé et une faible complexité de propagation et observation des vecteurs de test.

3.3 Segmentation

3.3.1 Introduction

La génération algorithmique d'une segmentation telle que le coût (longueur) du test pseudo-exhaustif associé soit inférieur à une borne B donnée est un problème difficile, dont la difficulté augmente rapidement avec la taille du circuit. En fait, nous allons montrer que c'est un problème NP-complet.

La classe des problèmes NP-complets a largement été étudiée au cours de ces dix dernières années [Garey 79]. Aucune méthode de résolution de ces problèmes avec un coût de calcul qui soit dans tous les cas borné par une fonction polynomiale de la taille N du problème, n'a été trouvée. Les meilleures méthodes exactes de résolution de ces problèmes requièrent un coût de calcul qui augmente exponentiellement avec N. Tous ces problèmes sont rangés dans la même classe car si une solution en un temps polynomial existait pour l'un de ces problèmes, elle pourrait être transcrite en une procédure en un temps polynomial pour chacun des problèmes de la classe NP.

Nous allons montrer que le problème décisionnel suivant est NP-complet:

Segmentation optimale d'un circuit (problème SOC):

Instance: Un graphe orienté G (comme défini précédemment), un entier positif $K =< |Sp|$ et une borne, entier positif, $B =< 2^{Se}$.

Question: Existe-t-il un recouvrement R de Sp, de taille K, tel que $C(S) =< B$?

Il est intéressant à ce point de noter que le problème SOC est fortement lié au problème de la recherche d'une segmentation résultant en un test de longueur minimale, problème STM defini comme suit:

Problème STM:

Trouver un recouvrement R, de taille au plus égale à K, qui minimise le coût C(R).

STM et SOC sont polynomialement liés: l'existence d'une solution algorithmique en un temps polynomial pour l'un implique l'existence d'une solution algorithmique en un temps polynomial pour l'autre, et réciproquement. En d'autres termes, si SOC est

NP-complet, STM l'est aussi. En effet, un algorithme Asoc pour SOC en temps O(T(n)), utilisé comme sous-programme, donne un algorithme Astm pour STM en un temps O((logB)T(n)), puisque il suffit à Astm d'effectuer une recherche binaire sur B en utilisant au plus logB+1 appels a Asoc. Donc Astm est un algorithme en temps polynomial si et seulement si Asoc l'est. Réciproquement, il suffit à un algorithme Asoc pour SOC de comparer le cout minimum, donné par un algorithme Astm, avec la borne B. Donc Asoc est un algorithme en temps polynomial si et seulement si Astm l'est. CQFD.

Theorème 3-3: Le problème SOC est NP-complet.

Démonstration: Donnée en Annexe 3 (ainsi que dans [Archambeau 85a]).

Corollaire 3-1: Le problème STM est NP-complet.

Démonstration: D'après la remarque précédente, corollaire direct du Théorème 3-3.

Notons qu'un certain nombre de sous-problèmes du problème SOC, qui peuvent être d'interêt pratique, sont aussi NP-complets.

Corollaire 3-2: Le problème SOC reste NP-complet pour les circuits dont les portes sont de sortance 1, mais les entrées primaires d'entrance quelconque.

Corollaire 3-3: Le problème SOC reste NP-complet pour les circuits dont les portes ont une sortance bornée par une constante c>1.

Corollaire 3-4: Le problème SOC reste NP-complet si K est égal à n'importe quelle constante c>1.

Démonstrations: Données en Annexe 3.

Les corollaires 3-3 et 3-4 indiquent que pour pratiquement tous les types de circuits (autres que les arbres et les forêts) le problème SOC est NP-complet.

Le corollaire 3-4 indique que la segmentation du circuit en un petit nombre de segments est aussi NP-complet.

La classe des problèmes NP-complets contenant de nombreux problèmes d'intérêt général [Garey 79], des méthodes de resolution heuristiques ont été développées. Pour ces méthodes, la complexité est proportionnelle à de faibles puissances de N. Pour resoudre notre problème de segmentation, nous allons donc adopter une de ces méthodes heuristiques.

Il existe deux stratégies de base pour résoudre de manière heuristique les problèmes NP-complets: la division du problème en sous-problèmes séparement solvables ("diviser pour régner" en langage d'intelligence artificielle) et l'amélioration itérative. Dans cette seconde stratégie, le point de départ est une configuration arbitraire du système. Un réarrangement est alors appliqué successivement à toutes les parties du système, jusqu'à ce que l'un d'eux diminue la fonction coût à minimiser. La nouvelle configuration du système devient le point de départ pour une nouvelle itération du processus de réarrangement. De nouvelles iterations sont effectuées jusqu'à ce que le coût ne diminue plus ou qu'aucun nouveau réarrangement ne soit possible. Une méthode particulièrement efficace, appartenant à cette deuxième classe de procedures heuristiques, a été récemment présentée pour resoudre le problème du bi-partitionnement des circuits electriques [Fiduccia 82]. La stratégie que nous avons choisie pour la segmentation des circuits pour le test pseudo-exhaustif est basée sur cette technique. Avant de

présenter cette procédure de "bi-segmentation" il convient de rappeler la définition du bi-partitionnement et de présenter la procedure heuristique de bi-partitionnement sus-mentionnée.

3.3.2 Bi-Partitionnement

3.3.2.1 Présentation

Le problème du bi-partitionnement peut etre formulé de la manière suivante:

Soit un graphe G=(S,A), dont chaque élément ai de A est pondéré par un coefficient Ci. Le bi-partitionnement consiste à partitionner les sommets du graphe en deux sous ensembles disjoints (S1,S2) tels que S2 = S-S1 et que le coût du partitionnement P défini par:

$$C(P) = \sum Ci$$ pour tout i tel que ai lie un element

de S1 à un élément de S2, soit minimum sous la contrainte:

$$L < |S1| < U .$$

L et U etant respectivement la borne inférieure et supérieure du cardinal de l' ensemble S1.

Definition: L'ensemble des arêtes a=(s1,s2) telles que s1 appartienne à S1 et s2 appartienne à S2 est appelé la coupe du bi-partitionnement.

Le problème du bipartitionnement peut être résolu de manière exhaustive, mais en un temps proportionnel a $O(|S|!)$. Aucun algorithme pouvant résoudre ce problème en temps polynomial n'a été trouvé, mais une procédure heuristique de bi-partitionnement des graphes a été introduite dans [Kernighan 70]. Une méthode pour appliquer cette procèdure aux circuits électriques a été

présentée dans [Schweikert 79]. Des travaux récents [Fiduccia 82] présentent un algorithme itératif pour le placement des cellules des circuits intégrés basé sur la technique de Kernighan dont le temps d' éxécution dans le pire cas varie linéairement avec la taille du circuit. Finalement Krishnamurthy dans [Krishnamurthy 84] généralise les idées introduites par Fiduccia.

Dans les paragraphes suivants nous présenterons l' application du bi-partitionnement au problème du placement des circuits électriques ainsi qu' une heuristique pour ce bi-partitionnement (présentée dans [Fiduccia 82]). Puis une application de cette heuristique à la segmentation des circuits pour le test pseudo-exhaustif sera présentée.

3.3.2.2 Procédure Heuristique De Bi-partitionnement

3.3.2.2.1 Présentation Du Problème

Le problème du bipartitionnement des circuits électriques, en vue du placement des cellules (composants), peut être formulé de la manière suivante:

Soit un circuit constitué d' un ensemble de cellules (composants) connectées par un ensemble de signaux (équipotentielles). Le bi-partionnement consiste à partitionner l'ensemble des cellules en deux blocs tels que le nombre de signaux connectant des cellules situées dans les deux blocs soient minimal.

Pratiquement seules les partitions satisfaisant une condition basée sur la cardinalité des deux blocs sont acceptées, sinon la solution du problème serait constituée d'un bloc contenant

toutes les cellules et d' un bloc vide.

3.3.2.2.2 Notations

- Dans tout ce qui suit, on supposera que toutes les cellules et signaux sont de poids identiques.

- Pour une partition (A,B) d' un circuit, un signal est dit coupé s'il a au moins un terminal (cellule) dans chaque bloc.

- L' ensemble des signaux coupés est appelé coupe de la partition.

- Une cellule est dite "fixe" lorsqu'elle ne peut changer de bloc.

- Soient un entier r (appelé rapport), tel que: $0 < r < 1$ et une partition (A,B) du circuit. On dira que la partition est équilibrée pour le rapport r si et seulement si:

$$|A| / (|A| + |B|) \tilde{-} r$$

|A| et |B| etant les cardinaux (nombre de cellules) des blocs A et B.

3.3.2.2.3 Description

Nous pouvons maintenant redéfinir plus formellement le problème du bipartitionnement d' un circuit électrique:

Soient un ensemble de cellules connectées par un ensemble de signaux et un rapport r. Le bi-partionnement consiste à diviser l' ensemble des composants en deux blocs tels que la partition

résultante soit équilibrée pour le rapport r et que le cardinal de la coupe soit minimal.

En fait pour le placement des cellules d'un circuit electrique, la définition ci-dessus n'est pas suffisante car les cellules ne sont pas toutes équivalentes (par exemple les plots se placent différemment des autres cellules) et les signaux ont des poids différents (un signal d'horloge distribué dans tout le circuit par exemple). Il convient alors de définir plusieurs rapports, plusieurs équilibres et une minimisation du cardinal de la coupe pondéré par les poids des signaux (mesure appelée taille). Mais ces détails nous entraineraient trop loin pour ce tour d'horizon rapide.

L' idée de base de la procédure proposée par Fiduccia est de déplacer une seule cellule à la fois (la cellule de base) dans le but de minimiser la taille de la coupe de la partition finale.

La cellule de base est choisie en fonction de l'influence sur la coupe de son passage d'un bloc à l'autre. La diminution du cardinal de la coupe lorsque la cellule i change de bloc est appelée gain du composant i et est noté g(i). g(i) est un nombre entier compris entre -p(i) et p(i) (où p(i) est le nombre de signaux connectant la cellule i). A noter que le gain peut être négatif.

Un critère lié à l'équilibre de la partition est utilisé pour déterminer le bloc dans lequel la cellule de plus haut gain est sélectionnée. Si le gain du composant de base est négatif (son échange fait augmenter la taille de la coupe), l'échange est tout de même effectué, en espérant que ces échanges,

d'apparence contraire au but recherché, permettront plus tard à l' algorithme de sortir de minima locaux potentiels.

Afin d' éviter d'éventuelles boucles dans le processus, la cellule de base est figée dans son nouveau bloc, aussitôt après son deplacement. Seules les cellules "libres" (non fixes dès le départ ou non figées après un déplacement) peuvent être échangées.

L' algorithme s' arrête lorsque toutes les cellules sont figées ou fixes, ou lorsque la condition liée à l'équilibre de la partition ne permet plus d' échange.

Lorsque tous les échanges possibles ont été effectués, la partition finale est choisie parmi toutes les partitions crées au cours du déroulement de l'algorithme: la partition dont la coupe a été de taille minimum est sélectionnée. L' algorithme peut ensuite être répété en prenant comme nouvelle partition initiale, la partition finale choisie. Ce processus itératif continue jusqu' à ce que la taille de la coupe ne diminue plus. Les résultats ont montré que 2 ou 3 passes sont généralement suffisantes.

Tableau 3-6: Organigramme du bi-partionnement

```
Entree de la partition initiale et du rapport ;
Entree de la liste d' interconnexions ;
Tant que la taille de la coupe diminue faire {
     Initialiser les tailles des segments ;
     Initialiser les gains des cellules ;
     Tant qu'il existe une cellule de base faire {
          Choisir la cellule de base ;
          Faire passer la cellule de base d'un segment
          a l'autre;
          Figer la cellule de base dans son nouveau segment ;
          Corriger les gains des composants ;
     }
     Choisir comme partition finale la partition de taille d
     coupe minimale;
}
Sortie de la partition finale ;
```

3.4 Bi-Segmentation

La stratégie de [Fiduccia 82] décrite précedemment a été adaptée ici pour la segmentation en vue du test pseudo-exhaustif. La procédure résultante procède aussi par affinements successifs: elle effectue itérativement des coupes dans le circuit, jusqu'à ce que la fonction coût devienne inférieure à une borne B donnée.

3.4.1 Définitions Et Notations

- Dans la partie combinatoire d'un circuit, il existe une correspondance unique entre les noeuds et les portes logiques. Le même nom est utilisé pour désigner le noeud ou la porte logique dont il est issu. Dans ce qui suit nous ne distinguerons pas entre un noeud et la porte logique dont ce noeud est l'unique sortie.

- Un noeud m est un dépendant direct d'un noeud n si et seulement si n est une entrée de m.

- Un noeud m est dépendant d'un noeud n si et seulement si:
  * m est un dépendant direct du noeud n, ou
  * il existe au moins un noeud y tel que m soit un dépendant direct du noeud y et y soit dépendant de n

- Le cone de dépendance d'un noeud n est l'ensemble de tous les noeuds dépendants du noeud n

- Dans la procédure de segmentation, le circuit initial est décomposé en deux segments (ou blocs). Puis chaque bloc est à son tour coupé en deux, et l'itération est répétée jusqu'à ce que le coût devienne inférieur à une borne prédéfinie. La coupe de deux blocs est l'ensemble des noeuds reliant des portes de l'un des blocs à l'autre bloc.

- Un circuit à plusieurs sorties sera considéré comme la combinaison des cones de dépendance originaires de chaque sortie primaire. La procédure de segmentation coupe le cone de dépendance d'une sortie primaire (le circuit entier pour les circuits mono-sortie) en sous-cones, un cone à la fois.

3.4.2 Description De La Procédure

Pour fractionner un cone de dépendance à une seule sortie (le cone originel), on part d'une segmentation initiale (A,B), où A est défini comme le segment qui contient la sortie OUT et B comme le complément de A dans le cone originel. On notera Si la coupe de (A,B). Le segment A peut être considéré comme un cone dont la base sont des entrées primaires du cone originel ou des pseudo-entrées, éléments de la coupe (A,B). Le segment B est une collection de sous-cones, dont les entrées sont des entrées primaires du cone originel et la sortie de chaque sous-cone est un élément de la coupe (A,B). Cette définition de A et B est illustrée par le schéma de la figure 3-18. Dans cette figure, le cone originel, de sortie OUT1 est segmenté en deux segments A et B, où A contient les portes 3 et 4, et B contient les sous-cones d'origine respective a1 et a2, qui contiennent respectivement la porte 1 et la porte 2 . La coupe de (A,B) est ici {a1,a2}.

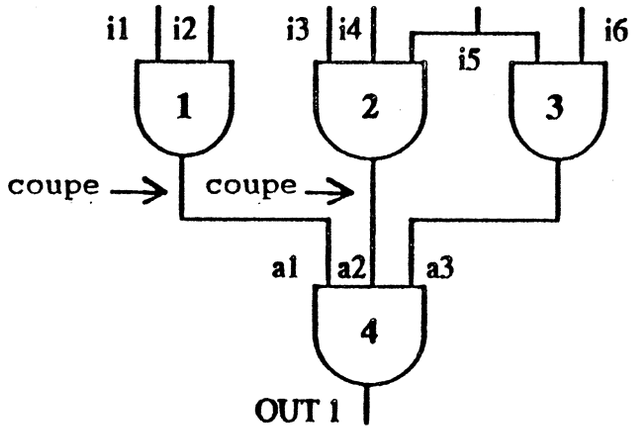


Figure 3-18: Exemple de segmentation

A = {3,4}	B = {1,2}
entrees: {a1,a2,i5,i6}	entrees: {i1,i2,i3,i4,i5}
sortie: {OUT1}	sortie des cones: {a1,a2}

Un test pseudo-exhaustif associé à la segmentation (A,B) est la réunion du test exhaustif de A et des tests exhaustifs de chacun des sous-cones constituant B. Dans l'exemple de la figure 3-18, le test pseudo-exhaustif associé à la segmentation (A,B) est la réunion du test exhaustif de A et du test pseudo-exhaustif de B. Le test exhaustif de A, un cone à 4 entrées est obtenu avec 16 vecteurs. Le test pseudo-exhaustif de B est la réunion des tests exhaustifs des deux cones de sorties respectives a1 et a2. Ces cones ont respectivement deux et trois entrées, et chacun nécessite donc respectivement 4 et 8 vecteurs pour un test exhaustif. La longueur du test pseudo-exhaustif de B est donc 4 + 8 = 12. Finalement, la longueur du test pseudo-exhaustif du circuit pour cette segmentation (A,B) est 16 + 12 = 28.

La fonction coût associée à une segmentation (A,B) est:

$$C = 2^{Ni} + \sum_{\text{coupe}} 2^{Ni}$$

où Ni est le nombre total d'entrées de A, c'est à dire le nombre d'entrées primaires dans A plus le nombre de noeuds dans la coupe (A,B).
Ni est, pour chaque noeud Si de la coupe (A,B), le nombre d'entrées primaires dans le cone de dependance de Si.

Remarque: Dans cette définition du coût d'un test pseudo-exhaustif, seule la longueur du test final apparait explicitement comme un facteur de coût. La difficulté de creer des chemins de contrôle et d'observation pour les entrées et les sorties des segments est aussi un facteur de coût important. Alors que la longueur du test influence le coût direct du test (la durée du test sur une machine de prix élevé), la difficulté de contrôler et d'observer les accès internes des segments influence directement le coût de la génération de la sequence de test. Comme nous le verrons dans le chapitre suivant, ce deuxième facteur de coût est implicitement inclus dans les règles heuristiques utilisées dans le programme qui implémente la procédure de segmentation.

Le problème de la segmentation est résolu à l'aide d'une procédure dérivée de la procédure de coupe minimale décrite dans le chapitre précédent. Cette procédure minimise à chaque étape la fonction coût, définie précédemment, en modifiant les segments A et B de facon à diminuer le coût C(S) à chaque pas.
Definitions:

(a) Le point de départ est une segmentation arbitraire du circuit {A,B}, la segmentation initiale.

(b) Une étape de la segmentation est définie comme étant l'état de la segmentation (A,B) entre deux déplacements de cellules.

(c) Pour chaque cellule ou porte x, le gain G(x) à une étape donnée est la différence entre la valeur de la fonction coût C avant de faire passer x de son segment actuel (A ou B) vers le segment opposé (B ou A) et la valeur de la fonction C après ce changement.
C'est à dire:

G(x) = coût (avant changement de x) - coût (après changement de x)

(d) Le meilleur changement à chaque étape est celui qui conduit au gain le plus élevé, c'est à dire la diminution du coût la plus grande. La cellule choisie pour passer d'un segment à l'autre à chacune des étapes de la procédure est appelée la cellule de base. La cellule de base est choisie parmi les cellules de sortance minimale permettant le meilleur changement à chaque étape. Cette procédure est décrite dans le chapitre suivant.

(e) Dans le cas envisagé ici, les cellules sont toutes des portes logiques à une seule sortie. Nous emploierons donc dans ce qui suit indifféremment les termes cellule ou porte.

(f) Dans un circuit combinatoire contenant des points à sortance multiple, une porte logique peut appartenir à plusieurs cônes de dépendance disjoints. De ce fait, une porte logique peut appartenir à la fois au segment A et au segment B si elle appartient à la fois au cone A et à l'un des cones issus de la coupe (A,B). Cette possibilité (qui n'existe pas dans le cas d'un partitionnement) est illustrée par la figure 3-19. De telles cellules sont dites appartenir à une zone de bi-appartenance ("zone grise") et ne sont pas candidates pour le choix de la cellule de base.

(g) La procédure itérative peut maintenant être définie comme suit: partant de la segmentation initiale, une cellule (la cellule de base) change de segment à chaque étape. Lorsque la cellule de base change de segment, son cone de dépendance spécifique change de segment avec elle. Les cellules qui, grâce à ce changement quittent la zone de bi-appartenance, passent aussi dans le nouveau segment. Toutes les cellules arrivant dans un nouveau segment sont "figées" dans ce segment. Pour éviter les oscillations, une cellule figée ne peut plus changer de segment. Si à une étape de la procédure tous les gains des cellules sont négatifs, une cellule de base est tout de même choisie et change de segment pour permettre à la procédure de sortir du minimum local. Lorsque plus aucun changement n'est possible, c'est à dire toutes les cellules sont fixes ou figées dans un segment ou se trouvent dans une zone de bi-appartenance, le

circuit est ramené dans l'état de la segmentation qui résultait en un coût minimum. Des passages additionels de la procédure peuvent être effectués sur les segments résultants, jusqu'à ce que le coût final devienne inférieur au maximum donné.

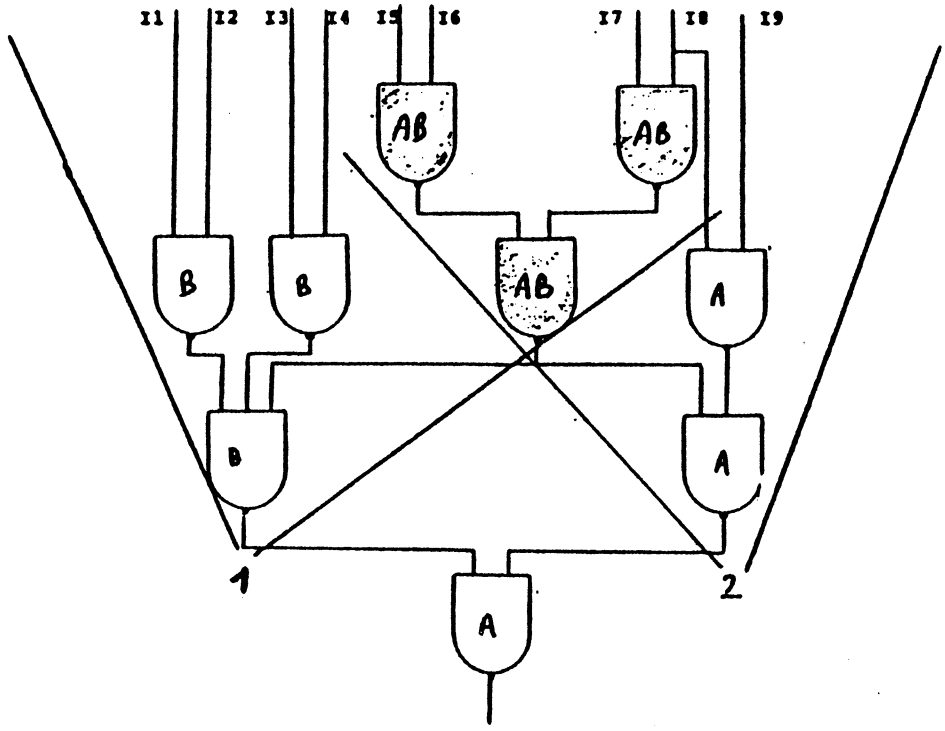


Figure 3-19: Une segmentation peut comprendre une zone de bi-appartenance ("zone grise") L'appartenance de chaque cellule est indiquée par A,B ou AB (pour la zone grise).

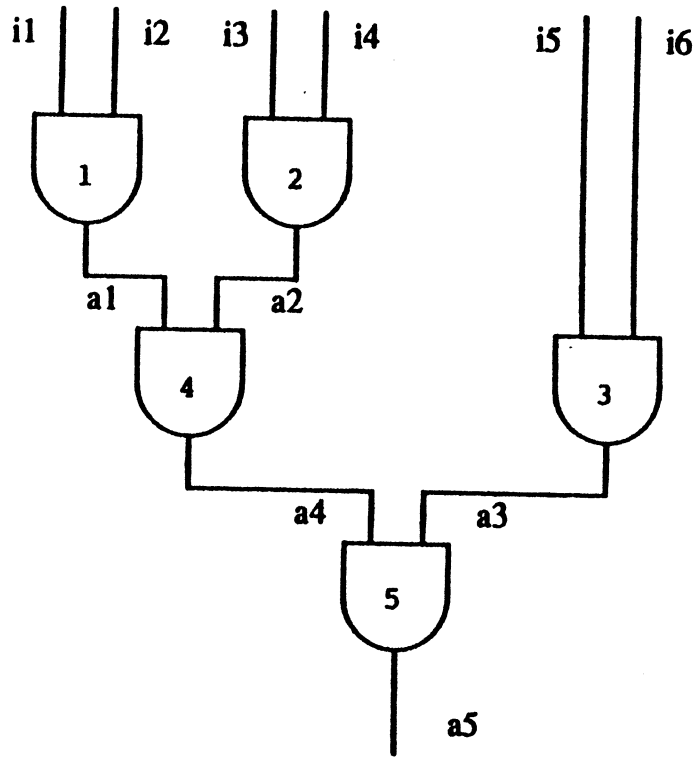
3.4.3 Organigramme De La Segmentation

Nous donnons ici un organigramme de la procédure de segmentation:

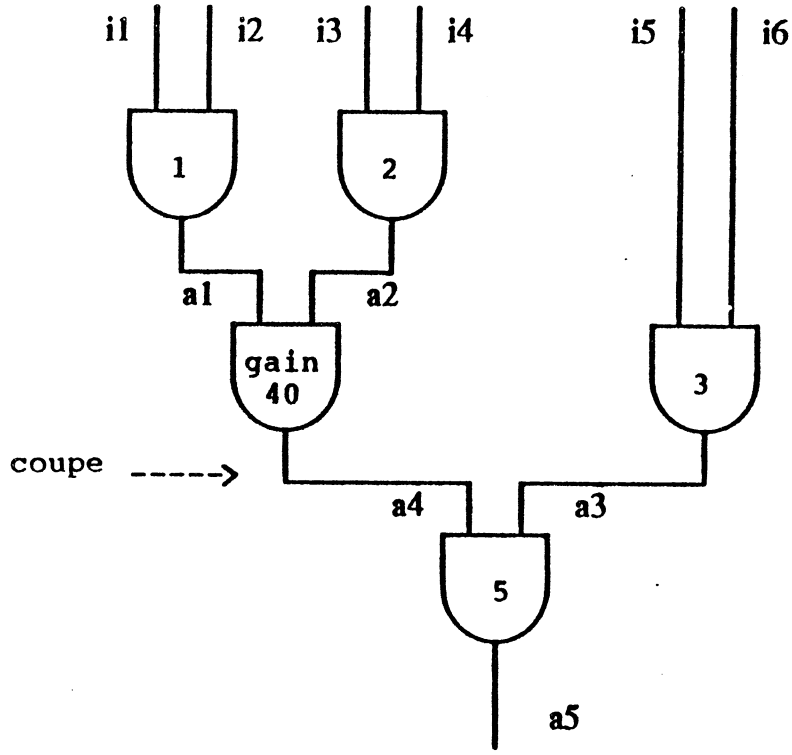
```
Entrer la description du circuit, de la segmentation
initiale et de la borne B;
Tant que le coût diminue faire {
        Initialiser le coût et la coupe
        de la segmentation;
        Initialiser les cones des cellules;
        Initialiser les gains des cellules;
        Tant qu'il existe une cellule libre faire {
             Choisir la cellule de base;
             Déplacer la cellule de base de son segment
             courant vers le segment opposé;
             Figer la cellule et son cone dans le
             nouveau segment;
             Réviser les gains des autres cellules;
        }
        Choisir comme nouvelle segmentation initiale
        la segmentation de coût minimal;
}
Sortir la segmentation finale;
```

3.4.4 Exemple De Segmentation

On considère le circuit à une sortie primaire, constitué de portes logiques à deux entrées, illustré ci-dessous:

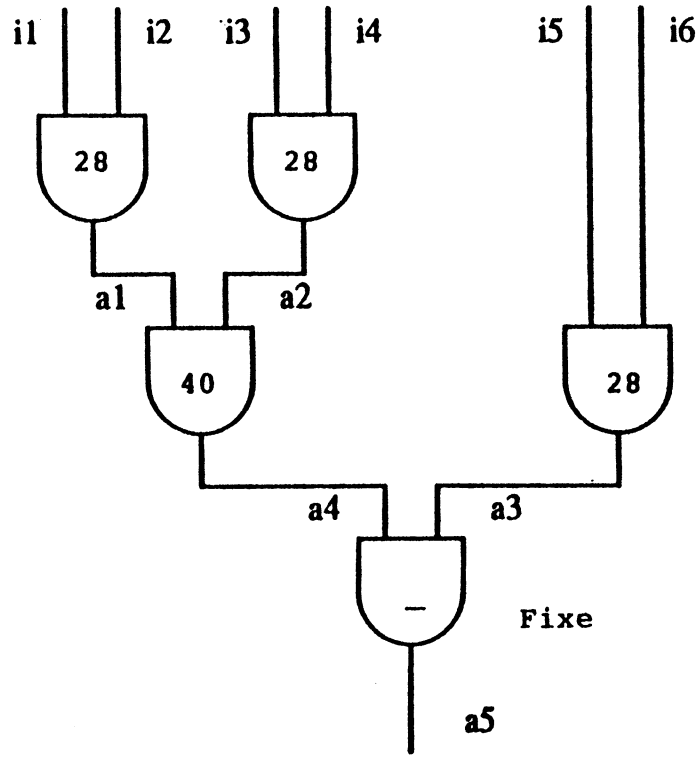


La segmentation initiale est la suivante: B, et donc la coupe (A,B) sont vides, et A est le cone {1,2,3,4,5} dont les 6 entrées sont {i1,i2,i3,i4, i5,i6} et la sortie {a5}. Le test pseudo-exhaustif associé à cette segmentation est aussi le test exhaustif de A. Le coût initial est donc 64.



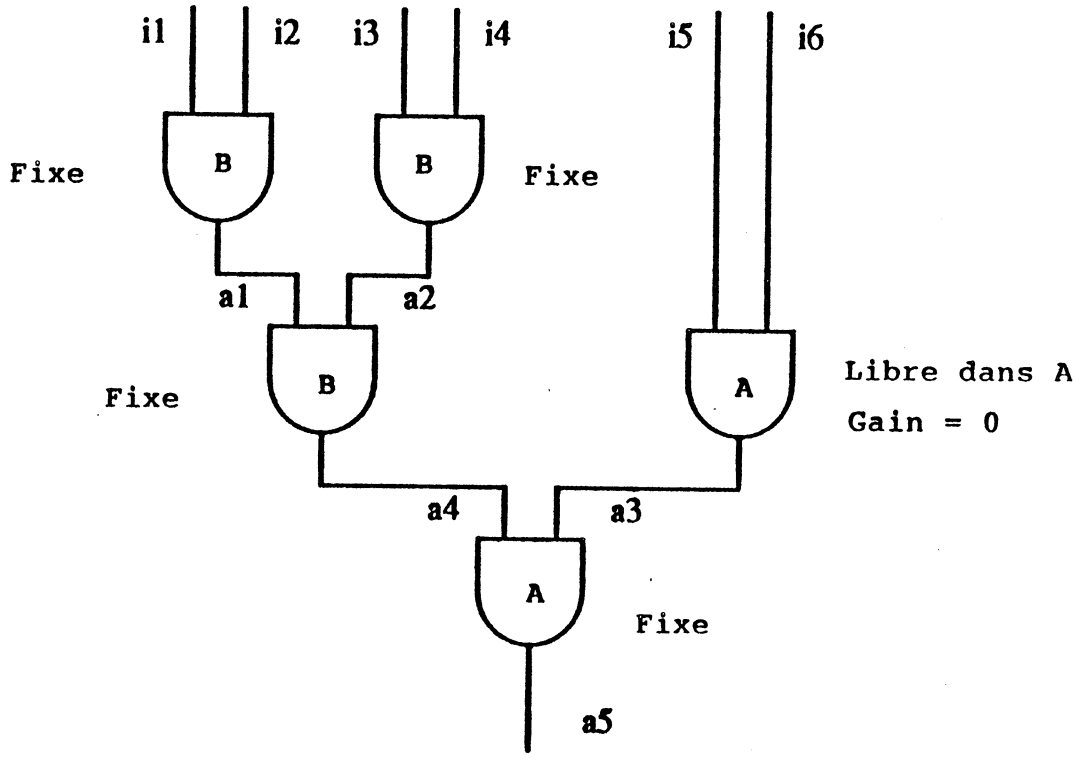
On calcule pour chaque cellule (ici des portes logiques), le gain pour cette étape, défini précédemment comme le gain en coût de test pseudo-exhaustif obtenu avec le passage de la cellule de son présent segment vers le segment opposé. Par exemple, transférer la porte 4 du bloc A vers le bloc B (ainsi que les portes de son cone de dépendance {1,2}) résulte en un nouveau bloc B: le cone à 4 entrées i1,i2,i3,i4 et de sortie a4. Le test de ce bloc nécessite donc 16 vecteurs. La nouvelle coupe de (A,B) est le singleton {a4}. Le nouveau bloc A est le cone {3,5} dont la sortie est a5 et les entrées sont a4,i5 et i6. Le test exhaustif du nouveau bloc A est donc de longueur 8. Le coût du test de la nouvelle segmentation est donc 16 + 8 = 24. L'ancien coût étant de 64, le gain associé à la porte 4 est donc 64 - 24 = 40.

Le nombre entier contenu dans chaque cellule libre du schema ci-dessous indique le gain de la cellule.

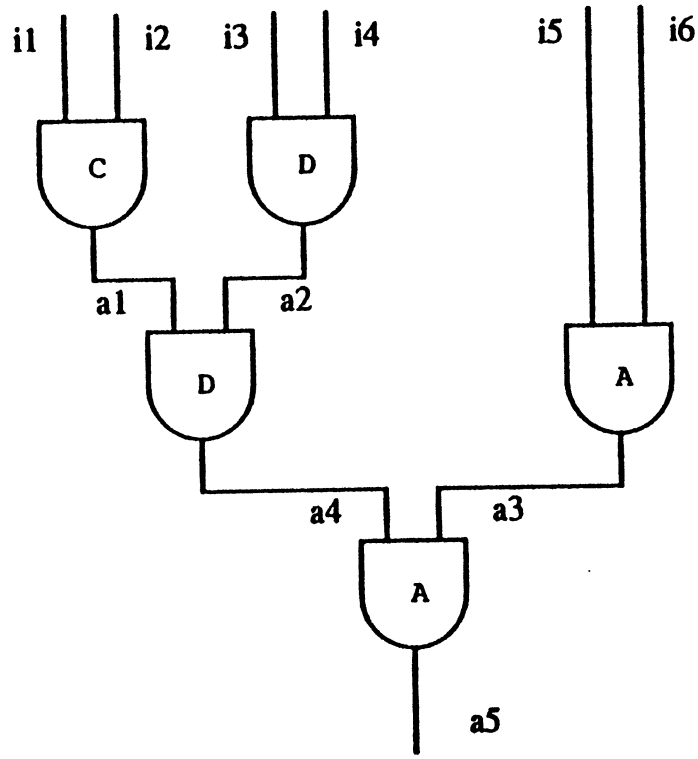


Le gain est calculé de la même manière pour chaque cellule libre. La porte 5 est, par définition de A, fixe dans A et ne peut passer dans B: son gain n'est donc pas calculé. La porte 4 a le gain maximum parmi les cellules libres et est choisie comme la cellule de base. A la fin de la première étape, la porte 4 et les portes de son cone de dépendance passent du bloc A vers le bloc B et deviennent figées dans B.

La figure suivante montre la segmentation à la fin de la deuxième étape: l'appartenance de chaque porte est indiquée dans la porte elle-même.



Au début de la deuxième étape, la porte 3 est la seule porte libre. Son gain est nul. Puisque elle a le gain le plus grand, c'est la cellule de base choisie pour cette étape. Lorsque la porte 3 change de bloc (passant de A vers B), le nouveau coût est toujours 24.



A la fin de la deuxième étape, il n'y a plus de cellules libres. Cela termine la passe de l'algorithme. L'étape 1 et l'étape 2 resultent en un coût identique, le coût minimum puisque inférieur au coût initial. L'étape 1 est choisie puisqu'elle résulte en une segmentation plus équilibrée que pour l'étape 2. Le coût final est donc 24. Si ce coût est encore trop important, une étape additionnelle peut être réalisée en partant de l'un des segments A ou B. En partant du bloc B, la segmentation de B en deux blocs C et D, comme illustré ci-dessus, conduit à une segmentation finale en 3 blocs (ici disjoints) avec un coût final de 20.

3.5 Implémentation De La Procédure

3.5.1 Structure Des Données

La procédure de partitionnement a été implementée en FORTRAN 77. La description détaillée du programme P.E.T.. correspondant est donnée en Annexe 4.

3.5.2 Synopsis Du Programme

```
     PROGRAMME P.E.T.
*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*
*      Segmentation pour le Test Pseudo-Exhaustif Fonctionnel  *
*                                                              *
*      VERSION ORIGINELLE :  4 Fevrier  1984                   *
*      VERSION 1-C        :  5 Decembre 1984                   *
*      VERSION VAX/VMS: FORTRAN 77 ; INTEGER*2                 *
*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*
```

Le programme est consistué des étapes suivantes:

- Pré-traitement du circuit et initialisation des tables

  Le pré-traitement du circuit est effectué par une procédure indépendante (appelée COMPIL) qui "compile" une liste d'interconnections donnée, produisant pour chaque noeud des pointeurs permettant d'accéder au nom, type, liste des entrées et liste des sorties de ce noeud.

- Partitionnement du circuit en couches logiques

  Les noeuds du circuit sont ordonnés en fonction de leur profondeur logique, appelé couche logique. Les entrées primaires sont par définition dans la couche zéro. Les couches des autres noeuds sont déterminées itérativement comme étant la valeur de couche maximum des entrées du noeud plus un.

- Evaluation de la dépendance des sorties primaires par rapport aux entrées primaires.

Pour chaque noeud du circuit, un vecteur de dépendance de ce noeud par rapport aux entrées primaires est évalué. Chaque bit de ce vecteur est une valeur binaire (0 ou 1): un 0 pour le bit i indique que le noeud ne dépend pas de l'entrée primaire # i, un 1 pour le bit i indique que le noeud dépend de l'entrée primaire # i. Les vecteurs sont initialisés de manière triviale pour les entrées primaires. Puis les vecteurs des autres noeuds sont déduits, en procédant par couche logique croissante.

- Partitionnement du circuit

La procédure heuristique de partitionnement décrite au chapitre précédent est utilisée ici pour réaliser le partitionnement du circuit pour le test pseudo-exhaustif par segments.

Ces étapes successives sont réalisées par des appels aux sous-programmes suivants:

1. Chercher la description du circuit pré-traité dans le fichier "circuit".SAW

```
call RDCOM(filnam,iersta)
```

2. Initialiser les tables

```
call INIT
```

3. Structurer le circuit par couches logiques (vérifier ainsi l'absence de rebouclages)

```
call LVLIZ

call CHECK (nogo)
```

4. Evaluer la dépendance des noeuds du circuit par rapport aux entrées primaires

```
call DEPEND
```

5. Chercher la liste des sorties primaires dans le fichier "circuit".SIM

```
call RDOUT
```

6. Imprimer la matrice de dépendance des sorties primaires D(N)

```
call PRTDN
```

7. Réaliser le partitionnement des cones de dépendance des sorties primaires (procédure "mincut")

```
call SEGMNT
```

3.5.3 Implémentation De La Bi-segmentation

Les différents ensembles (les segments A et B, la coupe, les frontières) sont initialisés, ainsi que les listes de gains. Pour évaluer le gain d'une cellule, la procédure suivante est utilisée (subroutine FNGAN1):

- Evaluer la nouvelle coupe A/B obtenue lorsque une cellule n est changée de bloc (subroutine NWCUST.) Si cet échange résulte en un changement dans la coupe A/B, un gain potentiel existe pour la somme des longueurs des tests exhaustifs des cones de dépendance de chaque noeud coupé. Ce gain potentiel est:

$$deltai = \sum_{\text{ancienne coupe}} 2^{NSi} - \sum_{\text{nouvelle coupe}} 2^{NSi}$$

  Où NSi est le nombre d'entrées primaires dont le noeud coupé Si dépend.

- Evaluer le nouvel ensemble d'entrées primaires entrant directement dans A lorsque la cellule n change de bloc (subroutine NWIA). Ce changement peut entrainer un gain lors du test de A, à cause de la modification potentielle du nombre d'entrées de A. Ce gain potentiel est:

$$delta2 = 2^{OIA + OCU} - 2^{NIA + NCU}$$

  avec: OIA l'ancien nombre d'EPs entrant directement dans A.

  OCU l'ancien nombre de noeuds dans la coupe A/B

  NIA le nouveau nombre d'EPs entrant directement dans A.

  NCU le nouveau nombre de noeuds dans la coupe A/B

- Ajouter les gains partiels deltal et delta2 pour trouver le gain total.

Modifier en conséquence la liste des valeurs de gains (subroutine MODGAN). Lorsque les gains de toutes les cellules libres ont été mis à jour, la cellule de base pour le pas suivant est choisie, en suivant les règles de sélection suivantes (subroutine BASCEL):

- REGLE 1

Choisir la cellule de base parmi les cellules libres de plus fort gain et parmi celles ci, choisir la cellule de sortance minimale (ce deuxième critère minimise le nombre d'E/S des blocs qui ne sont pas primaires, et minimise ainsi le coût de sensibilisation) : call FNLIS(i,list,mj)

- REGLE 2

S'il existe plusieurs candidats sélectionnés par la première règle, sélectionner, s'il en existe, les noeuds N dont les cones de dépendance n'ont pas de point de divergence (c'est à dire, tels que le cone de N soit un arbre de racine N).

- REGLE 3

S'il n'existe pas de tels noeuds, ou s'il subsiste plusieurs candidats satisfaisant à la deuxième règle, parmi ceux-ci, le noeud situé dans la couche logique la plus haute est alors choisi (ce qui correspond à la sensibilisation de la sortie du bloc la plus facile).

Lorsqu'une cellule X est changée de bloc, toutes les cellules dans son cone de dépendance changent de bloc avec elle. La cellule et son cone sont alors fixés dans leur nouveau bloc (subroutine CONLOK). Une exception existe lorsque une cellule Y dans le cone de dépendance de X appartient aussi au cone d'une autre cellule Z, qui ne se trouve pas dans le même cone que X. Dans ce cas, la cellule Y appartient aux deux blocs (il s'agit donc plus d'une "segmentation" que d'un "partitionnement" au sens mathématique du terme). La cellule Y entre alors dans ce que nous définirons comme la "zone grise". La cellule Y ne sortira de la zone grise que si toutes les cellules issues de Y sont à nouveau dans le même bloc. Les cellules de la zone grise sont fixées dans cette zone, et, par conséquent, leur gain n'est pas calculé.

Le coût courant du test pseudo-exhaustif (correspondant à la partition courante) et le coût du meilleur pas (c'est à dire le pas de la procédure pour lequel le coût du test est minimum - en cas d'ex-aequo le pas conduisant à la partition la plus équilibrée est choisi) sont mis à jour (subroutine MNCOST). Un nouveau pas est effectué, et ainsi de suite, jusqu'à ce qu'il n'y ait plus de cellules libres. A la fin de la passe, toutes les cellules qui ont été changées de bloc après le pas optimal, sont remises dans leur bloc initial (subroutine BESTPN) et la partition optimale est sortie dans le fichier "circuit".PAR (subroutine DISPAR). La procédure est alors répétée pour les blocs résultants, si l'opérateur le spécifie.

3.5.4 Format De Sortie

Le programme génère un fichier, "circuit".PAR, qui décrit la partition finale après chaque passe. Le tableau 3-7 donne un exemple de ce fichier de sortie ".PAR".

Tableau 3-7: Exemple de fichier de sortie de segmentation finale.

Description du test pseudo-exhaustif	Longueur du test:
Test pseudo-Exhaustive du segment A d'entrees: N1 N4 N2 NA1	16
Test exhaustif du cone d'origine N1 I1 I2 I6	8
Test exhaustif du cone d'origine N4 I1 I3 I4 I5	16
Test exhaustif du cone d'origine N2 I3 I4 I5	8
Test exhaustif du cone d'origine NA1 I3 I5 I6	8
Longueur du test final :	56
Longueur du test initial :	64

Un fichier supplémentaire est créé: "filnam".TET. Ce fichier contient des informations détaillées sur chaque pas de la procédure de partionnement. On y trouve en particulier les valeurs des gains de chaque cellule libre à chaque pas, la cellule de base à chaque pas, la coupe, les entrées de chaque bloc et l'appartenance des cellules aux différents blocs à chaque pas.

4 RESULTATS PRATIQUES

4.1 Performances

Le programme a été utilisé pour réaliser le partitionnement de plusieurs circuits combinatoires afin de générer un test pseudo-exhaustif. Les pages qui suivent exposent les résultats obtenus avec 2 circuits.

Le premier circuit utilisé comme illustration (voir figure 3-22) est un circuit de faible complexité, qui a la particularité d'être un pire cas pour le "random-testing" [Shedletsky 77]. En fait ce circuit est aussi un pire cas pour le test pseudo-exhaustif par segments, et les résultats de la segmentation sont donnés dans les tableaux 3-8, 3-9, 3-10. En superposant les segments résultants qui sont identiques entre ces 3 tableaux, la longueur du test pseudo-exhaustif final est ramenée à 56 (au lieu de 64 pour la longueur du test exhaustif). Ce résultat est dû à la structure extrèmenent reconvergente du circuit, et au faible rapport du nombre d'entrées primaires sur le nombre de points de sortance.

/ = coupe

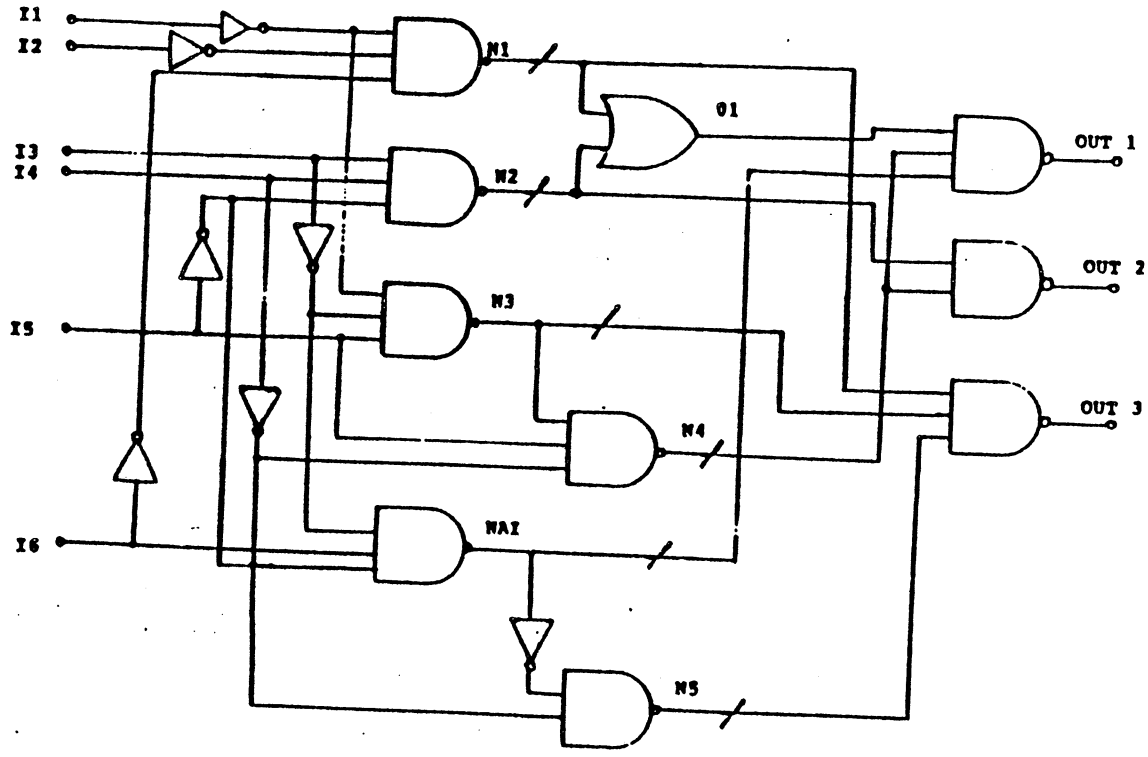


Figure 3-22: Exemple de segmentation

Tableau 3-8: Segmentation du cone d'origine OUTi

Description du test pseudo-exhaustif:	Longueur du test:
Test pseudo-exhaustif du segment A d'entrees: Ni N4 N2 NA1	16
Test exhaustif du cone d'origine N1 I1 I2 I6	8
Test exhaustif du cone d'origine N4 I1 I3 I4 I5	16
Test exhaustif du cone d'origine N2 I3 I4 I5	8
Test exhaustif du cone d'origine NA1 I3 I5 I6	8
Longueur du test final	56
Longueur du test initial	64

Tableau 3-9: Segmentation du cone d'origine OUT2

Description du test pseudo-exhaustif	Longueur du test:
Test pseudo-exhaustif du segment A d'entrees: I1 I3 I4 I5	16
Longueur du test final :	16
Longueur du test initial	16

Tableau 3-10: Segmentation du cone d'origine OUT3

Description du test pseudo-exhaustif	Longueur du test
Test pseudo-exhaustif du segment A d'entrees: Ni N5 N3	8
Test exhaustif du cone d'origine N1 I1 I2 I6	8
Test exhaustif du cone d'origine N5 I3 I4 I5 I6	16
Test exhaustif du cone d'origine N3 I1 I3 I5	8
Longueur du test final :	40
Longueur du test initial :	64

Le second exemple (voir figure 3-23) est le circuit LS74181 ULA/Générateur de fonction. Le programme, utilisé de manière itérative, permet d'obtenir une partition en 24 segments. La longeur du test est réduite de 16384 (test exhaustif) à 1076 (test exhaustif des segments en série). En fait, en utilisant les techniques de test pseudo-exhaustif de vérification [McCluskey 83] à l'ensemble de ces segments, le circuit est testé pseudo-exhaustivement (les segments sont testés en parallèle) avec 264 vecteurs (356 pour la segmentation obtenue à la main et donnée en figure 3-5-b et page 108).

Tableau 3-11: Segmentation du cone F3 (Passe Finale)

Description du test pseudo-exhaustif: Test:	Longueur du
Test exhaustif de la XOR F3: A5F3 A6F3	4
Test pseudo-exhaustif du cone A5F3:	76
Test pseudo-exhaustif du cone A6F3:	36
Test exhaustif parallèle des cones: BF0 BF1 BF2 BF3	16
Test exhaustif parallèle des cones: AF0 AF1 AF2 AF3	16

Longueur du test pseudo-exhaustif	148
Longueur du test exhaustif	16384

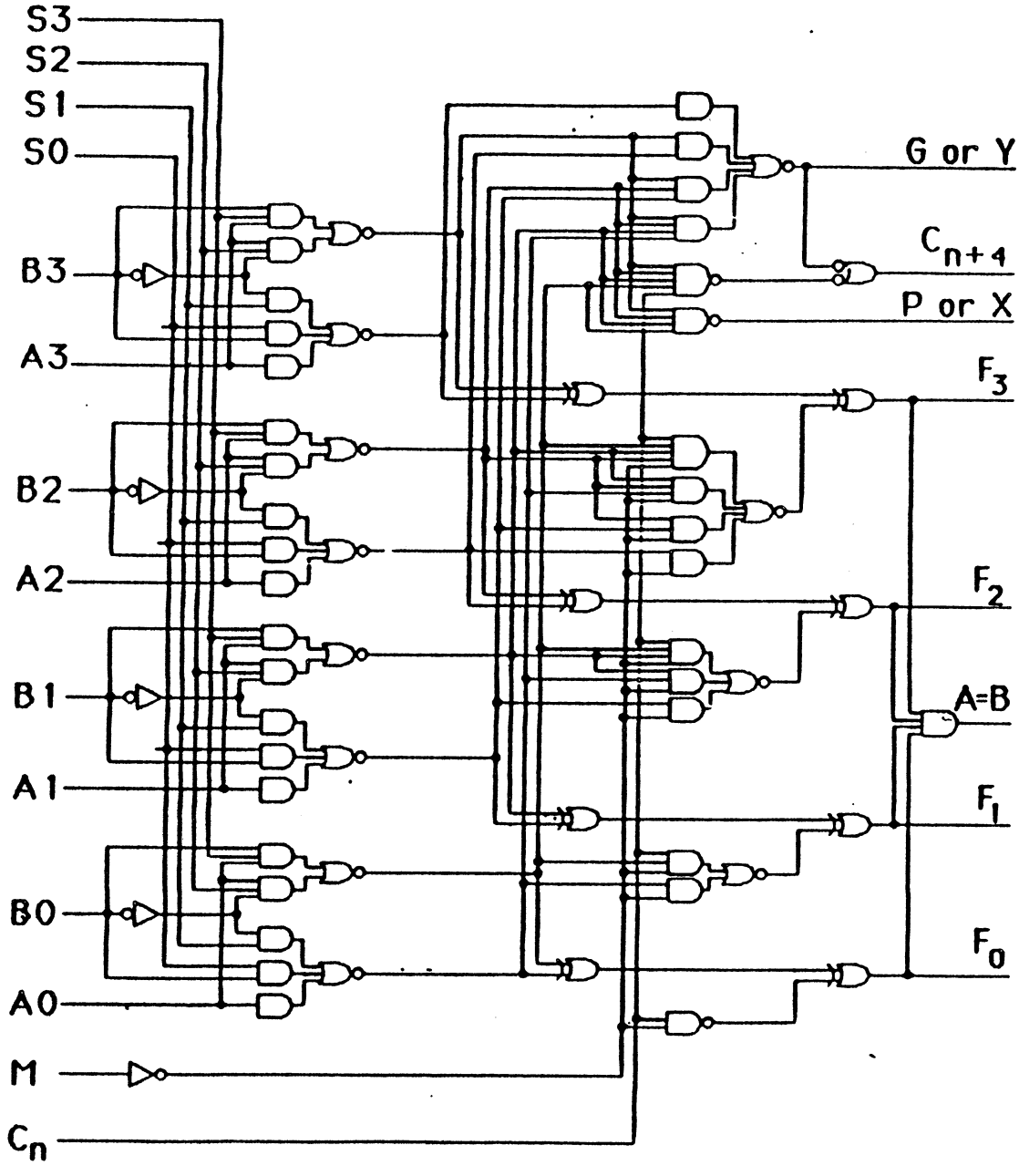


Figure 3-23: UAL 74181

Tableau 3-12: Segmentation du circuit LS748i (Segmentation Finale)

Description du test pseudo-exhaustif	Longueur du Test:
Test pseudo-exhaustif du segment F0: M CN A1F0	8
Test pseudo-exhaustif du segment F1: A3F1 A4F1	4
Test pseudo-exhaustif du segment F2: A4F2 A5F2	4
Test pseudo-exhaustif du segment F3: A5F3 A6F3	4
Test pseudo-exhaustif du segment X: AF3 AF2 AF1 AF0	16
Test pseudo-exhaustif du segment Y: BF0 BF3 AF3 BF2 BF1 AF1 AF2	128
Test pseudo-exhaustif du segment CN4: Y A6N	4
Test pseudo-exhaustif du segment AB: F0 F1 F2 F3	16
Test exhaustif du cone d'origine A6N AF0 AF1 AF3 AF3 CN	32
Test exhaustif du cone d'origine A1F0 AF0 BF0	4
Test exhaustif du cone d'origine A3F1 AF0 BF0 M CN	16
Test exhaustif du cone d'origine A4F1 AF1 BF1	4
Test exhaustif du cone d'origine A4F2 AF0 BF0 AF1 BF1 M CN	64
Test exhaustif du cone d'origine A5F2 AF2 BF2	4
Test exhaustif du cone d'origine A5F3 AF0 BF0 AF1 BF1 AF2 BF2 M CN	256
Test exhaustif du cone d'origine A6F3 AF3 BF3	4
Test exhaustif du cone d'origine BF3 A3 B3 S0 S1	16
Test exhaustif du cone d'origine AF3 A3 B3 S2 S3	16
Test exhaustif du cone d'origine BF2 A2 B2 S0 S1	16
Test exhaustif du cone d'origine AF2 A2 B2 S2 S3	16
Test exhaustif du cone d'origine BF1 A1 B1 S0 S1	16
Test exhaustif du cone d'origine AF1 A1 B1 S2 S3	16
Test exhaustif du cone d'origine AF0 A0 B0 S2 S3	16
Test exhaustif du cone d'origine BF0 A0 B0 S0 S1	16
Longueur du test P.E. (test en serie)	1076
Longueur initiale du test exhaustif	16384

/ = coupe

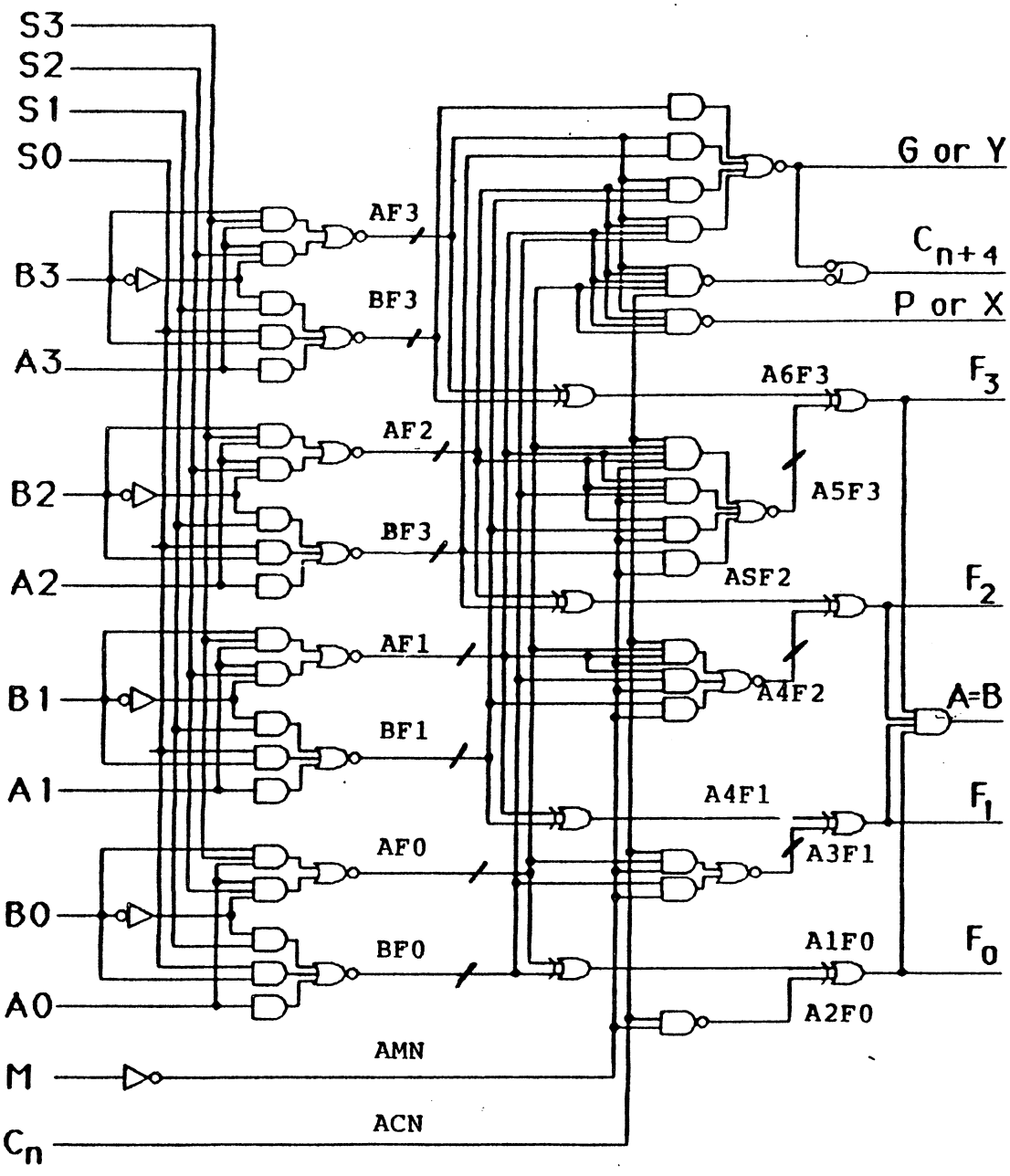


Figure 3-24: Segmentation de l'UAL 74181

4.2 Conclusion

Le test pseudo-exhaustif par segments est une technique de test pseudo-exhaustive qui garantit un très bon taux de couverture et élimine la nécessité d'utiliser un simulateur de fautes pour évaluer la performance des vecteurs de test, tout en réduisant considérablement l'effort associé à la génération de ces vecteurs de test.

Nous avons décrit une procédure heuristique, basée sur une technique de "mincut", qui permet de segmenter un circuit pour le test pseudo-exhaustif en un temps polynomial. Cette procédure a été implementée en Fortran-77 sur un VAX/750. Plusieurs circuits ont été segmentés pour le test pseudo-exhaustif par ce programme, et les résultats ont été décrits dans ce document.

Ces résultats peuvent être utilisés comme base d'une procédure plus générale qui permettrait de générer automatiquement les vecteurs de test pseudo-exhaustifs pour des circuits combinatoires ou des circuits qui ont été concus avec une technique de type "scan-path". Des travaux supplémentaires sont nécessaires pour atteindre ce but, en particulier afin de réaliser automatiquement la sensibilisation des entrées et sorties non-primaires des segments.

C O N C L U S I O N S

Deux méthodes de génération de vecteurs de tests pour le test fonctionnel des circuits intégrés digitaux ont été présentées.

La première méthode fait l'hypothèse de panne unique de type collage et génère un test qui détecte ces pannes. Cette méthode peut être considérée comme une extension des techniques d'analyse de controlabilité - observabilité, réalisant une analyse globale de la testabilité du circuit. La procédure décrite garde en particulier la caractéristique principale des programmes heuristiques d'analyse de testabilité: une grande rapidité d'exécution obtenue en évitant les retours en arrière. Pour des circuits de faible profondeur logique, l'algorithme garantit la génération d'un test pour toutes les pannes détectables du circuit. Pour les circuits de grande profondeur logique, le grand nombre de vecteurs générés pour chaque noeud entraine la sélection d'un sous-ensemble des vecteurs set, reset et moniteur. Nous avons exposé deux règles de sélection possible pour cette opération, et comparé leurs performances relatives. Une étude plus approfondie est nécessaire pour établir des règles de sélection "intelligentes" qui tiennent compte de la structure du circuit lors de la génération des vecteurs permettant le contrôle et l'observation des valeurs logiques des noeuds du circuit.

La deuxième méthode est une technique de test pseudo-exhaustif qui réalise, pour un ensemble de sous-circuits contenant collectivement le circuit considéré, le test exhaustif de chacun d'eux. Le taux de couverture des fautes est extrèmement élevé. En particulier toutes les pannes de type collage sont détectées ainsi que la plupart des pannes de type court-circuit qui sont détectables. Une procédure a été décrite pour réaliser la segmentation des circuits en vue du test pseudo-exhaustif. Cette procédure, basée sur une technique de "mincut", permet de segmenter un circuit pour le test pseudo-exhaustif en temps polynomial. Des travaux supplémentaires seront nécessaires pour permettre la génération automatique des vecteurs de test, en particulier créer les chemins de sensibilisation pour les entrées et les sorties des segments.

A N N E X E I

Les résultats détaillés de l'analyse de testabilité faite par le programme ARCOP (semblable à SCOAP) d'un circuit combinatoire sont donnés ci-après:

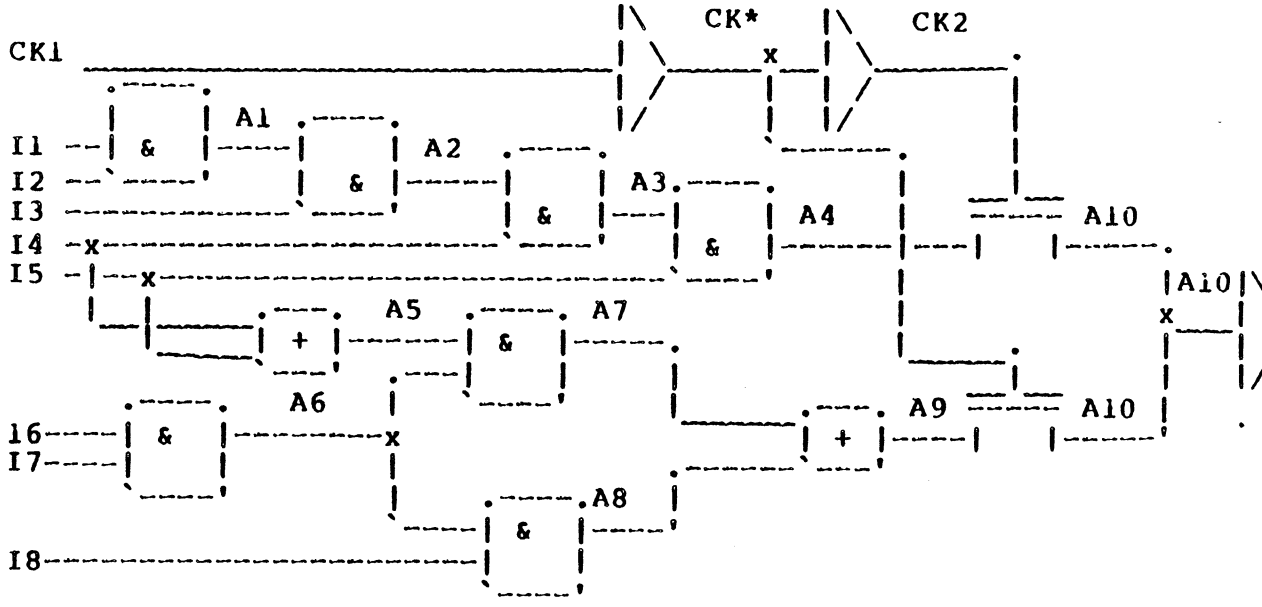


Figure A1-i: Description du circuit

Le programme donne tout d'abord les mesures de controlabilité à 0 et à 1 des noeuds du circuit:

```
 - ARCOP controllability and observability  analysis -

          **********PROGRAM ARCOP**********

     CC0 = COMBINATIONAL CONTROLLABILITY - ZERO
     CC1 = COMBINATIONAL CONTROLLABILITY - ONE
     SC0 = SEQUENTIAL CONTROLLABILITY - ZERO
     SC1 = SEQUENTIAL CONTROLLABILITY - ONE
```

NODE NAME	CC0	CC1	SC0	SC1
CK*	2	2	0	0
CK2	3	3	0	0
A1	2	3	0	0
A2	2	5	0	0
A3	2	7	0	0
A4	2	9	0	0
A5	2	3	0	0
A6	2	3	0	0
A7	3	7	0	0
A8	2	5	0	0
A9	6	6	0	0
A10	9	9	0	0
A10	6	13	0	0
A11	10	7	0	0
OUT	8	11	0	0
I1	1	1	0	0
I2	1	1	0	0
I3	1	1	0	0
I4	1	1	0	0
I5	1	1	0	0
I6	1	1	0	0
I7	1	1	0	0
I8	1	1	0	0
CK1	1	1	0	0

Puis le programme donne les mesures d'observabilité des noeuds du circuit:

**********PROGRAM ARCOP**********

CO = COMBINATIONAL OBSERVABILITY
SO = SEQUENTIAL OBSERVABILITY

NODE NAME	CO	SO
CK*	3	0
CK*-CK2	4	0
CK*-A10	3	0
CK2	3	0
A1	11	0
A2	9	0
A3	7	0
A4	5	0
A5	11	0
A6	10	0
A6-A7	11	0
A6-A8	10	0
A7	7	0
A8	8	0
A9	4	0
A10	2	0
A10	2	0
A11	1	0
OUT	0	0
I1	13	0
I2	13	0
I3	13	0
I4	13	0
I4-A3	13	0
I4-A5	13	0
I5	13	0
I5-A4	13	0
I5-A5	13	0
I6	12	0
I7	12	0
I8	12	0
CK1	4	0

Le programme donne ensuite la liste des noeuds qui paraissent difficiles à observer, c'est à dire dont la mesure d'observabilité, définie comme la moyenne des mesures CO plus 2 fois la variance des CO plus 1, est supérieure à la normale. Ces noeuds sont des candidats privilégiés pour l'insertion de points d'observation.

```
****************** PROGRAM ARCOP ********************

NORM = MEAN + 2*VARIANCE + 1

List of the nodes with higher observability than
the norm  n = 9  (for combinational circuits)
```

NODE NAME	CO	SO
A1	11	
A5	11	
A6	10	
I1	13	
I2	13	
I3	13	
I4	13	
I5	13	
I6	12	
I7	12	
I8	12	

Enfin le programme génère une liste de pannes de type collage unique et calcule une valeur de testabilité pour chacune des pannes en utilisant les mesures CC0, CC1 et CO comme décrit dans le chapitre 2-2:

fault number	element output name	fault origin	fault type	comb. test. meas.
1	CK*	CK1	INPUT S.A. 0	5
2	CK*	CK1	INPUT S.A. 1	5
3	CK2	CK*	INPUT S.A. 0	5
4	CK2	CK*	INPUT S.A. 1	5
5	A1	I1	INPUT S.A. 0	14
	A1	I2	INPUT S.A. 0	14
6	A1	I1	INPUT S.A. 1	14
7	A1	I2	INPUT S.A. 1	14
8	A2	A1	INPUT S.A. 0	14
	A2	I3	INPUT S.A. 0	14
9	A2	A1	INPUT S.A. 1	14
10	A2	I3	INPUT S.A. 1	14
11	A3	A2	INPUT S.A. 0	11
	A3	I4	INPUT S.A. 0	11
12	A3	A2	INPUT S.A. 1	14
13	A3	I4	INPUT S.A. 1	14
14	A4	A3	INPUT S.A. 0	9
	A4	I5	INPUT S.A. 0	9
15	A4	A3	INPUT S.A. 1	13
16	A4	I5	INPUT S.A. 1	13
17	A5	I4	INPUT S.A. 0	13
18	A5	I4	INPUT S.A. 1	13
	A5	I5	INPUT S.A. 1	13
19	A5	I5	INPUT S.A. 0	13
20	A6	I6	INPUT S.A. 0	13
	A6	I7	INPUT S.A. 0	14
21	A6	I6	INPUT S.A. 1	14
22	A6	I7	INPUT S.A. 1	14
23	A7	A5	INPUT S.A. 0	14
	A7	A6	INPUT S.A. 0	14
24	A7	A5	INPUT S.A. 1	14

(Suite...)

fault number	element output name	fault origin	fault type	comb. test. meas.	seq. test meas
25	A7	A6	INPUT S.A. 1	13	0
26	A8	I8	INPUT S.A. 0	13	0
	A8	A6	INPUT S.A. 0	13	0
27	A8	I8	INPUT S.A. 1	13	0
28	A8	A6	INPUT S.A. 1	13	0
29	A9	A7	INPUT S.A. 0	10	0
30	A9	A7	INPUT S.A. 1	14	0
	A9	A8	INPUT S.A. 1	14	0
31	A9	A8	INPUT S.A. 0	10	0
32	A10	CK*	ENABLE S.A.0	5	0
	A10	A10	INPUT S.OPEN	5	0
33	A10	CK*	ENABLE S.A.1	5	0
34	A10	A9	INPUT S.A. 0	10	0
35	A10	A9	INPUT S.A. 1	10	0
36	A10	CK2	ENABLE S.A.0	6	0
	A10	A10	INPUT S.OPEN	6	0
37	A10	CK2	ENABLE S.A.1	6	0
38	A10	A4	INPUT S.A. 0	7	0
39	A10	A4	INPUT S.A. 1	14	0
40	A11	A10	INPUT S.A. 0	8	0
41	A11	A10	INPUT S.A. 1	11	0
42	OUT	A11	INPUT S.A. 0	11	0
43	OUT	A11	INPUT S.A. 1	8	0

The average combinational
testability of the circuit is : 11
The average sequential
testability of the circuit is : 0

A N N E X E II

ANNEXE 2: Description du programme Reptil

1 STRUCTURE DU PROGRAMME

Le programme Reptil est constitué d'environ 8.000 lignes de C (dont environ 1500 lignes de commentair s) divisé en 91 modules dont la liste est donnée dans le Tableau A2-1. Le programme principal est structuré en 6 modules principaux:

- COMPIL le pre-processeur qui, transforme une description sous forme réseau prédiffusé du circuit (c'est-à-dire la liste des cellules utilisées) en une base de donnée sous forme de listes d'enregistrements et de pointeurs (un exemple de cette structure de données est donnée dans le tableau A2-2),
- LVLIZ qui effectue la structuration du circuit en couches logiques,
- CONTRL qui calcule les vecteurs de controlabilité,
- OBSERV qui calcule les vecteurs moniteurs,

- TSTPG qui génère la liste de pannes de collage unique circuit en utilisant des techniques d'équival structurelle et d'équivalence fonctionnelle, puis calcul compacte les vecteurs de test,

- SIMOUT qui sort les résultats dans deux fichiers: *.tst *.sm0.

Tableau A2-1: Liste des modules du programme

Module principal:

ATP.C;4

Pre-processeur:

CALCDLAY.C;3	CKLOD.C;3	CKMACRO.C;2	CMPERR.C;6
DELAY.C;1	DOMACRO.C;1	EQUIVNOD.C;4	EXPMACRO.C;1
FFLOP.C;3	GATE.C;1	GETEXPAN.C;3	GETINP.C;1
GETIOLIS.C;3	GETMACRO.C;3	GETSCANL.C;1	GT1OUTLS.C;2
GTMIOLIS.C;3	HILOTOLE.C;4	INNODE.C;7	IOLOOKUP.C;5
LOOKUP.C;4	MAKENODE.C;1	MAKEROM.C;24	MCKLOD.C;3
MDELAY.C;1	MDUMPIT.C;1	MEXPMACRO.C;7	MGATE.C;3
MGETEXPA.C;3	MINNODE.C;5	MKINPLST.C;1	MKINPWOR.C;8
MKMACRO.C;1	MKOUTLST.C;2	MKPO.C;1	MKPSEUDO.C;1
MMAKENOD.C;1	MMAKEROM.C;4	MTLOOKUP.C;5	NEWNAME.C;2
PRINTVER.C;1	PROCESS.C;1		

Generation des vecteurs:

ADDUNDET.C;1	CONTROL0.C;2	CONTROLI.C;1	CRFLTLST.C;2
DUMP.C;1	FLTCOLPS.C;1	GETEQUFL.C;1	GETFLTPT.C;1
JUNK.C;8	LEVELIZE.C;2	MKMFLTLS.C;1	MKSCANPA.C;1
MKTESTPA.C;1	OBSERVE.C;2	OBSERVEI.C;2	PATGEN.C;1
PIRISK.C;8	PODEPEND.C;10	PRINTFLT.C;1	PRTSIM.C;2
PSCONT.C;1	PSENOB.C;1	PSOBSERV.C;1	UTIL2.C;1
UTIL3.C;1	UTIL4.C;1	UTILS.C;5	

Utilitaires:

ATOFP.C;1	ATPERR.C;2	CHARPLUS.C;1	CMPERR.C;1
DBGIT.C;1	DEMUX.C;1	DUMPCONT.C;5	DUMPLEVE.C;1
DUMPMFLT.C;1	DUMPOBSE.C;3	FPTOA.C;1	GETINST.C;9
GETLINE.C;1	GETWORD.C;33	INVERT.C;1	ISAINPUT.C;1
ISAOUTPU.C;1	LODATOI.C;1	MDUMPIT.C;1	READDIGI.C;1
RELEASE.C;2			

Grand total of 4 directories, 91 files.

Tableau A2-2: Structure de données pour un noeud

1. Noeud:

```
struct snode {                      /* holds node (netlist of circuit)
        struct stern *input;        /* pointer to list of inputs
        struct stern *output;       /* pointer to list of outputs   */
        struct stern *bi;
        short type;                 /* type of gate                 */
        short fanout;               /* number of fanouts            */
        short parallel;
        short adrline;
        short dataline;
        float minrise,maxrise,minfall,maxfall;  /* delay times      */
        char name[MAXCHAR];         /* name of gate                 */
        char cellname[MAXCHAR];
        long lastcktim;
        unsigned state;             /* level of gate                */
        struct romtable *romptr, *masterptr;
        struct construct *c0, *c1;
        struct constlist *clist;
        /*struct snode *pseudo;*/
        int pseudo;
        unsigned lastckval : 4;
        unsigned macro : 1;         /* true = macro expansion       */
        unsigned ps0mark : 1;
        unsigned ps1mark : 1;
        unsigned obmark : 1;
        unsigned scanmark : 1;
        short mark;
        int level;                  /* level of node                */
        };
```

2. Structures d'E/S:

```
struct stern {                      /* holds list of inputs or outputs
        int node;                   /* node number of input or output
        char name[MAXCHAR];         /* name of input or output
        struct stern *next;         /* next node on list
        short invert;               /* 0 = normal, 1 = input is inverted.
        };
```

3. Tableau de structure (circuit) et pointeur vers le scan-path: .

```
struct snode *node[MAXNODE];        /* array to hold nodes
struct nodelist *scanpath;
```

1.1 Module LVLIZ

Ce module répartit le circuit en différentes couches logiques, comme défini précédemment. En addition, une vérification de la non-existence de bouclage à l'intérieur du circuit supposé combinatoire est effectuée.

1.2 Modules CONTRL Et OBSERV

Ces deux modules réalisent la génération des vecteurs de contrôle et d'observabilité d'une manière très proche de la technique décrite dans le chapitre précédent. Dans les sous-programmes réalisant l'intersection et la sélection des vecteurs, un branchement est fait suivant la méthode heuristique de tri des vecteurs choisie. Le tableau A2-3 donne le code pour le sous-programme réalisant l'intersection des vecteurs.

Tableau A2-3: Opération Intersection

1. Module principal:
Ce sous-programme retourne l'ensemble de vecteurs qui sont obtenus
par "intersection" de deux ensembles de vecteurs donnés:

```
struct construct *intersect(vector1,vector2)
struct construct *vector1, *vector2;
{
struct construct *cptr1, *cptr2, *result, *temp, *mergeconst(), *getconst
struct construct *intsct_1();
int j;
result = NULL;
for (cptr1 = vector1; cptr1 != NULL; cptr1 = cptr1->next) {
    for (cptr2 = vector2; cptr2 != NULL; cptr2 = cptr2->next) {
        if ((temp = intsct_1(cptr1,cptr2)) != NULL) {
            result = mergeconst(result,temp);
            rlsconst(temp);
            }
        }
    }
return (result);
}
```

2. Utilitaire de construction de vecteurs

```
struct construct *getconst()
{
struct construct *cptr;
char *malloc();
cptr = (struct construct *) malloc(sizeof(struct construct));
if (cptr == NULL) {
    atperr(FATAL,42,"construct","getconst");
    }
cptr->next = NULL;
cptr->nextseq = NULL;
cptr->risk = 0;
cptr->dc = 0;
cptr->pattern = (int *) malloc(sizeof(int) * pinum);
if (cptr->pattern == NULL) {
    atperr(FATAL,42,"pattern","getconst");
    }
return (cptr);
}
```

3. Ajoute un vecteur dans la liste de vecteurs associé
   à un noeud. Trie les vecteurs suivant la règle
   choisie (risque ou x):

```
struct construct *findmerge(invect,vector)
struct construct *invect, *vector;
{
struct construct *temptr;
if (vector == NULL) return (NULL);
if (flags.sortkey == SORTBYRISK) {
    if (vector->risk > invect->risk) return (NULL);
    else if (vector->risk == invect->risk && vector->dc < invect->dc)
        return (NULL);
    else {
        for (temptr = vector;
        temptr->next != NULL && temptr->next->risk < invect->risk;
        temptr = temptr->next) ;
        for (; temptr->next != NULL && temptr->next->risk == invect->risk
        && temptr->next->dc > invect->dc; temptr = temptr->next) ;
        return (temptr);
        }
    }
else {
    if (vector->dc < invect->dc) return (NULL);
    else if (vector->dc == invect->dc && vector->risk > invect->risk)
        return (NULL);
    else {
        for (temptr = vector;
        temptr->next != NULL && temptr->next->dc > invect->dc;
        temptr = temptr->next) ;
        for (; temptr->next != NULL && temptr->next->dc == invect->dc
        && temptr->next->risk < invect->risk; temptr = temptr->next) ;
        return (temptr);
        }
    }
}
```

4. Module pour effectuer l'opération intersection (symbole !) de deux vecteurs. Utilise une méthode de recherche en table ("table look-up") avec la table suivante:

!	0	1	x	#	Z
0	0	#	0	#	#
1	#	1	1	#	#
x	0	1	x	#	Z
#	#	#	#	#	#
Z	#	#	Z	#	Z

```
struct construct *intsct_1(vector1,vector2)
struct construct *vector1, *vector2;
{
struct construct *result, *getconst(), *cptr1, *cptr2, *cptr;
int dc = 0; float maxrisk = 0.0;
int i, mindc, j;  int done = FALSE;
static table[5][5] =  {
    ZERO,  CLASH, CLASH,   ZERO,    CLASH,
    CLASH, ONE,   CLASH,   ONE,     CLASH,
    CLASH, CLASH, HIGHIMP, HIGHIMP, CLASH,
    ZERO,  ONE,   HIGHIMP, DONTCARE, CLASH,
    CLASH, CLASH, CLASH,   CLASH,    CLASH  };
mindc = pinum;  cptr = result = getconst();
cptr1 = vector1; cptr2 = vector2;
while (!done) {      /* use lookup table to find intersection */
    for (i = 0; i < pinum; i++) {
        cptr->pattern[i] = table[cptr1->pattern[i]][cptr2->pattern[
        if (cptr->pattern[i] == CLASH) break;
        if (cptr->pattern[i] == DONTCARE) dc++;
        }
    if (i == pinum) {   /* no clashes found */
        cptr->risk = cptr1->risk + cptr2->risk;
        cptr->dc = dc;
        if (cptr->risk > maxrisk) maxrisk = cptr->risk;
        if (dc < mindc) mindc = dc;
        cptr->next = NULL;
        /* loop to find intersection of next vector in sequence */
        if (cptr1->nextseq == NULL && cptr2->nextseq == NULL) done =
        else {
            cptr->nextseq = getconst();
            cptr = cptr->nextseq;
            if (cptr1->nextseq != NULL) cptr1 = cptr1->nextseq;
            if (cptr2->nextseq != NULL) cptr2 = cptr2->nextseq;
            }
        }
    else {
        rlsconst(result);   return (NULL);
        }
    }
result->risk = maxrisk;  result->dc = mindc;
return (result);
}
```

1.3 Module TSTPG

Ce module génère tout d'abord la liste des pannes de collage unique du circuit. Puis, un test est généré pour chacune des classes d'équivalence de panne. Ces classes sont marquées au fur et à mesure que leurs tests sont générés, et la procédure s'arrête dès que toutes les classes ont été marquées.

Une procédure en n x log(n) est utilisée pour trier les vecteurs de test, créés pour chaque panne, suivant le nombre de x dans le vecteur. Puis les vecteurs égaux sont éliminés pour ne laisser qu'un représentant par vecteur. Enfin les vecteurs compatibles sont successivement combinés deux à deux, en partant des vecteurs contenant le plus de x.

1.4 Module OUTPUT

Les résultats sont mis en forme dans deux fichiers. Le fichier "tst" contient les informations relatives à l'analyse de testabilité et la génération des vecteurs de test durant chaque passe, ainsi que la liste des problèmes potentiels identifiés. Le fichier "sim" contient les vecteurs de test sous une forme directement utilisables par le simulateur de fautes de Matra Design Systems. Pour cela le pas de test (lu dans un fichier circuit) est utilisé pour incrémenter le temp d'application des vecteurs de test et créer une fenêtre d'observation pour émuler le testeur pendant la simulation. Enfin le fichier "tim" donne les performances du programme en temps cpu. Les tableaux A2-4, A2-5 et A2-6 donnent un exemple de ces fichiers de sortie.

Tableau A2-4: Exemple de fichier "tst"

de pannes irredondantes: 28
de pannes potentiellement indetectables: 0

Liste des pannes de collage: (15 classes)

	FAULT NUMBER	ELEMENT or OUTPUT NAME	FAULT ORIGIN	FAULT TYPE		FAULT CLASS
*	1	AND1	I1	Input	S.A.0	
	2	AND1	I1	Input	S.A.1	18
*	3	AND1	I2	Input	S.A.0	
	4	AND1	I2	Input	S.A.1	18
*	5	AND2	I3	Input	S.A.0	
	6	AND2	I3	Input	S.A.1	18
*	7	AND2	I4	Input	S.A.0	
	8	AND2	I4	Input	S.A.1	18
*	9	AOR1	AND1	Input	S.A.0	
	10	AOR1	AND1	Input	S.A.1	18
*	11	AOR1	AND2	Input	S.A.0	
	12	AOR1	AND2	Input	S.A.1	18
*	13	AOR2	OR1	Input	S.A.0	
	14	AOR2	OR1	Input	S.A.1	18
*	15	AOR2	OR2	Input	S.A.0	
	16	AOR2	OR2	Input	S.A.1	18
*	17	ANDOUT	AOR1	Input	S.A.0	
*	18	ANDOUT	AOR1	Input	S.A.1	
*	19	ANDOUT	AOR2	Input	S.A.0	
	20	ANDOUT	AOR2	Input	S.A.1	18
*	21	OR1	I5	Input	S.A.0	
	22	OR1	I5	Input	S.A.1	18
*	23	OR1	I6	Input	S.A.0	
	24	OR1	I6	Input	S.A.1	18
*	25	OR2	I7	Input	S.A.0	
	26	OR2	I7	Input	S.A.1	18
*	27	OR2	I8	Input	S.A.0	
	28	OR2	I8	Input	S.A.1	18

Detection des vecteurs (le nombre de droite indique le nombre de pannes detectees par chaque vecteur):

IIIIUUUU 12341234	
00000001	1
00010000	1
00000010	3
00000100	1
00001000	2
10000000	2
00000000	14
01000000	1
00100000	3

Tableau A2-5: Exemple de fichier "sim"

```
$ Sortie primaire:
$OUTPUT OUT
$
$
$PATTERN I1 I2 I3 I4 U1 U2 U3 U4
          0           00000001
        100           00010000
        200           00000010
        300           00000100
        400           00001000
        500           10000000
        600           00000000
        700           01000000
        800           00100000
$
$EOP
$
$TIME         900
$
$STROBE  70   80 100 R 0
$
$STATIS
```

Tableau A2-6: Exemple de fichier "tim"

```
TIMER_UTILITY                                   Reptil 2.a0

Circuit: or7


Test Generation Time :          1380 ms
```

A N N E X E III

ANNEXE 3: Complexité du partitionnement

Théorème 1: La segmentation optimale d'un circuit (problème SOC) est un problème NP-complet.

Démonstration: (d'après la méthode standard de [Garey 79]).

Notations: Pour un circuit combinatoire, on notera Vp l'ensemble des portes de ce circuit, Ve l'ensemble des entrées primaires du circuit et Vo l'ensemble de ses sorties primaires.

Le problème SOC est dans NP car, pour un circuit combinatoire donné et pour tout entier positif B donné:

1. Un algorithme non-déterministe peut trouver un ensemble S de au plus K sous-ensembles de Vp, et vérifier en temps polynomial que S recouvre Vp et que C(S), le coût du test pseudo-exhaustif associé à la segmentation S, est inférieur à B.

2. Il existe une transformation, dont le temps d'éxécution croit de façon polynomiale, du problème NP-complet connu "CLIQUE" en SOC, où le problème clique est defini comme suit:

   - INSTANCE: Un graphe $G = (V,E)$ et un entier positif $J =< |V|$.

   - QUESTION: Est-ce que G contient une clique de dimension J? C'est-à dire, existe-t-il un ensemble de J sommets tels que E contienne tous les $J(J-1)/2$ arcs possibles les reliants ?

La démonstration du théorème 1 revient donc a trouver une transformation en temps polynomial du problème "CLIQUE" en problème SOC. Il suffit pour cela de montrer qu'une instance Iclique du problème "clique" de dimension J est obtenue à partir d'une instance Isoc du problème SOC si et seulement si Isoc contient une couverture de Sp de dimension inferieure à K et de cout inferieur à B.

Lemme: L'entier J peut-etre pris tel que: $3|V|/4 < J < |V|$ sans restreindre la demonstration.

Si J est égal à |V|, il suffit d'ajouter à G un seul sommet, sans arc qui lui soit adjacent. Cette transformation est faite en temps polynomial, et le graphe original G contient une clique de dimension J si et seulement si le nouveau graphe G' en contient une.

Si J est inférieur ou egal à 3/4 de |V|, G est augmenté de 3|V| nouveaux sommets, et des $(3|V|)(3|V|-1)/2$ arcs existants entre les paires de nouveaux sommets, ainsi que des 3|V| arcs entre les sommets nouveaux et anciens. Cette transformation est faite en temps polynomial et le graphe G original contient une clique de taille J si et seulement si le graphe G' modifié contient une clique de taille $J + 3|V|$, qui est supérieure à 3/4 du nouveau |V'| (car $|V'| = 4|V|$).

Donc la supposition initiale peut-etre faite sans perte de généralité, ce qui assure que la clique aura au moins la moitié des arcs de E (ou encore que J(J-1) sera superieur à |E|).

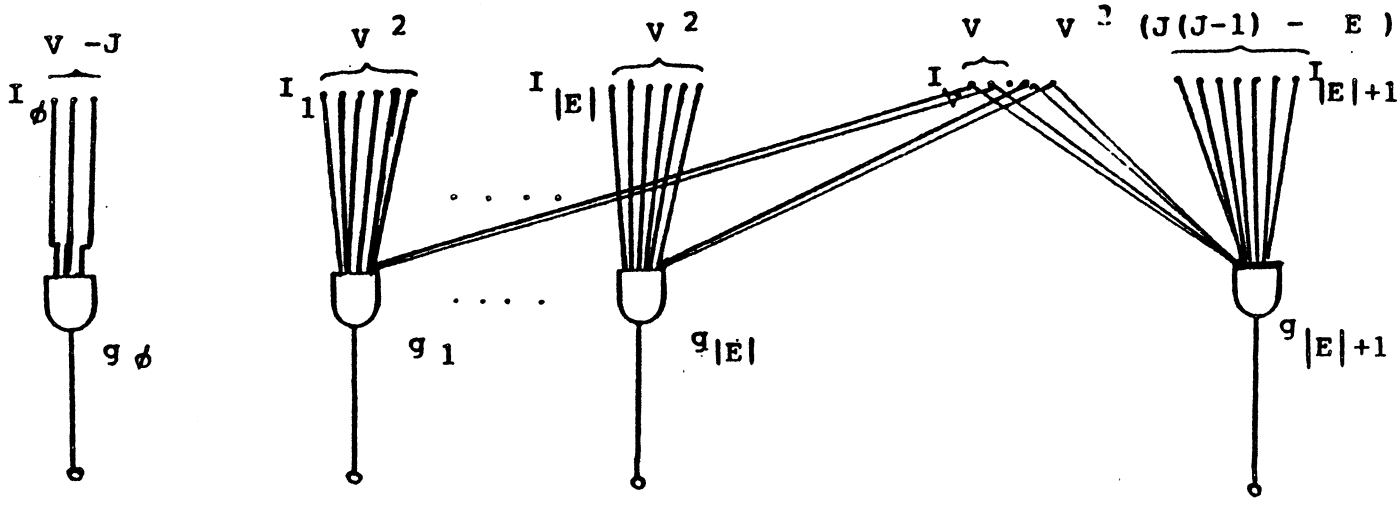


Figure A3-1: Le graphe oriente pour le problème SOC.

Une instance Isoc du problème SOC peut-etre definie à partir d'une instance Iclique du problème clique comme suit:

Dans ce qui suit le graphe associé à Isoc sera noté Gsoc=(Vsoc,A), avec Vsoc={Vp,Ve,Vs} (ensemble de sommets correspondants respectivement aux portes, entrées primaires et sorties primaires du circuit) et A l'ensemble des arcs du graphe. Le graphe correspondant à Iclique est noté G=(V,E). Le graphe Gsoc est illustré par la figure A3-1, et peut-être décrit comme suit:

- L'ensemble des portes Vp contient |E|+2 portes; les portes g1 à g|E| correspondant chacune à un arc de G; les portes g0 et g|E|+1 seront décrites dans la suite.

- L'ensemble Vs des sorties primaires contient |E|+2 sorties primaires, une par porte.

- L'ensemble Ve des entrées primaires consiste en la réunion des ensembles disjoints I0 à I|E|+1, plus IV. I0 contient |V|-J entrées primaires; les ensembles Ii (i=1 à |E|) ont chacun $|V|^2$ entrées primaires, I|E+1| a $|V|^2$ (J(J-1)-|E|) entrées primaires (ce dernier nombre est un entier positif grace aux conditions imposées initialement sur J - voir lemme), enfin IV contient |V| entrées primaires, chacune correspondant à un sommet de G.

- Finalement, l'ensemble A des arcs, contient:

1. |E|+2 arcs: un arc reliant chacune des portes à une sortie primaire

2. |V|-J arcs dans A0: un arc reliant chacune des entrées primaires de I0 à la porte g0

3. $|V|^2$ arcs dans chacun des ensembles Ai (i=1 à |E|): un arc reliant chaque entrée de Ii à la porte gi

4. $|V|^2$ (J(J-1)-|E|) + |V| arcs dans A|E|+1 (entrées de la porte g|E|+1): un arc pour chaque entrée primaire de I|E|+1 et de IV

5. et 2|E| arcs dans AV : AV contient un arc reliant chaque entrée ix de Iv à la porte gy si et seulement si l'arc de G corrrespondant à gy soit incident au sommet de G correspondant à ix

L'instance Isoc du problème SOC est completée en fixant la dimension maximale du recouvrement à K=2 et le cout maximum associe à $B=2^{|V|^2 (J(J-1)/2+|V|+1}$.

Le graphe dirigé Gsoc a $O(|V|^4)$ sommets et arcs, et puisque le nombre de bits dans B est une fonction polynomiale de la taille de G, la transformation de Ie en Isoc est faite en temps polynomial. Si Iclique contient une clique Vj de taille J, Isoc est définie par le recouvrement S={Sa,Sb} où Sa consiste en la porte g0 et les J(J-1)/2 portes gi (i=1,|E|) correspondantes aux

arcs entre paires de sommets dans Vj, et où Sb consiste en g|E|+1 et toutes les portes gi (i=1 à |E|+1) restantes. Le nombre d'entrées primaires dans Sa est donc la somme de:

1. |V|-J entrées primaires dans g0.
2. $|V|^2 J(J-1)/2$ entrées primaires pour les portes gi (i=1,|E|) (chacune a |V|2 entrées primaires).
3. J entrées primaires dans Iv (puisque Vj a exactement J sommets).

Ainsi le nombre total d'entrées primaires dans Sa est: $|V|^2 J(J-1)/2+|V|$.

De manière équivalente, le nombre d'entrées primaires dans Sb est la somme de:

1. $|V|^2$ $(J(J-1)/2 - E)$ entrées primaires pour la porte $g|E|+1$.
2. $|V|^2$ $(|E| - J(J-1)/2)$ entrées primaires pour les portes g_i $(i=1,|E|+1)$ (chacune a exactement $|V|2$ entrées primaires).
3. J entrées primaires dans Iv.

Le nombre total d'entrées primaires dans Sb est donc: $|V|^2 J(J-1)/2+|V|$, le même nombre que pour Sa. Ainsi C(S), le coût en nombre de vecteurs du test exhaustif en série de Sa et Sb est exactement $2^{|V|^2 J(J-1)/2+|V|+1}$, c'est à dire la borne B. Il s'ensuit que Isoc a un recouvrement S de taille 2 et de coût $C(S)=<B$.

Réciproquement, supposons maintenant que Isoc ait un recouvrement S={Sa,Sb} avec C(S)=<B. Les portes g1 à g|E|+1 ont des arcs provenant des $|V|^2 J(J-1)$ entrées primaires dans Ii (i=1,|E|+1), et puisque chacunes de ces entrées primaires a une sortance égale à 1, le nombre des entrées primaires dans Sa et dans Sb doivent chacun être egal à la moitié du nombre total des entrées primaires, c'est à dire $|V|^2 J(J-1)/2$. Sinon, puisque I(Sa) et I(Sb) (respectivement l'ensemble des entrées primaires dans Sa et Sb) doivent chacun avoir un cardinal multiple de $|V|^2$, lorsque restreints à Ii (i=1,|E|+1), l'un des deux aurait au moins $|V|^2 (J(J-1)/2+1)$ éléments, et C(T) serait supérieur à la borne B. Donc l'ensemble contenant g|E|+1, soit Sa, doit contenir exactement |E|-J(J-1)/2 des portes gi (i=1,|E|), et Sb doit contenir les J(J-1)/2 portes restantes. Mais il y a |V| entrées primaires additionnelles dans Sa provenant des ensembles Iv à g|E|+1. Le nombre total d'entrées primaires dans Sb est donc déjà: $|V|^2 J(J-1)/2+|V|$. Ce nombre ne peut être plus grand car C(S) deviendrait plus grand que la borne B, et puisqu'il existe au moins une entrée dans g0 (grace à l'hypothese J<|V|), g0 ne peut être dans Sa; il s'en suit que Sb doit contenir g0. Le nombre total d'entrées primaires de Sb dans les ensembles Ii (i=0,|E|) est donc $|V|^2 J(J-1)/2+|V|-J$.

Par conséquent, Sb ne peut avoir plus de J entrées primaires dans Iv. Mais, si les J(J-1)/2 portes gi (1<i<|E|) que Sb contient ont des arcs incidents à au plus J sommets dans Iv, alors les arcs correspondants dans E sont collectivement incident à au plus J sommets dans A, et ces sommets doivent former une clique de taille J. En conclusion, Iclique a une clique de taille J si et seulement si Isoc a un recouvrement de taille au plus K et de coût au plus B. CQFD.

Corollaire 1: Le problème SOC reste NP-complet pour les graphes dont les portes (mais pas necessairement les entrées primaires) sont de sortance égale à 1.

La démonstration du théorème 1 ne considère que des portes de sortance égale à 1. La démonstration de ce corollaire est donc identique à celle du theoreme 1.

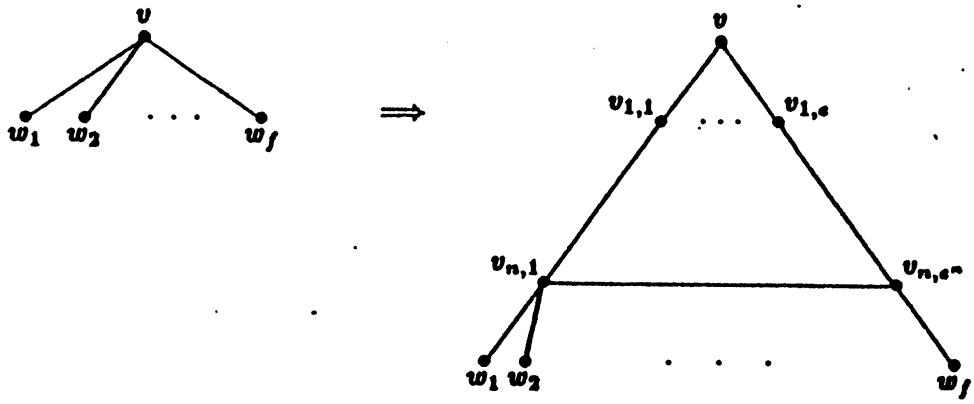


Figure A3-2: Remplacement des sommets par des arbres de niveau c

Corollaire 2: Le problème SOC reste NP-complet pour les graphes dont les sommets ont une sortance bornée par une constante c > 1.

Une demonstration similaire à celle du théorème 1 peut être construite en ne changeant, dans la définition de Isoc, que le graphe Gsoc, en remplacant tout sommet s avec une sortance égale à f > c, par un arbre d'ordre c et de racine s, tel que exactement f arcs quittent cet arbre et que tous les sommets de l'arbre (sauf peut-être ceux du dernier niveau) aient une sortance égale à c (voir Figure A3-2).

Puisqu'il existe un entier n tel que: $cn < f < cn+1$, cette construction ajoute $(cn+1 - c)/(c - 1)$ nouveaux sommets à chaque sommet s de sortance > c. Donc la transformation du graphe originel est effectuée en temps polynomial. A noter que si la transformation est faite à l'interieur de chacune des parties du recouvrement, le coût associé au nouveau recouvrement S' reste inchangé. Donc une occurence a un recouvrement satisfaisant les contraintes de taille et de coût si et seulement si l'autre occurence les satisfait. CQFD.

Corollaire 3: Le problème SOC reste NP-complet si K est égal à n'importe quelle constante c>1.

Dans la démonstration du Théorème 1, K est égal à 2. Mais une démonstration similaire peut être construite sur le même modèle avec K égal à n'importe quelle constante c, avec les modifications suivantes:

1. en ajoutant c-2 nouvelles portes, chacune ayant un arc dirigé vers une nouvelle sortie primaire et chacunes avec $|V|^2 J(J-1) + |V|$ arcs provenant d'entrées primaires additionnelles.

2. en rendant B égal à $c(2^{|V|^2 J(J-1)+|V|})$.

Le nouveau recouvrement à considérer consiste en c-2 parties, chacune contenant l'une des c-2 nouvelles portes, plus les 2 parties Sa et Sb déjà définies. Ce recouvrement est de dimension c et de coût B. Et puisque la transformation précédente est faite en temps polynomial, la nouvelle occurence de SOC correspond à un recouvrement de dimension au plus c et de coût au plus B si et seulement si l'occurence de CLIQUE a une clique de taille J, ce qui achève la démonstration de ce corollaire.

A N N E X E IV

ANNEXE 4: Description du programme P.E.T.

1 STRUCTURE DES DONNÉES

La procédure de partitionnement a été implementée en FORTRAN 77. La description détaillée du programme P.E.T. correspondant est donnée en Annexe 4. Les tableaux et pointeurs sont contenus dans des commons, regroupés par fonction dans des fichiers "common.inc". La structure de ces fichiers peut être décrite comme suit:

```
*       FILE = [ARQ.PET]PARA.INC
*       Contient les parametres du programme

 parameter (MAXCEL=3000) !nombre maxi de cellules
 parameter (MAXNOD=3000) !nombre maxi de noeuds
 parameter (MAXPTN=3001) !nombre maxi de pointeurs
                         !vers les tableaux de noeuds
 parameter (MAXPI=100)   !nombre maxi d'entrees primaires
 parameter (LVLIM=200)   !nombre maxi de niveaux logiques
 parameter (MAXEVT=1500) !nombre maxi de noeuds par niveau
 parameter (MAXIOL=10000)!nombre maxi d'elements dans le tableau d'E/
 parameter (NPOUT=50)    !nombre maxi de colonnes
                         !dans la matrice de dependance

*       FILE = ARQ$PET:NODE.INC
```

```
*       Contient les tableaux de description du circuit

        integer *2 point2(MAXPTN),point1(MAXPTN)  ! pointeurs vers les sorti
                                                  ! et entrees de chaque noe
        integer *2 type(MAXNOD),level(MAXNOD)     ! type et niveau logique
        integer *2 input(MAXIOL),fanout(MAXIOL)   ! liste des E/S
        integer *2 name(5,MAXNOD),knom(6,MAXNOD)  ! tableaux des noms
        integer *2 ncod,mxcod,ngat                ! pointeurs de fin de list
```

Pour les circuits combinatoires, un noeud est uniquement défini comme étant la sortie d'une porte logique. Dans le programme nous n'aurons donc qu'une seule entité pour décrire une cellule, une porte ou un noeud.

Chaque noeud est repéré par un numero de noeud, qui permet de référer son type (type) , son niveau (level) son nom (name), l'expansion de ce nom dans le cas d'une macro (knom) et les pointeurs vers la liste d'entrées (point1) et la liste de sorties (point2). Par construction, point1(n) pointe dans la table input(MAXIOL) vers la première entrée de la porte n. Point1(n+1)-1 pointe vers la dernière entrée de la porte n. Point2 est construit de manière symetrique pour le tableau des sorties fanout(MAXIOL).

```
*       FILE = ARQ$PET:GAIN.INC
*       Contient la structure de donnee des gains du mincut

        integer *4 gain(MAXCEL)
        integer *2 fircel(MAXCEL),lascel(MAXCEL),largan,lowgan
        integer *2 nxtlow(MAXCEL),nxtlar(MAXCEL),headvid,ngain
        common /cgain/gain,fircel,lascel,nxtlow,nxtlar,largan,
     +   lowgan,hedvid,ngain,mxrec
```

La figure A4-1 montre la structure de donnée des gains du "mincut" qui peut être détaillée comme suit:

- largan: tête de la liste des gains, c'est à dire pointeur vers l'enregistrement correspondant au gain le plus fort dans la liste (symetriquement lowgan est le pointeur vers le gain le plus faible dans la liste).

L'enregistrement du gain n contient:

- gain(n) : valeur du gain n
- nxtlow(n): pointeur vers le prochain gain de valeur plus faible dans la liste
- nxtlar(n): pointeur vers le prochain gain de valeur plus forte dans la liste
- fircel(n): pointeur vers la première cellule dans la liste des cellules de même gain gain(n)
- lascel(n): pointeur vers la dernière cellule dans la liste des cellules de même gain gain(n)

En FORTRAN, les listes dynamiques n'existent pas. La liste des enregistrements des gains est donc contenue dans un tableau de taille fixe, la place pour chaque enregistrement étant alloué à l'interieur de ce tableau. La première place libre dans le tableau est accessible par le pointeur "hedvid". La figure A4-2 montre la structure de cet tableau d'enregistrements vides.

```
*       FILE = ARQ$PET:CELL.INC
*       Contient les enregistrements des gains de chaque cellule

        integer *2 celgan(MAXCEL)
        integer *2 nxtcel(MAXCEL),antcel(MAXCEL)
        integer *2 firnod(MAXCEL),lasnod(MAXCEL)
        integer *2 ncell
        common /ccell/celgan,nxtcel,antcel,firnod,lasnod,ncell
```

Comme montré en figure A4-1, pour chaque cellule dans une liste de cellules de même gain (vu des cellules, la structure globale apparait comme une liste de listes de cellules), on a :

- celgan: pointeur vers l'enregistrement contenant le gain de la cellule

- nxtcel: pointeur vers la cellule suivante dans la liste

- antcel: pointeur vers la cellule précédente dans la liste

```
*       FILE = ARQ$PET:DEPEND.INC
*       Contient les tableaux de dependance des cellules

        integer *2 ranged(MAXNOD),dpend(MAXPI,MAXNOD)
        integer *2 tranche(MAXEVT)
        integer *2 order(LVLIM),wor(LVLIM)
        integer *2 limit
        common /clvliz/ limit,wor,order,ranged
        common /cevt/ tranche !temporary array
        common /cdpnd/ dpend
```

Ranged(maxnod) contient la liste de tous les noeuds, ordonnés par couche logique croissante.

Order(L) contient le pointeur vers le premier noeud de la couche L dans le tableau Ranged.

dpend(MAXPI,MAXNOD) contient pour chaque noeud du circuit, un vecteur de dépendance de ce noeud par rapport aux entrées primaires du circuit. Ce vecteur contient un 0 en colonne i si le noeud ne dépend pas de l'entrée i, 1 dans le cas contraire.

```
*       FILE = ARQ$PET:ERROR.INC
*       Contient les indicateurs d'erreur.

        integer *2 nogo,maxngo
        common /cerror/ nogo,maxngo


*       FILE = ARQ$PET:IOPIN.INC
*       Contient les listes des entrees et des sorties primaires

        integer *2 lprim(1000),output(1000)
        integer *2 nprim,mxprim,nout,mxout
        common /cin/ nprim,mxprim,lprim
        common /cout/ nout,mxout,output

        Lprim: liste des entrées primaires
        Output: liste des entrées primaires

*       FILE = ARQ$PET:SEGMNT.INC
*       Contient les tableaux utilisees par le "mincut"

        integer *2 frontA(MAXCEL)
        integer *2 frontB(MAXCEL)
        integer *2 cutset(MAXNOD)
        integer *2 piA(MAXPI),piB(MAXPI)
        integer *2 nfron,ncut


        common /csegmn/nfron,ncut,frontA,frontB,cutset
        common /csegpi/piA,piB
```

FrontA(maxcel): liste des noeuds qui sont à la frontière de A. Un noeud de A est à la frontière de A si et seulement si tous les chemins de ce noeud vers la sortie primaire passent par un noeud élément de la coupe A/B. Notons que le nombre de noeuds dans la coupe ne change pas lorsque qu'un noeud frontière est changé de bloc.

FrontB(maxcel): liste des noeuds à la frontière de B.

Cutset(maxnod): liste des noeuds coupés (éléments de la coupe A/B)

piA(maxpi) (resp. piB(maxpi)): liste des entrées primaires entrant directement dans le bloc A (resp. bloc B).

nfron: nombre de cellules frontières

ncut: nombre de noeuds coupés.

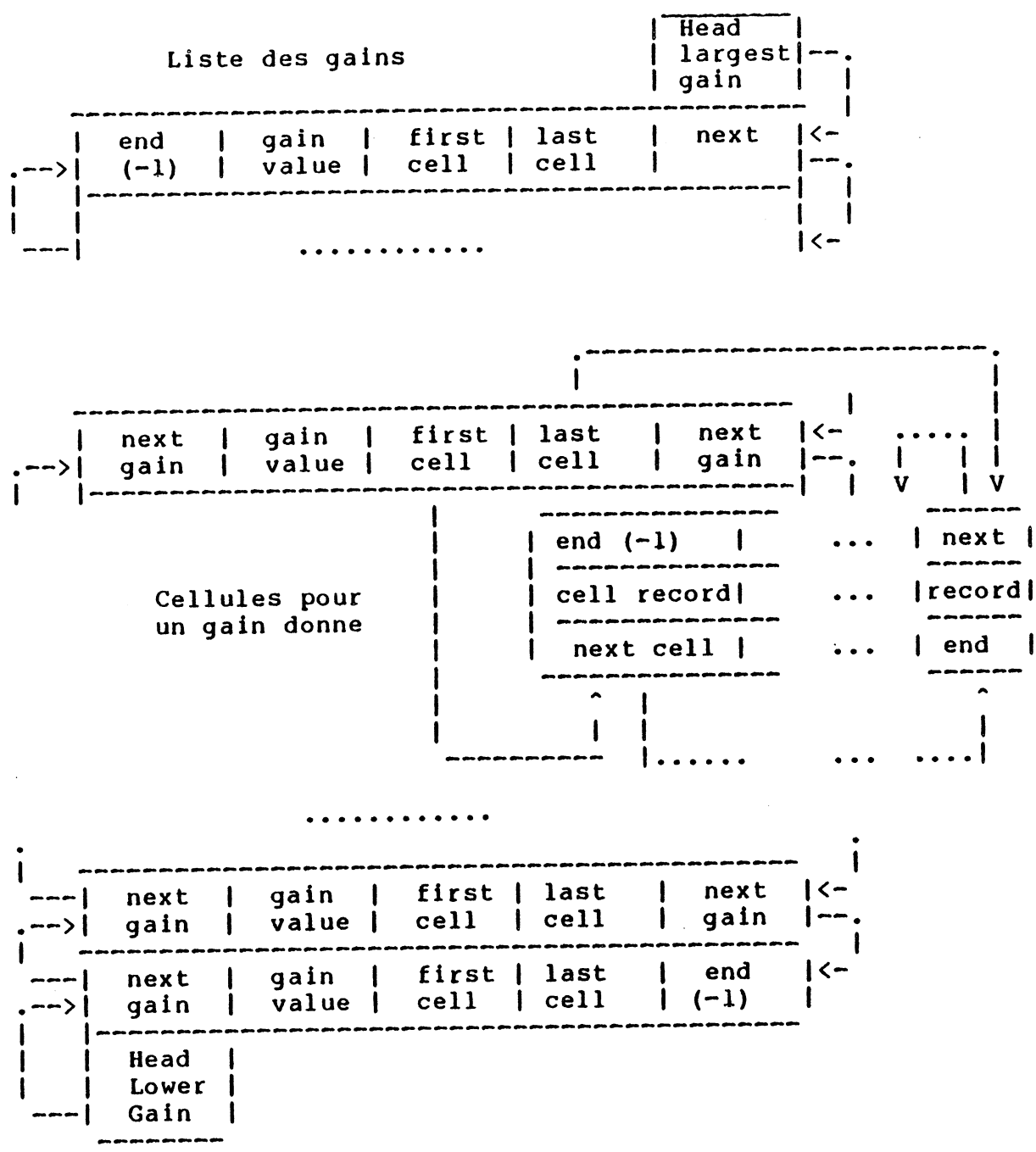


Figure A4-1: Structure contenant les listes de cellules de gains egaux pour le "mincut"

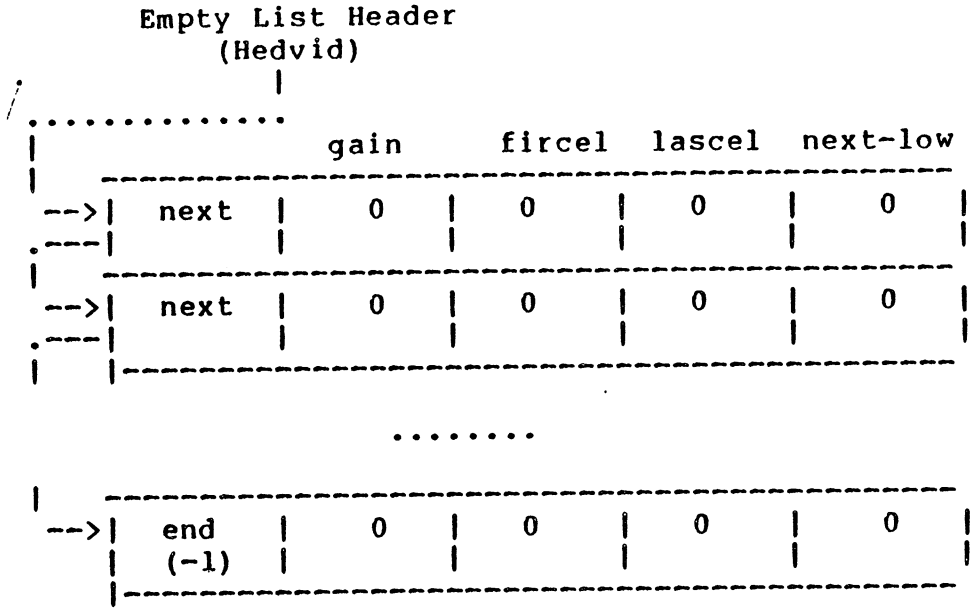


Figure A4-2: Structure des enregistrements vides libres.

B I B L I O G R A P H I E

[Acken 83] Acken, J.M., "Testing for Bridging Faults in CMOS Circuits," Proceedings 19th Design Automation Conference, pp. 717-718, Miami, Juin 27-29 1983.

[Agrawal 82] Agrawal, V.A. and M.R. Mercer, "Testability Measures: What Do They Tell Us ?", Proceedings 1982 International Test Conference, pp. 391-396, Philadelphia, Pennsylvania, Novembre 1982.

[Agrawal 84] Agrawal, V.A., S.K. Jain and D.M.Singer, "A CAD System for Design for Testability," VLSI Design Magazine, pp. 46-54, Octobre 1984.

[Ando 80] Ando H., "Testing VLSI with Random Access Scan," Digest of Papers, COMPCON 80, pp. 50-52, San Fransisco, California, Fevrier 1980.

[Archambeau 83a] Archambeau, E.C., "A Computed Aided Engineering System for CMOS Gate Arrays," Proceedings 3rd International Conference on Semi-Custom ICs, London, England, Novembre 1-3 1983.

[Archambeau 83b] Archambeau, E.C., "Test Pattern Generation for Small Digital Gate Arrays," Proceedings Wescon Technical Session #33, San Francisco, Novembre 11 1983.

[Archambeau 83c] Archambeau, E.C. and C. Coupal, "ARCOP (A Rapid Controllability Observability Program) User's Manual," Matra Design Systems, Santa Clara, California, Aout 1983.

[Archambeau 84a] Archambeau, E.C., "Pseudo-Exhaustive Testing," 6th Annual Workshop on Design for Testability, Vail, Colorado, Avril 1984.

[Archambeau 84b] Archambeau, E.C. and E.J. McCluskey, "Fault Coverage of Pseudo-Exhaustive Testing," Proceedings 14th International Conference on Fault-Tolerant Computing, pp. 141-145, Kissimee, Florida, Juin 20-22 1984.

[Archambeau 85a] Archambeau, E.C., "Network Segmentation for Pseudo-Exhaustive Testing," CRC Technical Report (Pending Publication), Center for Reliable Computing, Computer Systems Laboratory, Departments of Electrical Engineering and Computer Science, Stanford University, Stanford, California, Juillet 1985.

[Archambeau 85b] Archambeau, E.C., "Heuristics-based Functional TPG for Semi-Custom Circuits," Proceedings CADMAT-Asia International Conference, Singapore, Singapore, Juillet 9-11 1985.

[Armstrong 66] Armstrong, D.B., "On Finding a Nearly Minimal Set of Fault Detection Tests for Combinational Logic Nets." IEEE Transactions on Electronic Computers, Vol. EC-15, Feb. 1966, pp. 66-73.

[Arzoumanian 81] Arzoumanian Y. and J. Waicukauski, "Fault Diagnosis in an LSSD Environment," Proceedings 1981 International Test Conference, pp. 397-401, Philadelphia, Pennsylvania, Novembre 1981.

[Avizienis 82] Avizienis, A., "The Four-Universe Information System Model for the Study of Fault-Tolerance," Proceedings 12th International Conference on Fault-Tolerant Computing, pp. 6-13, Santa-Monica, California, Juin 1982.

[Barzilai 81] Barzilai Z. et al., "The Weighted Syndrome Sums Approach to VLSI Testing," IEEE Trans. on Computers, Vol C-30, pp. 996-1000, Decembre 1981.

[Baschiera 85] Baschiera, D. and B. Courtois, "Testing CMOS and DFT of CMOS," 7th Annual Workshop on Design for Testability, Beaver Creek, Colorado, Avril 1985.

[Bellon 82] Bellon, C., et al., "Automatic Generation of Microprocessor Test Programs," Proceedings 19th Design Automation Conference, Las Vegas, Nevada, Juin 14-16, 1982.

[Bellon 84] Bellon, C. and R. Velasco, "Taking Into Account Asynchronous Signals In Functional Test of Complex Circuits," Proceedings 21st Design Automation Conference, pp. 175-181, Albuquerque, New Mexico, Juin 14-16, 1984.

[Bennets 80] Bennets, R. G., G. D. Robinson and C. Maunder, "Computer-Aided Measure for Logic Testability," Proceedings IEEE International Conference on Circuits and Computers, pp. 1162-1165, Port Chester, New-York, Octobre 1980.

[Bennetts 84] Bennetts R. G., in "Design of Testable Logic Circuits," Addison-Wesley Publishers, 1984.

[Berg 82] Berg, W. C. and R. D. Hess, "COMET: A Testability Analysis and Design Modification Package," Proceedings 1982 International Test Conference, pp. 364-368, Philadelphia, Pennsylvania, Novembre 1982.

[Botorff 77] Botorff P.S. et al., "Test Generation for Large Logic Networks," Proceedings 14th Design Automation Conference, pp. 479-485, New Orleans, Juin 20-22 1977.

[Botorff 79] Botorff P.S., R.E. France and H.C. Godoy, "Automatic Test Generation for LSI Chips and Printed Circuit Boards," Proceedings International Solid-State Conference, pp. 252-253, New York, New-York, Fevrier 1979.

[Botorff 80] Botorff P.S., "Computer Aids to Design - An Overview," in Computer Design Aids for VLSI Circuits, Sigthoff & Noordhoff Publishers, The Hague, 1980.

[Botorff 81] Botorff P.S., "Partitioning Large LSSD Networks for Test Generation," 4th Annual Workshop on Design for Testability, Vail, Colorado, Avril 1981.

[Bozorgui 84] Bozorgui-Nesbat, S. and E.J. McCluskey, " Design for Delay Testing of PLAs," Proceedings ICCAD Conference, pp. 146-148, Santa-Clara, California, Novembre 1984.

[Breuer 76] Breuer, M.A. and A.D. Friedmann, in Diagnosis & Reliable Design of Digital Systems, Computer Science Press, Maryland, 1976.

[Breuer 78] Breuer, M. A., "New Concepts in Automated Testing of Digital Circuits," Proceeedings Symposium on Computer-Aided Design of Digital Electronic Circuits and Systems, pp. 57-80, Bruxelles, Belgique, Novembre 1978.

[Breuer 79] Breuer, M.A. and A.D. Friedmann, "TEST/80 - A Proposal for an Advanced Automatic Test Generation System," Proceedings IEEE Autotestcon Conference, pp. 205-312, Novembre 1979.

[Buric 85] Buric, M. R. and T. G. Matheson, "Silicon Compilation Environments," Proceeedings Custom Integrated Circuits Conference, pp. 57-80, Portland, Oregon, Mai 1985.

[Carter 82] Carter, W.C., "Signature Testing with Guaranteed Bounds for Fault Coverage," Proceedings 1982 International Test Conference, pp. 75-82, Philadelphia, Pennsylvania, Novembre 1982.

[Chiang 83] Chiang, K. W. and Z. G. Vrazenic, " On Fault Detection in CMOS Logic Networks," Proceedings 20th Design Automation Conference, pp. 50-56, Miami Beach, Florida, Juin 27-29 1983.

[Crouzet 78] Crouzet, Y., "Conception de Circuits a Large Echelle d'Integration Totalement Autotestables," These de Docteur Ingenieur, Toulouse, France, Novembre 1978.

[DasGupta 82] DasGupta, S., P. Goel, R. Walther and T.W. Williams, "A Variation of LSSD and Its Implication on Design and Test Pattern Generation in VLSI," Proceedings 1982 International Test Conference, pp. 63-66, Philadelphia, Pennsylvania, Novembre 1982.

[David 75] David, R., "Paradoxe du Test des Circuits Combinatoires," Revue Digital Processes, pp. 333-336, Avril 1975.

[Dussault 78] Dussault, J.A., "A Testability Measure," Proceedings IEEE International Test Conference, pp. 113-116, Philadelphia, Pennsylvania, Novembre 1978.

[Eichelberger 77] Eichelberger, E.B. and T.W. Williams, "A Logic Design Structure for LSI Testability," Proceedings 14th Design Automation Conference, pp. 462-468, New Orleans, Juin 20-22 1977.

[Eichelberger 78] Eichelberger, E.B. , in Keynote Speech, 3rd USA-Japan Computer Conference, Tokyo, Japan, Novembre 1978.

[El Ziq 81] El Ziq, Y., "Automatic Test Generation for for Stuck-Open Faults in CMOS VLSI," Proceedings 18th Design Automation Conference, pp. 347-354, Nashville, Tenessee, Juin 20-22 1981.

[El Ziq 83] El Ziq, Y., "Classifying, Testing and Eliminating VLSI MOS Failures," VLSI Design Magazine, pp. 30-35, Septembre 1983.

[Fiduccia 82] Fiduccia, C.M. and P.M. Mattheyses, "A Linear Time Heuristic For Improving Network Partitions," Proceedings 19th Design Automation Conference, pp. 175-181, Las Vegas, Nevada, Juin 14-16, 1982.

[Funatsu 75] Funatsu, S. N. Wakatsuki and T. Arima, "Test Generation Systems in Japan," Proceedings 12th Design Automation Conference, pp. 114-122, Juin 1975.

[Funatsu 78] Funatsu, S., N. Wakatsuki and A. Yamada, "Designing Digital Circuits with Easily Testable Consideration," Proceedings 1978 Semiconductor Test Conference, pp. 98-102, Cherry-Hill, New Jersey, Novembre 1978.

[Galiay 80] Galiay, J., Y. Crouzet and M. Vergniault, "Physical versus Logical Fault Models in MOS LSI Circuits: Impact on their Testability," IEEE Trans. on Computers, Vol. C-29, No. 6, pp. 527-531, Juin 1980.

[Garey 79] Garey, M.R., and D.S. Johnson, in "Computers and Intractability: A Guide to the Theory of NP-Completeness," W.H. Freeeman and Co., Publ., San Francisco, California, 1979.

[Godoy 77] Godoy, H.C., G. Franklin and P. Botorff, "Automatic Checking of Logic Design Structures for Compliance with Testability Ground Rules," Proceedings 14th Design Automation Conference, pp. 469-478, New Orleans, Louisiana, Juin 20-22 1977.

[Goel 81] Goel, P. and B.C. Rosales, "PODEM-X: An Automatic Test Generation System for VLSI Logic Structures," Proceedings 18th Design Automation Conference, pp. 260-268, Nashville, Tenessee, Juin 20-22 1981.

[Goel 82] Goel, P. and T. McMahon, "Electronic Chip-in-Place Test," Proceedings IEEE International Test Conference, pp. 83-90, Philadelphia, Pennsylvania, Novembre 1982.

[Golstein 79] Goldstein, L.H., "Controllability - Observability Analysis for Digital Circuits," IEEE Trans. on Circuits and Systems, Vol. CAS-26, No. 9, pp. 685-693, Septembre 1979.

[Golstein 80] Goldstein, L.H., "SCOAP: Sandia Controllability - Observability Analysis Program," Proceedings 17th Design Automation Conference, pp. 190-196, Minneapolis, Minnesota, Juin 1980.

[Grason 79] J. Grason, "TMEAS, A Testability Measurement Program," Proceedings 16th Design Automation Conference, pp. 156-161, San Diego, California, Juin 1979.

[Hassan 83] Hassan, S.Z., D.J. Lu and E.J. McCluskey, "Parallel Signature Analyzers," Digest of Papers, COMPCON Spring 83, San Fransisco, California, Fevrier 1983.

[Hassan 84] Hassan, S.Z. and E.J. McCluskey, "Increased Fault Coverage Through Multiple Signatures," Proceedings 14th International Conference on Fault-Tolerant Computing, pp. 354-359, Kissimee, Florida, Juin 20-22 1984.

[Hild 84] Hild, M., "Converting PALs into CMOS Gate Arrays," VLSI Design Magazine, pp. 58-64, Mars 1984.

[Hild 85] Hild, M., and J.O. Piednoir, "Efficient Placement Algorithms for VLSI," VLSI Design Magazine, pp. 46-50, Avril 1985.

[Hughes 84] Hughes, J.L.A. and E.J. McCluskey "An Analysis of the Multiple Fault Detection Capabilities of Single Stuck-At Fault Test Sets," Proceedings 1984 International Test Conference, pp. 52-58, Philadelphia, Pennsylvania, Novembre 1984.

[Ibarra 75] Ibarra, O.H. and S.K. Sahni, "Polynomially Complete Fault Detection Problems," IEEE Transactions on Computers, Vol. C-24, pp. 242-249, Mars 1975.

[Jensen 82] Jensen, L., "Investigation of Commercially Available Programs and Algorithms for Automatic Test Pattern Generation," C. Rousing Technical Report, Copenhagen, Danemark, Septembre 1982.

[Kernighan 70] Kernighan, B.W. and S. Lin, "An Efficient Heuristic Procedure for Partitioning Graphs," Bell System Technical Journal, Vol. 49, pp. 291-307, Fevrier 1970.

[Koenemann 79] Koenemann, B., J. Mucha and G. Zwiedoff, "Built-In Logic Obervation Techniques," Proceedings IEEE International Test Conference, pp. 37-41, Philadelphia, Pennsylvania, Octobre 1979.

[Koenemann 80] Koenemann, B., J. Mucha and G. Zwiedoff, "Built-In Test for Complex Digital Integrated Circuits," IEEE Journal of Solid-State Circuits, Vol. SC-15, No. 3, pp. 315-319, Juin 1980.

[Kovijanic 81] Kovijanic, P. G., "Single Testability Measure of Merit," Proceedings IEEE International Test Conference, pp. 521-529, Philadelphia, Pennsylvania, Novembre 1981.

[Krishnamurthy 84] Krishnamurthy, B., "An Improved Min-Cut Algorithm for Partitioning VLSI Networks," IEEE Trans. on Comp.,/Vol. C-33, No.5, pp. 438-446, Mai 1984.

[Lowden 79] Lowden, R.P., "Testing a High Density Logic Masterslice," Proceedings International Solid-State Conference, pp. 250-251, New York, New York, Fevrier 1979.

[Malaiya 82] Malayia, K. and Y. H. Su, "A New Fault Model and Testing Technique for CMOS Devices," Proceedings IEEE International Test Conference, Philadelphia, Pennsylvania, Novembre 1982.

[McCluskey 81] McCluskey, E.J. and S. Bozorgui-Nesbat, "Design for Autonomous Test," IEEE Trans. on Comp., Vol. C-30, No.11, pp. 866-875, Novembre 1981.

[McCluskey 82a] McCluskey, E.J., "Verification Testing," Proceedings 19th Annual Design Automation Conference, pp. 449-500, Las Vegas, Nevada, Juin 14-16, 1982.

[McCluskey 82b] McCluskey, E.J., "Built-In Verification Test," Proceedings IEEE International Test Conference, pp. 183-190, Philadelphia, Pennsylvania, Novembre 1982.

[McCluskey 83] McCluskey, E.J., "Exhaustive and Pseudo-Exhaustive Test," Int. Test Conference 83 Tutorial on Built-in Test - Concepts and Techniques, Philadelphia, Pennsylvania, Octobre 17, 1983.

[McCluskey 84] McCluskey, E.J. "A survey of Design for Testability Scan Techniques," VLSI Design Magazine, pp. 38-61, Decembre 1984.

[McCluskey 85] McCluskey, E.J., "Design for Testability," in Recent Developments in Fault Tolerant Computing, D.K. Pradham, Ed., Prentice-Hall, Englewoods Cliff, New Jersey, 1985.

[Mei 74] Mei, K.C., "Bridging and Stuck-at Faults," IEEE Trans. on Computers, Vol. C-23, pp. 720-727, Juillet 1974.

[Mercer 81] Mercer, M. R., V. D. Agrawal and C. M. Roman, "Test Generation for Highly Sequential Scan-Testable Circuits Through Logic Transformation," Proceedings IEEE International Test Conference, pp. 561-565, Philadelphia, Pennsylvania, Octobre 1981.

[Muehldorf 81] Muehldorf, E.I. and D.V. Savkar, "LSI Logic Testing - An Overview," IEEE Trans. on Computers, Vol. C-30, pp. 1-17, Janvier 1981.

[Ratiu 81] Ratiu, I.M., "Macromodels for Testability Analysis," 4th Annual Workshop on Design for Testability, Vail, Colorado, Avril 1981.

[Ratiu 82] Ratiu, I.M., "VICTOR: A Fast VLSI Testability Analysis Program," Proceedings 1982 International Test Conference, pp. 397-401, Philadelphia, Pennsylvania, Novembre 1982.

[Saucier 84] Saucier, G. and C. Bellon, "Classification and Comparison of Different Placement Strategies," Proceedings IEEE International Conference on Computer Design, pp. 745-751, Port-Chester, New York, Octobre 1984.

[Savir 83] Savir, J., G. Ditlow and P.H. Bardell, "Random Pattern Testability," Proceedings 13th International Conference on Fault-Tolerant Computing, pp. 80-89, Milano, Italy, Juin 28-30 1983.

[Schneider 67] Schneider, P. R., "On the Necessity to Examine D-chains in Diagnostic Test Generation - An Example," IBM Journal of Research and Development, Vol. 11, pp.114, Janvier 1967.

[Schweikert 79] Schweikert, D.G. and B.W. Kernighan, "A Proper Model for the Partitioning of Electrical Circuits," Proceedings 9th Annual Design Automation Workshop, pp. 57-62, Dallas, Texas, Juin 1979.

[Shedletsky 77] Shedletsky, J.J., "Random Testing: Practicality vs. Verified Effectiveness," Proceedings 7th International Conference on Fault-Tolerant Computing, Los Angeles, California, pp. 175-179, Juin 28-30, 1977.

[Smith 78] Smith, J.E., "On the Existence of Combinational Circuits Exhibiting Multiple Redundancy," IEEE Transactions on Computers, Vol. C-27, pp. 1221-1225, Decembre 1978.

[Susskind 81] Susskind, A. "Testability and Reliability of LSI," RADC Technical Report, RADC-TR-80-384, Janvier 1981, pp. 99-102.

[Tang 83] Tang, D.T. and L.S. Woo, "Exhaustive Test Pattern Generation With Constant Weight Vectors," IEEE Trans. on Computers, Vol. C-32, pp. 1145-1150, Decembre 1983.

[Ulrich 73] Ulrich, E.G. and T. Baker, "The Concurrent Simulation of Nearly Identical Networks," Proceedings 10th Annual Design Automation Conference, pp. 145-150, Juin 1973.

[VLSI 85a] VLSI SYSTEMS DESIGN Staff, "A perspective on CAE workstations," VLSI Design Magazine, pp. 52-74, Avril 1985.

[VLSI 85b] VLSI SYSTEMS DESIGN Staff, "Survey of IC Layout CAD Systems," VLSI Design Magazine, pp. 45-51, Septembre 1985.

[Wadsack 78] Wadsack, R. L., "Fault Modelling and Logic Simulation of CMOS and MOS Integrated Circuits," Bell System Technical Journal, Vol. 57, No. 4, pp. 1449-1474, Mai-Juin 1978.

[Wang 84] Wang, L.T. and E. Law, "DTA: Daisy Testability Analysis," Proceedings IEEE International Conference on Computer-Aided Design, pp. 143-145, Santa-Clara, California, Novembre 1984.

[Williams 73] Williams, M.J.Y. and J.B. Angell, "Enhancing Testability of Large Scale Integrated Circuits Via Test Points and Additional Logic," IEEE Trans. on Computers, Vol. C-22, pp. 46-60, Janvier 1973.

[Williams 83] Williams, T.W. and K.P. Parker, "Design for Testability: A Survey," Proceedings IEEE, Vol. 71, No. 1, pp. 98-112, Janvier 1983.

[Yamada 77] Yamada, A., N. Wakatsuki, H. Shibano, O. Itoh, K. Tomita and S. Funatsu, "Automatic Test Generation for Large Digital Circuits," Proceedings 14th Design Automation Conference, pp. 78-83, New Orleans, Juin 20-22 1977.

INPG département des études doctorales

AUTORISATION de SOUTENANCE

VU les dispositions de l'article 15 Titre III de l'arrêté du 5 juillet 1984 relatif aux études doctorales

VU les rapports de présentation de Messieurs
. J.C GEFFROY, Professeur
. C. LANDRAULT, Chargé de recherche

Monsieur Eric ARCHAMBEAU

est autorisé à présenter une thèse en soutenance en vue de l'obtention du diplôme de DOCTEUR de L'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE, Spécialité "Microélectronique".

Fait à Grenoble, le 25 septembre 1985

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

P.O. le Vice-Président.