Contribution au test des circuits intégrés logiques

Jacques Caillat

THESE

présentée à

Université Scientifique et Médicale de Grenoble
Institut National Polytechnique de Grenoble

pour obtenir le grade de

Docteur de 3ème cycle
« Informatique »

par

Jacques CAILLAT

CONTRIBUTION AU TEST DES CIRCUITS INTEGRES LOGIQUES

Thèse soutenue le 8 octobre 1976 devant la Commission d'Examen :

Président : Monsieur J. KUNTZMANN

Examinateurs : Monsieur L. BOLLIET
Madame G. SAUCIER
Monsieur J.P. LUSINCHI
Monsieur J.C. RONCIN

TABLE DES MATIERES

Cette table est sauvegardée en même temps que les fichiers REPRISE et RESULTATS. Elle est donc inchangée tant que l'instruction NEW ou une nouvelle commande * DAD n'est pas rencontrée.

Option : TOUS signifie que la table TDD contient tous les défauts (identique à TDR).

L'exécution de la commande *DAD provoque l'impression de la liste des défauts de la nouvelle table TDD en indiquant pour chacun d'eux s'il a déjà ou non été détecté.

3. ANALYSE D'UNE SOUS-SEQUENCE

Activée par la commande
* ANALYSE
suivie de

A) REPRISE
ou NOREPRISE
ou BLOC = K

REPRISE signifie que la sous-séquence à analyser doit être accolée à la fin de la séquence précédente : l'état initial du circuit est pris sur le fichier REPRISE.

NOREPRISE signifie le début d'une nouvelle séquence : l'état initial du circuit est totalement indéterminé.

B) Les options REPRISE et NOREPRISE doivent être suivies d'un ensemble d'instructions précisant, sous forme d'une liste de 0 et de 1, les états logiques appliqués aux entrées (une par affichage). Cette liste peut éventuellement être précédée d'un indicateur NT, qui signifie que la mesure des sorties ne sera pas effectuée pour cet affichage. Aucun défaut ne sera donc considéré comme détecté par un affichage comportant l'indicateur NT.

Un bloc est l'ensemble des affichages générés pour une même table de défauts à détecter. (Il peut contenir plusieurs séquences).
BLOC = K signifie que l'on désire connaître les défauts détectés sur une nouvelle table de défauts TDD par la ou les séquences générées précédemment pour une autre table. K est le numéro d'occurence de la sous-commande * DAD correspondante.

4. VALIDATION DE LA DESCRIPTION TOPOLOGIQUE

La validation de la description topologique est une simulation préalable, sans propagation de liste de défauts. Son emploi est facultatif. Le résultat de la validation est sans effet sur la suite du déroulement d'un programme.

Le module de validation est activé par :

** VALIDATION

suivie de

A) NOMCIRCUIT (obligatoire)

nom sous lequel le circuit à valider est stocké en bibliothèque.

B) Liste de valeurs

On donne ainsi la grille des entrées-sorties du circuit : la simulation sera faite en appliquant les entrées et une vérification des sorties calculées par rapport aux sorties attendues est faite, provoquant l'impression du résultat.

DESCRIPTION VALIDE

ou DESCRIPTION NON VALIDE

La grille d'entrées-sorties est donnée par une suite d'instructions (une par affichage).

Chaque instruction est une suite de valeurs logiques 0, 1 ou X séparées par des virgules, autant que le nombre total de pattes du circuit : en tête les entrées, puis les sorties, dans l'ordre de la description topologique (première instruction).

Une valeur X en entrée ou en sortie n'a pas la même signification :

- En entrée, X est traitée comme une véritable valeur logique.
- En sortie, X signifie que la valeur n'est pas à prendre en compte pour la validation .

B.9 - BLOCS : cette instruction (facultative) permet de recopier, dans l'ensemble de défauts en cours de définition, certains des ensembles de défauts relatifs aux blocs qui ont été incorporés.

La carte BLOC est suivie d'une liste de noms de modules (donnés ou générés) indicés par le numéro de l'ensemble de défauts à considérer.

Ainsi, si la description topologique de NOMCIRCUIT a utilisé l'instruction NOM-TYPE (liste 1 - liste 2) signifiant l'insertion d'un module dont le nom bibliothèque est TYPE et identifié par NOM à l'intérieur de NOMCIRCUIT, NOM (1) signifie : pour le module NOM, incorporation du I^e ensemble de défauts de TYPE.

C) Une carte FIN termine la description des défauts.

B.2 - COLLAGES à 0

puis liste d'identificateurs de connexions.

B.3 - COLLAGES à 1

puis liste d'identificateurs de connexions

B.4 - Les instructions COLLAGES à 0 et COLLAGES à 1 peuvent être remplacées par une instruction COLLAGES suivie d'une liste. COLLAGES est équivalente aux deux instructions COLLAGES à 0 et COLLAGES à 1 relatives à cette liste.

B.5 - C/C ET

puis une liste de couples d'identificateurs de connexions sous la forme A1, A2 - A3, A4, définissant les courts-circuits de type ET (0 dominant).

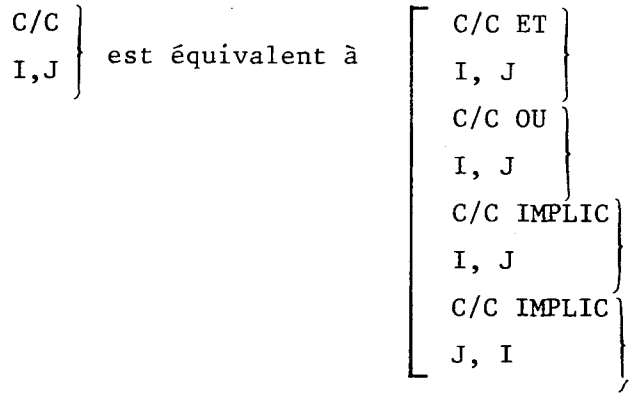
B.6 - C/C OU

puis une liste de couples d'identificateurs de connexions (même forme que ci-dessus) définissant les courts-circuits de type OU (1 dominant).

B.7 - C/C IMPLIC

puis une liste de couples d'identificateurs de connexions (même forme que ci-dessus) définissant les courts-circuits de type implication. Dans ce cas, l'ordre des connexions est significatif : la première connexion force sa valeur sur la seconde.

B.8 - Les instructions C/C ET, C/C OU et C/C IMPLIC peuvent être remplacées par une seule instruction C/C. Dans ce cas, pour chaque couple de connexions de la liste qui suit C/C, quatre défauts de court - circuit seront définis.



Si l'instruction EMPLOI est présente, elle est suivie d'une suite d'instructions donnant la condition logique correspondant à chacune des restrictions.

Ces conditions sont données sous forme d'une expression booléenne, en somme de produits éventuellement complémentée, dont les termes sont des identificateurs de connexions, à maintenir à la valeur 1 en permanence.

Symboles : + : ou logique
. : et logique
- : complémentation : ne porte que sur l'identificateur suivant ou sur l'ensemble de l'expression qui est alors précédée d'un point
$: à considérer en valeur initiale : ne porte que sur l'identificateur suivant et ne peut être appliqué qu'à une sortie d'élément séquentiel standard.

D) Une carte FIN termine la description topologique.

3. DESCRIPTION DES DEFAUTS

Activée par l'instruction
* DEFAUTS
suivie obligatoirement de

A) NOMCIRCUIT : nom du circuit pour lequel on décrit les défauts. La description topologique de NOMCIRCUIT doit obligatoirement être stockée en bibliothèque.

B) On retrouve ensuite une ou plusieurs fois une séquence d'instructions définissant des ensembles de défauts. S'il n'y a qu'un ensemble défini et si l'on effectue une commande d'ENCHAINEMENTS sur le circuit, la table TDD ne pourra qu'être vidée ou identique à TDR.

B.1 - ENSEMBLE : cette instruction indique la définition d'un nouvel ensemble de défauts. (Les différents ensembles seront ultérieurement répérés par leur numéro relativement à l'ordre dans lequel ils ont été définis).

Eléments séquentiels standards :

	Nombre d'entrées	Nombre de sorties
BNAND	2	2
BNOR	2	2
BINTER	3	1
BPNOR	3	2
BPINTER	5	1
BRSNOR	4	2
BMINT	2	1

B.3 - Il est possible d'utiliser une description itérative d'une suite de modules de même fonction (module standard ou bloc). Les instructions précédentes deviennent :

B.3.1 - TYPE/n,m/ (liste des entrées - liste des sorties).

B.3.2 - NOM/n,m/ - TYPE (liste des entrées - liste des sorties)

NOM est limité à 2 caractères. Ce sera l'identificateur du bloc itératif.

n et m sont les bornes de variation de l'indice d'itération.

n peut être omis, il sera alors pris égal à 1.

Les listes d'entrées et de sorties peuvent comporter :

. soit des identificateurs de connexions habituels (cas où la connexion entre dans tous les modules de l'itération)

. soit des identificateurs répétitifs, sous la forme

NOM ## I

ou NOM ## I + K

I symbolise l'indice de répétition

K est un nombre entier positif, à ajouter à l'indice.

Les identiticateurs itératifs sont générés et peuvent être utilisés dans une autre partie de la description.

B.4 - Une carte FIN termine la description des modules.

C) EMPLOI : cette instruction (facultative) indique le début de l'introduction des restrictions d'emploi.

Si l'instruction EMPLOI est absente, aucune restriction à l'emploi du circuit n'est introduite, autre que celles internes aux blocs appelés, qui sont automatiquement recopiées.

B) Une suite d'instructions décrit les modules constituant le circuit sous l'une des deux formes :

B.1 - TYPE (liste des entrées - liste des sorties)

ou

B.2 - NOM-TYPE (liste des entrées - liste des sorties)

- TYPE est le nom d'un module standard ou celui d'un bloc déjà stocké en bibliothèque.

- NOM est un identificateur (facultatif) individualisant le module dans le circuit (uniquement dans le cas des blocs bibliothèques). Si le nom est omis (forme B.1), un nom correspondant au numéro de module sera généré.

- Liste des entrées et liste des sorties : suite d'identificateurs de connexions séparés par des virgules. L'ordre relatif à l'intérieur d'une liste est sans signification pour les opérateurs combinatoires, mais est représentatif du rôle particulier des entrées ou des sorties pour les éléments séquentiels.
  Dans le cas des éléments séquentiels et des blocs bibliothèques, les nombres d'entrée-sortie sont fixés.
  Il est toutefois possible de ne pas utiliser une ou plusieurs sorties, à condition :
  . qu'une au moins des sorties du module existe
  . que l'emplacement de la (ou des) sortie(s) manquante(s) soit repéré par deux séparateurs consécutifs.
  Ex : BNAND (A,B-C); BNAND (A,B-,D)

Liste des modules standards :

Opérateurs combinatoires :

	Nombre d'entrées	Nombre de sorties
ET	≥ 1	1
ETNON	≥ 1	1
OU	≥ 1	1
OUNON	≥ 1	1
OUEX (disjonction)	2	1
EGAL (conjonction)	2	1
RAMIF (ramification)	1	≥ 1

2. DESCRIPTION DES CIRCUITS

1. INTRODUCTION

Le module de description est activé par la commande

** DESCRIPTION

La description d'un circuit comporte deux parties :

. Description topologique : blocs logiques constituant le circuit et restrictions d'emploi.

. Description des défauts : permet de définir explicitement les défauts potentiels du circuit.

La description topologique d'un circuit est stockée en bibliothèque si

. aucune erreur n'a été détectée lors de l'analyse de cette description

. Aucun circuit de même nom n'est déjà présent dans la bibliothèque.

La description des défauts d'un circuit est facultative (en l'absence de cette commande, aucun défaut ne pourra être considéré sur le circuit). Elle n'est possible que pour un circuit dont la description topologique est déjà stockée en bibliothèque. Une nouvelle description de défauts pour un même circuit efface la précédente.

2. DESCRIPTION TOPOLOGIQUE

Activée par l'instruction :

* TOPOLOGIE

Les instructions suivantes sont :

A) NOMCIRCUIT (liste des entrées - liste des sorties)

NOMCIRCUIT : identificateur du circuit. Le circuit sera stocké sous ce nom en bibliothèque.

Liste des entrées : suite d'identificateurs de connexions, séparés par des virgules, précisant les entrées primaires du circuit.

Liste des sorties : suite d'identificateurs de connexions, séparés par des virgules, précisant les sorties primaires du circuit.

Chacune des trois commandes de test manipule une ou plusieurs sous-séquences. On dispose de deux ordres (REPRISE et NOREPRISE) pour les organiser en séquences.
D'autre part, il est possible de fixer la longueur maximumd'une séquence. Le programme se charge alors de gérer les indéterminations d'état initial.

5. REGLES SYNTAXIQUES GENERALES

- Les commandes du programme sont organisées en plusieurs niveaux :
  . Les commandes d'appel d'un module débutent par **
  . Les commandes de traitement internes aux modules débutent par *
- Les blancs ne sont pas significatifs, quel que soit le type de carte.
- Une liste d'éléments est une suite d'éléments séparés par des virgules.
- Les cartes commençant par un $ en colonne 1 sont uniquement reproduites (cartes commentaires).
- Un ";" comme dernier caractère non blanc d'une carte indique qu'il y a une carte suite.
- Tout programme commence par une carte de titre et s'achève par une carte ** FIN.
- Un identificateur comporte au plus huit caractères alphanumériques sauf indication contraire.

3. GESTION DES TABLES DE DEFAUTS

Pour simplifier leur manipulation en cours d'exécution, les défauts sont regroupés dans deux tables :

. La première (TDR) est la table de référence. Elle comporte tous les défauts qui pourront affecter le circuit. Elle est définie une fois pour toutes lors de la description du circuit.

. La seconde (TDD) est une partie de la précédente : elle comporte la sous-liste des défauts dont on souhaite la détection à un moment donné en cours d'enchaînements. Cette table TDD peut être modifiée en cours d'enchaînements à partir des éléments (ensembles de défauts) qui auront été préparés lors de la description.

Pour assurer la gestion de TDD, lors de la description des défauts d'un circuit, on définit un certain nombre d'ensembles de défauts (non nécessairement exclusifs). La réunion de tous les ensembles de défauts d'un même circuit forme automatiquement la table TDR. Quant à la table TDD, elle sera formée (puis éventuellement modifiée) en cours d'ENCHAINEMENTS en indiquant une liste d'ensembles parmi ceux décrits.

4. GESTION DES SEQUENCES DE TEST

Définitions

Une combinaison d'entrées est un ensemble de valeurs logiques appliquées simultanément aux entrées primaires du circuit.
Une séquence est une suite ordonnée de combinaisons d'entrées, pour laquelle l'état des sorties a été calculé en supposant l'état initial du circuit totalement indéterminé (état X sur toutes les connexions).

Une sous-séquence est une suite ordonnée de combinaisons d'entrées pour laquelle l'état des sorties a été calculé, en partant d'un état initial qui est le dernier état atteint par le circuit au cours du déroulement du programme. La sous-séquence n'a donc pas de sens par elle-même.

Vis-à-vis du module d'édition (donc du résultat final), le test est un ensemble de séquences (éventuellement réduit à une seule). L'état indéterminé étant supposé en tête des séquences, chacune d'entre elles pourra être appliquée isolément au circuit; l'ordre des séquences pourra être permuté. Par contre, aucune modification n'est possible à l'intérieur d'une séquence.

B.9 - BLOCS : cette instruction (facultative) permet de recopier, dans l'ensemble de défauts en cours de définition, certains des ensembles de défauts relatifs aux blocs qui ont été incorporés.

La carte BLOC est suivie d'une liste de noms de modules (donnés ou générés) indicés par le numéro de l'ensemble de défauts à considérer.

Ainsi, si la description topologique de NOMCIRCUIT a utilisé l'instruction NOM-TYPE (liste 1 - liste 2) signifiant l'insertion d'un module dont le nom bibliothèque est TYPE et identifié par NOM à l'intérieur de NOMCIRCUIT, NOM (1) signifie : pour le module NOM, incorporation du I^e ensemble de défauts de TYPE.

C) Une carte FIN termine la description des défauts.

3. GESTION DE LA BIBLIOTHEQUE

Le module de gestion de bibliothèque est activé par la commande :

** BIBLIOTHEQUE

Les commandes sont

1. * CREATION

   Initialise une bibliothèque vide (efface toute l'ancienne bibliothèque si celle-ci existait). Cette commande doit avoir été utilisée avant la première description.

2. * EDITE

   Permet l'impression de la description des circuits de la bibliothèque.

3. * REORGANISER

   Permet de retasser la bibliothèque.

4. * EFFACER

   Suivie d'une liste de noms de circuits.

   Permet de supprimer la description de certains circuits de la bibliothèque. Le nom des circuits effacés est alors disponible.

   Une carte * FIN termine la sous-commande de * EFFACER.

30) ROTH J.P. : "Programmed Algorithm to Compute Test to Detect and Distinguish Between Failures in Logic Circuits", IEEE Trans. Comp. Vol. EC 16 n° 5, pp. 567-580, Oct. 1967.

31) SUSSKIND A.M. : "Diagnosis for Logic Networks", IEEE Spectrum, pp.40-47 Oct. 1973.

32) SZYGENDA S.A. : "Tegas 2 : Anatomy of a General Purpose Test Generation and Simulation System for Digital Logic", 9th Design Automation Workshop, pp. 116-127 (1972).

33) TABARLY J.H. : Contribution à la Synhtèse de Tests Complets de Circuits Digitaux par Observation - Commande - Programme DEDALE", Thèse EEA, Univ. P. Sabatier, Toulouse, juillet 1974.

34) THOMAS J.J. "Automated Diagnostic Test Programs for Digital Networks", Computer Design, Vol. 10 n° 8, pp. 63-67, Aug. 1971.

35) THUEL J. : "Etude de l'Influence des Pannes Multiples dans les Réseaux Logiques et Application à leur Détection", Thèse Docteur Ingénieur USMG et INPG à Grenoble, janvier 1974.

36) TOTH A., CHIP HOLT. : "Automated Data Base Driven Digital Testing", Computer, pp. 13-19, Jan. 1974.

37) TULLOUE R., ZIRPHILE J. :"Simulation Déductive Etendue à trois Valeurs Logiques", Note Interne à paraître, Thomson-CSF/DIS, 33, rue de Vouillé, 75015 PARIS.

38) TYURIN A.V. : "Design of Diagnostic Test for Synchronous Sequential Circuits", Automation and Remote Control, Vol. C35 n° 7, pp. 1132-1140, Dec. 1974.

39) VERDILLON A. : "Pannes dans les Réseaux Acycliques", Thèse de 3e Cycle USMG, Décembre 1972.

4. VALIDATION DE LA DESCRIPTION TOPOLOGIQUE

La validation de la description topologique est une simulation préalable, sans propagation de liste de défauts. Son emploi est facultatif. Le résultat de la validation est sans effet sur la suite du déroulement d'un programme.

Le module de validation est activé par :

** VALIDATION

suivie de

A) NOMCIRCUIT (obligatoire)

nom sous lequel le circuit à valider est stocké en bibliothèque.

B) Liste de valeurs

On donne ainsi la grille des entrées-sorties du circuit : la simulation sera faite en appliquant les entrées et une vérification des sorties calculées par rapport aux sorties attendues est faite, provoquant l'impression du résultat.

DESCRIPTION VALIDE

ou DESCRIPTION NON VALIDE

La grille d'entrées-sorties est donnée par une suite d'instructions (une par affichage).

Chaque instruction est une suite de valeurs logiques 0, 1 ou X séparées par des virgules, autant que le nombre total de pattes du circuit : en tête les entrées, puis les sorties, dans l'ordre de la description topologique (première instruction).

Une valeur X en entrée ou en sortie n'a pas la même signification :

- En entrée, X est traitée comme une véritable valeur logique.
- En sortie, X signifie que la valeur n'est pas à prendre en compte pour la validation .

3. GESTION DES TABLES DE DEFAUTS

Pour simplifier leur manipulation en cours d'exécution, les défauts sont regroupés dans deux tables :

. La première (TDR) est la table de référence. Elle comporte tous les défauts qui pourront affecter le circuit. Elle est définie une fois pour toutes lors de la description du circuit.

. La seconde (TDD) est une partie de la précédente : elle comporte la sous-liste des défauts dont on souhaite la détection à un moment donné en cours d'enchaînements. Cette table TDD peut être modifiée en cours d'enchaînements à partir des éléments (ensembles de défauts) qui auront été préparés lors de la description.

Pour assurer la gestion de TDD, lors de la description des défauts d'un circuit, on définit un certain nombre d'ensembles de défauts (non nécessairement exclusifs). La réunion de tous les ensembles de défauts d'un même circuit forme automatiquement la table TDR. Quant à la table TDD, elle sera formée (puis éventuellement modifiée) en cours d'ENCHAINEMENTS en indiquant une liste d'ensembles parmi ceux décrits.

4. GESTION DES SEQUENCES DE TEST

Définitions

Une combinaison d'entrées est un ensemble de valeurs logiques appliquées simultanément aux entrées primaires du circuit.
Une séquence est une suite ordonnée de combinaisons d'entrées, pour laquelle l'état des sorties a été calculé en supposant l'état initial du circuit totalement indéterminé (état X sur toutes les connexions).

Une sous-séquence est une suite ordonnée de combinaisons d'entrées pour laquelle l'état des sorties a été calculé, en partant d'un état initial qui est le dernier état atteint par le circuit au cours du déroulement du programme. La sous-séquence n'a donc pas de sens par elle-même.

Vis-à-vis du module d'édition (donc du résultat final), le test est un ensemble de séquences (éventuellement réduit à une seule). L'état indéterminé étant supposé en tête des séquences, chacune d'entre elles pourra être appliquée isolément au circuit; l'ordre des séquences pourra être permuté. Par contre, aucune modification n'est possible à l'intérieur d'une séquence.

Cette table est sauvegardée en même temps que les fichiers REPRISE et RESULTATS. Elle est donc inchangée tant que l'instruction NEW ou une nouvelle commande * DAD n'est pas rencontrée.

Option : TOUS signifie que la table TDD contient tous les défauts (identique à TDR).

L'exécution de la commande *DAD provoque l'impression de la liste des défauts de la nouvelle table TDD en indiquant pour chacun d'eux s'il a déjà ou non été détecté.

3. ANALYSE D'UNE SOUS-SEQUENCE

Activée par la commande
* ANALYSE
suivie de

A) REPRISE
ou NOREPRISE
ou BLOC = K

REPRISE signifie que la sous-séquence à analyser doit être accolée à la fin de la séquence précédente : l'état initial du circuit est pris sur le fichier REPRISE.

NOREPRISE signifie le début d'une nouvelle séquence : l'état initial du circuit est totalement indéterminé.

B) Les options REPRISE et NOREPRISE doivent être suivies d'un ensemble d'instructions précisant, sous forme d'une liste de 0 et de 1, les états logiques appliqués aux entrées (une par affichage). Cette liste peut éventuellement être précédée d'un indicateur NT, qui signifie que la mesure des sorties ne sera pas effectuée pour cet affichage. Aucun défaut ne sera donc considéré comme détecté par un affichage comportant l'indicateur NT.

Un bloc est l'ensemble des affichages générés pour une même table de défauts à détecter. (Il peut contenir plusieurs séquences).
BLOC = K signifie que l'on désire connaître les défauts détectés sur une nouvelle table de défauts TDD par la ou les séquences générées précédemment pour une autre table. K est le numéro d'occurence de la sous-commande * DAD correspondante.

21) MANGE D. : "Modèles Asynchrones des Bascules Bistables", Cahiers de la C.S.L, Ecole Polytechnique de Lausanne n° 5, pp. 256-286, Oct. 1973.

22) MEY K.C.Y. : "Bridging and Stuck at Faults", International Symposium on Fault Tolerant Computing, pp. 91-94, (1973).

23) MUTH P. : "A Nine Valued Circuit Model to Generate tests for Sequential Circuits", International Symposium on Fault Tolerant Computing, pp. 43-49, (1975) et "A Nine Valued Circuit Model for Test Generation", IEEE Trans. Comp. Vol. C25 n° 6, pp. 630-636, june 1976.

24) PIGNAL P. et all : "Une Méthode de Test Automatique pour les Ensembles Logiques", l'Onde Electrique, Vol. 48 n° 500, pp. 997-1003, Nov. 1968 et n° 501, pp. 1081-1088, Dec. 1968.

25) PLITMAN A.D. : "Design of a Sequential Test for a Multiple Fault in Asynchronous Circuits", Automation and Remote Control, pp. 1123-1131, July 1974.

26) PUTZOLU G.R., ROTH J.P. : "A Heuristic Algorithm for the Testing of Asynchronous Circuits", IEEE Trans. Comp. Vol. C20, June 1971.

27) RAULT J.C. : "La Détection et la Localisation des Défauts dans les Circuits Logiques", International Symposium on Fault Tolerant Computing, pp. 17-23 (1975).

28) RAULT J.C. : "Bibliographie sur la Détection et la Localisation des Défauts dans les Circuits Logiques", Rapport Thomson CSF, 33, rue de Vouillé 75015 PARIS.

29) ROBISON J.M. : "Applications of Logic Simulation in Design Automation at Texas Instrument", 9th Design Automation Workshop, pp. 138-143 (1972).

10) CHANG H.Y. et all : "LAMP System Description", B.S.T.J. Vol. 53 n° 8, pp. 1431-1449, Oct. 1974.

11) CHANG H.Y. : "Comparison of Parallel and Deductive Fault Simulation Methode", IEEE Trans. Comp., Vol. C23 n° 11, pp.1132-1138, Nov.1974.

12) CHAPELL S.G. : "Automatic Test Generation for Asynchronous Digital Circuits", B.S.T.J., pp. 1417-1503, Oct. 1974.

13) FRIEDMAN A.D. : "Diagnosis of Short-Circuit Faults in Combinational Circuits", IEEE Trans. Comp. Vol. C23 n° 7, July 1974.

14) FRIEDMAN A.D., MENON P.R. : "Fault Detection in Digital Circuits", Prentice Hall 1971.

15) FUNATSU S. et all : "Test Generation in Japan", 12th Design Automation Workshop, pp. 114-122 (1975).

16) KAMAL S. : "An Approach to the Diagnosis of Intermittent Faults", IEEE Trans. Comp. Vol. C24 n° 5, May 1976.

17) KARIBSKII V.V. : "Construction of an Input Sequence that Detects a Given Failure of aDiscrete Device", Automation and Remote Control Vol. 33 n° 5, pp. 843-851, May 1972.

18) KUBO H. : "A Procedure for Generating Test Sequences to Detect Sequential Failures", NEC Research and Development, n° 12, pp. 69-78, Oct. 1968.

19) KREUWELS WG.J. :"Structural Testing of Digital Circuits", Philips Technical Review, Vol. 35 n° 10, pp. 261-270 (1975).

20) PREISS R.J. : "Fault Test Generation", Chapter 7 Design Automation of Digital Systems Prentice Hall, M.A. BREUER Ed. (1972).

3. GESTION DE LA BIBLIOTHEQUE

Le module de gestion de bibliothèque est activé par la commande :

** BIBLIOTHEQUE

Les commandes sont

1. * CREATION

   Initialise une bibliothèque vide (efface toute l'ancienne bibliothèque si celle-ci existait). Cette commande doit avoir été utilisée avant la première description.

2. * EDITE

   Permet l'impression de la description des circuits de la bibliothèque.

3. * REORGANISER

   Permet de retasser la bibliothèque.

4. * EFFACER

   Suivie d'une liste de noms de circuits.

   Permet de supprimer la description de certains circuits de la bibliothèque. Le nom des circuits effacés est alors disponible.

   Une carte * FIN termine la sous-commande de * EFFACER.

BIBLIOGRAPHIE

1) AKERS S.B. Jr : "A Logic System for Fault Test Generation", International Symposium on Fault Tolerant Computing, pp.37-42, may 1975 et IEEE Trans. Comp. Vol C25 n° 6, pp.613-620, june 1976.

2) ARIMA T. et all : "A New Heuristic Test Generation Algorithm for Sequential Circuits", 11th Design Automation Workshop, pp.169-176, (1974).

3) BENNETS G., LEWIN D.W. : "Fault Diagnosis of Digital Systems - A Review", COMPUTER, pp. 12-20, Jul/Aug 1971.

4) BOUTE R., MC CLUSKEY E.J. : "Fault Equivalence in Sequential Machines", COMPUTER and Automata pp.483-507, Vol. C21, New York, April 1971.

5) BOURICIUS W.G. et all : "Algorithms for Detection of Faults in Logic Circuits", IEEE Trans. Comp. Vol. C20 n° 11, pp.1258-1264, Nov 1971 et IBM Thomas J. Watson Research Center, Oct. 1970.

6) BREUER M.A : "The Effect of Races, Delays and Delay - Faults on Test Generation", IEEE Trans. Comp. Vol. C23 n° 10, pp.1078-1092, Oct. 1974.

7) BREUER M.A. : "Generation of Fault Detection Tests for Sequential Circuits", International Symposium on Fault-Tolerant Computing, pp. 18-21, March 1971.

8) BREUER M.A. et all : "Identification of Multiple Stuck Type Faults in Combinational Networks", IEEE Trans. Comp. Vol. C25, n° 1, pp. 44-54, Jan. 1976.

9) BREUER M.A. : "Testing for Intermittent Faults in Digital Circuits", IEEE Trans. Comp., Vol C22 n° 3, pp. 241-246, March 1973.

5. ENCHAINEMENT DES COMMANDES DE TEST

1. INTRODUCTION

Le module d'enchaînement est activé par :

** ENCHAINEMENTS

suivi de

A) NOMCIRCUIT

puis

B) New

LONGUEUR = K (facultatif)

ou OLD

NOMCIRCUIT est le nom sous lequel le circuit étudié est stocké en bibliothèque. Les deux inscriptions (topologie et défauts) doivent être présentes.

La carte NEW initialise les fichiers de REPRISE et de RESULTATS : elle indique donc le début de l'étude du test d'un circuit.

Si LONGUEUR est spécifié, le test sera automatiquement découpé en séquences de K affichages.

La carte OLD indique que les fichiers de REPRISE et de RESULTATS existent déjà et sont à conserver. Il s'agit donc de la continuation d'une étude précédente. L'état de ces fichiers correspond à la dernière opération signalée par un message d'exécution au cours du passage précédent (voir ci-dessous).

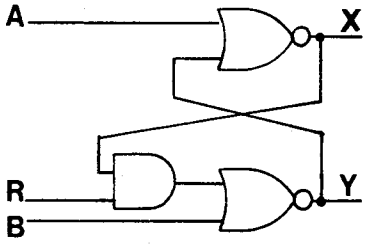
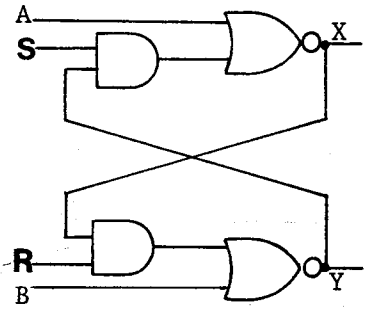
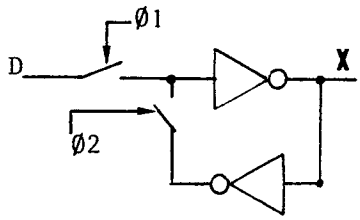
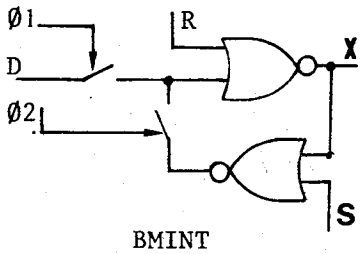
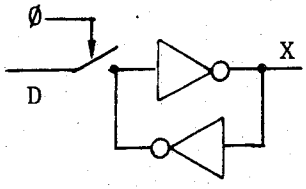
2. CONSTRUCTION DE LA TABLE DES DEFAUTS A DETECTER

Est activée par la commande

* DAD

suivie d'une liste de numéros d'ensembles de défauts

La table des défauts à détecter (TDD) est alors constituée par la réunion de tous les défauts présents dans ces ensembles.

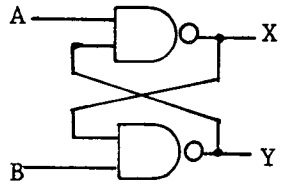
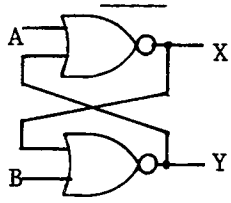
Opérateur	Blocage entre l'entrée	et la sortie	Condition de blocage
BPNOR	A	X	$\overline{B}.(\overline{R}+\overline{X^*}) = 1$
	A	Y	$B + Y^* + \overline{R} = 1$
	B	X	$A + R.X^* = 1$
	B	Y	$\overline{A}.R.\overline{Y^*} = 1$
	R	X	$A + B + \overline{X^*} = 1$
	R	Y	$A + B + Y^* = 1$
BRSNOR	A	X	$S.\overline{B}.(\overline{R} + \overline{X^*} = 1$
	A	Y	$B + S.Y^* + \overline{R} = 1$
	B	X	$A + R.X^* + \overline{S} = 1$
	B	Y	$R.\overline{A}.(\overline{S} + \overline{Y^*}) = 1$
	R	X	$A + B + \overline{S} + \overline{X^*} = 1$
	R	Y	$A + B + S.Y^* = 1$
	S	X	$A + B + R.X^* = 1$
	S	Y	$A + B + \overline{R} + \overline{Y^*} = 1$
BINTER	D	X	$Ø2.\overline{Ø1} = 1$
	Ø1	X	$D.\overline{X^*} + \overline{D}\ X^* = 1$
	Ø2	X	$D.\overline{X^*} + \overline{D}\ X^* = 1$
BPINTER	D	X	$R + Ø2.\overline{Ø1} = 1$
	Ø1	X	$R + S.\overline{D} + \overline{D}\ \overline{X^*} + \overline{D}\ X^* = 1$
	Ø2	X	$R + S.\overline{D} + \overline{D}\ X^* + D\ \overline{X^*} = 1$
	R	X	$Ø1.\overline{Ø2}\ D + Ø1\ Ø2(S + X^*)=1$
	S	X	$R + Ø1\ Ø2\ X^* = 1$
BMINT	D	X	$\overline{Ø} = 1$
	Ø	×	$D\ \overline{X^*} = \overline{D}\ X^* = 1$

Pour les portes élémentaires, les conditions de blocage sont triviales : Nous considérons la condition de blocage entre l'entrée A et la sortie S d'une porte à n entrées A, B, C...

Porte	Condition de blocage
ET, ET NON	$\overline{B} + \overline{C} + \ldots = 1$
OU, OU NON	$B + C + \ldots = 1$

Il n'y a pas de blocage possible sur les opérateurs de disjonction et de conjonction ainsi que sur les opérateurs à une seule entrée.

Pour les opérateurs standards séquentiels, les conditions de blocage sont plus sophistiquées.
Nous noterons X^* la valeur initiale d'une variable interne.

Opérateur	Blocage entre l'entrée	et la sortie	Condition de blocage
BNAND 	A	X	$B\ X^* = 1$
	A	Y	$\overline{B} + \overline{Y^*} = 1$
	B	X	$\overline{A} + \overline{X^*} = 1$
	B	Y	$A\ Y^* = 1$
BNOR 	A	X	$\overline{B}.\overline{X^*} = 1$
	A	Y	$B + Y^* = 1$
	B	X	$A + X^* = 1$
	B	Y	$\overline{A}.\overline{Y^*} = 1$

4. GENERATION ALEATOIRE

Activée par la commande

* GEAL

suivie de

A) REPRISE — Même signification
ou NOREPRISE — que ci-dessus

B) TAUX =...

Indique le pourcentage de couverture des défauts à atteindre, relativement à la table TDD actuelle. Le pourcentage correspond au cumul des défauts détectés depuis le début de l'étude (instruction NEW), il inclut donc les défauts éventuellement déjà détectés avant * GEAL.

C) LIM =...

Indique le nombre maximum de répétitions de la sous-séquence (provoque l'arrêt de GEAL même si le taux n'est pas atteint). (Instruction facultative).

D) Une suite d'instructions, donnant l'état logique à appliquer aux entrées sous forme d'une liste de valeurs 0, 1 ou A, définit la sous-séquence.

A signifie que la valeur (0 ou 1) doit être tirée aléatoirement.
Chaque affichage peut être précédé de NT comme précédemment.

5. SYNTHESE - ANALYSE

Activée par la commande

* SYAN

suivie de

A) REPRISE — Comme
ou NOREPRISE — précédemment

B) Temps =...

Indique le temps CPU maximum (en secondes) alloué à chaque synthèse (instruction facultative).

C) LIM = ...

Indique un nombre maximum de combinaisons d'entrées autorisé pour chacune des sous-séquences synthétisées. (Instruction facultative).

ANNEXE II

CONDITIONS DE BLOCAGE D'UN CYCLE PAR UN OPERATEUR

D) CONSTANT (instruction facultative)

Si l'instruction est présente, elle est suivie de l'une au moins des instructions

ETAT 0

puis liste des entrées primaires à maintenir constamment à 0

ETAT 1

puis liste des entrées primaires à maintenir constamment à 1.

E) RETESTER (instruction facultative)

suivi de : liste de numéros de défauts

Cet ordre permet de redemander la synthèse du test de certains défauts difficiles non détectés lors d'un premier passage. (On peut alors donner à la synthèse des paramètres limites plus grands).

EXEMPLE :

```
** ENCHAINEMENTS
   NOMCIRCUIT
   NEW
 * DAD
   1
 * SYAN
   NOREPRISE        Ce premier passage laisse par exemple les
   TEMPS = 5        défauts 15 et 20 non détectés.
** FIN


** ENCHAINEMENT
   NOMCIRCUIT
   OLD
 * SYAN
   REPRISE
   TEMPS = 10       On accorde dans ce deuxième passage plus de
   RETESTER         temps à la synthèse pour les deux défauts
   20, 15           difficiles.
** FIN
```

* LISTER

BLOC = K (facultatif)

. impression de la table des défauts à détecter du bloc considéré

. impression pour le bloc considéré, et pour chaque défaut, de l'historique du test :

- défauts détectés ou non
- défauts étudiés ou non
- défauts échoués en synthèse (échec total, échec dû au temps, échec dû au nombre de vecteurs).

* FUSIONNER

BLOC = K, L

ou

BLOC = K

Cette commande n'est utilisable que si deux blocs contiennent les mêmes affichages pour des ensembles de défauts différents. On a alors création d'un nouveau bloc par fusion pour chaque affichage et chaque sortie des listes de défauts correspondantes. Le résultat est donc le même que si le nouveau bloc avait été généré sur la réunion des deux ensembles de défauts.

6. EDITION DES RESULTATS

1. INTRODUCTION

Le module d'édition est activé par

** EDITION

suivi de

NOMCIRCUIT

Les commandes décrites dans les paragraphes suivants concernent uniquement l'impression des divers résultats.

Vis-à-vis du module d'édition, les résultats sont organisés par séquences, qui sont des parties de la séquence de test pouvant être isolées. La manière dont une séquence a été obtenue (types de commandes utilisées, en un ou plusieurs passages dans ENCHAINEMENTS) est ignorée.

On rappelle qu'un bloc est l'ensemble des affichages générés pour une même table de défauts à détecter (TDD) : la définition d'un nouveau bloc correspond à la sous-commande * DAD. Le numéro d'un bloc est le numéro d'occurence de sa commande * DAD correspondante.

Dans toutes les sous-commandes suivantes, lorsqu'un numéro de bloc n'est pas spécifié, il s'agit du dernier généré.

2. COMMANDES

* GRILLE E/S

  BLOC = K (facultatif)

  . impression des connexions d'entrée sortie

  . impression pour le bloc considéré des affichages avec la valeur des sorties.

* DIAGNOSTIC

  BLOC = K (facultatif)

  . impression pour le bloc considéré, pour chaque affichage et pour chaque sortie, des défauts détectés et de ceux provoquant une indétermination.

UNIVERSITE SCIENTIFIQUE
ET MEDICALE DE GRENOBLE

Monsieur Gabriel CAU : Président
Monsieur Pierre JULLIEN : Vice-Président

MEMBRES DU CORPS ENSEIGNANT DE L'U.S.M.G.

PROFESSEURS TITULAIRES

MM.	ARNAUD Paul	Chimie
	AUBERT Guy	Physique
	AYANT Yves	Physique approfondie
Mme	BARBIER Marie-Jeanne	Electrochimie
MM.	BARBIER Jean-Claude	Physique Expérimentale
	BARBIER Reynold	Géologie appliquée
	BARJON Robert	Physique nucléaire
	BARNOUD Fernand	Biosynthèse de la cellulose
	BARRA Jean-René	Statistiques
	BARRIE Joseph	Clinique chirurgicale
	BEAUDOING André	Clinique de Pédiatrie et Puériculture
	BERNARD Alain	Mathématiques Pures
Mme	BERTRANDIAS Françoise	Mathématiques Pures
MM.	BERTRANDIAS Jean-Paul	Mathématiques Pures
	BEZES Henri	Pathologie chirurgicale
	BLAMBERT Maurice	Mathématiques Pures
	BOLLIET Louis	Informatique (IUT B)
	BONNET Georges	Electrotechnique
	BONNET Jean-Louis	Clinique ophtalmologique
	BONNET-EYMARD Joseph	Clinique gastro-entérologique
Mme	BONNIER Marie-Jeanne	Chimie générale
MM.	BOUCHERLE André	Chimie et toxicologie
	BOUCHEZ Robert	Physique nucléaire
	BOUSSARD Jean-Claude	Mathématiques Appliquées
	BOUTET DE MONTVEL Louis	Mathématiques Pures
	BRAVARD Yves	Géographie
	CABANEL Guy	Clinique rhumatologique et hydrologique
	CALAS François	Anatomie
	CARLIER Georges	Biologie végétale
	CARRAZ Gilbert	Biologie animale et pharmacodynamie
	CAU Gabriel	Médecine légale et toxicologie
	CAUQUIS Georges	Chimie organique
	CHABAUTY Claude	Mathématiques Pures
	CHARACHON Robert	Clinique Oto-rhino-laryngologique
	CHATEAU Robert	Clinique de neurologie
	CHIBON Pierre	Biologie animale
	COEUR André	Pharmacie chimique et chimie analytique
	CONTAMIN Robert	Clinique gynécologique
	COUDERC Pierre	Anatomie pathologique
Mme	DEBELMAS Anne-Marie	Matière médicale
MM.	DEBELMAS Jacques	Géologie générale
	DEGRANGE Charles	Zoologie
	DELORMAS Pierre	Pneumophtisiologie

MM.	DEPORTES Charles	Chimie minérale
	DESRE Pierre	Métallurgie
	DESSAUX Georges	Physiologie animale
	DODU Jacques	Mécanique appliquée (IUT A)
	DOLIQUE Jean-Michel	Physique des plasmas
	DREYFUS Bernard	Thermodynamique
	DUCROS Pierre	Cristallographie
	DUGOIS Pierre	Clinique de dermatologie et syphiligraphie
	GAGNAIRE Didier	Chimie physique
	GALLISSOT François	Mathématiques Pures
	GALVANI Octave	Mathématiques Pures
	GASTINEL Noël	Analyse numérique
	GAVEND Michel	Pharmacologie
	GEINDRE Michel	Electroradiologie
	GERBER Robert	Mathématiques Pures
	GERMAIN Jean-Pierre	Mécanique
	GIRAUD Pierre	Géologie
	JANIN Bernard	Géographie
	KAHANE André	Physique générale
	KLEIN Joseph	Mathématiques Pures
	KOSZUL Jean-Louis	Mathématiques Pures
	KRAVTCHENKO Julien	Mécanique
	KUNTZMANN Jean	Mathématiques Appliquées
	LACAZE Albert	Thermodynamique
	LACHARME Jean	Biologie végétale
Mme	LAJZEROWICZ Janine	Physique
MM.	LAJZEROWICZ Joseph	Physique
	LATREILLE René	Chirurgie générale
	LATURAZE Jean	Biochimie pharmaceutique
	LAURENT Pierre-Jean	Mathématiques Appliquées
	LEDRU Jean	Clinique médicale B
	LLIBOUTRY Louis	Géophysique
	LOISEAUX Pierre	Sciences nucléaires
	LONGEQUEUE Jean-Pierre	Physique nucléaire
	LOUP Jean	Géographie
Mlle	LUTZ Elisabeth	Mathématiques Pures
MM.	MALGRANGE Bernard	Mathématiques Pures
	MALINAS Yves	Clinique obstétricale
	MARTIN-NOEL Pierre	Clinique cardiologique
	MAZARE Yves	Clinique médicale A
	MICHEL Robert	Minéralogie et Pétrographie
	MICOUD Max	Clinique maladies infectieuses
	MOURIQUAND Claude	Histologie
	MOUSSA André	Chimie nucléaire
	MULLER Jean-Michel	Thérapeutique (Néphrologie)
	NEEL Louis	Physique du Solide
	OZENDA Paul	Botanique
	PAYAN Jean-Jacques	Mathématiques Pures
	PEBAY-PEYROULA Jean-Claude	Physique
	RASSAT André	Chimie systématique
	RENARD Michel	Thermodynamique
	REVOL Michel	Urologie
	RINALDI Renaud	Physique
	DE ROUGEMONT Jacques	Neuro-chirurgie
	SEIGNEURIN Raymond	Microbiologie et Hygiène
	SENGEL Philippe	Zoologie

MM.	SIBILLE Robert	Construction mécanique (IUT A)
	SOUTIF Michel	Physique générale
	TANCHE Maurice	Physiologie
	TRAYNARD Philippe	Chimie générale
	VAILLANT François	Zoologie
	VALENTIN Jacques	Physique nucléaire
	VAUQUOIS Bernard	Calcul électronique
Mme	VERAIN Alice	Pharmacie galénique
MM.	VERAIN André	Physique
	VEYRET Paul	Géographie
	VIGNAIS Pierre	Biochimie médicale
	YOCCOZ Jean	Physique nucléaire théorique

PROFESSEURS ASSOCIES

MM.	CLARK Gilbert	Spectrométrie physique
	CRABBE Pierre	CERMO
	ENGLMAN Robert	Spectrométrie physique
	HOLTZBERG Frédéric	Basses températures
	DEMBICKI Eugéniuz	Mécanique
	MATSUSHIMA Yozo	Mathématiques Pures

PROFESSEURS SANS CHAIRE

Mlle	AGNIUS-DELORD Claudine	Physique pharmaceutique
	ALARY Josette	Chimie analytique
MM.	AMBROISE-THOMAS Pierre	Parasitologie
	BELORIZKY Elie	Physique
	BENZAKEN Claude	Mathématiques Appliquées
	BIAREZ Jean-Pierre	Mécanique
	BILLET Jean	Géographie
	BOUCHET Yves	Anatomie
	BRUGEL Lucien	Energétique (IUT A)
	BUISSON René	Physique (IUT A)
	BUTEL Jean	Orthopédie
	COHEN ADDAD Pierre	Spectrométrie physique
	COLOMB Maurice	Biochimie
	CONTE René	Physique (IUT A)
	DEPASSEL Roger	Mécanique des fluides
	FONTAINE Jean-Marc	Mathématiques Pures
	GAUTHIER Yves	Sciences Biologiques
	GAUTRON René	Chimie
	GIDON Paul	Géologie et Minéralogie
	GLENAT René	Chimie organique
	GROULADE Joseph	Biochimie médicale
	HACQUES Gérard	Calcul numérique
	HOLLARD Daniel	Hématologie
	HUGONOT Robert	Hygiène et Médecine préventive
	IDELMAN Simon	Physiologie animale
	JOLY Jean-René	Mathématiques Pures
	JULLIEN Pierre	Mathématiques Appliquées
Mme	KAHANE Josette	Physique
MM.	KRAKOWIAK Sacha	Mathématiques Appliquées
	KUHN Gérard	Physique (IUT A)
	LE ROY Philippe	Mécanique (IUT A)
	LUU DUC Cuong	Chimie organique

MM.	MAYNARD Roger	Physique du solide
Mme	MINIER Colette	Physique (IUT A)
MM.	PELMONT Jean	Biochimie
	PERRIAUX Jean-Jacques	Géologie et Minéralogie
	PFISTER Jean-Claude	Physique du solide
Mlle	PIERY Yvette	Physiologie animale
MM.	RAYNAUD Hervé	M.I.A.G.
	REBECQ Jacques	Biologie (CUS)
	REYMOND Jean-Charles	Chirurgie générale
	RICHARD Lucien	Biologie végétale
Mme	RINAUDO Marguerite	Chimie macromoléculaire
MM.	ROBERT André	Chimie papetière
	SARRAZIN Roger	Anatomie et chirurgie
	SARROT-REYNAULD Jean	Géologie
	SIROT Louis	Chirurgie générale
Mme	SOUTIF Jeanne	Physique générale
MM.	STREGLITZ Paul	Anesthésiologie
	VIALON Pierre	Géologie
	VAN CUTSEM Bernard	Mathématiques Appliquées

MAITRES DE CONFERENCES ET MAITRES DE CONFERENCES AGREGES

MM.	AMBLARD Pierre	Dermatologie
	ARMAND Gilbert	Géographie
	ARMAND Yves	Chimie (IUT A)
	BACHELOT Yvan	Endocrinologie
	BARGE Michel	Neuro-chirurgie
	BARJOLLE Michel	M.I.A.G.
	BEGUIN Claude	Chimie organique
Mme	BERIEL Hélène	Pharmacodynamie
MM.	BOST Michel	Pédiatrie
	BOUCHARLAT Jacques	Psychiatrie adultes
Mme	BOUCHE Liane	Mathématiques (CUS)
MM.	BRODEAU François	Mathématiques (IUT B)
	CHAMBAZ Edmond	Biochimie médicale
	CHAMPETIER Jean	Anatomie et organogénèse
	CHARDON Michel	Géographie
	CHERADAME Hervé	Chimie papetière
	CHIAVERINA Jean	Biologie appliquée (EFP)
	CONTAMIN Charles	Chirurgie thoracique et cardio-vasculaire
	CORDONNIER Daniel	Néphrologie
	COULOMB Max	Radiologie
	CROUZET Guy	Radiologie
	CYROT Michel	Physique du solide
	DELOBEL Claude	M.I.A.G.
	DENIS Bernard	Cardiologie
	DOUCE Roland	Physiologie végétale
	DUSSAUD René	Mathématiques (CUS)
Mme	ETERRADOSSI Jacqueline	Physiologie
MM.	FAURE Jacques	Médecine légale
	FAURE Gilbert	Urologie
	GAUTIER Robert	Chirurgie générale
	GENSAC Pierre	Botanique
	GIDON Maurice	Géologie
	GROS Yves	Physique (IUT A)

MM. GUITTON Jacques	Chimie
HICTER Pierre	Chimie
IVANES Marcel	Electricité
JALBERT Pierre	Histologie
JUNIEN-LAVILLAVROY Claude	O.R.L.
KOLODIE Lucien	Hématologie
LE NOC Pierre	Bactériologie-virologie
LEROY Philippe	IUT A
MACHE Régis	Physiologie végétale
MAGNIN Robert	Hygiène et médecine préventive
MALLION Jean-Michel	Médecine du travail
MARECHAL Jean	Mécanique (IUT A)
MARTIN-BOUYER Michel	Chimie (CUS)
MICHOULIER Jean	Physique (IUT A)
NEGRE Robert	Mécanique (IUT A)
NEMOZ Alain	Thermodynamique
NOUGARET Marcel	Automatique (IUT A)
PARAMELLE Bernard	Pneumologie
PECCOUD François	Analyse (IUT B)
PEFFEN René	Métallurgie (IUT A)
PERRET Jean	Neurologie
PERRIER Guy	Géophysique - Glaciologie
PHELIP Xavier	Rhumatologie
RACHAIL Michel	Médecine interne
RACINET Claude	Gynécologie et obstétrique
RAMBAUD André	Hygiène et hydrologie
RAMBAUD Pierre	Pédiatrie
Mme RENAUDET Jacqueline	Bactériologie
MM. ROBERT Jean-Bernard	Chimie Physique
ROMIER Guy	Mathématiques (IUT B)
SHOM Jean-Claude	Chimie générale
STOEBNER Pierre	Anatomie pathologique
VROUSOS Constantin	Radiologie

MAITRE DE CONFERENCES ASSOCIES

M. COLE Antony	Sciences nucléaires

Fait à SAINT MARTIN D'HERES, AVRIL 1976.

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Président : M. Philippe TRAYNARD

Vice-Président : M. Pierre-Jean LAURENT

PROFESSEURS TITULAIRES

MM. BENOIT Jean	Radioélectricité
BESSON Jean	Electrochimie
BLOCH Daniel	Physique du solide
BONNETAIN Lucien	Chimie Minérale
BONNIER Etienne	Electrochimie et Electrométallurgie
BOUDOURIS Georges	Radioélectricité
BRISSONNEAU Pierre	Physique du solide
BUYLE-BODIN Maurice	Electronique
COUMES André	Radioélectricité
DURAND Francis	Métallurgie
FELICI Noël	Electrostatique
FOULARD Claude	Automatique
LESPINARD Georges	Mécanique
MOREAU René	Mécanique
PARIAUD Jean-Charles	Chimie-Physique
PAUTHENET René	Physique du solide
PERRET René	Servomécanismes
POLOUJADOFF Michel	Electrotechnique
SILBER Robert	Mécanique des Fluides

PROFESSEUR ASSOCIE

M. ROUXEL Roland	Automatique

PROFESSEURS SANS CHAIRE

MM. BLIMAN Samuel	Electronique
BOUVARD Maurice	Génie Mécanique
COHEN Joseph	Electrotechnique
LACOUME Jean-Louis	Géophysique
LANCIA Roland	Electronique
ROBERT François	Analyse numérique
VEILLON Gérard	Informatique Fondamentale et Appliquée
ZADWORNY François	Electronique

MAITRES DE CONFERENCES

MM. ANCEAU François	Mathématiques Appliquées
CHARTIER Germain	Electronique
GUYOT Pierre	Chimie Minérale
IVANES Marcel	Electrotechnique
JOUBERT Jean-Claude	Physique du solide
MORET Roger	Electrotechnique Nucléaire
PIERRARD Jean-Marie	Mécanique
SABONNADIERE Jean-Claude	Informatique Fondamentale et Appliquée
Mme SAUCIER Gabrièle	Informatique Fondamentale et Appliquée

MAITRE DE CONFERENCES ASSOCIE

M. LANDAU Ioan	Automatique

CHERCHEURS DU C.N.R.S. (Directeur et Maître de Recherche)

MM. FRUCHART Robert	Directeur de Recherche
ANSARA Ibrahim	Maître de Recherche
CARRE René	Maître de Recherche
DRIOLE Jean	Maître de Recherche
MATHIEU Jean-Claude	Maître de Recherche
MUNIER Jacques	Maître de Recherche

Je tiens à exprimer toute ma reconnaissance à Monsieur le Professeur J. KUNTZMANN de l'honneur qu'il me fait en présidant ce jury.

Je remercie vivement :

- *Monsieur le Professeur BOLLIET d'avoir bien voulu s'intéresser à ce travail,*
- *Madame SAUCIER, Maître de Conférences à l'ENSIMAG qui a dirigé cette étude et permis son bon achèvement,*
- *Monsieur LUSINCHI, Chef du Service Informatique de la SESCOSEM qui a assuré l'environnement de ce travail et accepté de faire partie du jury,*
- *Monsieur RONCIN, Ingénieur en Chef du Département Fiabilité, Mesure et Instrumentation au Centre National d'Etude des Télécommunications de Lannion, d'avoir bien voulu faire partie du jury.*

Je tiens à remercier tout spécialement Monsieur ZIRPHILE, Ingénieur à la SESCOSEM, sans lequel ce travail n'aurait pu aboutir ainsi que Monsieur TULLOUE, Ingénieur à la Société THOMSON-CSF/DIS.

Je remercie également tout le personnel du Service Informatique de la SESCOSEM pour l'accueil qu'il m'a fait et tous les membres de l'équipe "Conception et Sécurité des Systèmes Logiques" de l'ENSIMAG pour les échanges fructueux qui ont jalonné ce travail.

Enfin, je remercie Madame DUFFOURD, Secrétaire à l'ENSIMAG ainsi que toutes les personnes ayant participé à la réalisation matérielle de cet ouvrage.

Le présent travail a été réalisé au Service Informatique de la Division SESCOSEM de la Société THOMSON-CSF à Saint-Egrève (38120).

CHAPITRE III : CARACTÉRISATION DES PANNES

1. TYPES DE DEFAUTS

2. PANNES DE COLLAGES

   2.1. Relation d'équivalence
   2.2. Relation de couverture

3. PANNES DE COURTS-CIRCUITS

   3.1. Types de courts-circuits logiques
   3.2. Relations entre collages et courts-circuits
   3.3. Modélisation des courts-circuits
      a) Définition
      b) Courts-circuits type implication
      c) Courts-circuits type ET et type OU

4. RESUME DES HYPOTHESES DE PANNES

CHAPITRE IV : STRUCTURATION D'UN PROGRAMME DE TEST

1. STRATEGIE DE TEST D'UN CIRCUIT

   1.1. Rappels
   1.2. Principe de synthèse-analyse
   1.3. Choix d'une panne

2. STRATEGIE DE TEST D'UNE PANNE

   2.1. Rappels des méthodes de chemin sensible
   2.2. Adaptation au cas des circuits séquentiels
   2.3. Probleme des implications

CHAPITRE V : DÉTERMINATION DES VALEURS DES CONNEXIONS

1. DEFINITIONS

   1.1. Introduction
   1.2. Domaine des valeurs de connexion
   1.3. Opérations sur le domaine de valeurs
      a) *Extension des opérations définies sur* B^2
      b) *Implications*

2. RESTRICTION DU DOMAINE DES VALEURS DE CONNEXIONS

   2.1. Assignation initiale et stabilité
   2.2. Sous-domaines de valeurs
   2.3. Propriétés du sous-domaine $\mathcal{D}_1$
      a) *Structure de treillis*
      b) *Simplification des opérations d'implication*
      c) *Prise en compte des effets répétés d'une panne*
      d) *Exemples d'application*

CHAPITRE VI : ALGORITHME DE SYNTHESE

1. DEFINITIONS

2. ALGORITHME

   2.1. Représentation des pannes
      a) *Courts-circuits de type ET et de type OU*
      b) *Court-circuit de type IMPLICATION*
   2.2. Implications
      a) *Implications dans un vecteur d'état*
      b) *Implications dans une famille de vecteurs d'état*
   2.3. Propagation
   2.4. Prolongation aval de la sous-séquence

INTRODUCTION

La vérification de la fonction réalisée par un circuit intégré nécessite de plus en plus une aide informatique.

Jusqu'à présent, la simplicité des circuits que l'on intégrait (circuits MSI) ou l'aspect répétitif de leur structure (mémoires) rendaient suffisante l'élaboration manuelle de séquences de test.

Avec l'avènement des circuits LSI à structure non répétitive (microprocesseurs, circuits à la demande...), la nécessité d'une automatisation de cette élaboration se fait de plus en plus sentir.

Or, une étude de l'abondante littérature concernant les diverses formes du test fait apparaître l'absence d'une méthode pouvant répondre de manière satisfaisante aux besoins industriels dans le domaine des circuits séquentiels.

Nous avons donc été amenés à concevoir une méthode de génération automatique de séquences de test qui se prête à la mise en oeuvre réaliste d'un programme industriel.

Le présent travail décrit cette étude et le programme qui en a résulté. Ce dernier est destiné à s'insérer dans le système SIGMA de conception assistée par ordinateur de la Société THOMSON-CSF.

Le premier chapitre a pour but de considérer le problème dans son cadre industriel et de rappeler que seules les méthodes basées sur une étude de la structure des circuits peuvent le résoudre (méthodes structurelles).

Dans le cadre de ces méthodes, nous constatons que des deux principes retenus à ce jour (approche synchrone et approche asynchrone), le premier simplifie trop le problème alors que le second ne semble pas pouvoir supporter une automatisation efficace.

Dans le deuxième chapitre, une étude du problème de la représentation des circuits (qui est le principal obstacle aux approches asynchrones) nous conduit à étudier une classe de circuits et un type de fonctionnement de ceux-ci qui se prête à une représentation efficace.

Nous définissons alors une caractérisation du fonctionnement des circuits lors de leur test qui permet une modélisation des fonctions séquentielles sous forme d'éléments standards.

Cette caractérisation répondra au problème du test des circuits intégrés classiques.

Le troisième chapitre considère le problème de la représentation des défauts susceptibles d'affecter les circuits intégrés.

En fonction des besoins industriels, nous sommes ainsi amenés à considérer, en plus des habituelles pannes de collages, les pannes de courts-circuits pour lesquelles nous définissons une modélisation sous forme de panne multiple.

Le quatrième chapitre étudie le principe des algorithmes heuristiques des méthodes structurelles.

Le problème du test comportant de nombreux degrés de liberté, le caractère réaliste d'un programme est conditionné par la bonne utilisation d'un certain nombre de critères heuristiques.

Nous mettons donc en évidence les points où l'insertion de tels critères est indispensable et nous proposons certaines améliorations.

Le cinquième chapitre a pour but la formalisation de l'évolution des états des connexions d'un circuit lors de l'élaboration du test d'une panne de ce circuit.

La définition des différentes valeurs que peut prendre une connexion est en effet fondamentale car elle conditionne l'orientation des critères heuristiques principaux.

Le sixième chapitre est consacré à l'exposé de l'algorithme que les considérations précédentes ont permis de concevoir.

Enfin, le septième chapitre donne les caractéristiques du programme obtenu ainsi que des exemples d'applications.

Nous considérons ainsi trois circuits MSI (bascule D, registre à décalage, mémoire 16 bits) et un circuit LSI (microprocesseur).

CHAPITRE I

PRÉSENTATION

1. TRI INDUSTRIEL

1.1 - Introduction

Le tri des circuits intégrés industriels est un paramètre influençant fortement le coût final des produits.

Le degré de précision des tests de validation pouvant être variable, deux points de vue se dégagent :

- *Celui du fabricant :* il lui faut à priori un test général mais minimal.
- *Celui du client :* il veut un test correspondant à l'utilisation future de ses circuits.

Aussi, le premier propose-t-il un certain nombre de normes d'acceptation possibles (militaires, standards...) que le second peut modifier selon ses besoins.

D'après ces normes, les contrôles opérés seront plus ou moins sévères et, pour certaines applications spécifiques, les pièces subiront préalablement des épreuves plus ou moins dures (variations rapides de températures, chocs, vieillissements, bains, fortes accélérations...).

1.2 - Types de tests

On distingue trois grandes familles :

- *Test paramétrique :* contrôle des paramètres statiques permettant de réaliser la compatibilité avec d'autres circuits.
  Les mesures concernent les caractéristiques d'entrée, de transferts et de sortie, la consommation et la puissance.
- *Test dynamique :* contrôle des caractéristiques de commutation.
  Les mesures portent sur les temps de propagation et de transition, sur les fréquences maximales dans des conditions données, sur la puissance consommée selon la fréquence.

- *Test logique :* vérification de la fonction logique réalisée par le circuit.

Les problèmes posés par le test paramétrique sont généralement ceux de l'électronique conventionnelle (mesure de faibles courants...) sauf dans certaines technologies (CMOS) pour lesquelles il n'y a pas de distinction réelle entre tests paramétrique et logique (certaines caractéristiques dépendent de l'état du circuit).

Le test paramétrique est généralement le plus coûteux parce que le plus lent.

Le problème du test dynamique est la définition du "pire cas" qui demande une très bonne connaissance du circuit pour savoir déterminer les configurations les plus défavorables à un paramètre donné.

Pour le test logique, cadre de cette étude, il n'existe pas, à l'heure actuelle, de méthode générale applicable à tous les circuits.

1.3 - Aspect matériel du tri

Les pièces sont triées à deux niveaux de leur fabrication :

- *Sur plaquette :* c'est-à-dire avant les opérations de découpage et de mise en boîtier qui sont onéreuses car individuelles.
- *Sur boîtier :* c'est-à-dire avant stockage ou livraison du produit fini.

Les systèmes physiques de test se composent généralement :

- *D'un petit calculateur* avec ses périphériques (exemple : PDP8, télétype, imprimante, console de visualisation et lecteur de cassettes).
- *D'un ensemble de machines de test* permettant :
  - . le test sur plaquette
  - . le test sur boîtier
  - . le test en étuve.

- *D'un interface calculateur-machinesde test traitant les informations et mettant en forme les signaux.*

A l'aide d'un tel ensemble on peut, à l'heure actuelle :

- *Mettre en oeuvre différentes familles de test*
- *Gérer différents lots de pièces refusées ou acceptées*
- *Acquérir des résultats de test partiels ou finaux*
- *Générer des séquences de test câblées pour certains circuits particuliers (mémoires)*
- *Calculer en cours d'opérations des valeurs moyennes de paramètres pour un ensemble de pièces donné*
- *Appliquer les tests à des fréquences de l'ordre de quelques mégahertz.*

2. TEST LOGIQUE

2.1 - Définitions et rappels

Nous appellerons défaut une malformation physique et panne l'effet qu'elle produit sur le circuit.

Les deux phases classiques du test logique -Détection et Localisation des défauts- n'ont pas, en circuits intégrés, la même importance. En effet, puisqu'il n'est pas question de réparation, la localisation des défauts est simplement un outil d'analyse permettant de contrôler les processus de fabrication et d'améliorer la connaissance de l'effet des défauts.

Notre problème principal est donc de savoir décider de la validité de la fonction d'un circuit. Nous rappelons que, matériellement, celui-ci n'est accessible que par ses entrées et sorties primaires.

Un vecteur d'entrée (ou vecteur de test) sera l'ensemble des valeurs à appliquer à un instant du test sur les entrées primaires.

L'ensemble ordonné des vecteurs d'entrée élaborés pour le test d'un circuit sera appelé séquence de test.

Nous demanderons que la séquence de test d'un circuit séquentiel respecte l'hypothèse d'un fonctionnement en mode fondamental :

> Toute entrée primaire ne pourra changer d'état qu'une fois le circuit dans un état stable.

L'interdiction de faire varier les entrées lorsque le circuit est dans un état instable provient de l'imperfection de la caractérisation temporelle donnée avec les circuits à tester. Il n'est pas possible dans la pratique de prévoir correctement les instants d'apparition exacts des états instables. L'hypothèse de mode fondamental n'est donc plus liée à la technologie des machines de test : il est fort possible d'en concevoir une appliquant des vecteurs d'entrée à des intervalles de temps précis, si l'on accepte de supporter le coût en matériel correspondant.

Rappelons brièvement les principales méthodes de test actuellement connues |3, 14, 20, 27, 28|.

Le test logique a été initialement étudié pour les circuits combinatoires. Les méthodes pour ce type de circuits sont essentiellement de deux types :

- *Probabilistes :* un ensemble de vecteurs d'entrée est appliqué simultanément au circuit à tester et à un modèle de référence. Toute différence de comportement dénonce une erreur.
- *Déterministes :* les vecteurs d'entrée sont déterminés par examen du circuit.

Outre la méthode dite "manuelle", cette deuxième catégorie regroupe les méthodes fonctionnelles (ne considérant que la fonction du circuit) et les méthodes structurelles (qui examinent la structure du réseau réalisant la fonction logique du circuit).

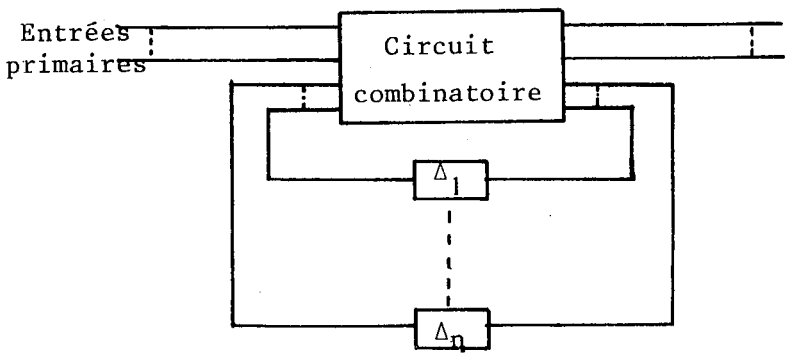
Enfin, on peut également diviser cette dernière approche en méthodes algébriques (qui manipulent des équations sur les divers composants du circuit) et en méthodes heuristiques qui cherchent à faire progresser l'effet d'une panne vers une sortie primaire en lui créant un "chemin" dans le circuit (algorithmes dits de chemin sensible).

Intermédiaire entre les approches probabilistes et déterministes, se trouve la génération aléatoire : les vecteurs d'entrée sont générés aléatoirement, puis une simulation du circuit permet de connaître les pannes que la séquence peut détecter.

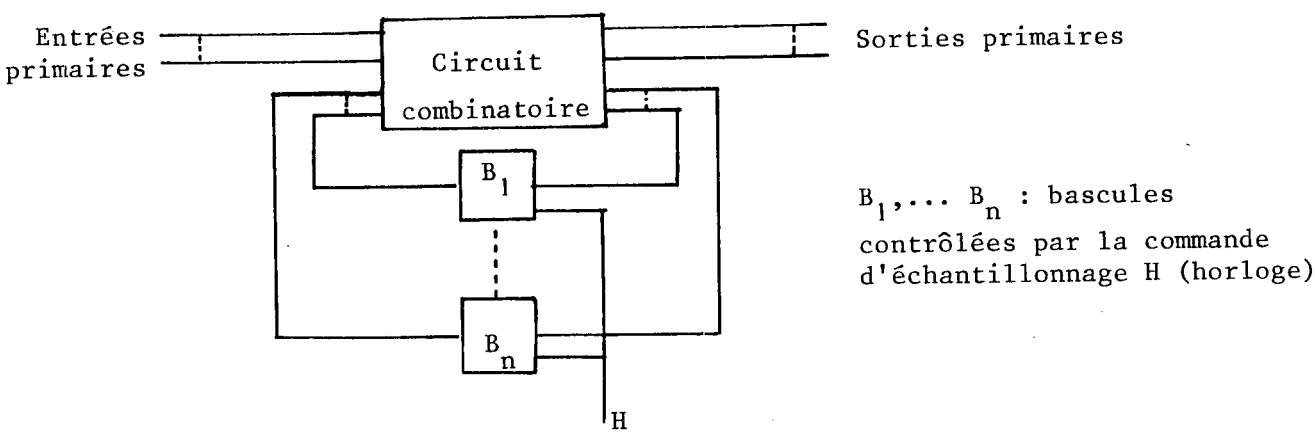
Toutes ces méthodes sont basées sur une connaissance plus ou moins fine des pannes potentielles du circuit. Le problème immédiat est donc de connaître l'ensemble des défauts possibles et d'étudier leur effet pour les modéliser de manière utilisable par un programme. Nous verrons que dans la pratique, peu de types de défauts se prêtent à une modélisation efficace.

2.2 - Le problème séquentiel

Nous rappelons le modèle de Huffman qui est la représentation classique des circuits séquentiels asynchrones :



Pour les circuits synchrones, cette représentation devient :



Il existe certaines méthodes de test spécifiques aux circuits séquentiels. Elles sont basées sur l'utilisation de leur table d'état. Cependant, dans la pratique, la dimension que ces tables pourraient atteindre et la difficulté de les obtenir rendent cette approche inutilisable dans le cas général.

La seule extension fructueuse de méthode de test combinatoire au cas séquentiel concerne les approches structurelles.

Dans les méthodes de chemin sensible, initialement caractérisées par le D-algorithme de Roth |30|, le cas des circuits séquentiels a très vite été envisagé |5, 18, 26|.

Citons également pour les approches algébriques une méthode s'apparentant à la dérivée booléenne |10, 12|.

Cependant, bien que l'on sache actuellement tester les circuits séquentiels synchrones, le cas asynchrone quelconque pose encore de gros problèmes |2, 15, 19, 29, 32, 34, 36, 41|.

Les rares programmes traitant ce cas ne sont satisfaisants ni du point de vue hypothèses de pannes, ni du point de vue coût, et cela particulièrement pour les circuits intégrés. Il semble, à l'heure actuelle, qu'une solution satisfaisante au problème asynchrone ne puisse que passer par une approche très conversationnelle intégrant correctement les capacités humaines de décision.

La mise en oeuvre des approches synchrones s'explique surtout par la facilité du modèle à considérer : la connaissance des boucles et leur contrôle par horloge permettent de traiter ces circuits de manière analogue aux circuits combinatoires|7, 17, 18, 24, 38|.

Cependant, cela sous-entend que l'on isole ces commandes d'échantillonnage et que l'on ne prenne pas en compte les défauts pouvant y figurer.
Ce sont malheureusement les plus graves car ils conditionnent le fonctionnement de tout le circuit. Faire l'hypothèse dangereuse qu'une commande d'horloge est sans défaut, ou vérifiable par ailleurs, est inapplicable en circuits intégrés.

Il est donc nécessaire de banaliser toutes les connexions d'un circuit et, comme |5|, de considérer son comportement comme étant asynchrone même s'il est synchrone.

Notons que, dans le cadre des approches synchrones, Breuer |7| a apporté une amélioration : bien qu'il conserve la synchronisation par horloge des bascules, il autorise la présence dans le circuit combinatoire de certains points mémoire.

Nous allons nous attacher à démontrer, dans cette étude, que dans le domaine des circuits intégrés, une génération efficace de séquences de test peut dépasser largement le cadre des approches synchrones, tout en prenant en compte des hypothèses de pannes plus satisfaisantes que celles considérées généralement.

La méthode de génération de vecteurs de test que nous étudierons pour cela sera structurelle et de type chemin sensible.

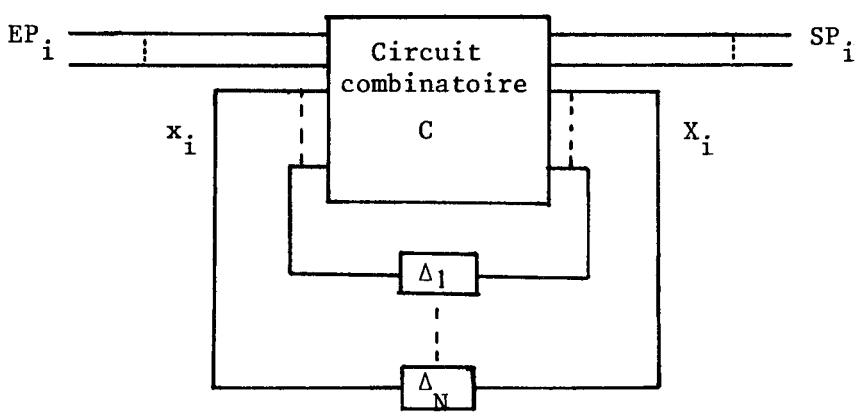
CHAPITRE II

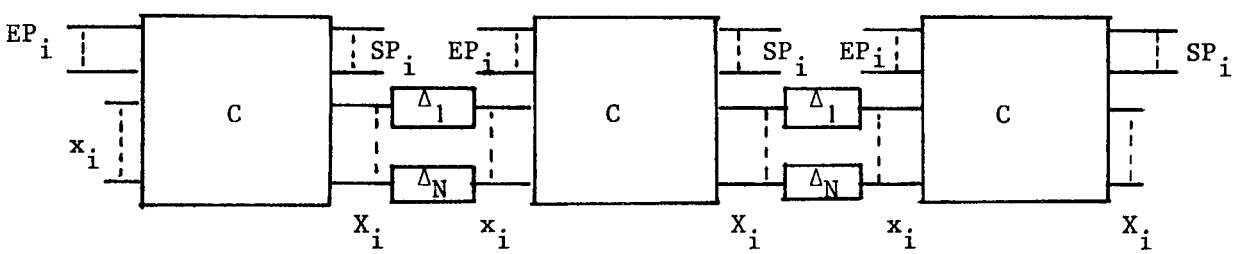
CARACTÉRISATION DES CIRCUITS EN VUE DU TEST

1. PROBLEME DE REPRESENTATION

Dans les approches asynchrones, le principal problème est celui du manque de précision du modèle utilisé pour représenter le circuit lors de la génération de test.

Le plus connu de ces modèles est la représentation combinatoire itérative de Roth [5, 26] : après recherche des boucles de réaction du circuit, certaines sont coupées et le circuit initial est alors transformé en un circuit combinatoire itératif où chaque itération représente l'application d'un vecteur d'entrée.



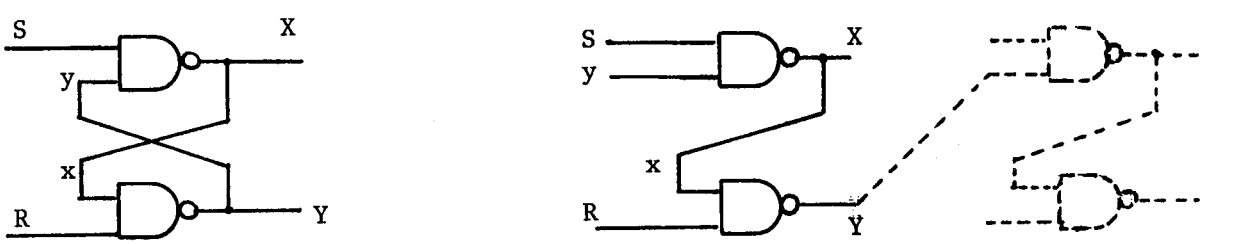


Les limites de validité de ce modèle sont strictes mais, en l'absence d'une solution plus satisfaisante, il a été abondamment utilisé [2, 15, 19, 23, 31, 32, 33, 36, 41...]. Les problèmes qui lui sont liés sont de deux ordres :

- *Inefficacité :* en pratique on est amené à limiter très vite le nombre P d'itérations que l'on peut prendre en compte car, si le circuit initial comporte N connexions, le programme travaillant sur le modèle itéré doit en considérer N x P. Or, le facteur P représente également le nombre maximum de vecteurs d'entrée que l'on pourra construire pour tester une panne donnée. On voit donc que le test des pannes demandant plus de P vecteurs d'entrée ne sera pas assuré.

- *Résultats potentiels :* l'aspect temporel n'étant pas correctement pris en compte, les séquences de test sont parfois erronées. C'est le problème du choix des points de coupure des boucles : la localisation des variables internes sur ces points traduit, plus ou moins bien, le fonctionnement réel du circuit.

Exemple : *La figure suivante représente l'effet d'une coupure de boucle sur un point-mémoire classique :*



x, y : valeurs précédentes de X, Y

(instant suivant)

Nous obtenons les tables suivantes : (avec $U \in \{0,1\}$)

xy	SR	XY
UU	01	10
UU	10	01
UU	00	11
01	11	01
10	11	10
11	11	?

xy	SR	XY
UU	01	10
U0	10	11
U1	10	01
UU	00	11
01	11	01
10	11	10
11	11	01

Mis à part le problème de l'indétermination apparaissant sur la dernière ligne du tableau que nous considérerons par la suite, nous constatons une anomalie dans le comportement du circuit itéré en présence de la configuration d'entrée SR = 10 : contrairement au circuit initial, l'état atteint dépend de l'état précédent.

Afin de résoudre ce problème, Breuer [6] propose une méthode qui, en dehors des conditions de course critique, traduit correctement le fonctionnement du circuit. Malheureusement, elle conduit à une itération bidimensionnelle du réseau, le rendant impropre à un traitement efficace.

Le manque de solution satisfaisant à la représentation des circuits en vue du test, nous conduit à envisager le problème sous un autre angle.

Plutôt que de tenter de modéliser tout le circuit, nous nous attacherons à ne le faire que pour les modules rendant le circuit séquentiel.

Grâce à la localisation des variables internes, que nous allons définir, nous modéliserons chaque opérateur séquentiel sous forme d'un macro-bloc standard séquentiel.

2. CARACTERISATION DES CIRCUITS A TESTER

Contrairement aux approches habituelles, les critères que nous allons prendre, afin de caractériser les circuits à tester, concernent non seulement leur définition mais également leur comportement lors de l'application de la séquence de test.

2.1 - Définition des circuits à tester

Les circuits dóivent vérifier les trois conditions :

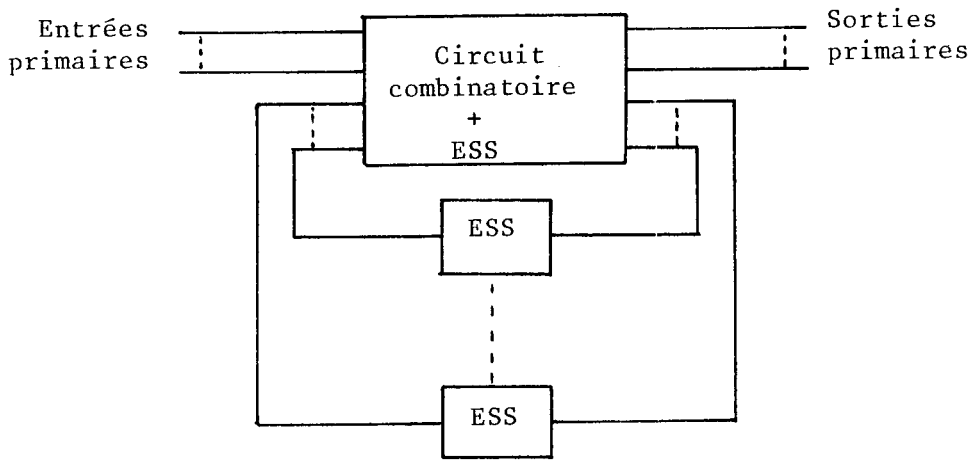
a) N'être composés que d'opérateurs combinatoires et d'éléments séquentiels standards, répertoriés dans une liste de référence.

b) Etre conçus de telle sorte que leur fonctionnement en l'absence de défauts est indépendant des temps de propagation des opérateurs (circuits "bien conçus").

c) Tout cycle (au sens de la théorie des réseaux) comporte au moins un élément standard séquentiel.

Le modèle de circuit que nous considérerons sera donc :

Avec
ESS = élément
standard séquentiel



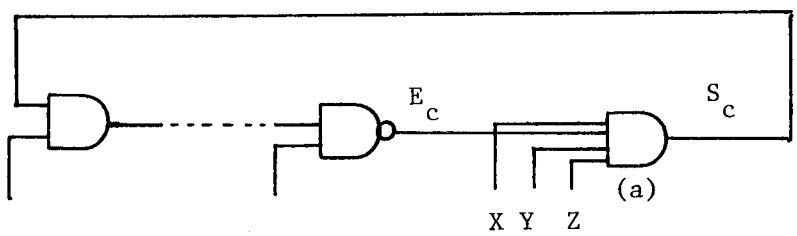
2.2 - Comportement des circuits lors du test

a) Définitions

Pour un circuit à tester, nous appellerons restriction d'emploi toute limitation apportée à son fonctionnement en cours de test. Une restriction d'emploi sera donnée par une équation logique liant les valeurs de certaines connexions et devant être vérifiée lors du test.

Soit un opérateur logique dont une entrée E_c et une sortie S_c appartiennent à un cycle C.

Nous appellerons condition de blocage du cycle par cet opérateur une équation logique liant les valeurs des entrées de l'opérateur et dont la solution implique que les valeurs des connexions E_c et S_c sont indépendantes.



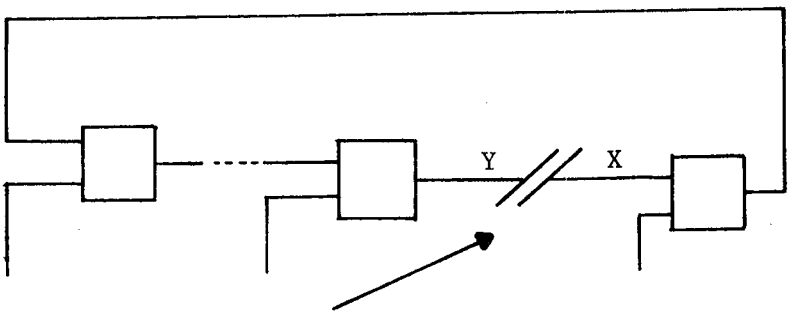
La condition de blocage du cycle sur la porte(a) est : $\overline{X} + \overline{Y} + \overline{Z} = 1$.

Les conditions de blocage de cycle par les opérateurs standards sont résumées dans l'annexe II. On remarquera que, dans le cas des opérateurs séquentiels, les valeurs initiales des variables internes seront assimilées à des valeurs d'entrées.

b) Condition de non rebouclage instantané

Nous imposons que lors du test d'un circuit, tout cycle, en présence ou non d'une panne, vérifie la propriété suivante :

> Pour tout état du circuit, si l'on considère une coupure fictive en un point quelconque du cycle, la valeur logique en entrée de la coupure est indépendante de celle en sortie.



Coupure fictive : Y est indépendant de X

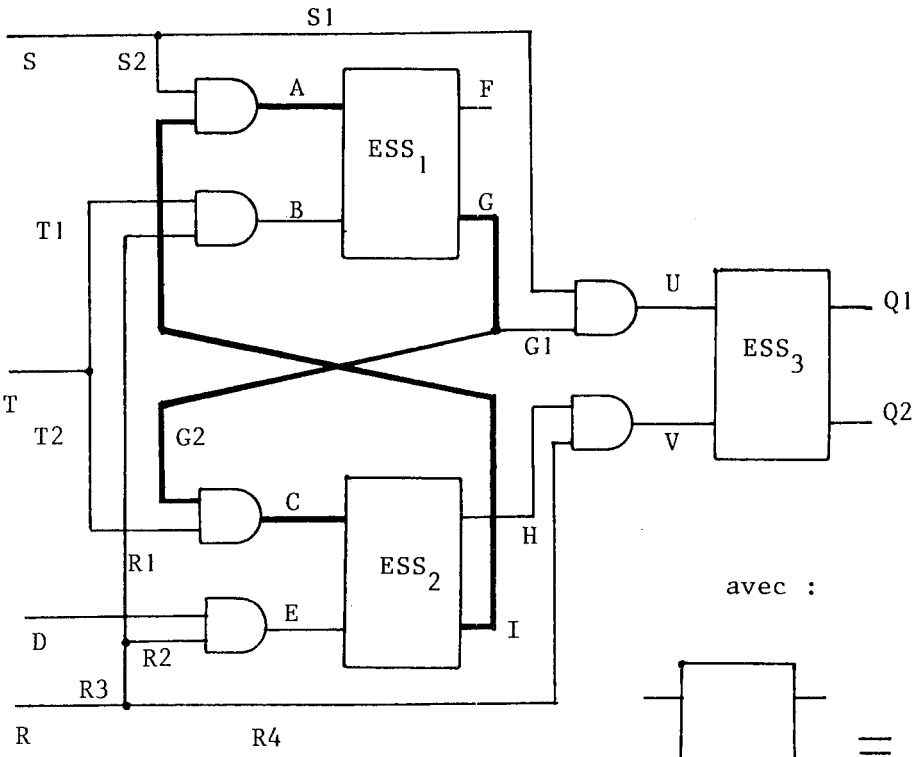
Il faut donc que sur un cycle il y ait toujours, au moins, un opérateur vérifiant sa condition de blocage du cycle afin d'inhiber la circulation de l'information sur celui-ci.

Nous remarquerons que la topologie de certains cycles leur assure la vérification de cette condition, mais cela simplement en l'absence de certaines pannes. Par exemple, dans un cycle comportant une bascule maître-esclave, la commande d'horloge assurera le blocage de la circulation de l'information à condition qu'elle ne soit pas affectée par certaines pannes.

Ne pouvant faire cette hypothèse, nous considèrerons que, à priori pour aucun cycle, la condition de non rebouclage instantané n'est assurée par la topologie.

Aussi, pour chaque cycle, nous établirons la restriction d'emploi rassemblant les conditions de blocage du cycle par chacun des opérateurs qu'il contient.
Grâce à cette restriction d'emploi, l'algorithme de génération de vecteurs d'entrée pourra choisir à chaque instant la condition de blocage du cycle la plus appropriée au bon déroulement des opérations.

Exemple : *La bascule de type D est un circuit MSI très intéressant sur le plan de sa modélisation. Nous donnons son schéma dans sa représentation correspondant à nos hypothèses.*



Nous remarquons un cycle extérieur aux éléments standards séquentiels. Il se compose des connexions A, G, G2, C, I. *Nous introduisons donc une restriction d'emploi qui demande le positionnement à l'état logique "0" d'au moins une des connexions* S2, T2, B, E, I, G.

Les connexions I *et* G *sont considérées en tant que valeurs initiales de sorties d'éléments standards séquentiels et proviennent des conditions de blocage sur ces opérateurs (voir annexe 2). Si nous notons* I^* *et* G^* *ces valeurs initiales, la restriction d'emploi s'écrit :*

$$S2.\ T2.\ B.\ E.\ I^*.\ G^* = 0$$

L'algorithme de génération de vecteurs d'entrée garantira donc que lors du test il y aura toujours au moins une des connexions précédentes à la valeur logique "0".

2.3 - Propriété des circuits lors du test

Le comportement lors du test de tout circuit satisfaisant aux conditions précédentes aura la propriété :

> Pour un état donné du circuit et une variation de ses entrées primaires, toute connexion, mises à part celles internes aux éléments standards séquentiels, ne peut changer qu'une seule fois d'état.

En effet, d'après l'hypothèse (b) de la définition des circuits, seul un cycle peut introduire un changement multiple d'état sur une connexion. La condition de non rebouclage instantané assure alors qu'à tout instant les valeurs, sur ce cycle, sont calculables directement à partir des valeurs figurant sur les entrées primaires et les éléments standards séquentiels.

Corollaire : *Les seules variables internes figurant dans un cycle sont celles des éléments standards séquentiels qu'il contient.*

L'état interne du circuit est donc caractérisable par les seules variables internes des éléments standards séquentiels.

REMARQUES : *Les conditions précédentes semblent très restrictives, elles permettent cependant le test de tous les circuits intégrés classiques dont la structure n'est pas essentiellement du type mémoire.*
D'autre part, l'emploi de composants bibliothécaires dans lesquels les restrictions ont déjà été incorporées simplifiera, dans la pratique, l'identification des cycles.

3. MODELISATION DES ELEMENTS STANDARDS SEQUENTIELS

Deux types d'opérateurs séquentiels correspondant à deux niveaux de complexité ont été considérés :

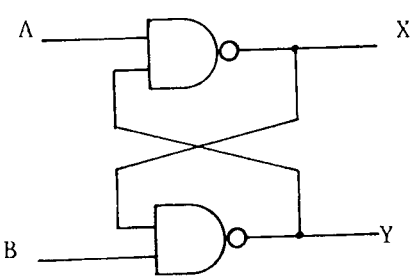
- *bascules maître - esclave*
- *points mémoire type ETNON entrecroisés.*

La première solution s'est avérée irréalisable pour des raisons de coût de stockage et de traitement des informations caractérisant ces opérateurs.

Nous avons donc choisi la deuxième solution et défini une famille d'éléments standards séquentiels permettant la représentation aisée de plus gros ensembles.

3.1 - Comportement asynchrone

Nous étudions le comportement asynchrone d'un point mémoire ETNON entrecroisés :



L'étude fine de ce comportement montre que sous certaines configurations, l'état atteint par le circuit est fonction de ses paramètres temporels |21|. Cela se traduit sur sa table d'état par la présence d'un état indéterminé :

		A B = 00	01	11	10
	00	11	11	11	11
xy	01	11	11	(01)	(01)
	11	(11)	10	X	01
	10	11	(10)	(10)	11

Avec x, y les variables d'état associées à X, Y

La configuration AB xy = 1111 laisse le point mémoire dans un état indéterminé.

Remarquons que, si la théorie prévoie le cas où,les temps de traversée des opérateurs étant égaux, le système se met à osciller, dans la pratique cette condition n'est jamais réalisée. Le point mémoire bascule après passage par un état intermédiaire (xy = 00) dans une configuration qui ne serait prévisible que par la donnée de ses caractéristiques temporelles.
Ces caractéristiques n'étant jamais disponibles, nous avons bien une indétermination. Nous sommes donc amenés à restreindre le fonctionnement du point-mémoire.
Nous interdirons pour cela l'apparition de la configuration critique (ABxy = 1111) à l'aide d'une restriction d'emploi.

Celle-ci s'écrira : A.B.x.y = 0

Grâce à la propriété caractéristique de nos circuits appliquée aux connexions d'entrée du point mémoire, cette condition ne sera imposée que sur les états stables.

La table d'état que nous considérerons pour le point-mémoire sera donc :

		A B = 00	01	11	10
	01	11	11	(01)	(01)
xy	11	(11)	10	I	01
	10	11	(10)	(10)	11

Avec I : impossible

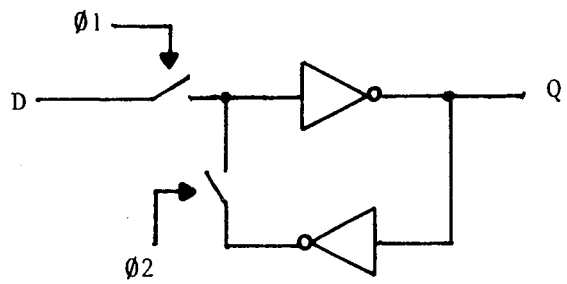
Respectivement, pour chaque sortie du point mémoire, le fonctionnement synchrone sera représenté par les tables de Karnaugh :

		A B			
	X	00	01	11	10
	01	1	1	0	0
xy	11	1	1	I	0
	10	1	1	1	0

		A B			
	Y	00	01	11	10
	01	1	0	1	1
xy	11	1	0	I	1
	10	1	0	0	1

L'étude précédente est applicable aux autres éléments standards séquentiels basés sur la même structure. On aboutit alors à des résultats similaires.

Le cas de certains opérateurs spécifiques à la technologie MOS est différent : certains points mémoire demandent la présence de deux signaux complémentaires (Ø1 et Ø2) qui sont élaborés extérieurement à la porte :



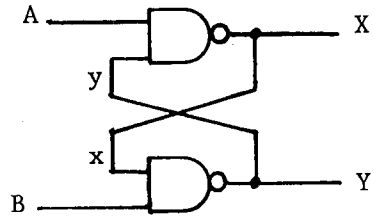
Il faut donc introduire une restriction d'emploi pour interdire la non-complémentarité de ces deux signaux qui agirait de manière imprévisible sur le comportement du point mémoire.

La restriction d'emploi sera donc :

$$Ø1 . \overline{Ø2} + \overline{Ø1} . Ø2 = 1$$

3.2 - Modélisation

L'étude précédente des éléments de type ETNON entrecroisés aboutit pour la résolution de leur modélisation à une fonction ϕ-booléenne pour chacune des sorties.



Pour la sortie X nous avons la table :

		A B			
	X	00	01	11	10
	00	ϕ	ϕ	ϕ	ϕ
xy	01	1	1	0	0
	11	1	1	ϕ	0
	10	1	1	1	0

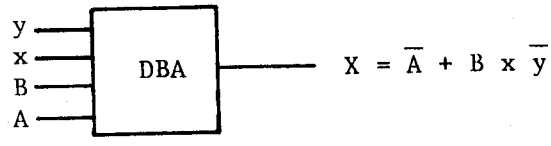
Avec : x, y = anciennes valeurs de X, Y

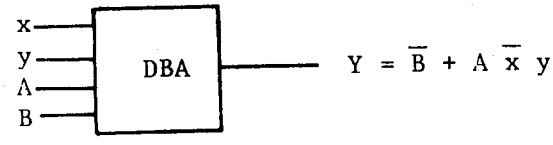
Sans oublier la condition : ABxy = 0.

L'expression de X que nous considèrerons sera : $X = \overline{A} + B \times \overline{y}$.

Elle sera réalisée par un opérateur spécifique (DBA) de fonction :
$F(y, x, B, A) = \overline{A} + B \times \overline{y}$.

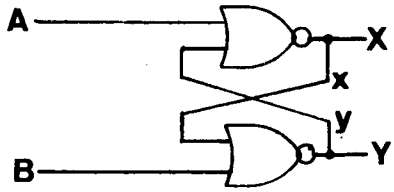
La modélisation complète du point mémoire est donc :



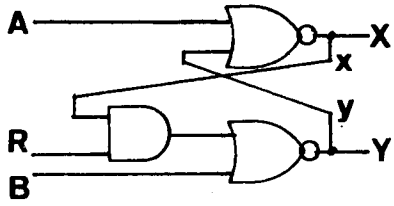


Avec : A B x y = 0

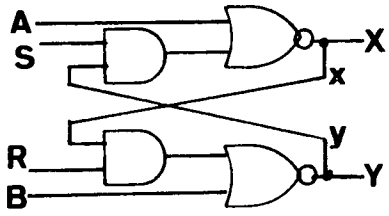
Pour les autres éléments standards séquentiels, nous obtenons de manière similaire :



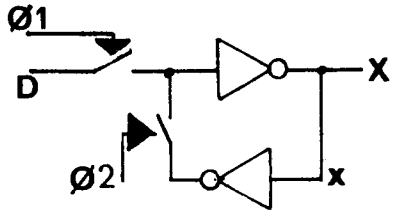
$$\begin{cases} X = \overline{A}(B + x\overline{y}) \\ Y = \overline{B}(A + \overline{x}y) \\ A + B + x + y = 1 \end{cases}$$



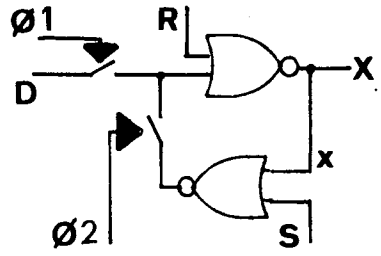
$$\begin{cases} X = \overline{A}(B + R\overline{y}x) \\ Y = \overline{B}(\overline{R} + A + \overline{x}y) \\ \overline{R} + A + B + x + y = 1 \end{cases}$$



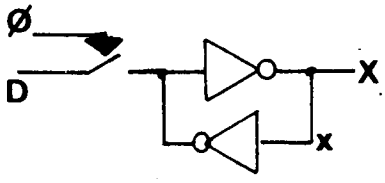
$$\begin{cases} X = \overline{A}(B + R\overline{y}x + \overline{S}) \\ Y = \overline{B}(A + Sxy + \overline{R}) \\ \overline{R} + \overline{S} + A + B + x + y = 1 \end{cases}$$



$$\begin{cases} X = Ø1.\ \overline{D} + Ø2.\ x \\ Ø1.\ \overline{Ø2} + \overline{Ø1}.\ Ø2 = 1 \end{cases}$$



$$\begin{cases} X = Ø1.\ \overline{R}.\ \overline{D} + Ø2.\ \overline{R}(S + x) \\ (Ø1.\ \overline{Ø2} + \overline{Ø1}.\ Ø2)\ \overline{R} = 1 \end{cases}$$



$$X = Ø.\ D + \overline{Ø}.\ x$$

CHAPITRE III

CARACTÉRISATION DES PANNES

1. TYPES DE DEFAUTS

Nous rappelons que les pannes sont les conséquences de défauts physiques d'un circuit sur son fonctionnement.
On distingue généralement les défauts par :

- *leur durée :* défauts permanents ou intermittents.
- *leur nombre :* défauts uniques ou multiples, selon qu'un ou plusieurs défauts peuvent affecter le circuit simultanément.
- *leur conséquence :*
  - . défauts affectant la fonction d'un opérateur
  - . défauts affectant la valeur logique d'une équipotentielle
  - . défauts faisant intervenir plusieurs équipotentielles (courts-circuits)
  - . (.../...)

Un défaut peut être indétectable s'il ne perturbe pas le fonctionnement du circuit. Plusieurs défauts peuvent être indiscernables si leur effet sur le circuit est le même.

Globalement, sur un circuit, un défaut peut :

- *lui faire perdre son caractère logique* (cas de certains courts-circuits avec stabilisation à des tensions intermédiaires).
- *affecter les temps de propagation,* ce qui se traduit pas l'apparition d'impulsions ou de transitions parasites en sortie (pannes de propagation).
- *modifier sa fonction logique* (pannes logiques).

Jusqu'à présent, seules certaines pannes logiques se sont prêtées à une mise en oeuvre effective de génération de vecteurs de test.

Bien qu'un certain nombre d'études permettent d'envisager les défauts multiples |8, 25, 35, 39, 40, 42| ou intermittents |9, 16|, et ceux affectant les temps de propagation |6|, leur utilisation reste théorique ou limitée aux simples circuits combinatoires.

En ce qui concerne les défauts multiples, la gravité de cette carence est diminuée par le fait qu'un test élaboré pour des défauts uniques peut détecter de nombreux défauts multiples.

Le problème auquel on se heurte pour ces différents types de défauts est un problème de modélisation : comment représenter leur effet de manière utilisable par un programme de génération de vecteurs de test. Cette modélisation doit, d'autre part, être faite en liaison étroite avec la technologie des circuits car la méconnaissance de celle-ci peut rendre inadéquat les hypothèses de pannes et donc invalider les résultats obtenus.

2. PANNES DE COLLAGE

L'hypothèse de panne logique habituelle est le collage à zéro ou à un d'une connexion (C.A.0/C.A.1). Cette panne ne signifie pas que le défaut est sur la connexion : il faut considérer que tous se passe comme si celle-ci était constamment à la valeur logique 0 ou 1.

Dans toutes les technologies, un collage peut être représentatif des courts-circuits avec une alimentation.
En technologie TTL, la coupure d'une connexion se traduit par un collage à 1 vis-à-vis des opérateurs suivants. Les défauts internes aux opérateurs se traduisent par un collage à 0 ou à 1 de leur sortie.
En technologie MOS, les collages sont représentatifs des défauts internes aux transistors. Ce n'est pas forcément le cas des coupures de connexion qui peuvent nécessiter un modèle plus élaboré.

2.1 - Relation d'équivalence

Définitions

> Une fonction élémentaire en x est une fonction pouvant se mettre sous la forme $F = \tilde{X} + G$ ou $F = \tilde{X} \cdot G$ avec $\tilde{X} = X$ ou $\overline{X}$, et G indépendant de X.
> Une fonction élémentaire est une fonction qui est élémentaire par rapport à chacune de ses variables.

Les portes élémentaires seront donc les portes dont la fonction est élémentaire.

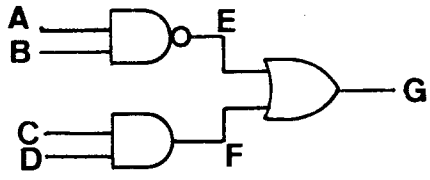
Exemples : *Les portes de base ET, OU, ETNON, OUNON sont élémentaires mais pas l'opérateur de DISJONCTION (OU exclusif).*

Dans un circuit, certaines pannes de collage sont indiscernables, aussi introduit-on généralement la relation d'équivalence : |40|.

Deux pannes P1 et P2 sont <u>équivalentes</u> (P1 $\sim$ P2) si tout vecteur de test de l'une est un vecteur de test de l'autre.

L'application de cette relation aux portes élémentaires permet de décrire dans un réseau logique des chaînes de pannes indiscernables.

Exemple :



Notons X^0/X^1 le collage à 0/1 d'une connexion X.
Nous avons : $\cdot A^0 \sim B^0 \sim E^1 \sim F^1 \sim G^1$
$\cdot C^0 \sim D^0 \sim F^0$

Il suffit donc pour chaque classe d'équivalence d'en tester un représentant.

Cette relation n'est utilisable que sur les portions de circuit combinatoires arborescentes :
D'une part la présence d'une <u>divergence</u> (une même connexion entrant dans plusieurs portes) interdit d'établir à priori cette relation entre son amont et son aval.
D'autre part, l'étude exhaustive des éléments standards séquentiels montre qu'il n'existe pas, en général, d'équivalence de panne entre leurs entrées et leurs sorties.

Nous ne mettrons donc en oeuvre cette relation que sur les parties arborescentes et combinatoires des circuits.

Remarquons qu'il existe une théorie des équivalences de pannes pour les circuits séquentiels |4|. Elle est cependant très lourde et repose sur la connaissance de tables d'état qui nous sont inaccessibles.

2.2 - Relation de couverture

Une deuxième relation entre pannes est parfois introduite : |40|.
Une panne P1 est couverte par une panne P2 si tout vecteur de test de P1 est un vecteur de test de P2.

Dans un circuit combinatoire, cette relation (qui est un pré-ordre) peut aboutir à des chaînes similaires aux précédentes mais orientées.

Ici encore, la présence de divergences, lorsqu'elles sont suivies de reconvergences (les chemins divergents sont à nouveaux réunis en entrée d'une porte), limite son utilisation.

Il en est de même avec nos opérateurs séquentiels possédant deux sorties : les relations de couverture pouvant exister entre les collages de certaines entrées et sorties demandent l'hypothèse de non reconvergence de ces dernières.

L'utilisation de cette relation a, par ailleurs, comme conséquence la diminution de la localisation des défauts (que nous verrons être restreinte par d'autres points de notre algorithme). Nous ne la mettrons donc pas en oeuvre.

3. PANNES DE COURTS-CIRCUITS

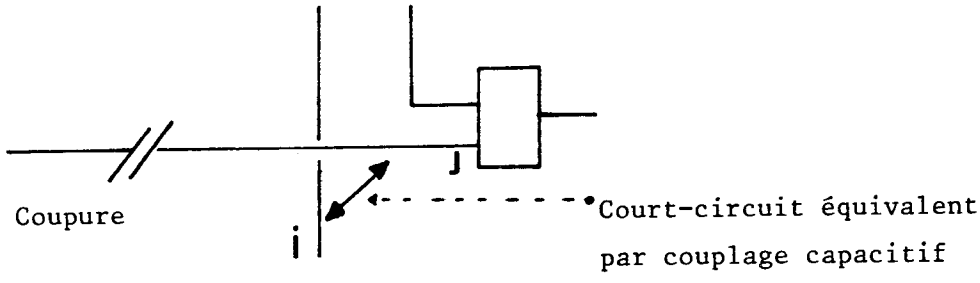
Un défaut de court-circuit est la liaison accidentelle de plusieurs connexions entre elles.
L'état que prennent alors les connexions est fonction des circuits électriques associés et des valeurs des connexions en amont de la liaison.
Cet état peut être :

- *un état logique,*
- *une tension intermédiaire.*

Nous ne nous intéresserons qu'au premier cas : les courts-circuits que nous considèrerons conserveront au circuit son caractère logique (courts-circuits logiques).

Ces courts-circuits peuvent cependant altérer profondément le fonctionnement du circuit : il peut y avoir création de nouveaux cycles. Si ceux-ci n'ont pas été répertoriés, ils ne satisferont pas nécessairement à la condition de non rebouclage instantané. En ce sens, les tests élaborés pour les courts-circuits pourront n'être que potentiels : leur validité devra parfois être prouvée.

La classe de défauts dont le comportement est de type court-circuit n'est pas limitée aux courts-circuits physiques. En technologie MOS, par exemple, les coupures de connexion peuvent ne pas être modélisables par des collages : l'état logique en entrée d'un opérateur est parfois déterminé par un couplage capacitif dû à un croisement de connexions en aval de la coupure.



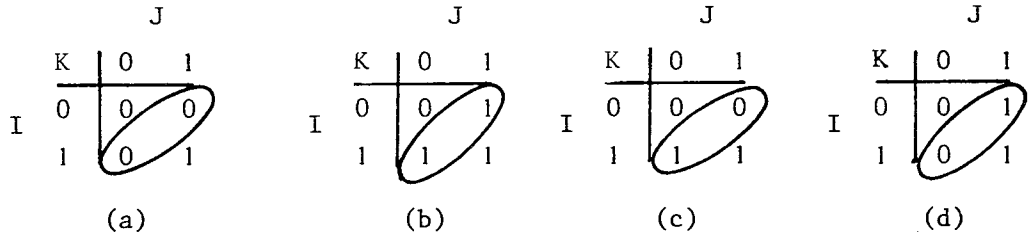
Nous voyons que dans ce cas, la valeur de la connexion I force celle de J.

3.1 - Types de courts-circuits logiques

Le comportement d'un court-circuit logique entre deux connexions obéit aux règles suivantes :

- *les deux connexions en aval du court-circuit sont forcées à la même valeur,*
- *si les connexions en amont du court-circuit ont une même valeur, alors elles ont également cette valeur en aval.*

Nous pouvons donc résumer toutes les possibilités de court-circuit logique sur les tables suivantes :
(I et J représentent les valeurs des connexions en amont du court-circuit, K leur valeur en aval).



Nous définissons donc trois types de courts-circuits logiques :

- *type ET (0 dominant) : table (a)*
- *type OU (1 dominant) : table (b)*
- *type IMPLICATION : tables (c) et (d)*
  (une connexion donne sa valeur à l'autre)
  . I force J : table (c)
  . J force I : table (d)

C'est la connaissance de la technologie qui permet de décider laquelle de ces formes prendra un court-circuit.

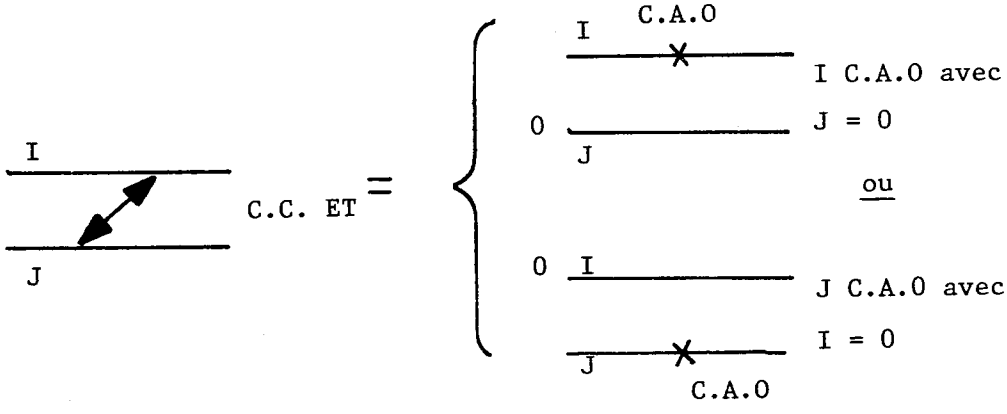
3.2 - Relation entre collages et courts-circuits

Les travaux de Friedman |13| et de Mey |22| permettent d'ignorer les courts-circuits de type ET/OU entre les entrées d'une porte élémentaire. Ceux-ci sont, soit indétectables (si leur type se confond avec la fonction de la porte), soit détectés par le test des collages des entrées.

Mey démontre également que certains courts-circuits, créant des cycles, peuvent être détectés par un test complet des collages, sous certaines conditions. Ces résultats ne sont cependant applicables qu'aux sous-réseaux combinatoires de nos circuits.

Friedman propose, pour tester un court-circuit, de considérer des collages sous contrainte :

Le court-circuit ET entre deux connexions I et J peut être détecté par le test du collage de I à 0 sous la contrainte J = 0 ou du collage de J à 0 sous la contrainte I = 0.



Le court-circuit est donc couvert par chacun des collages sous contrainte.

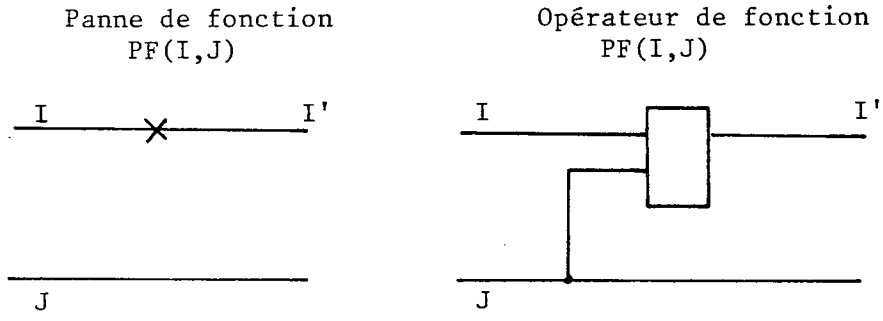
- *Pour un circuit combinatoire* il n'existe pas d'autres possibilités de tester le court-circuit, aussi suffit-il de tenter la génération d'un vecteur de test pour l'un des collages sous contrainte puis, si elle échoue, de le faire pour l'autre.
- *Pour un circuit séquentiel* cette méthode est inapplicable car il existe d'autres possibilités de tester le court-circuit : pour ces circuits il faut, à priori, pour tester une panne, plusieurs vecteurs de test. L'examen exhaustif des possibilités de tester le court-circuit devrait donc envisager l'un ou l'autre des collages sous contrainte, indépendamment dans chacun des vecteurs.

3.3 - Modélisation des courts-circuits

a) Définition

Etant donné deux connexions I et J, nous dirons que la connexion I porte une panne de fonction PF (I,J), si son état logique en aval de la panne est donné par la fonction PF (I,J).

La présence d'une panne de fonction sur une connexion I est donc analogue à la création sur celle-ci d'un opérateur logique particulier de fonction PF (I,J).



Si nous notons V(K) la valeur d'une connexion K, la panne sera mise en évidence lorsque l'on aura :

$$V(I) \neq V(I') = PF(I,J)$$

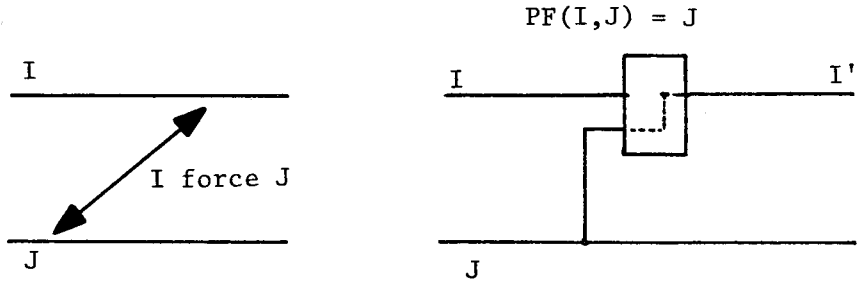
REMARQUE : *On peut considérer les pannes de collage comme des cas particuliers de ces pannes (la fonction PF étant identiquement nulle ou égale à un, selon le collage).*

b) Courts-circuits de type implication

Dans ce type de court-circuit, une seule connexion est affectée par la panne : celle qui est forcée. En effet, la table d'un court-circuit implication I force J peut se résumer à :

I	J
0	0
1	1

Nous modéliserons donc le court-circuit implication d'une connexion I sur une connexion J par une panne de fonction PF (I,J) = J.



Donc, pour ce type de court-circuit, tout se passe comme si, en présence de la panne, la connexion impliquée ne recevait plus que la valeur de la connexion impliquante.

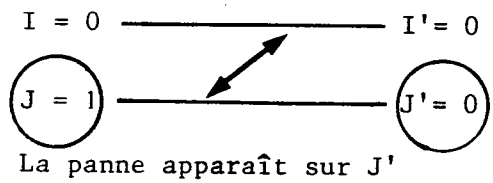
c) Courts-circuits de type ET et de type OU

Nous rappelons que, pour mettre en évidence un court-circuit entre deux connexions, il faut nécessairement que celles-ci soient à des valeurs logiques opposées en amont de la panne.

Dans le cas des courts-circuits ET (OU), chacune des connexions peut, selon la répartition de ces valeurs logiques, être affectée par la panne, bien qu'une seule puisse l'être à la fois :

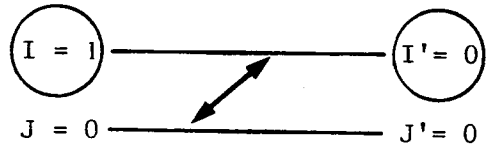
Pour manifester un court-circuit de type ET entre deux connexions I et J, nous aurons :

soit



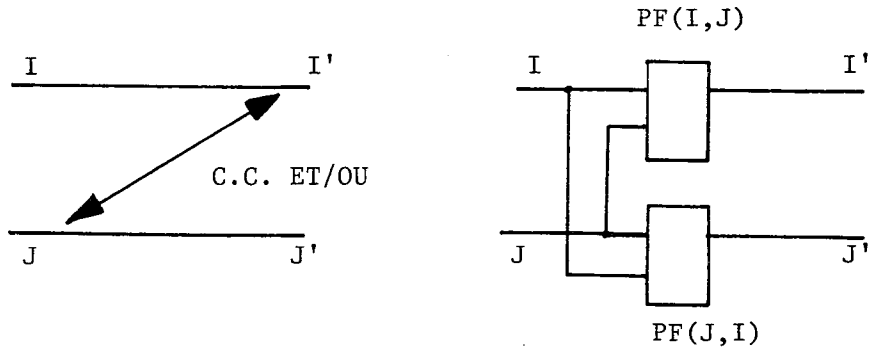
La panne apparaît sur J'

soit



La panne apparaît sur I'

Nous modéliserons donc ces types de courts-circuits par une panne double : chacune des connexions I et J du court-circuit portera une panne de fonction PF (I,J) correspondant au type du court-circuit.



avec PF (I,J) = PF (J,I).

- *Pour un court-circuit ET* nous aurons donc :

$$I' = J' = I.J$$

- *Et pour un court-circuit OU*

$$I' = J' = I + J$$

Nous retrouvons donc bien les tables de définition de ces courts-circuits vues précédemment.

Nous approfondirons la mise en oeuvre de ces définitions lorsque nous aborderons la réalisation de notre programme.

REMARQUE : *Parmi les 16 types de pannes de fonction possibles entre 2 connexions, nous n'en avons considéré que 6 :*

. *2 pour les collages*
. *2 pour les courts-circuits implication*
. *2 pour les courts-circuits ET/OU*

Il en reste donc 10 qui trouveront leur emploi si elles peuvent correspondre à des comportements réels des circuits en présence de certains défauts.

La modélisation de ces défauts pourra utiliser la composition sous forme de panne multiple comme nous l'avons fait pour les courts-circuits ET et OU. Il est également possible de généraliser la notion de panne de fonction, afin de pouvoir considérer des défauts mettant en jeu plus de deux connexions.

4. RESUME DES HYPOTHESES DE PANNES

Les pannes que nous considérerons seront donc les collages et les courts-circuits logiques (type ET, OU, IMPLICATION).

Elles pourront affecter les entrées et les sorties des opérateurs ainsi que les branches d'équipotentielles.

Tout défaut est supposé permanent.

Actuellement, l'hypothèse de défaut unique est faite, mais une version ultérieure du programme pourra considérer les défauts multiples (certains courts-circuits sont déjà assimilés à des pannes doubles).

CHAPITRE IV

STRUCTURATION D'UN PROGRAMME DE TEST

Nous avons défini le type de circuits et les pannes que nous devons considérer. Il reste donc à déterminer comment élaborer une séquence de test dans le cadre des approches structurelles précédemment choisies.

1. STRATEGIE DE TEST D'UN CIRCUIT

1.1 - Rappels

Une analyse est la simulation d'un circuit et de l'ensemble de ses pannes potentielles, cette simulation ayant pour but de déterminer toutes les pannes détectées par un ensemble de vecteurs de test.

Dans cette étude, nous ne considèrerons ce procédé classique des systèmes de test logique que comme un outil efficace et relativement bien rodé, |14, 27, 28|.

Rappelons cependant que sa relative simplicité algorithmique, comparée aux difficultés inhérentes à toute méthode de génération de vecteurs de test, a fait, qu'initialement, elle fût l'unique aide apportée au problème du test logique. Son rôle était alors de valider des séquences de test élaborées manuellement. (Notons que cette approche demeure actuellement la seule utilisée pour les cas difficiles).

On distingue généralement deux grandes familles d'analyse :

- *La simulation parallèle :* simulation logique du circuit sain et parallèlement des N circuits erronés correspondant aux N pannes.
- *La simulation déductive :* simulation logique du circuit sain puis déduction, à partir de son comportement, des pannes détectées.

Les performances de ces deux approches sont assez semblables. Il semble cependant |11| que la deuxième soit préférable pour de gros circuits non fortement séquentiels. Il faut noter, également, que l'utilisation de la simulation déductive est relativement récente et que certaines améliorations intéressantes peuvent encore lui être apportées |37|.

1.2 - Principe de synthèse - analyse

Considérant un circuit à tester, la question immédiate qui se pose est : peut-on élaborer une séquence de test globalement en considérant à chaque instant et simultanément l'effet de tous ses défauts potentiels ?
La réponse est malheureusement non : il est simplement possible de générer des vecteurs de test détectant un sous-ensemble de pannes et d'itérer ce procédé jusqu'à couverture de tout l'ensemble de pannes.

Nous appellerons synthèse ce procédé d'élaboration de vecteurs de test d'un sous-ensemble de pannes, et sous-séquence de test les séquences ainsi déterminées.

Bien que certains travaux |38| tendent à l'élaboration de vecteurs de test détectant un sous-ensemble donné de pannes, on ne sait, dans la pratique, que constater les pannes détectées par une sous-séquence. Celles-ci ne sont donc pas, en général, une donnée des phases de synthèse.

Cette constatation peut être faite en cours de synthèse : c'est le cas du programme LASAR |34|. Il ne résoud cependant pas entièrement ce problème et aboutit à une séquence de test très redondante (pour avoir généré certaines sous-séquences inutilement). La longueur de cette séquence peut être très utile à la localisation des défauts, mais elle est une charge lorsque l'on se préoccupe essentiellement de détection. D'autre part, les hypothèses de pannes nécessitées par cette approche sont limitatives (collages des entrées et sorties d'opérateurs).

Plus généralement, la constatation des pannes détectées par une sous-séquence est faite à postériori : on utilise alors une méthode d'analyse qui peut correspondre à deux besoins :

- *Connaître les pannes non encore détectées.*
- *Vérifier que la sous-séquence est valide : nous avons vu que dans certains cas, les sous-séquences élaborées en phase de synthèse ne sont que potentielles et nécessitent leur validation.*

Si, comme dans le programme LASAR vu précédemment, le rôle de l'analyse se limite à ce deuxième point, elle se situera en fin d'algorithme et sans rétroaction sur la synthèse.
Si elle doit, également, satisfaire le premier point, elle se situera entre deux opérations de synthèse successives. Nous parlerons alors de méthode de synthèse-analyse.

Dans cette méthode, chaque phase de synthèse est caractérisée par une panne dont on recherche des vecteurs de test et est suivie de la phase d'analyse constatant toutes les pannes détectées par ces vecteurs.

Le critère de choix de la panne, pour laquelle on demande chaque génération de vecteurs de test, peut alors influer sur le nombre de sous-séquences, sur leur longueur et sur leur coût d'élaboration. Il a donc trait à l'optimisation du schéma global de génération de séquences de test.

1.3 - Choix d'une panne

Il est possible d'orienter ce choix par des critères reliés à la topologie des circuits. Ils sont cependant insuffisants car, très efficaces en début de test, ils tendent à perdre leur signification au bout d'un certain nombre d'itérations. Citons, par exemple, le programme ATG |12| dont les performances dépendent du degré de vérification de l'hypothèse : des vecteurs de test élaborés pour les pannes des entrées-sorties du circuit détectent le maximum de pannes internes.

Les critères de choix basés sur la topologie des circuits sont donc peu satisfaisants. Celui que nous proposons sera lié à l'état initial du circuit en début de chaque phase de synthèse : lorsque cet état est supposé inconnu, il faut débuter chaque sous-séquence par des vecteurs positionnant le circuit dans un certain état.

La séquence de test, union de telles sous-séquences, a alors la propriété de pouvoir être réagencée : un réordonnancement des sous-séquences permettra d'obtenir une bonne localisation des défauts.

Si, comme dans notre cas, on s'intéresse principalement à la détection des défauts, cette propriété n'est plus nécessaire.
L'élaboration de vecteurs positionnant le circuit en début de chaque sous-séquence peut être alors évitée et l'état initial de chacune de celles-ci correspond à l'état final de la précédente.

Cependant, cette méthode n'est valide que si le comportement du circuit, en présence des vecteurs d'entrée déjà générés, a été étudié en fonction de la panne en cours. Cela peut être assuré par l'analyse si celle-ci considère toujours toutes les pannes potentielles du circuit : elle sera alors capable de donner, en fonction de l'une quelconque de ces pannes, l'état interne dans lequel l'application des vecteurs d'entrée déjà générés laisse le circuit.

Pour une phase de synthèse donnée, on utilisera donc potentiellement l'ensemble des vecteurs d'entrée précédemment élaborés comme une séquence de positionnement du circuit.
Il suffira de choisir une panne dont l'effet est de modifier l'état atteint par le circuit sous l'application de cette séquence, pour qu'une partie du travail incombant normalement à la phase de synthèse soit ainsi déjà effectuée.

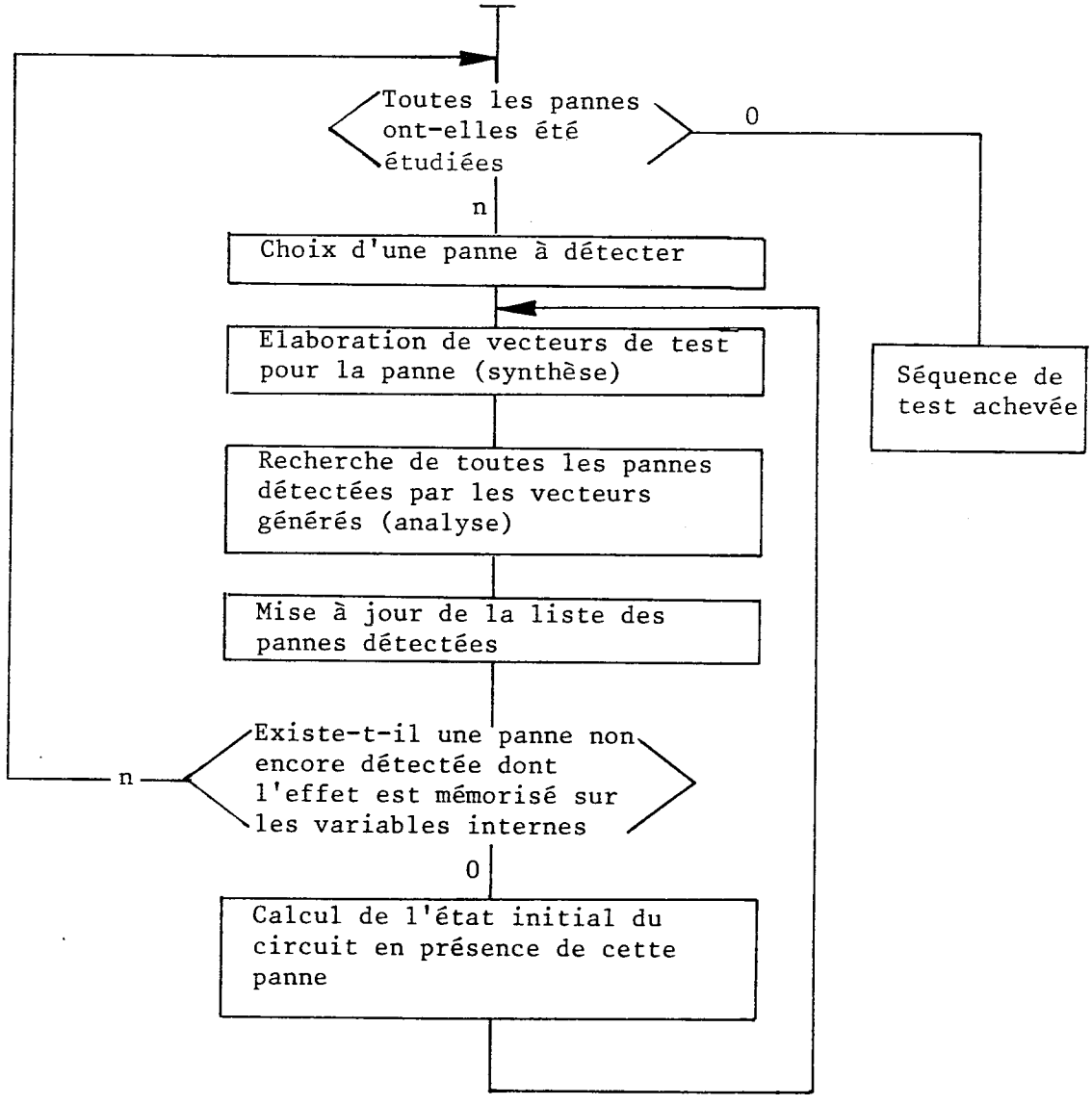
Notre politique de choix de la prochaine panne sera donc :

En fin d'analyse d'une sous-séquence, on recherchera une panne non encore détectée, dont l'effet est mémorisé sur les variables internes des éléments standards séquentiels. Cette panne, et l'état des variables internes correspondant, seront alors donnés en paramètre de la prochaine synthèse.

Lorsqu'on ne trouvera pas de panne par cette méthode, le choix considèrera alors des critères topologiques et l'état initial de synthèse sera considéré comme inconnu. Ce sera, en particulier, le cas pour la première sous-séquence de test du circuit.

Le gain attendu par la méthode précédente est une diminution de la longueur des sous-séquences et de leur coût d'élaboration.

En résumé, l'algorithme de principe de notre programme sera :



2. STRATEGIE DE TEST D'UNE PANNE

2.1 - Rappels des méthodes de chemin-sensible

Le principe des méthodes de chemin-sensible, pour un circuit combinatoire, est de créer dans celui-ci un chemin logique reliant une connexion en panne et une sortie primaire.

Ce chemin doit être tel que, si le circuit est en panne, les valeurs logiques des connexions appartenant à ce chemin seront différentes de celles du circuit sain.

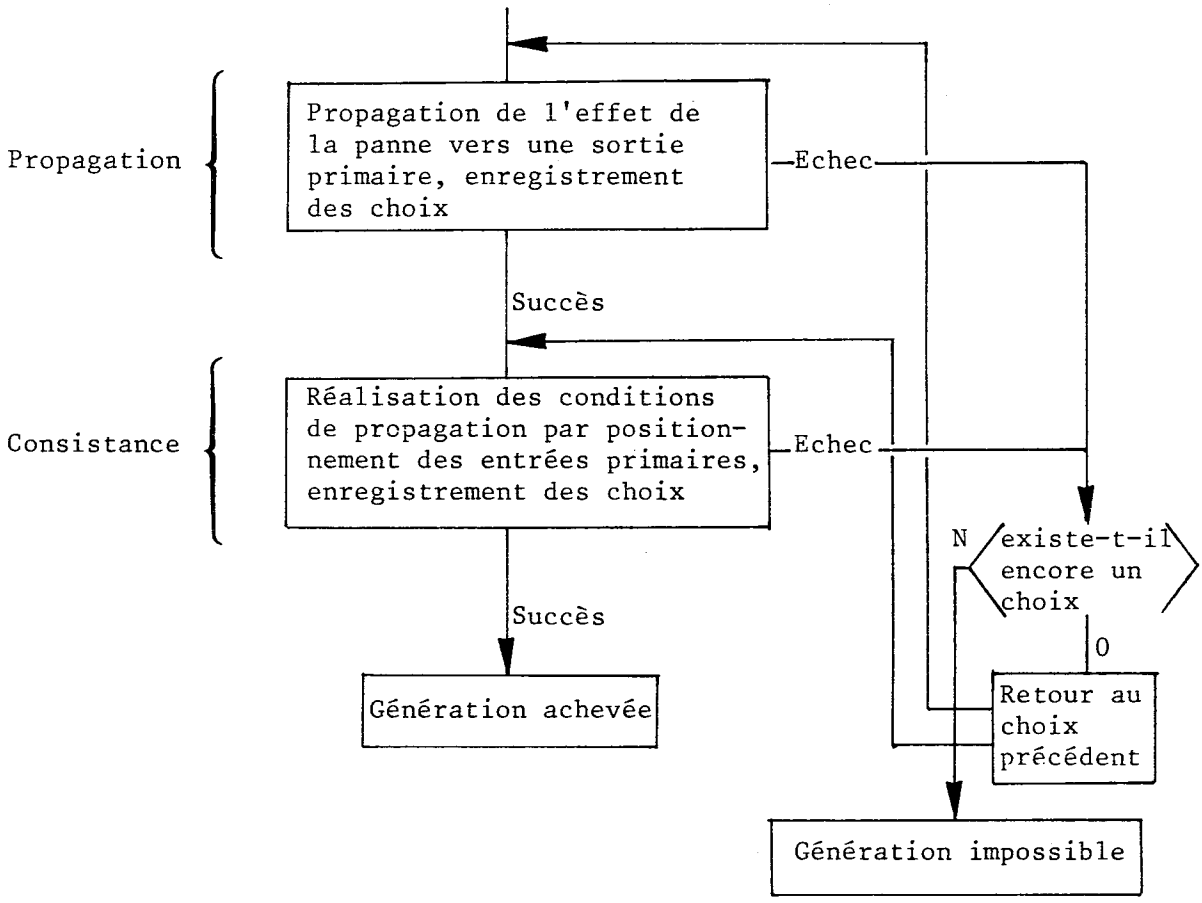
La phase de création d'un tel chemin est habituellement appelée propagation de la panne et nécessite l'assignation de valeurs logiques à des connexions internes du circuit. Il faut donc ensuite assurer ces assignations par le positionnement des entrées primaires : c'est la phase généralement appelée consistance, dont le résultat est un vecteur d'entrée testant la panne donnée.

Les phases de propagation et de consistance sont basées sur une étude de différents choix. L'ordre d'étude de ceux-ci est généralement dicté par des critères heuristiques et certaines opérations de ces phases peuvent échouer. On est alors en présence d'un blocage de l'algorithme :

- *propagation impossible :* la panne ne peut plus être propagée vers une sortie,
- *incohérence :* le fonctionnement demandé au circuit est impossible (par exemple, les deux extrêmités d'un inverseur non en panne à la même valeur logique).

Afin de pouvoir reconsidérer chacune des décisions de propagation et de consistance, les différents choix sont enregistrés et leur environnement est restitué lorsqu'un blocage est constaté : c'est le mécanisme de retour au choix précédent.

Le principe de la méthode de chemin sensible peut être résumé par l'algorithme suivant :



Notons qu'un autre procédé peut être de ramener sur les entrées primaires les conditions de propagation dès que celles-ci apparaissent. Les phases de propagation et de consistance seront alors imbriquées.

2.2 - Adaptation au cas des circuits séquentiels

Le principe de la méthode reste applicable pour les circuits séquentiels. Cependant, la présence des variables internes donne une dimension de plus au problème : les phases de propagation et de consistance doivent être généralisées afin de pouvoir envisager la présence de plusieurs vecteurs d'entrée pour tester la panne.

Chacune de ces deux phases a ainsi un aspect spatial et un aspect temporel.

L'aspect spatial ne concerne que les valeurs logiques à mettre sur un vecteur d'entrée pour que, sous leur action, l'effet de la panne atteigne ou se "rapproche" d'une sortie primaire.

L'aspect temporel concerne la manière d'utiliser ou de positionner les variables internes du circuit.

Pour pouvoir assurer certaines valeurs logiques en sortie d'un élément mémorisant, il peut être nécessaire que celui-ci soit dans un état initial déterminé. La phase de consistance devra assurer cet état.

L'application d'un vecteur d'entrée peut laisser les variables internes des éléments mémorisants dans un état dépendant de la présence de la panne dans le circuit. La phase de propagation devra donc considérer la possibilité d'utiliser les valeurs de ces variables internes comme état initial d'un nouveau vecteur d'entrée qui succèdera au précédent dans la sous-séquence de test.

La définition d'un algorithme de chemin-sensible pour circuits séquentiels doit donc résoudre un certain nombre de points propres à ces circuits. Ces points concernent essentiellement la définition de prépondérance entre les deux aspects des phases de propagation et de consistance.
On peut, en effet, se poser les questions suivantes :

- *Selon quels critères doit-on prendre la décision de demander un nouveau vecteur d'entrée et abandonner la propagation de la panne par le vecteur d'entrée précédent ?*
- *Lorsque cette décision est prise, doit-on achever de spécifier le vecteur d'entrée précédent ou, au contraire, laisser la possibilité de l'utiliser pour assurer l'état initial de certaines variables internes du nouveau vecteur d'entrée ?*
- *La phase de consistance doit-elle s'attacher d'abord à créer des vecteurs d'entrée assurant les valeurs demandées sur les variables internes ou, au contraire, doit-elle d'abord ramener toutes les conditions possibles sur les entrées primaires du circuit pour les vecteurs d'entrée déjà existants ?*

La caractérisation d'un algorithme de chemin-sensible pour circuits séquentiels est fonction des réponses apportées à ces questions.

2.3 - Problème des implications

Les performances de tout algorithme de chemin-sensible dépendent directement du nombre de choix étudiés. Il importe donc, particulièrement pour le test des circuits séquentiels, de minimiser ce nombre.

L'utilisation de critères heuristiques plus ou moins sophistiqués permet généralement de résoudre une partie de ce problème.
Leur définition ne doit cependant intervenir qu'après une étude précise des informations que ces critères peuvent considérer.

Toute décision de propagation ou de consistance doit être prise en fonction du maximum de renseignements sur l'état des connexions du circuit, afin de minimiser le nombre de décisions conduisant à un blocage de l'algorithme.

Il est d'autre part crucial de s'apercevoir de ces blocages le plus rapidement possible car, lorsque l'un de ceux-ci intervient, rien ne permet d'affirmer qu'il soit dû au dernier choix effectué. Il n'existe pas de méthode permettant d'indiquer le choix fautif et de revenir directement le réactiver sans décrire l'arbre de ceux qui lui ont succédé.

On voit donc que les performances d'un algorithme de chemin-sensible seront déterminées par la manière dont sont répercutées, sur l'ensemble du circuit, les décisions prises en phase de propagation ou de consistance.

On appelle généralement implications ces opérations de répercussion. Leur mise en oeuvre doit être considérée comme le point central de tout programme si l'on veut que celui-ci conserve un caractère réaliste.

La définition des implications est étroitement mêlée à celle des informations qu'elles transmettent.

Les algorithmes de chemin-sensible travaillent sur l'algèbre booléenne $B^2 = \{0,1\} \times \{0,1\}$ car ils considèrent simultanément le circuit sain et le circuit affecté d'une panne.

Nous allons voir que, sur cette algèbre, les différentes informations concernant une connexion peuvent s'exprimer par un ensemble de valeurs de la définition duquel dépendra la finesse des informations transmises par les implications.

CHAPITRE V

DÉTERMINATION DES VALEURS DES CONNEXIONS

I. DEFINITIONS

1.1 - Introduction

Soit $\mathcal{B}^2$ l'algèbre booléenne $\{0,1\} \times \{0,1\}$.
Soit (i,j) le doublet représentant l'état d'une connexion respectivement en l'absence et en présence d'une panne dans le circuit. Nous dirons que i est l'état de la connexion dans le circuit juste et j celui dans le circuit faux.

Au moment du test, chacun des états i et j ne peut être que "0" ou "1". La valeur (circuit juste - circuit faux) d'une connexion lors de l'application des vecteurs de test sera donc prise sur l'algèbre $\mathcal{B}^2$ et sera l'une des 4 valeurs :

$$B_1 = (0,0) \qquad B_2 = (1,1)$$
$$B_3 = (1,0) \qquad B_4 = (0,1)$$

Nous appellerons valeurs de base ces valeurs dont la notation classique est respectivement 0, 1, D, $\overline{D}$

On définit généralement sur l'algèbre $\mathcal{B}^2$ des opérations correspondant aux portes logiques élémentaires. Les tables régissant ces opérations sont bien connues :

Z = ET(X,Y)

X \ Y	0	1	D	$\overline{D}$
0	0	0	0	0
1	0	1	D	$\overline{D}$
D	0	D	D	0
$\overline{D}$	0	$\overline{D}$	0	$\overline{D}$

Z = OU(X,Y)

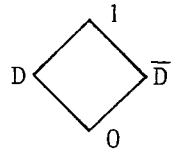
X \ Y	0	1	D	$\overline{D}$
0	0	1	D	$\overline{D}$
1	1	1	1	1
D	D	1	D	1
$\overline{D}$	$\overline{D}$	1	1	$\overline{D}$

Z = NON(X)

X	Z
0	1
1	0
D	$\overline{D}$
$\overline{D}$	D

Nous appellerons une telle table, table fondamentale associée à une opération.

REMARQUE : *Les opérations ET et OU définissent sur B^2 une structure de treillis. Nous pouvons donc résumer les tables fondamentales correspondantes par :*



avec pour définition des bornes supérieures et inférieures :

$$SUP(B_i,B_j) = OU(B_i,B_j) = B_i+B_j$$
$$INF(B_i,B_j) = ET(B_i,B_j) = B_i\ B_j$$
$$\forall\ B_i,B_j \in B^2$$

1.2 - Domaine des valeurs de connexion

En début de génération d'une sous-séquence de test, une connexion a, sauf cas particulier, une valeur totalement indéterminée : elle est susceptible d'être affectée à l'une quelconque des valeurs de base 0, 1, D, $\overline{D}$. La génération a pour effet de choisir l'une de ces valeurs.
Cependant, cette spécification peut être faite pas à pas par diminution progressive de l'ensemble des valeurs de base auxquelles la connexion peut être affectée.

Exemple : *La spécification de la valeur d'une connexion pourra considérer successivement les ensembles de valeurs possibles suivants :* {0, 1, D, $\overline{D}$} {0, 1, $\overline{D}$} {1, $\overline{D}$} et enfin {$\overline{D}$}

Soit $\mathcal{D} = P(B^2)$ l'ensemble des parties de l'ensemble {0, 1, D, $\overline{D}$}. Nous appellerons ensemble des valeurs de base possibles d'une connexion tout élément de cet ensemble.

Les opérations ensemblistes classiques d'union et d'intersection définissent sur cet ensemble un treillis dont les **U**-générateurs sont {0}, {1}, {D}, {$\overline{D}$}.

Considérant un ensemble de valeurs possibles V tel que $V \in \mathcal{D} - \{\emptyset\}$ nous avons donc :

$$V = \{B_1 \cup B_2 \ldots \cup B_p\}$$

avec $B_i \in \{\{0\}, \{1\}, \{D\}, \{\overline{D}\}\}$ pour tout i

Nous appellerons valeur d'une connexion l'ensemble de ses valeurs de base possibles.

Dans un but de simplification des écritures, nous identifierons parfois un ensemble de valeurs possibles avec son contenu en notant

$B_1\ B_2 \ldots\ B_p$ la valeur $\{B_1 \cup B_2 .. \cup B_p\}$ d'une connexion.

La valeur d'une connexion sera donc prise sur l'ensemble $\mathcal{D}$ à 16 éléments suivant, qui correspond à celui défini par Akers |1|.

0	D0	01	0D$\overline{D}$
1	D1	D$\overline{D}$	1D$\overline{D}$
D	$\overline{D}$0	01D	01D$\overline{D}$
$\overline{D}$	$\overline{D}$1	01$\overline{D}$	$\emptyset$

Nous appellerons cet ensemble $\mathcal{D}$ domaine des valeurs de connexions.

01D$\overline{D}$ représente l'indétermination totale (I).

$\emptyset$ (ensemble vide) représente l'incohérence : la connexion ne peut prendre aucune des valeurs de base.

Nous conserverons sur cet ensemble les opérations d'union et d'intersection et la relation d'inclusion telles qu'elles sont définies sur le treillis.

1.3 - Opérations sur le domaine de valeurs

a) Extension des opérations définies sur $\mathcal{B}^2$

Soit une opération binaire notée $\star$ et définie sur l'ensemble {0, 1, D, $\overline{D}$}. Nous définirons l'extension de cette opération à l'ensemble $\mathcal{D} = P(\{0, 1, D, \overline{D}\})$ par distributivité de l'opération $\star$ par rapport à l'union ($\cup$) :

Soit $V_1 \in \mathcal{D} - \{\emptyset\}$

et $V_2 \in \mathcal{D} - \{\emptyset\}$

avec $V_1 = \{V_1^1 \cup V_1^2 \ldots \cup V_1^p\}$

et $V_2 = \{V_2^1 \cup V_2^2 \ldots \cup V_2^q\}$

Nous définirons

$V_1 \ast V_2 = \{V_1^1 \cup V_1^2 \ldots \cup V_1^p\} \ast \{V_2^1 \cup V_2^2 \ldots \cup V_2^q\}$

par

$V_1 \ast V_2 \equiv \{(V_1^1 \ast V_2^1) \cup (V_1^1 \ast V_2^2) \ldots \cup (V_1^1 \ast V_2^q) \ldots \cup (V_1^p \ast V_2^1) \ldots \cup (V_1^p \ast V_2^q)\}$

C'est-à-dire :

$V_1 \ast V_2 \equiv \{(V_1^i \ast V_2^j)\}$ pour tout couple i, j.

Exemple :

$ET(D\overline{D}, 1D\overline{D}) =$

$ET(D,1) \cup ET(D,D) \cup ET(D,\overline{D}) \cup ET(\overline{D},1) \cup ET(\overline{D},D) \cup ET(\overline{D},\overline{D})$

$= D \cup D \cup 0 \cup \overline{D} \cup 0 \cup \overline{D}$

$= 0D\overline{D}$

L'extension de l'opération unaire NON définie sur $\{0, 1, D, \overline{D}\}$ à l'ensemble $\mathcal{D}$ sera :

soit $V \in \mathcal{D} - \{\emptyset\}$

avec $V = \{V^1 \cup V^2 \ldots \cup V^p\}$

alors $NON(V) \equiv \{NON(V^1) \cup NON(V^2) .. \cup NON(V^p)\}$

Exemple :

$NON(D0) = NON(D) \cup NON(0) = \overline{D} \cup 1 = \overline{D}1$

REMARQUE : *L'associativité des opérations définies sur $\{0, 1, \mathcal{D}, \overline{\mathcal{D}}\}$ permettra sur $\mathcal{D}$ la généralisation des opérations binaires à des opérations n-aires.*

Aspect pratique

Pour le traitement des opérations binaires il faut considérer, dans la table fondamentale associée à l'opération, la sous-matrice obtenue en ne considérant dans cette table respectivement que les lignes et les colonnes correspondant aux valeurs de base possibles des entrées. L'union des termes de cette sous-matrice donne alors le résultat de l'opération.

Exemple :

$$Z = ET(X,Y)$$

$$V_X = D\overline{D}$$

$$V_Y = 1D\overline{D}$$

La sous-matrice correspondante est :

X \ Y	1	D	$\overline{D}$
D	D	D	0
$\overline{D}$	$\overline{D}$	0	$\overline{D}$

Nous avons donc

$ET(D\overline{D}, 1D\overline{D}) = D \cup D \cup 0 \cup \overline{D} \cup 0 \cup \overline{D} = 0D\overline{D}$

b) Implications

Nous appellerons restriction de la valeur d'une connexion toute diminution de l'ensemble de ses valeurs de base possibles.

Soit un opérateur logique réalisant $Z = X \star Y$ et V_Z, V_X, V_Y la valeur de ces variables prises sur $\mathcal{D} - \{\emptyset\}$.
Une implication aval des entrées X, Y sur la sortie Z sera la restriction de la valeur V_Z de celle-ci à la valeur V'_Z définie par :

$$V'_Z = V_Z \wedge (V_X \star V_Y)$$

Lorsque cette intersection sera vide nous serons en présence d'une incohérence.

Inversement, nous pouvons définir une restriction des valeurs des entrées d'un opérateur : soit $V_Z = V_X \star V_Y$, nous cherchons à connaître les restrictions pouvant être apportées aux valeurs V_X et V_Y lorsque V_Z devient $V'_Z \subset V_Z$.

Une implication amont de la sortie Z, passant de la valeur V_Z à V'_Z, sur les entrées X et Y sera la restriction d'au moins une des valeurs V_X ou V_Y à une valeur respective V'_X ou V'_Y avec

$$V'_X = \bigcup_i V_X^i \; t_q(V_X^i * V_Y) \wedge V'_Z \neq \emptyset$$

et

$$V'_Y = \bigcup_j V_Y^j \; t_q(V_X * V_Y^j) \wedge V'_Z \neq \emptyset$$

Le traitement pratique d'une implication amont sur un opérateur considèrera dans la table fondamentale associée à celui-ci la même sous-matrice que celle précédemment définie. Les valeurs V'_X et V'_Y seront obtenues dans cette sous-matrice en leur affectant toute valeur de base dont respectivement la ligne ou la colonne correspondante contient une valeur de base incluse dans la valeur V'_Z de la sortie.

Exemple :

$Z = ET(X,Y)$

$V_X = 1D\overline{D}$

$V_Y = 01D$

$V'_Z = \overline{D}1$

La sous-matrice correspondant aux valeurs des entrées est :

X \ Y	0	1	D
1	0	1	D
D	0	D	D
$\overline{D}$	0	$\overline{D}$	0

Nous avons donc :

$V'_X = \overline{D}1$: seules les lignes (1) et (3) contiennent des valeurs de base incluses dans V'_Z.

$V'_Y = 1$: seule la colonne (2) contient des valeurs de base incluses dans V'_Z.

REMARQUES :

- Le traitement des opérateurs NON sera donné par la simplification de ce qui précède au cas des opérateurs unaires.
- La décomposition des opérateurs à n entrées en réseaux d'opérateurs à deux entrées permettra l'extension des opérations d'implication au cas général.

2. RESTRICTION DU DOMAINE DES VALEURS DE CONNEXIONS

2.1 - Assignation initiale et stabilité

Le traitement des implications sur le domaine $\mathcal{D}$ peut être relativement onéreux.

On peut alors se poser les deux questions :

- *Dans quelle mesure les 16 valeurs de ce domaine sont-elles nécessaires à un algorithme de génération de vecteurs de test ?*
- *Dans quelle mesure une réduction de ce domaine peut-elle diminuer le coût de traitement des implications ?*

Pour tenter de répondre à ces questions, nous allons étudier un critère de stabilité.

Soit $\mathcal{D}'$ un sous-ensemble de valeurs inclus dans le domaine $\mathcal{D}$.

Considérant le principe des algorithmes de chemin-sensible, nous dirons que $\mathcal{D}'$ doit nécessairement contenir les valeurs de base 0, 1, D, $\overline{D}$ et la valeur indéterminée I.

Nous appellerons assignation initiale l'inclusion de ces valeurs au sous-ensemble $\mathcal{D}'$.

Nous dirons qu'un sous-ensemble $\mathcal{D}'$ du domaine de valeurs $\mathcal{D}$ est un sous-domaine, s'il est stable pour les opérations précédemment définies sur $\mathcal{D}$.

La condition de stabilité n'est pas strictement nécessaire : si le résultat d'une opération sur un sous-ensemble $\mathcal{D}'$ est une valeur ne faisant pas partie de ce sous-ensemble, il est toujours possible de l'appeler valeur indéterminée I. On voit cependant que l'on perd alors de l'information.

Exemple : *Le D-algorithme classique considère le sous-ensemble D' réduit à {0, 1, D, $\overline{D}$, I}. Si nous étudions un opérateur ET ayant en entrée deux quelconques de ces valeurs, nous obtenons la table :*

	0	1	D	$\overline{D}$	I
0	0	0	0	0	0
1	0	1	D	$\overline{D}$	I
D	0	D	D	0	D0
$\overline{D}$	0	$\overline{D}$	0	$\overline{D}$	$\overline{D}$0
I	0	I	D0	$\overline{D}$0	I

Nous constatons la non stabilité de cette opération sur ce sous-ensemble de valeurs :
Nous avons ET(D,I) = D0 (dans le circuit faux, la valeur logique "0" en entrée de l'opérateur détermine la valeur logique "0" en sortie) et ET($\overline{D}$,I) = $\overline{D}$0 (même raisonnement appliqué au circuit juste). Ne pouvant exprimer ces valeurs, le D-algorithme les remplace par la valeur indéterminée I.

Certains auteurs |7, 17| tentent de palier à ces pertes d'informations en considérant deux types de valeurs indéterminées :

- *valeur I = valeur totalement indéterminée*
- *valeur U = valeur non totalement indéterminée mais intraduisible.*

L'adjonction de cette dernière valeur pour tenter de modéliser plus finement le comportement du circuit n'est, en fait, qu'un pis aller pour tenter de palier à la perte d'information dûe à la non stabilité. Son intérêt ne motive pas sa mise en oeuvre.

Nous voyons donc que le critère de stabilité proposé a pour but de ne perdre aucune des informations transmises par les opérations d'implication. Il peut également influer sur la facilité de la mise en oeuvre de celles-ci : leur traitement pour un sous-ensemble de valeurs non stable doit obéir à certaines règles d'exception qui peuvent alourdir leur mise en oeuvre.

2.2 - Sous-domaines de valeurs

Nous venons de voir que la stabilité des sous-domaines correspondant à l'assignation initiale pour les opérations d'implication nécessite la présence des valeurs D0 et $\overline{D}0$.
Il faut adjoindre également les valeurs $\overline{D}1$ et D1 obtensibles par implication sur les opérateurs de complémentation.
Nous obtenons alors le sous-domaine $\mathcal{D}_1$ correspondant aux valeurs définies par Muth |23|.

$\mathcal{D}_1 = \{0, 1, D, \overline{D}, D0, D1, \overline{D}0, \overline{D}1, I\}$

Une vérification exhaustive peut montrer que ce sous-ensemble de $\mathcal{D}$ est bien stable pour toutes les opérations d'implication.

La recherche des autres sous-ensembles de valeurs stables correspondant à l'assignation initiale nous donne les sous-domaines de valeurs suivants :

$\mathcal{D}_2 = \mathcal{D}_1 \cup \{0D\overline{D}, 1D\overline{D}\}$

$\mathcal{D}_3 = \mathcal{D}_1 \cup \{01D, 01\overline{D}\}$

$\mathcal{D}_4 = \mathcal{D}_2 \cup \{D\overline{D}\}$

$\mathcal{D}_5 = \mathcal{D}_3 \cup \{01\}$

La stabilité d'un sous-domaine garantit que jamais il n'y aura naissance d'une nouvelle valeur lors des opérations d'implication.
La présence de valeurs ne faisant pas partie du sous-domaine ne peut donc provenir que d'un forçage d'une connexion à cette valeur.
On peut se demander alors dans quelle mesure ces forçages peuvent être utiles. Il faut pour cela étudier la signification des valeurs de $\mathcal{D}$ non présentes dans $\mathcal{D}_1$.

- *Valeur $D\overline{D}$*

Sa signification est "valeur sensible à la panne", son utilisation peut permettre d'éviter la génération d'un choix de propagation. On peut, en effet, imposer cette valeur à une connexion et laisser les implications décider laquelle des deux valeurs de base D ou $\overline{D}$ la connexion pourra effectivement prendre.

- *Valeurs $0D\overline{D}$ et $1D\overline{D}$*

Leur signification respective est :"valeur de base 1/0 impossible". Elles peuvent permettre la transcription de contraintes particulières.

- *Valeur 01*

Sa signification est "valeur non sensible à la panne". Elle est utilisable pour transmettre la notion d'entrée primaire. Ces connexions sont, en effet, les seules devant nécessairement être au même état logique en présence ou non d'une panne dans le circuit.

- *Valeurs $01D$ et $01\overline{D}$*

Leur signification respective est : "valeur de base $\overline{D}$/D impossible" et leur utilisation est similaire à celle de $0D\overline{D}$ et $1D\overline{D}$.

Nous voyons donc que deux de ces valeurs peuvent être intéressantes dans notre cas : les valeurs 01 et $D\overline{D}$.
Leur adjonction à $\mathcal{D}_1$ nécessite cependant, pour satisfaire le critère de stabilité, l'adjonction des valeurs $0D\overline{D}$, $1D\overline{D}$, 01D et $01\overline{D}$. Nous aurions alors le domaine $\mathcal{D} - \{\emptyset\}$ (l'incohérence n'est pas à considérer pour le critère de stabilité car les opérations d'implication ne sont pas définies pour cette valeur).

Nous choisirons donc pour représenter les valeurs des connexions le sous-domaine :

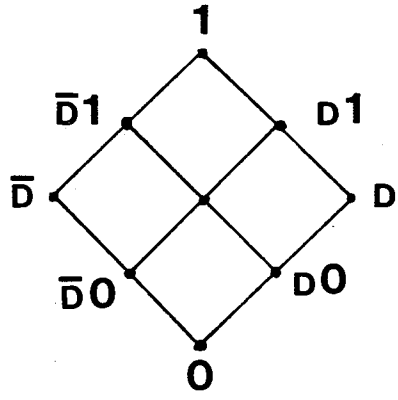
$\mathcal{D}_1 = \{0, 1, D, \overline{D}, D0, D1, \overline{D}0, \overline{D}1, I\}$

dont nous allons considérer les propriétés.

2.3 - Propriété du sous-domaine $\mathcal{D}_1$

a) Structure de treillis

Les opérations ET et OU, précédemment définies, induisent sur $\mathcal{D}_1$ une structure de treillis :



Ces opérations sont donc résumées sur le treillis par :

$SUP(V_1, V_2) = OU(V_1, V_2)$
$INF(V_1, V_2) = ET(V_1, V_2)$ $\quad \forall\ V_1, V_2 \in \mathcal{D}_1$

b) Simplification des opérations d'implication

Soit l'ensemble {0, 1, x} avec $x \in \{0,1\}$
Le sous-domaine $\mathcal{D}_1$ est isomorphe au produit cartésien {0, 1, x} x {0, 1, x} par la bijection :

0 = (0,0)	D0 = (x,0)
1 = (1,1)	D1 = (1,x)
D = (1,0)	$\bar{D}0$ = (0,x)
$\bar{D}$ = (0,1)	$\bar{D}1$ = (x,1)

I = (x,x)

(i,j) représentant les valeurs d'une connexion respectivement dans le circuit juste et dans le circuit faux. (valeurs prises sur {0, 1, x}).

La bijection pour les valeurs de base est évidente. Pour les autres valeurs elle est mise en évidence par décomposition.

Exemple : $D0 = D \cup 0 = (1,0) \cup (0,0) = (1 \cup 0,\ 0 \cup 0) = (x,0)$

Nous avons donc la propriété suivante :

> Pour résoudre les implications il suffit de traiter séparément les deux circuits et de considérer que dans chacun de ceux-ci la valeur d'une connexion est prise sur l'ensemble {0, 1, x}.

Nous sommes donc ramenés à utiliser l'algèbre ternaire {0, 1, x}. Sur cette algèbre, les opérations élémentaires sont très proches de celles de la logique habituelle :

$Z = ET(A,B)$

Z	B=0	B=1	B=x
A=0	0	0	0
A=1	0	1	x
A=x	0	x	x

$Z = OU(A,B)$

Z	B=0	B=1	B=x
A=0	0	1	x
A=1	1	1	1
A=x	x	1	x

$Z = NON(A)$

A	Z
0	1
1	0
x	x

Pour les deux premières opérations, ces tables peuvent être résumées par la relation d'ordre $0 \leq x \leq 1$ et les règles :

$$\left.\begin{array}{l} ET(a,b) = MIN(a,b) \\ OU(a,b) = MAX(a,b) \end{array}\right\} \forall\ a,b \in \{0,1,x\}$$

Le traitement des implications est alors très simplifié.

Exemple : Soit $Z = ET(A,B)$

Avec $V_A = D0 = (x,0)$

$V_B = \overline{D}1 = (x,1)$

Sur l'ensemble $\mathcal{D}_1$ nous devrions considérer :

$V_Z = ET(D,\overline{D}) \cup ET(D,1) \cup ET(0,\overline{D}) \cup ET(0,1) = D0.$

Sur l'ensemble $\{0,1,x\}^2$ nous avons simplement :

$V_Z = (ET(x,x),\ ET(0,1)) = (x,0).$

Nous identifierons dorénavant le sous-domaine $\mathcal{D}_1$ à l'ensemble $\{0,1,x\}^2$

Notre algorithme de génération de vecteurs de test considèrera toujours indépendamment le circuit juste et le circuit faux, sauf lors des opérations de propagation et lorsque l'on accèdera à une entrée primaire. Nous verrons que cette discrimination englobera également l'aspect temporel de la génération de vecteurs de test.

REMARQUE : *Si l'on recherche une caractérisation des valeurs du domaine initial D qui ne sont pas exprimables sur le produit cartésien* $\{0,1,x\}^2$, *on s'aperçoit alors que ces valeurs introduisent toutes une relation entre le circuit juste et le circuit faux. Par exemple, la valeur 01 qui signifie "état dans les deux circuits nécessairement égaux", ne serait traduisible que par le couple de doublets* $\{\begin{matrix}(0,0)\\(1,1)\end{matrix}$.

De même, la valeur $1D\overline{D}$, *qui signifie "état dans les deux circuits non simultanément nul ", ne serait traduisible que par les trois doublets :* $\{\begin{matrix}(1,1)\\(0,1)\\(1,0)\end{matrix}$

c) Prise en compte des effets répétés d'une panne

L'aspect séquentiel des circuits rend possible la présence d'une manifestation de la panne en amont de la connexion la portant. Il faut alors pouvoir déterminer l'effet d'une telle configuration.

Grâce à la séparation circuit juste - circuit faux, ce cas sera banalisé avec le traitement normal des pannes :

- *Dans le circuit juste, il n'en est pas tenu compte.*
- *Dans le circuit faux, la valeur en sortie de la connexion en panne est donnée par son type et éventuellement (courts-circuits) par des valeurs de connexion.*

Exemple : *Notons* J_c *et* F_c *les valeurs respectives d'une connexion dans le circuit juste et dans le circuit faux.*
Si X et X' sont les parties amont et aval d'une connexion dont on teste le collage à V (0 ou 1), nous aurons toujours :

$F_{X'} = V$

$J_{X'} = J_X$

F_X = *inopérant.*

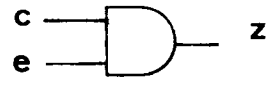
Donc, si une manifestation de la panne est présente sur X, nous n'en considèrerons que sa valeur dans le circuit juste. Selon celle-ci, une manifestation de la panne apparaîtra ou non sur X'.

Nous voyons que le sous-domaine de valeurs $\mathcal{D}_1$ permet de prendre en compte les effets répétés d'une panne et, par extension, les pannes multiples.

d) Exemples d'application

Nous allons comparer sur quelques exemples l'utilisation du sous-domaine $\mathcal{D}_1$ et celle du sous-ensemble $\mathcal{D}_A = \{0,1,D,\overline{D},I\}$ du D-algorithme classique.

- *Implication aval*



Sur $\mathcal{D}_1$ **c** = (0,1), **e** = (X,0) } => **z** = (0,0)

Sur $\mathcal{D}_A$ nous aurions

e = I = (X,X) et **z** = I = (X,X).

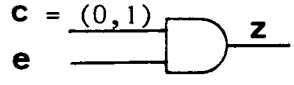
- *Implication amont*

Pour le même opérateur :

Sur $\mathcal{D}_1$ **c** = (1,0), **z** = (0,0) } => **e** = (0,X)

Sur $\mathcal{D}_A$ nous aurions **e** = (0,0) et la condition demandée sur **e** dans le circuit faux serait donc abusive.

- *Diminution du nombre de choix de propagation*



Pour manifester la panne sur la sortie **z**, nous demanderons :

Sur $\mathcal{D}_1$: **e** = (X,1)

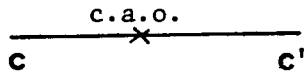
Sur $\mathcal{D}_A$: il faudrait étudier indépendamment les deux possibilités

e = (1,1)

e = (0,1)

et donc générer un choix.

- *Mise en évidence des pannes*

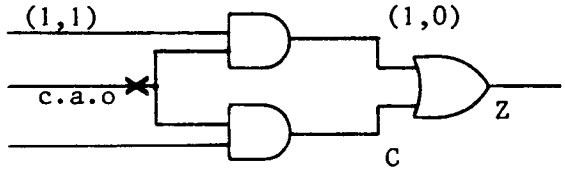


Pour avoir **c'** = (1,0) = D nous demanderons :

Sur $\mathcal{D}_1$: **c** = (1,X)

Sur $\mathcal{D}_A$: **c** = (1,1) : la condition demandée sur **c** dans le circuit faux serait donc abusive.

- *Traitement des reconvergences*



Pour manifester la panne sur Z on demande sur $\mathcal{D}_A$:

C = (1,1) ce qui est impossible d'où échec.

Sur $\mathcal{D}_1$ la panne est manifestée sur la sortie Z par simple implication car la connexion C prend la valeur (X,0).

Toutes ces améliorations deviennent très efficaces lorsqu'on les considère globalement sur l'ensemble d'un circuit.

En résumé

Le choix du sous-domaine de valeurs $\mathcal{D}_1$ = {0,1,D,$\overline{D}$,D0,D1,$\overline{D}$0,$\overline{D}$1,I} a été motivé par :

- *la facilité de la mise en oeuvre des opérations d'implication sur ce sous-domaine,*
- *l'ensemble de ses propriétés qui constitue l'essentiel de ce qui manquait à l'ensemble de valeurs du D-algorithme.*

CHAPITRE VI

ALGORITHME DE SYNTHESE

1. DEFINITIONS

Nous appellerons vecteur d'état un vecteur dont les composantes représentent respectivement les valeurs, à un instant donné, des connexions d'un circuit.

Ces valeurs seront prises sur l'ensemble {0, 1, X}.

Initialement, les composantes d'un vecteur d'état seront à la valeur X.

Nous appellerons positionnement d'une composante le remplacement de sa valeur X par une valeur 0 ou 1.

La spécification d'un vecteur d'état sera la positionnement progressif de certaines de ses composantes.

Nous considèrerons deux familles de vecteurs d'états correspondant respectivement au circuit juste et au circuit faux.
Entre ces deux familles nous établirons la correspondance suivante :
Considérant un vecteur d'état dans chacune des familles, nous dirons qu'ils forment un couple de vecteurs d'état s' ils représentent respectivement l'état de chacun des circuits pour l'application du même vecteur d'entrée de la sous-séquence en construction.

Pour un couple de vecteurs d'état, nous aurons donc égalité entre les valeurs respectives correspondant à chacune des entrées primaires; ces valeurs détermineront le vecteur d'entrée associé au couple.

Lors des différentes phases de l'algorithme, les vecteurs d'état pourront être considérés par couple, par famille ou indépendamment.

Afin de prendre en compte l'état initial du circuit, chaque vecteur d'état comportera des composantes appelées pseudo-entrées correspondant aux valeurs initiales des variables internes (les variables x et y des éléments standards séquentiels).

Les valeurs finales des variables internes seront celles des sorties des éléments standards séquentiels. Nous appellerons celles-ci pseudo-sorties.

Le nombre de vecteurs d'entrée d'une sous-séquence sera supposé initialement être égal à un. Nous appellerons prolongation aval (amont) de la sous-séquence le rallongement de celle-ci par adjonction d'un vecteur d'entrée suivant (précédent) ceux déjà pris en compte.

Nous remarquerons que, lorsqu'un état initial est pris en compte par la synthèse en même temps que la panne à tester, seule la prolongation aval de la sous-séquence est possible.

Nous avons vu, précédemment, la définition des opérations élémentaires d'implication.
Nous appellerons implications dans un vecteur d'état l'ensemble de toutes les implications possibles sur ses composantes.

Nous appellerons implications dans une famille de vecteurs d'état l'ensemble de toutes les implications possibles dans chacun de ses constituants et de toutes les implications possibles entre ceux-ci.

Ces dernières implications considèreront donc l'adjacence des vecteurs d'état déterminée par la correspondance des pseudo-entrées et pseudo-sorties.

Une phase d'implications sera l'ensemble de toutes les implications possibles dans les deux familles de vecteurs d'état.

Lorsqu'une phase d'implications aura été effectuée sans rencontrer d'incohérence, nous dirons que les vecteurs d'état sont cohérents.

Nous appellerons assignation tout positionnement d'une composante d'un vecteur d'état cohérent.

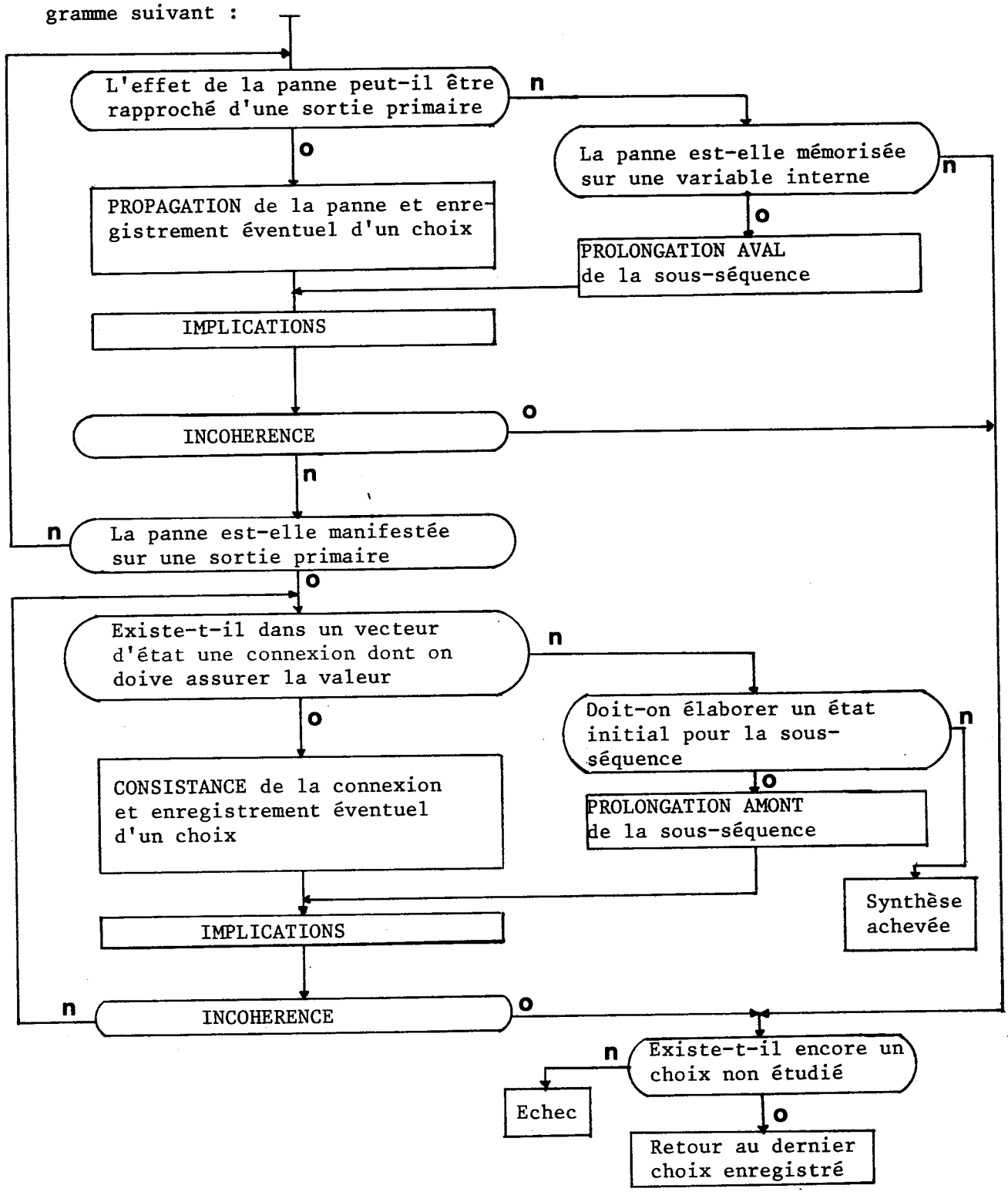
Dans un premier temps, nous considèrerons, pour les opérations de propagation et de consistance, les définitions suivantes :
Une opération de propagation est une assignation dont le rôle est de "rapprocher" l'effet de la panne d'une sortie primaire.

Une opération de consistance est une assignation dont le rôle est d'assurer la valeur en sortie d'un opérateur par celle de ses entrées. Chaque opération de consistance visera donc à "rapprocher" des entrées du circuit les conditions logiques demandées sur ses connexions internes.
Nous approfondirons ultérieurement ces deux définitions.

Nous remarquerons que la définition des opérations d'implication précédemment donnée sous-entend que l'élaboration d'une sous-séquence de test est une opération concernant, à chaque pas, tous les vecteurs d'état. Ceux-ci ne seront, en effet, pas spécifiés successivement mais au contraire simultanément.

2. ALGORITHME

Le schéma de principe de l'algorithme de synthèse est donné par l'organigramme suivant :



L'ensemble de l'algorithme est contrôlé par le mécanisme de gestion des choix : lorsqu'en propagation ou consistance plusieurs possibilités se présentent simultanément, l'ensemble des vecteurs d'état est sauvegardé avant qu'une décision soit prise (enregistrement d'un choix).

Lorsqu'une incohérence est constatée, l'ensemble des vecteurs d'état correspondant au dernier choix est restauré et une nouvelle décision est prise (retour au dernier choix enregistré).

2.1 - Représentation des pannes

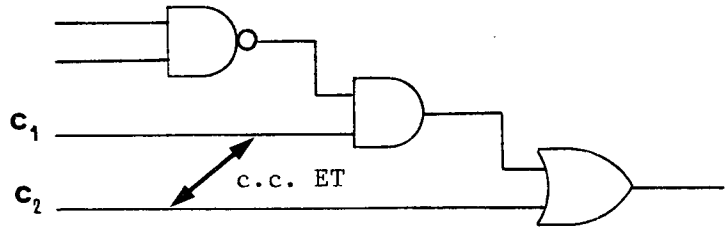
Une panne est représentée par un ou plusieurs blocs panne qui sont des opérateurs au même titre que les autres composants du circuit, mais dont la fonction dans le circuit juste est l'identité.

Pour une panne de collage, ce bloc est un simple opérateur, à une entrée et une sortie, qui a rôle de transmission dans le circuit juste et de positionnement à 0 ou 1 selon le collage dans le circuit faux.

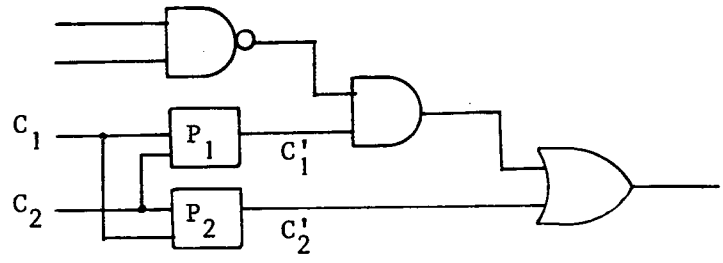
a) Courts-circuits de type ET et de type OU

Ils sont modélisés chacun par deux blocs panne ayant, en entrée, les deux connexions en court-circuit et, en sortie, respectivement l'une et l'autre de ces connexions.
Comme pour les collages, ces blocs ont une fonction de transmission simple dans le circuit juste.
Dans le circuit faux ils assurent le ET (OU) des entrées.

Exemple :



Le court-circuit ET entre C_1 et C_2 sera modélisé par :



P_1 et P_2 étant les blocs panne réalisant respectivement :

$C'_1 = C_1$
$C'_2 = C_2$ } dans le circuit juste

$C'_1 = ET\ (C_1, C_2)$
$C'_2 = ET\ (C_1, C_2)$ } dans le circuit faux

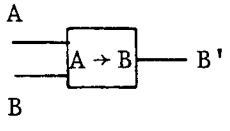
Selon les valeurs de C_1 et C_2 dans le circuit juste, la panne pourra se manifester sur C'_1 ou sur C'_2 ou être inhibée.

Le traitement de l'un des blocs panne (que ce soit en implication, propagation ou consistance) sera toujours fait indépendamment de l'autre : ces blocs seront considérés comme les constituants d'une panne multiple.

b) Court-circuit de type IMPLICATION

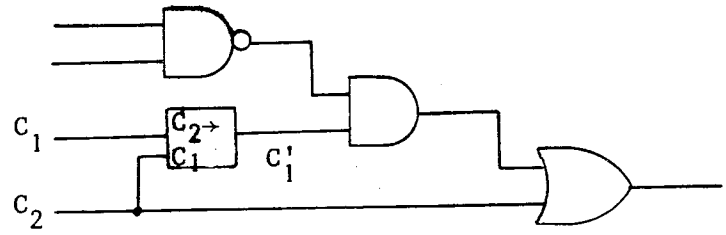
Il est modélisé par un seul bloc panne (seule la connexion forcée est affectée par la panne).
Ce bloc aura, en entrée les deux connexions en court-circuit et, en sortie la connexion forcée :

Court-circuit A force B (A → B)



B' = B dans le circuit juste
B' = A dans le circuit faux

Exemple : *Le circuit précédent, pour un court-circuit de type* C_2 *force* C_1, *sera représenté par :*



2.2 - Implications

a) Implications dans un vecteur d'état

La propriété (chapitre II) concernant le fonctionnement des circuits lors du test nous permet d'appeler incohérence toute tentative de passage d'une composante d'un vecteur d'état de la valeur 0 à la valeur 1 ou inversement.

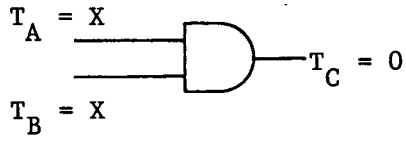
La mise en oeuvre des implications obéit à la règle suivante :

- *Tout opérateur dont l'une des connexions d'entrée-sortie a été positionnée est étudié.*
- *La comparaison entre la valeur de la sortie et celle calculée à partir des entrées permet de :*
  - . détecter les incohérences,
  - . décider du type d'implication (amont ou aval).

L'algorithme de traitement d'un opérateur de fonction Z = F(A,B,C...) est le suivant :

- *Calcul de la valeur* T'_Z *de la sortie à partir des valeurs des entrées :* $T'_Z = F(T_A, T_B, T_C \ldots)$
- *Comparaison de* T'_Z *avec la valeur* T_Z *de la sortie*
  - . si $T_Z = \overline{T}'_Z$: incohérence
  - . si $T_Z = T'_Z$: aucune action
  - . si $T_Z \neq X$ et $T'_Z = X$: tenter une implication amont de la valeur T_Z sur les entrées
  - . si $T_Z = X$ et $T'_Z \neq X$: tenter une implication aval de la valeur T'_Z sur les opérateurs successeurs de Z.

Ce procédé nous permet, en particulier, de prendre en compte les implications amont indirectes |7| qui sont caractérisées par l'influence de la variation d'une entrée sur les autres :



Si A passe à la valeur 1, l'algorithme déduit que B doit prendre la valeur 0.

b) Implications dans une famille de vecteurs d'état

Le traitement des implications entre les vecteurs d'état d'une même famille obéit à la règle :
Tout positionnement d'une pseudo-entrée (pseudo-sortie) amène l'étude de ses répercussions sur le vecteur d'état précédent (suivant).

Par ce processus, on balayera éventuellement dans une même phase d'implications tous les vecteurs d'état d'une famille, parfois même plusieurs fois.

Les implications sont effectuées indépendamment dans chaque famille de vecteurs d'état. Cependant, lorsque dans un vecteur d'état d'une famille une entrée primaire est positionnée, sa valeur est transmise au vecteur d'état correspondant de l'autre famille.

2.3 - Propagation

Les opérations de propagation doivent nécessairement considérer simultanément le circuit juste et le circuit faux, aussi considèreront-elles toujours un couple de vecteurs d'état. Les constituants de ce couple devront être cohérents.

Soit un couple de vecteurs d'état cohérents.

Nous dirons qu'une connexion est sensibilisée si les valeurs des composantes qui lui correspondent dans chacun des deux vecteurs d'état sont opposées.

Nous dirons qu'une connexion est propageable si :
- *au moins une des entrées de l'opérateur dont elle est issue est sensibilisée,*
- *sa valeur dans au moins un des deux vecteurs d'état du couple est X.*

Nous dirons avoir un choix de propagation si :
- *plusieurs connexions sont propageables*

ou si
- *une connexion propageable peut être sensibilisée de deux manières (ce cas peut se rencontrer sur les sorties des éléments standards, séquentiels ou des opérateurs OU-EXCLUSIF ou EGAL).*

Une opération de propagation sur une connexion sera la sensibilisation de celle-ci par assignation de sa valeur dans au moins un des deux vecteurs d'état du couple.

Les connexions issues de blocs panne sont envisagées au même titre que les autres connexions propageables. En début de synthèse et lorsqu'aucun état initial n'est donné, ce sont les seules connexions propageables. Dans tous les autres cas elles ne seront qu'une possibilité que l'algorithme utilisera éventuellement (si les implications ne les ont pas positionnées).

L'ensemble des connexions propageables relatif à un choix de propagation est ordonné selon un critère de "distance" entre chacune de celles-ci et une sortie primaire (sens croissant).
Le mécanisme de gestion des choix permet d'envisager chacune des possibilités selon cet ordre. La décision de revenir considérer la possibilité suivante est cependant prise extérieurement aux opérations de propagation.

L'algorithme correspondant à une opération de propagation est le suivant :
- *recherche des connexions propageables*
- *si aucune connexion n'est propageable, le contrôle est passé à l'algorithme de prolongation aval de la sous-séquence*
- *s'il y a choix de propagation :*
    - ordonnancement des connexions propageables
    - élimination des propagations déjà tentées

. enregistrement éventuel d'un choix
. propagation

- *s'il n'y a pas choix de propagation :*
. propagation

Dans ces deux derniers cas, le contrôle est ensuite passé aux implications.

Les opérations de propagation ne sont effectuées que sur le dernier couple de vecteurs d'état : celui-ci a été pris en compte lorsqu'il n'était plus possible d'effectuer une propagation sur le précédent. Il est donc raisonnable d'estimer peu probable la sensibilisation d'une sortie primaire dans les couples de vecteurs d'état autres que le dernier.

Nous remarquons qu'à aucun moment on ne prédétermine le chemin sensible : les implications rendent un certain nombre de connexions propageables et chaque pas de propagation considère virtuellement l'ensemble de celles-ci avant de prendre une décision.

2.4 - Prolongation aval de la sous-séquence

Lorsqu'aucune propagation n'est possible, deux solutions se présentent :
- *adjoindre un vecteur d'entrée suivant, si cela est possible, sinon reconsidérer les choix de propagation*
- *reconsidérer d'abord les choix de propagation*

Cette deuxième solution est préférable si l'on vise à obtenir des séquences de test de longueur minimale. Elle demande cependant de supporter le coût correspondant au balayage des choix, lequel coût peut être prohibitif si le circuit comporte desparties combinatoires importantes.
Il semble que, sauf exception, il faille se contenter d'une séquence de test de longueur "raisonnable" si l'on désire que son coût d'élaboration reste réaliste. Notons pour mémoire que les moyens de test industriel des circuits intégrés autorisent des fréquences de quelques mégahertz et une durée de test unitaire de l'ordre de la seconde.

Nous avons donc choisi la première solution en considérant que souvent il n'est pas possible de faire passer l'effet d'une panne à travers un élément séquentiel sans avoir plusieurs vecteurs d'entrée.

Ce choix est d'autant plus raisonnable que la politique de choix de la prochaine panne que nous avons définie diminue grandement la longueur des sous-séquences. L'algorithme contrôlera cependant que, d'un vecteur d'entrée à l'autre, l'effet de la panne mémorisé sur les variables internes se "rapproche" d'une sortie primaire du circuit.

Si ce n'est pas le cas, il refusera de prolonger la sous-séquence de test et rendra le contrôle au mécanisme de gestion des choix.

2.5 - Consistance

La consistance a pour but de positionner, dans chaque vecteur d'état, les entrées et pseudo-entrées-primaires nécessaires à assurer les valeurs demandées sur les autres composantes. La consistance considèrera indépendamment les deux familles de vecteurs d'état et chacune de ses opérations s'effectuera sur un vecteur d'état cohérent.

Nous dirons qu'une connexion est non justifiée si sa valeur n'est ni indéterminée (X) ni assurée par les valeurs des entrées de l'opérateur dont elle est issue.

Considérant une connexion non justifiée, une opération de consistance sera donc l'assignation d'au moins une des entrées de l'opérateur dont elle est issue afin de la justifier.

Il y a nécessairement au moins deux possibilités pour effectuer une opération de consistance : si ce n'était pas le cas, la justification aurait été déduite par les implications.

Comme pour la propagation, les différentes possibilités d'un choix de consistance sont ordonnées (ordre croissant d'une "distance" de chaque entrée de l'opérateur en cause aux entrées primaires du circuit). Le mécanisme de gestion des choix permet de les envisager l'une après l'autre.

L'algorithme de consistance d'un vecteur d'état est :

- *recherche d'une connexion non justifiée*
- *si échec : passer à un autre vecteur d'état*
- *recherche des différentes possibilités de consistance et ordonnancement*
- *élimination des possibilités déjà tentées*

- *enregistrement éventuel d'un choix*
- *consistance de la connexion*
- *implications*
- *retour en début d'algorithme.*

REMARQUE : *Un certain nombre de justifications sont généralement obtenues grâce à la phase d'implication qui suit chaque opération de consistance.*

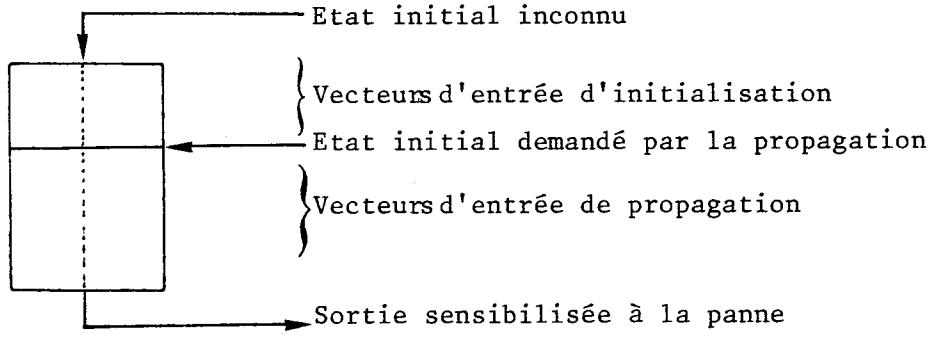
2.6 - Prolongation amont de la sous-séquence

Lorsque les opérations de consistance sur les vecteurs d'état correspondant à la propagation sont achevées, il se pose la question de l'état initial de la sous-séquence de test. S'il a été pris en compte par la synthèse en même temps que la panne, la génération est achevée. Sinon, il peut être nécessaire d'assurer des valeurs sur les pseudo-entrées d'un ou des constituants du premier couple de vecteurs d'état.

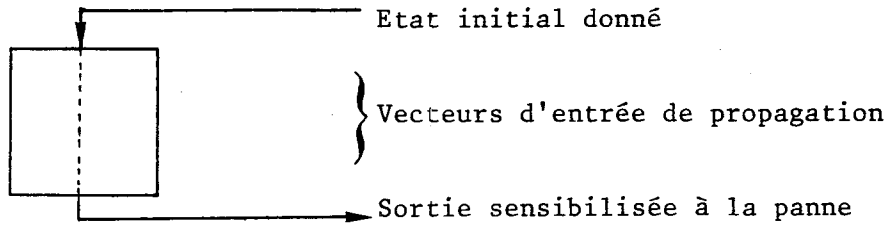
Cette initialisation est faite par adjonctions successives de vecteurs d'entrées précédents, jusqu'à ce que l'état initial demandé soit assuré à partir d'un état totalement inconnu.

Chaque adjonction est suivie du nombre de phases de consistance et d'implications nécessaires à ramener le maximum de conditions sur les entrées primaires.
Le processus s'arrête lorsque plus aucune valeur initiale n'est demandée.

Une sous-séquence, résultat final d'une synthèse, après les opérations de prolongation amont se présente de la manière suivante :



Lorsque l'état initial est donné en même temps que la panne, la sous-séquence a la forme :



Nous remarquerons que les vecteurs d'entrée d'initialisation tiennent compte de la panne.

2.7 - Critères heuristiques

Les critères heuristiques incorporés à l'algorithme de synthèse peuvent se ranger en deux catégories.

a) Critères ordonnant les différentes possibilités

Ce sont :

- *Le critère d'étude des différentes possibilités de manifester une panne sur une sortie primaire. Ce critère est déterminé par :*
  - . L'ordre d'étude des choix de propagation selon une "distance" aux sorties primaires de chaque connexion propageable.
  - . La prépondérance des opérations de prolongation aval de la sous-séquence sur la remise en cause des différents choix de propagation lorsque aucune propagation n'est plus possible.
- *Le critère d'étude des différentes possibilités d'assurer les conditions nécessaires à la propagation. Ce critère est déterminé par :*
  - . L'ordre d'étude des choix de consistance selon une "distance" aux entrées primaires des entrées des opérateurs dont la sortie n'est pas justifiée.
  - . La prépondérance des opérations de consistance sur celles de prolongation amont de la sous-séquence.
  - . L'ordre d'étude des connexions non justifiées d'un vecteur d'état selon une "distance" de celles-ci aux sorties primaires.
  - . L'ordre d'étude des vecteurs d'état selon l'ordre inverse de la sous-séquence.

. La prépondérance des opérations de consistance du circuit faux sur celles du circuit juste.

- *Le critère d'étude des vecteurs d'état en phase d'implication : prépondérance des vecteurs d'état précédents.*

Ces critères agissent sur l'ordre de déroulement des opérations et ont une influence prépondérante sur le temps de calcul.
Notons cependant que le problème du test d'une panne n'a généralement pas une solution unique. Aussi, une modification des critères précédents, bien que ne pouvant pas influer sur le type (succès ou échec) du résultat, peut modifier la sous-séquence obtenue.

b) Critères diminuant le nombre de possibilités

Ce sont :

- *Le critère d'autorisation de la prolongation aval de la sous-séquence; il demande que, sous l'action des vecteurs d'entrée, l'effet de la panne mémorisé sur les variables internes se "rapproche" d'une sortie primaire.*
- *Le critère d'autorisation de la prolongation amont de la sous-séquence; il demande que l'adjonction d'un vecteur d'entrée précédent ne nécessite pas d'assurer la présence d'une manifestation de la panne sur les variables internes du couple de vecteurs d'état correspondant.*

Ces critères correspondent à l'élimination de possibilités jugées "peu intéressantes". En conséquence, nous constatons que l'échec d'une synthèse ne signifiera pas nécessairement que la panne correspondante n'est pas testable.

Certains critères de la première catégorie peuvent avoir un effet semblable : la complexité de l'arbre des choix rend l'étude des possibilités défavorisées par les critères correspondants très coûteuse en temps.
Ces critères sont fondamentaux pour l'algorithme car, d'un point de vue pratique, il est parfois préférable de décider d'abandonner une synthèse avant d'avoir exploré tout l'arbre. On espèrealors que la panne correspondante pourra être détectée par une sous-séquence élaborée pour une autre panne.

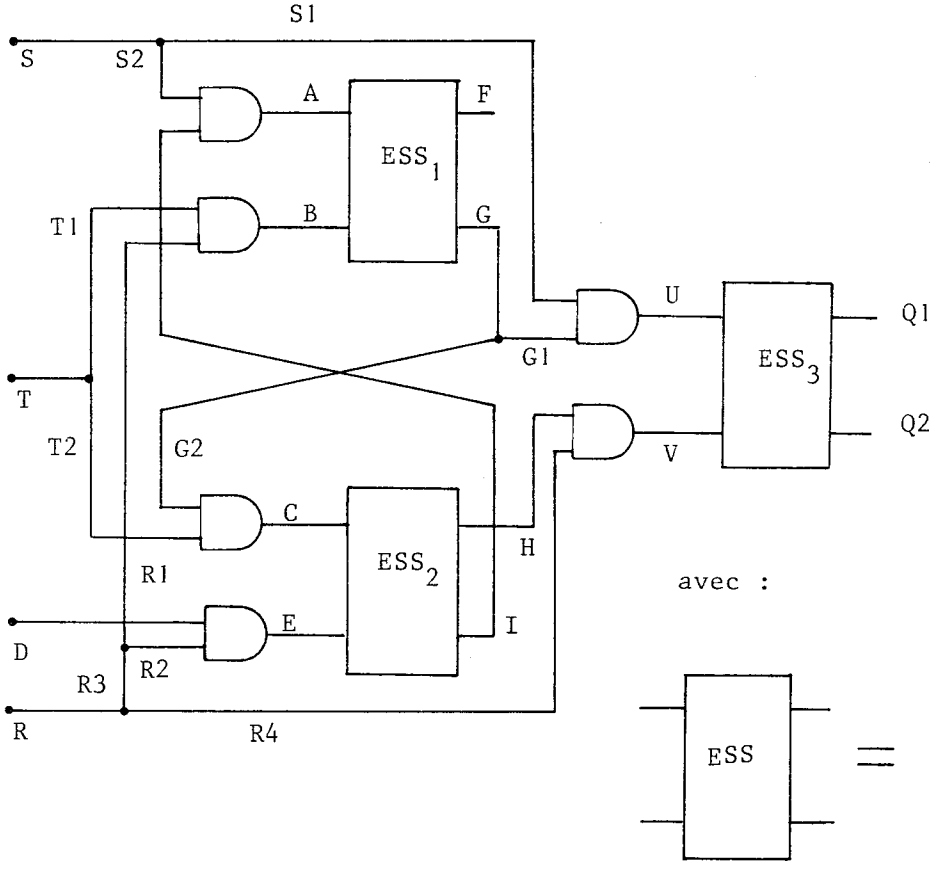
Nous avons donc introduit un autre critère heuristique appartenant à la deuxième catégorie et programmable par l'utilisateur : celui-ci peut décider du temps maximum alloué à chaque synthèse.

Lorsqu'une synthèse est abandonnée pour cause de dépassement de ce temps, le déroulement du programme continue avec l'étude d'autres pannes. Si les sous-séquences générées par celles-ci ne détectent pas la panne dont la synthèse a été abandonnée, il est alors possible d'affecter à celle-ci un temps plus long.

2.8 - Exemples

Nous présentons ici quatre exemples simples de déroulement de l'algorithme de synthèse.

Le circuit considéré est la bascule D vue précédemment :



a) Collage de B à zéro

Cet exemple met en évidence l'utilisation par la synthèse d'un état initial donné avec la panne à tester.
La variable interne portant une manifestation de la panne appartient à l'élément ESS_1.

Le déroulement de l'algorithme est le suivant :

SYNTHESE AVEC ETAT INITIAL

PROPAGATION : G (CHOIX)
IMPLICATIONS
PROPAGATION : U (CHOIX)
IMPLICATIONS
PROPAGATION : QI (CHOIX)
IMPLICATIONS
PANNE MANIFESTEE EN SORTIE

SYNTHESE ACHEVEE

On remarquera dans cet exemple, qu'aucune opération de consistance n'est nécessaire.

b) Collage de E à 1

Cet exemple met en évidence la prolongation aval d'une sous-séquence (ainsi que l'utilisation d'une manifestation de la panne sur une variable interne de l'élément ESS_2).

Le déroulement de l'algorithme est le suivant :

SYNTHESE AVEC ETAT INITIAL

PROPAGATION : H (CHOIX)
IMPLICATIONS
PROPAGATION : A (PAS DE CHOIX)
IMPLICATIONS
BLOCAGE DE PROPAGATION
PROLONGATION AVAL DE LA SOUS-SEQUENCE
IMPLICATIONS

PROPAGATION : H (CHOIX)
IMPLICATIONS
PROPAGATION : V (CHOIX)
IMPLICATIONS
PANNE MANIFESTEE EN SORTIE

SYNTHESE ACHEVEE

Ici encore, on remarquera qu'aucune opération de consistance n'est nécessaire.

c) Court-circuit OU entre G et R_2

Cet exemple met en évidence le traitement des courts-circuits et la prolongation amont d'une sous-séquence (aucun état initial n'est donc donné à la synthèse).

Le déroulement de l'algorithme est le suivant :

SYNTHESE SANS ETAT INITIAL

PROPAGATION : MANIFESTATION PANNE SUR G (CHOIX)
IMPLICATIONS
PROPAGATION : U (CHOIX)
IMPLICATIONS
PROPAGATION : Q_1 (CHOIX)
IMPLICATIONS
PANNE MANIFESTEE EN SORTIE

CONSISTANCE : Q_1 -CIRCUIT FAUX (CHOIX)
IMPLICATIONS
CONSISTANCE : H-CIRCUIT FAUX (CHOIX)
IMPLICATIONS
CONSISTANCE : G-CIRCUIT JUSTE (CHOIX)
IMPLICATIONS
CONSISTANCE VECTEURS DE PROPAGATION ACHEVEE
VALEURS INITIALES A ASSURER
PROLONGATION AMONT DE LA SOUS-SEQUENCE
IMPLICATIONS

CONSISTANCE : G-CIRCUIT JUSTE (CHOIX)
IMPLICATIONS
CONSISTANCE : A-CIRCUIT JUSTE (CHOIX)
IMPLICATIONS

SYNTHESE ACHEVEE

d) Collage de V à zéro

Cet exemple met en évidence le fonctionnement du système de gestion des choix.

Le déroulement de l'algorithme est le suivant :

SYNTHESE SANS ETAT INITIAL

PROPAGATION : MANIFESTATION PANNE SUR V
IMPLICATIONS
PROPAGATION : Q_1 (CHOIX)
IMPLICATIONS
PANNE MANIFESTEE EN SORTIE
CONSISTANCE : Q_1-CIRCUIT JUSTE (CHOIX)
IMPLICATIONS
CONSISTANCE : G-CIRCUIT JUSTE (CHOIX)
IMPLICATIONS
INCOHERENCE
RETOUR CHOIX : CONSISTANCE G-CIRCUIT JUSTE (CHOIX)
IMPLICATIONS
INCOHERENCE
RETOUR CHOIX : CONSISTANCE G-CIRCUIT JUSTE (PLUS DE CHOIX)
IMPLICATIONS
INCOHERENCE
RETOUR CHOIX : CONSISTANCE Q_1-CIRCUIT JUSTE (PLUS DE CHOIX)
IMPLICATIONS
CONSISTANCE : H-CIRCUIT JUSTE (CHOIX)
IMPLICATIONS
CONSISTANCE : G-CIRCUIT FAUX (CHOIX)
IMPLICATIONS
CONSISTANCE VECTEURS DE PROPAGATION ACHEVEE
VALEURS INITIALES A ASSURER

PROLONGATION AMONT DE LA SOUS-SEQUENCE

IMPLICATIONS

CONSISTANCE : Q_1-CIRCUIT JUSTE (CHOIX)

IMPLICATIONS

CONSISTANCE : U-CIRCUIT JUSTE (CHOIX)

IMPLICATIONS

CONSISTANCE : H-CIRCUIT JUSTE (CHOIX)

IMPLICATIONS

CONSISTANCE : C-CIRCUIT JUSTE (CHOIX)

IMPLICATIONS

SYSTHESE ACHEVEE

CHAPITRE VII

RÉSULTATS PRATIQUES

1. CARACTERISTIQUES DU PROGRAMME

La méthodologie précédente a permis de concevoir le programme TAU-1 de génération de séquences de test pour circuits intégrés à SESCOSEM.

Ce programme [43], actuellement opérationnel, est implanté sur un ordinateur IRIS 80. Il est écrit en FORTRAN et occupe 120 K-octets d'instructions en mémoire centrale.

Sa programmation est telle que sa taille globale est fonction de celle des circuits à traiter. Son implantation sur de plus petits ordinateurs pour des besoins restreints est donc possible.

A titre d'exemple, la taille mémoire supplémentaire demandée pour le test d'une bascule D est de 5 K-octets. Pour le microprocesseur considéré dans le paragraphe 4.4, elle est de 80 K-octets.

On trouvera dans l'Annexe I un résumé de la notice d'emploi du programme.

2. UTILISATION DU PROGRAMME

Le programme est capable de générer des séquences de test de manière autonome. Cependant, il est conçu afin de laisser à l'utilisateur la possibilité de donner une orientation fonctionnelle au déroulement des opérations et la possibilité d'intervenir au cours de celles-ci.

L'ensemble des pannes potentielles du circuit est décomposable en sous-ensembles dont les vecteurs de test peuvent être élaborés plus ou moins séparément.

Chaque sous-ensemble de pannes peut être traité à l'aide d'un ou plusieurs des types de génération suivants :

- *Enchaînement de synthèses et d'analyses.*
- *Enchaînement de générations pseudo-aléatoires de vecteurs d'entrée et d'analyses.*
- *Analyse de vecteurs d'entrée donnés.*

Parallèlement est introduite la possibilité d'une certaine interactivité entre le programme et l'utilisateur : celui-ci peut générer la séquence de test en un seul passage-machine ou, au contraire, en plusieurs et en choisissant à chaque étape le sous-ensemble de pannes et le type de génération les plus appropriés à la bonne continuation des opérations. Il a pour cela accès entre chaque étape à l'ensemble des résultats déjà obtenus.

2.1 - Enchaînement de synthèses et d'analyses

Une des contraintes des programmes de synthèse est de n'offrir aux utilisateurs que des commandes dont l'action sur l'algorithme leur est accessible.

Aussi, les phases de synthèse ne sont contrôlables que par le paramètre de temps maximum vu précédemment et, éventuellement, par un paramètre maximisant le nombre de vecteurs d'entrée de chaque sous-séquence.

L'arrêt d'un enchainement de synthèses et d'analyses se fera lorsque toutes les pannes du sous-ensemble correspondant auront été étudiées. (Nous rappelons qu'il sera possible ultérieurement de reconsidérer les pannes dont le test n'est pas assuré à l'issu de cette phase).

2.2 - Enchaînement de générations pseudo - aléatoires et d'analyses

L'utilisation d'une génération pseudo-aléatoire peut permettre de débuter économiquement l'élaboration d'une séquence de test.
Le contrôle de cet enchaînement est donné par :

- *La longueur de la sous-séquence à générer aléatoirement (avec éventuellement certaines valeurs prédéterminées).*
- *Le nombre maximum de répétition de la génération de cette sous-séquence.*
- *Le pourcentage de détection de pannes recherché par cette méthode.*

L'obtention des valeurs pseudo-aléatoires est assurée par un classique registre à décalage rebouclé.

2.3 - Analyse de vecteurs d'entrée donnés

L'introduction dans le programme de cette commande répond à plusieurs besoins :

- *Validation des sous-séquences utilisateurs.*
  Les concepteurs de circuits connaissent généralement suffisamment leurs produits pour pouvoir élaborer eux-mêmes des vecteurs de test de certaines pannes.
  Il suffit alors de pouvoir valider ces vecteurs par une phase d'analyse puis de complèter la séquence de test par un enchaînement de synthèse-analyse.
  Cette méthode est particulièrement efficace lorsque l'aspect fonctionnel de certaines zones d'un circuit, ou certaines expériences préalables, permettent la déduction rapide de vecteurs de test.

- *Positionnement initial du circuit.*
  L'état initial d'un circuit lors de sa mise sous tension est généralement indéterminé. Il est parfois préférable de spécifier au programme des vecteurs d'entrée de positionnement plutôt que de les lui laisser élaborer.
  L'analyse initiale d'une sous-séquence utilisateur élaborée dans ce sens permettra donc cette initialisation tout en déduisant les pannes détectées par cette sous-séquence.

- *Fusion des résultats de test partiels.*
  L'élaboration de vecteurs de test pour des sous-ensembles de pannes distincts nécessite la fusion des résultats partiels : si nous considérons un sous-ensemble de vecteurs d'entrée V_1 élaboré pour le test d'un sous-ensemble de pannes P_1, il faut pouvoir déterminer quelles pannes d'un sous-ensemble P_2 peuvent être détectées par les vecteurs d'entrée de V_1.

L'analyse des vecteurs d'entrée déjà générés lorsque l'on débutera le test d'un nouveau sous-ensemble de pannes permettra donc de ne pas générer inutilement des vecteurs d'entrée.

L'utilisation de toutes les possibilités précédentes n'est motivée que par le test des gros circuits. Nous en verrons un exemple avec un microprocesseur.
Pour les circuits de complexité moyenne, on pourra se contenter, en général, d'un enchaînement de synthèses et d'analyses (éventuellement précédé de l'analyse d'une sous-séquence donnée de positionnement du circuit) effectué en un seul passage-machine et considérant simultanément toutes les pannes potentielles du circuit.

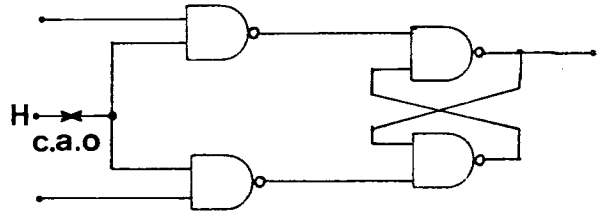
3. PANNES NON DETECTABLES

Pour les circuits combinatoires, la non détectabilité d'une panne est liée à la redondance des circuits.
Celle-ci se traduit lors de la génération d'un vecteur de test par la disparition complète de toute manifestation de la panne.
L'état du circuit juste et du circuit faux reste cependant connu.

Pour les circuits séquentiels, les causes de non détectabilité sont plus nombreuses : outre la redondance précédente, il faut considérer tout ce qui rend le comportement du circuit imprévisible en présence d'une panne.

Les causes d'imprévisibilité sont de deux sortes :

- *Causes d'initialisation :* Lorsqu'un point mémoire ne peut être positionné que d'une seule manière et que celle-ci est affectée par une panne :



Le collage à zéro de l'horloge H rend le point-mémoire inaccessible. Ne pouvant le positionner par ailleurs, nous ne pourrons donc pas observer l'effet de la panne à travers lui.

- *Causes de non définition de certains fonctionnements :* Comme nous l'avons vu au chapitre II, le comportement d'un point-mémoire n'est généralement pas défini pour toutes les configurations de ses entrées et variables internes. En présence de certaines pannes, il peut donc y avoir apparition d'une configuration interdite qui invalide la sous-séquence de test en construction.

4. EXEMPLES

4.1 - Bascule D SFC 474

Le schéma de ce circuit a été vu précédemment.

Les pannes envisagées sont :

- *Tous les collages.*
- *Deux courts-circuits :*
  - Type OU entre G et R_2
  - Type ET entre F et I

Le programme fournit en 21 secondes une séquence de test de 13 vecteurs d'entrée détectant toutes les pannes envisagées, sauf le collage à 1 de la connexion R_2 et le court-circuit entre F et I dont le test conduirait à un fonctionnement du circuit interdit par les hypothèses du programme.

4.2 - Mémoire 16 bits

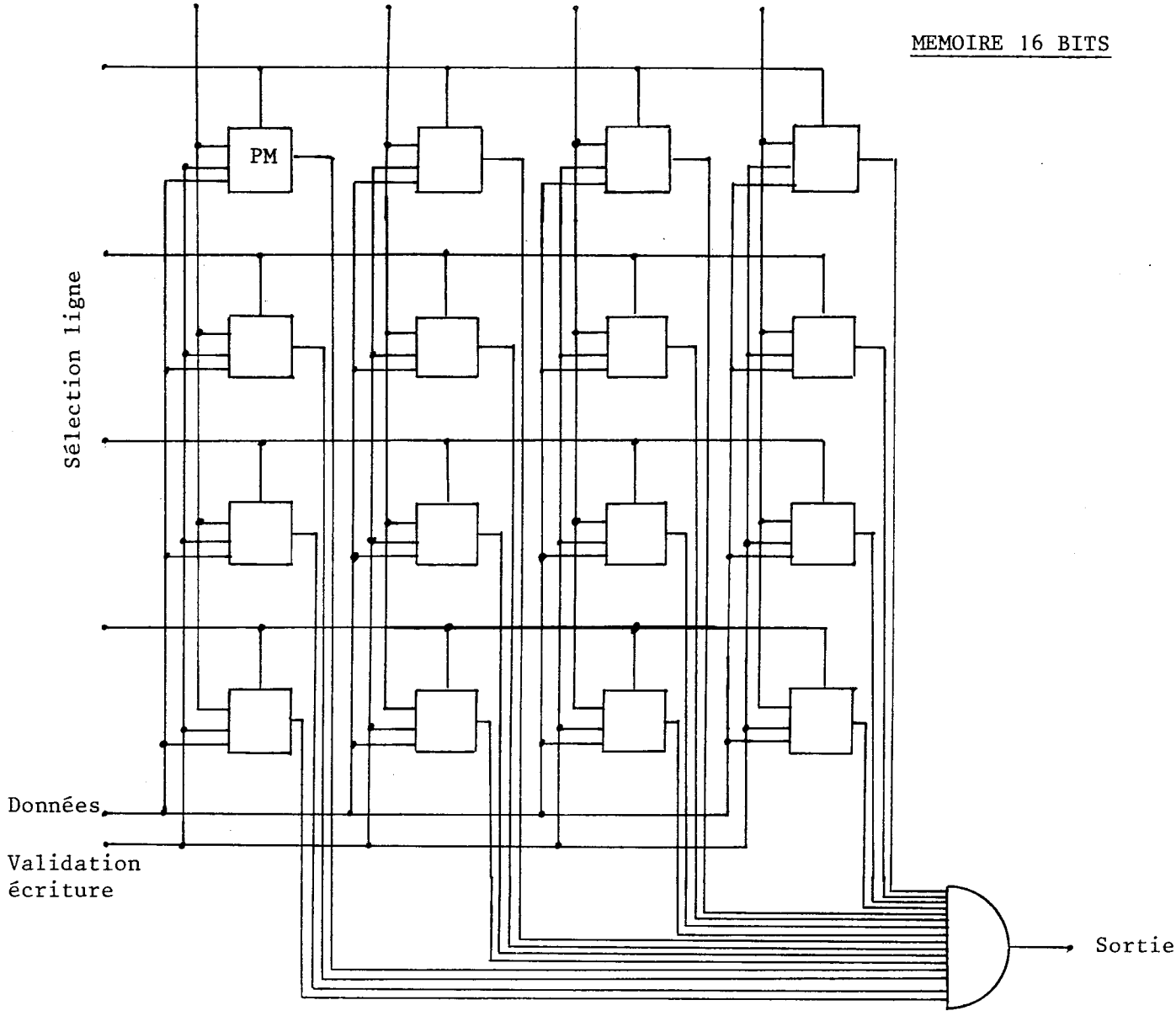
Ce circuit comporte 316 connexions. On se propose de ne tester que les collages de ces connexions. Il ne s'agit donc pas d'un test complet de mémoire comme on l'entend généralement.

Une génération automatique de vecteurs de test fournit en 8 mn une séquence de 179 vecteurs.

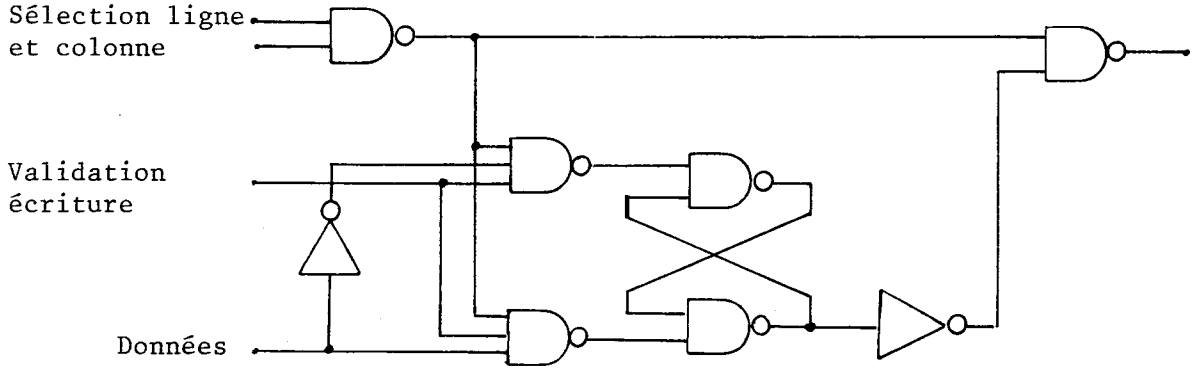
L'analyse préalable d'une sous-séquence utilisateur (écriture d'un champ de zéro) composée de 32 vecteurs d'entrée permet de ramener à 114 la dimension totale de la séquence et à 4 mn 30 son temps d'élaboration. (Il faut cependant tenir compte du fait que la structure répétitive de la mémoire rend très facile l'obtention manuelle des 32 premiers vecteurs de test).

Dans les deux cas, le pourcentage de pannes détectées est de 91%. Les 9% d'échec sont dûs à des causes d'initialisation et de fonctionnement interdit.

MEMOIRE 16 BITS



ORGANISATION DE LA MEMOIRE

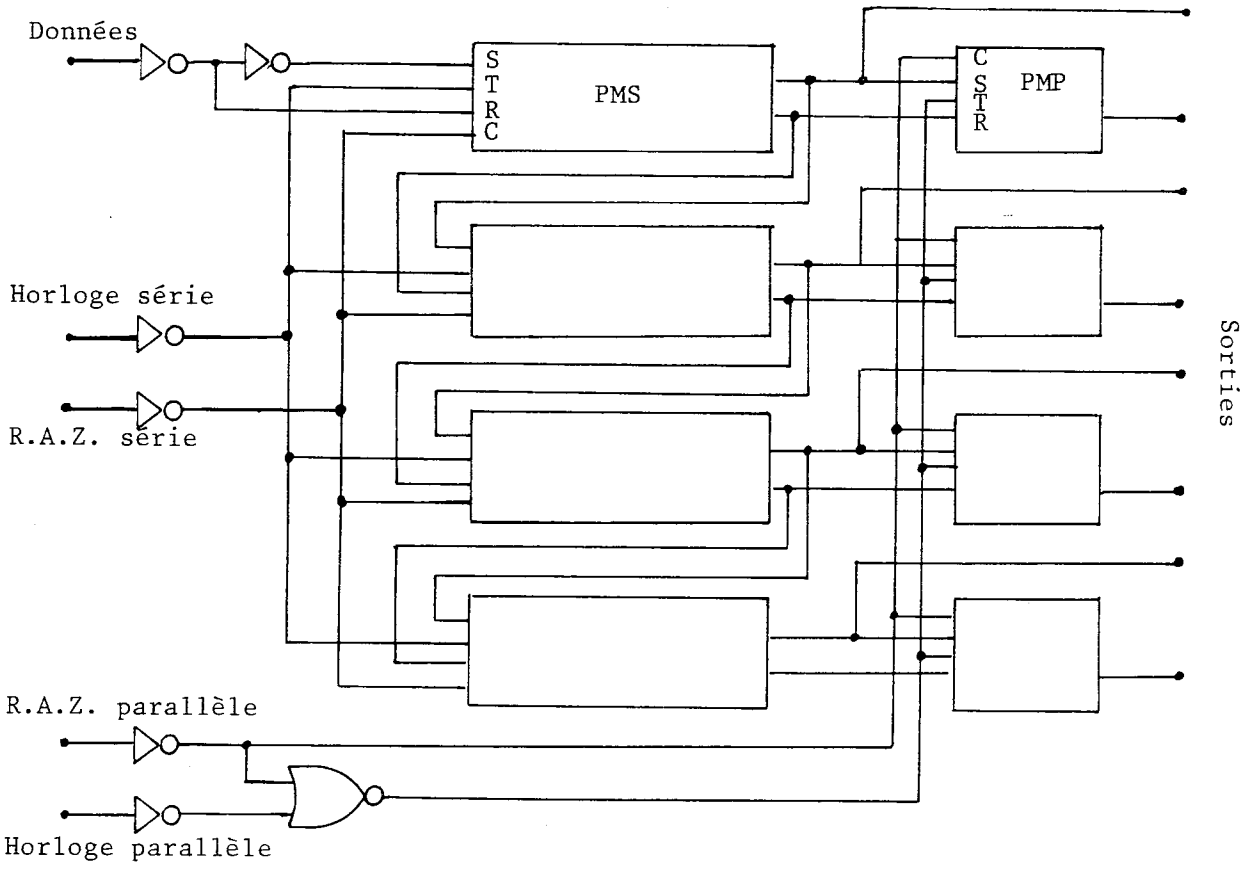


Point-Mémoire (PM)

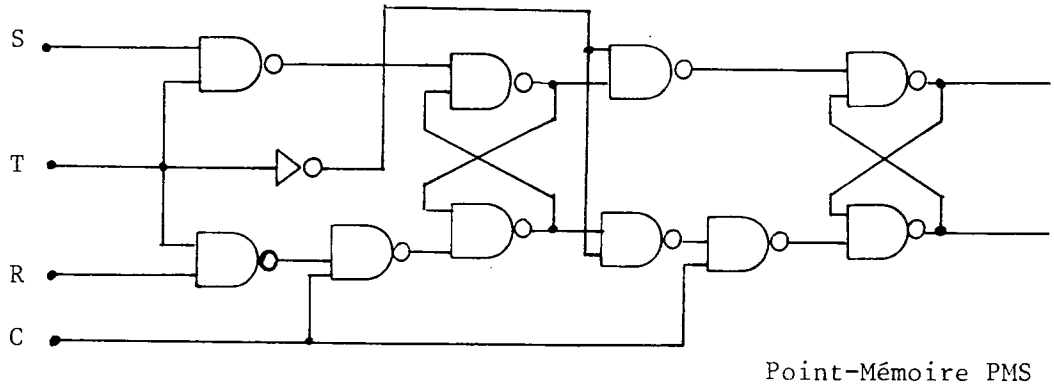
4.3 - Registre à décalage série-parallèle 4 étages

Ce circuit comporte 152 connexions dont on veut tester les collages (304 pannes potentielles).
La séquence de test obtenue comporte 86 vecteurs d'entrée et est obtenue en 5 mn 30.
Elle détecte 95% des pannes considérées. Notons qu'un vecteur d'entrée d'initialisation est donné au programme en début de génération.

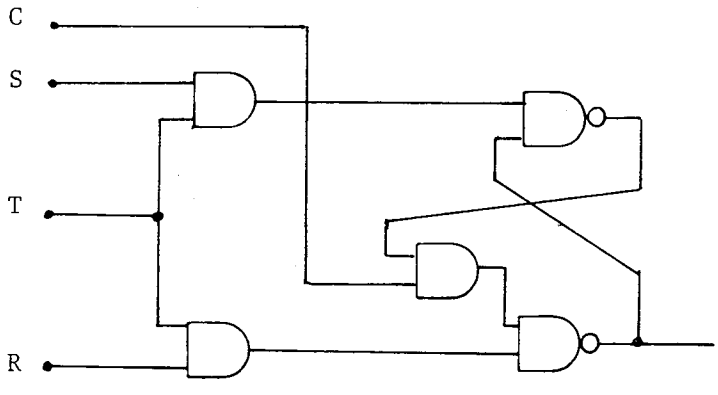
Les causes de non détectabilité sont les mêmes que précédemment (avec cependant certains échecs dûs à des redondances).



ORGANISATION DU REGISTRE



Point-Mémoire PMS



Point-Mémoire PMP

4.4 - Microprocesseur SFC 92901

Ce circuit comporte 528 connexions. Nous ne prenons pas en compte le test de la mémoire. Celle-ci est donc ramenée de 16 x 4 bits à 2 x 4 bits.

770 pannes potentielles (collages et courts-circuits) sont considérées.

La séquence de test obtenue comporte 78 vecteurs d'entrée qui détectent 92% de ces pannes.
La durée totale de la génération de la séquence est de 18 mn.

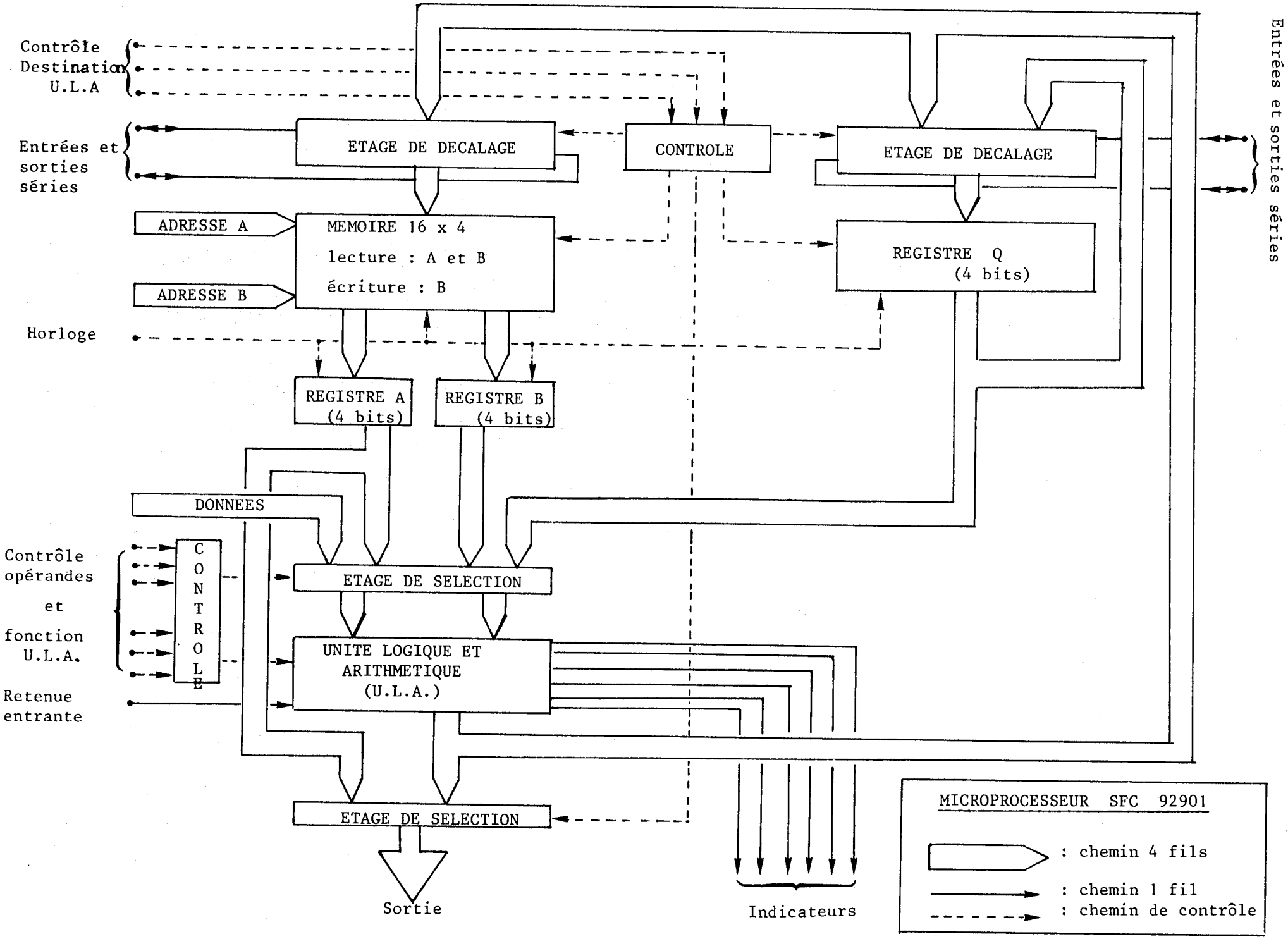
Les échecs sont dûs aux mêmes causes que précédemment et à l'incompétence du programme dans certains cas à générer une sous-séquence de test en un temps raisonnable.

Organisation du test :

Les pannes sont divisées en 5 sous-ensembles.
- *Pannes sur les entrées-sorties.*
- *Pannes sur les parties de contrôle.*
- *Pannes sur les registres et étages de décalage.*
- *Pannes sur les interconnexions.*
- *Pannes de l'unité logique et arithmétique (U.L.A) et des étages de sélection.*

Le déroulement du test est le suivant :
- *Pannes sur les entrées-sorties*
  - . Analyse d'une sous-séquence utilisateur de positionnement du circuit.
  - . Enchaînement de synthèses et analyses. On obtient ainsi en 108 secondes le test de 91% des pannes de ce bloc.

- *Pannes sur les parties de contrôle*
  - . Analyse des vecteurs de test précédemment élaborés (élimination des pannes du nouvel ensemble qui sont détectées par les vecteurs d'entrée précédents).
  - . Enchaînement de synthèses et analyses. On obtient ainsi en 46 secondes le test de 86% des pannes de ce bloc.


Contrôle Destination U.L.A
Entrées et sorties séries
ETAGE DE DECALAGE
CONTROLE
ETAGE DE DECALAGE
Entrées et sorties séries
ADRESSE A
ADRESSE B
MEMOIRE 16 x 4
lecture : A et B
écriture : B
REGISTRE Q
(4 bits)
Horloge
REGISTRE A
(4 bits)
REGISTRE B
(4 bits)
DONNEES
Contrôle opérandes et fonction U.L.A.
CONTROLE
ETAGE DE SELECTION
UNITE LOGIQUE ET ARITHMETIQUE
(U.L.A.)
Retenue entrante
ETAGE DE SELECTION
Sortie
Indicateurs
MICROPROCESSEUR SFC 92901
: chemin 4 fils
: chemin 1 fil
: chemin de contrôle

- *Pannes sur les registres et étages de décalage*
  - . Analyse des vecteurs de test déjà élaborés.
  - . Enchaînement de synthèses et analyses. On obtient ainsi en 154 secondes le test de 79% des pannes de ce bloc.

- *Pannes sur les interconnexions*
  - . Analyse des vecteurs de test précédemment élaborés.
  - . Enchaînement de synthèses et analyses. On obtient ainsi en 163 secondes le test de 87% des pannes de ce bloc.

- *Pannes sur l'unité logique et arithmétique et sur les étages de sélection*
  - . Analyse des vecteurs de test déjà élaborés.
  - . Analyse d'une sous-séquence utilisateur (propagation et inhibition de reports).
  - . Enchaînement de synthèses et analyses. On obtient ainsi en 325 secondes le test de 91% des pannes de ce bloc.

- *Fusion des résultats partiels*
  - . Pour les quatre premiers sous-ensembles de pannes, des vecteurs d'entrée ont été générés ultérieurement à leur étude. Il faut donc voir si ces vecteurs ne détectent pas certaines des pannes préalablement non détectées.

On procède donc à l'analyse de tous les vecteurs d'entrée sur la réunion de ces sous-ensembles de pannes.

On obtient alors en 190 secondes les pourcentages de détection suivants :

. Pannes d'entrée-sortie	95%
. Pannes sur les parties de contrôle	96%
. Pannes sur les registres et étages de décalage	88%
. Pannes de l'unité logique et arithmétique et des étages de sélection	91% (inchangé)
. Pannes sur les interconnexions	99%

La figure suivante reprend la stratégie précédente dans le langage de commande du programme (voir annexe I) Il n'y apparaît cependant pas l'aspect interactif du programme : il faut considérer que chacune des étapes a donné lieu à un ou plusieurs passages-machine avec étude des résultats intermédiaires et choix de la stratégie la plus adaptée.

```
$          MICROPROCESSEUR  2901 : COMMANDES DE TEST
$
$
$
$
** ENCHAINEMENT
C2901
NEW
$
$
$                    ENTREES -SORTIES
*DAD
1
*ANALYSE
NOREPRISE
$      INITIALISATION REGISTRES
0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0
0,0,0,0,1,0,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0
0,0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0
0,0,0,0,1,0,1,1,1,1,1,0,0,0,0,0,1,0,0,0,0
0,0,1,0,1,0,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0
1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0
*SYAN
REPRISE
TEMPS=7
LIM=6
$
$
$                    PARTIE CONTROLE
*DAD
2
*ANALYSE
BLOC=1
*SYAN
REPRISE
TEMPS=7
LIM=6
$
$
$                    REGISTRES ET ETAGES DE DECALAGE
*DAD
3
*ANALYSE
BLOC=2
*SYAN
REPRISE
TEMPS=7
LIM=6
$
$
$                    INTERCONNEXIONS
*DAD
4
*ANALYSE
BLOC=3
*SYAN
REPRISE
TEMPS=7
LIM=6
$
$
$                    U.L.A.  ET SELECTIONS
```

```
*DAD
5
*ANALYSE
BLOC=4
*ANALYSE
REPRISE
$      PROPAGATION ET INHIBITION DE REPORTS
1,1,0,1,1,0,0,0,1,1,1,1,1,1,1,0,0,0,0,0,0
1,1,0,1,1,0,0,0,1,1,1,1,1,1,1,1,0,0,0,0,1
0,0,0,1,1,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0
0,0,0,1,1,0,0,0,1,1,1,0,0,0,0,1,0,0,0,0,1
0,0,0,1,1,0,0,0,1,0,1,0,0,0,1,1,0,0,0,0,0
0,0,0,1,1,0,0,0,1,0,1,0,0,0,1,1,0,0,0,0,1
0,0,0,1,1,0,0,0,1,0,1,0,0,1,1,1,0,0,0,0,0
0,0,0,1,1,0,0,0,1,0,1,0,0,1,1,1,0,0,0,0,1
0,0,0,1,1,0,0,0,1,0,1,1,1,1,1,1,0,0,0,0,0
0,0,0,1,1,0,0,0,1,0,1,1,1,1,1,1,0,0,0,0,1
1,0,0,1,1,0,1,0,0,0,1,0,0,0,0,1,0,0,0,0,0
1,0,0,1,1,0,1,0,0,0,1,0,0,0,0,1,0,0,0,0,1
1,0,0,1,1,0,0,1,0,0,1,0,0,0,0,1,0,0,0,0,0
1,0,0,1,1,0,0,1,0,0,1,0,0,0,0,1,0,0,0,0,1
0,0,0,0,0,1,1,0,1,1,1,1,1,1,1,0,0,0,0,0,0
0,0,0,0,1,0,0,0,1,1,0,0,0,0,1,1,0,0,0,0,0
0,0,0,0,1,0,0,0,1,1,0,0,0,1,0,1,0,0,0,0,0
0,0,0,0,1,0,0,0,1,1,0,0,1,0,0,1,0,0,0,0,0
0,0,0,0,1,0,0,0,1,1,0,1,0,0,0,1,0,0,0,0,0
0,0,0,0,1,0,0,0,0,1,1,1,1,1,1,1,0,0,0,0,1
0,0,0,0,1,0,0,0,0,1,1,1,1,1,1,1,0,0,0,0,0
0,0,0,0,1,0,0,0,1,1,1,1,1,0,1,1,0,0,0,0,1
0,0,0,0,1,0,0,0,1,1,1,1,0,1,1,1,0,0,0,0,1
0,0,0,0,1,0,0,0,1,1,1,0,1,1,1,1,0,0,0,0,1
*SYAN
REPRISE
TEMPS=6
LIM=2
$
$
*DAD
1,2,3,4
*ANALYSE
BLOC=5
$
$
** FIN
```

CONCLUSION

Le problème posé a été étudié avec un constant souci d'adaptation de sa solution aux besoins industriels.

Afin de concevoir un produit réellement utilisable dans ce cadre, il s'est avéré nécessaire de concilier efficacité et facilité d'emploi.

L'efficacité a été obtenue d'une part, grâce à une caractérisation fine des circuits à tester et des pannes pouvant les affecter et d'autre part, grâce à une recherche systématique des paramètres modulant les temps de calcul.

Il ne faut pas oublier, en effet, que le problème pratique principal de tout algorithme heuristique de génération de tests (comme de tout algorithme de recherche combinatoire) est la diminution du coût correspondant à l'étude d'un grand nombre de possibilités différentes. Dans ce domaine, des recherches ultérieures concernant la définition d'outils de comparaison de critères heuristiques (recherches éventuellement basées sur des considérations statistiques) devraient permettre encore certaines améliorations.

Un programme de génération de séquences de test a été élaboré à partir de l'étude précédente.
Son utilisation, élémentaire pour les petits circuits, est associée, dans le cas de gros circuits, à un mode de traitement interactif qui n'intègre que des connaissances humaines relevant de l'aspect fonctionnel des circuits, aspect généralement bien connu des utilisateurs de programmes de test.

Par ailleurs, ce programme conçu dans le cadre des circuits intégrés reste utilisable sur des circuits non intégrés de taille et de complexité semblables.

Les limites pratiques du programme ne peuvent être données à priori : seule son expérimentation sur de nombreux exemples permettra de les connaître.
On peut cependant, dès à présent, distinguer deux types de limitations :
- *Les limitations dues à un nombre insuffisant de pannes détectées*
- *Les limitations dues à un coût prohibitif*

La première borne est atteinte lorsque les restrictions apportées au fonctionnement d'un circuit lors de son test ne permettent d'assurer la détection que de peu (ou pas) de pannes. Il faut remarquer que ce type d'echec se rencontrera plutôt pour des circuits relativement petits : pour des raisons de fiabilité de conception, il est très difficile d'élaborer un gros circuit dont le fonctionnement normal se distingue nettement du fonctionnement demandé par notre programme.

On peut espérer, compte tenu du temps de calcul relativement faible demandé par le test du microprocesseur SFC 92901, que la deuxième borne ne sera atteinte que par une nouvelle génération de circuits.

ANNEXE I

RESUME DE LA NOTICE D'EMPLOI DU PROGRAMME (*)

I - GENERALITES

II - DESCRIPTION DES CIRCUITS

III - GESTION DE LA BIBLIOTHEQUE

IV - VALIDATION DE LA DESCRIPTION

V - ENCHAINEMENT DES COMMANDES DE TEST

VI - EDITION DES RESULTATS

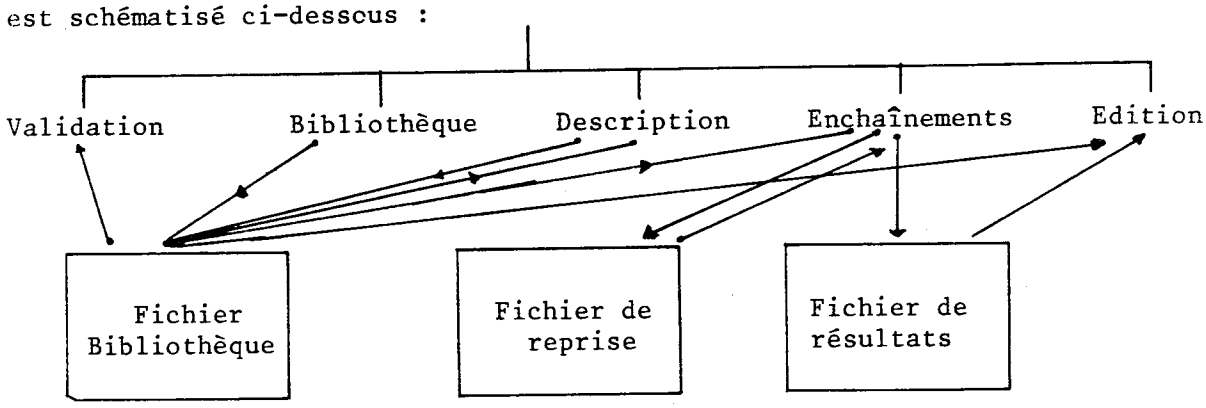
(*) Extrait de la notice d'emploi du programme, avec l'autorisation de SESCOSEM.

1. GENERALITES

1. Structure générale du programme

Le programme est organisé sous forme essentiellement modulaire. La communication entre les modules est réalisée par l'intermédiaire de fichiers. Cette conception du programme permet, soit une utilisation totalement automatique, soit une utilisation intéractive. Il est ainsi possible de construire une séquence de test en assemblant des parties connues (données par l'utilisateur) et des parties calculées.

Le principe de communication entre les modules principaux du programme est schématisé ci-dessous :



Les trois fichiers (Bibliothèque, Reprise, Résultats) sont des fichiers permanents, leur contenu est donc maintenu d'un passage-machine à un autre.

Description

Le module de description permet l'introduction des données relatives au circuit : topologie et défauts. Lorsqu'aucune erreur n'a été détectée lors de la description d'un circuit, celle-ci est stockée en bibliothèque.

Bibliothèque

Ce module regroupe un certain nombre de commandes permettant de connaître le contenu ou de restructurer une bibliothèque.

Validation

Le module de validation -d'emploi facultatif- permet de vérifier la description topologique d'un circuit par une simulation préalable sur une séquence donnée par l'utilisateur.

Enchaînements

Le module d'enchaînement comporte une commande destinée à la gestion de la table des défauts à détecter et trois commandes de test proprement dit.
Chacune des commandes de test comporte l'analyse (simulation) d'une suite de combinaisons d'entrées dont l'objet est de déterminer l'ensemble des défauts détectés. Cette suite de combinaisons est :

- donnée complètement par l'utilisateur (commande ANALYSE),
- donnée incomplètement par l'utilisateur, les entrées non spécifiées sont tirées aléatoirement (commande GEAL),
- synthétisée par le programme pour détecter un défaut particulier (commande SYAN).

Edition

Le module d'édition permet la récupération et la sélection des informations utiles à partir du fichier Résultats.

Dans sa version initiale, le programme comporte uniquement une sortie impression des résultats. Des interfaces spécialisés permettant la communication directe vers une machine de test seront réalisés ultérieurement.

2. ETATS LOGIQUES

L'état logique sur une connexion dans un circuit est 0, 1 ou X. L'état X correspond à une indétermination totale. Il peut provenir :

- de l'indétermination dûe à l'initialisation du circuit,
- du fonctionnement incorrect d'un élément séquentiel (transition interdite) : ici la valeur X est forcée sur les sorties de l'élément séquentiel conformément au modèle de celui-ci.

3. GESTION DES TABLES DE DEFAUTS

Pour simplifier leur manipulation en cours d'exécution, les défauts sont regroupés dans deux tables :

. La première (TDR) est la table de référence. Elle comporte tous les défauts qui pourront affecter le circuit. Elle est définie une fois pour toutes lors de la description du circuit.

. La seconde (TDD) est une partie de la précédente : elle comporte la sous-liste des défauts dont on souhaite la détection à un moment donné en cours d'enchaînements. Cette table TDD peut être modifiée en cours d'enchaînements à partir des éléments (ensembles de défauts) qui auront été préparés lors de la description.

Pour assurer la gestion de TDD, lors de la description des défauts d'un circuit, on définit un certain nombre d'ensembles de défauts (non nécessairement exclusifs). La réunion de tous les ensembles de défauts d'un même circuit forme automatiquement la table TDR. Quant à la table TDD, elle sera formée (puis éventuellement modifiée) en cours d'ENCHAINEMENTS en indiquant une liste d'ensembles parmi ceux décrits.

4. GESTION DES SEQUENCES DE TEST

Définitions

Une combinaison d'entrées est un ensemble de valeurs logiques appliquées simultanément aux entrées primaires du circuit.
Une séquence est une suite ordonnée de combinaisons d'entrées, pour laquelle l'état des sorties a été calculé en supposant l'état initial du circuit totalement indéterminé (état X sur toutes les connexions).

Une sous-séquence est une suite ordonnée de combinaisons d'entrées pour laquelle l'état des sorties a été calculé, en partant d'un état initial qui est le dernier état atteint par le circuit au cours du déroulement du programme. La sous-séquence n'a donc pas de sens par elle-même.

Vis-à-vis du module d'édition (donc du résultat final), le test est un ensemble de séquences (éventuellement réduit à une seule). L'état indéterminé étant supposé en tête des séquences, chacune d'entre elles pourra être appliquée isolément au circuit; l'ordre des séquences pourra être permuté. Par contre, aucune modification n'est possible à l'intérieur d'une séquence.

Chacune des trois commandes de test manipule une ou plusieurs sous-séquences. On dispose de deux ordres (REPRISE et NOREPRISE) pour les organiser en séquences.
D'autre part, il est possible de fixer la longueur maximumd'une séquence. Le programme se charge alors de gérer les indéterminations d'état initial.

5. REGLES SYNTAXIQUES GENERALES

- Les commandes du programme sont organisées en plusieurs niveaux :
  . Les commandes d'appel d'un module débutent par **
  . Les commandes de traitement internes aux modules débutent par *
- Les blancs ne sont pas significatifs, quel que soit le type de carte.
- Une liste d'éléments est une suite d'éléments séparés par des virgules.
- Les cartes commençant par un $ en colonne 1 sont uniquement reproduites (cartes commentaires).
- Un ";" comme dernier caractère non blanc d'une carte indique qu'il y a une carte suite.
- Tout programme commence par une carte de titre et s'achève par une carte ** FIN.
- Un identificateur comporte au plus huit caractères alphanumériques sauf indication contraire.

2. DESCRIPTION DES CIRCUITS

1. INTRODUCTION

Le module de description est activé par la commande
** DESCRIPTION

La description d'un circuit comporte deux parties :
. Description topologique : blocs logiques constituant le circuit et restrictions d'emploi.
. Description des défauts : permet de définir explicitement les défauts potentiels du circuit.

La description topologique d'un circuit est stockée en bibliothèque si
. aucune erreur n'a été détectée lors de l'analyse de cette description
. Aucun circuit de même nom n'est déjà présent dans la bibliothèque.

La description des défauts d'un circuit est facultative (en l'absence de cette commande, aucun défaut ne pourra être considéré sur le circuit). Elle n'est possible que pour un circuit dont la description topologique est déjà stockée en bibliothèque. Une nouvelle description de défauts pour un même circuit efface la précédente.

2. DESCRIPTION TOPOLOGIQUE

Activée par l'instruction :
* TOPOLOGIE

Les instructions suivantes sont :
A) NOMCIRCUIT (liste des entrées - liste des sorties)

NOMCIRCUIT : identificateur du circuit. Le circuit sera stocké sous ce nom en bibliothèque.

Liste des entrées : suite d'identificateurs de connexions, séparés par des virgules, précisant les entrées primaires du circuit.

Liste des sorties : suite d'identificateurs de connexions, séparés par des virgules, précisant les sorties primaires du circuit.

B) Une suite d'instructions décrit les modules constituant le circuit sous l'une des deux formes :

B.1 - TYPE (liste des entrées - liste des sorties)

ou

B.2 - NOM-TYPE (liste des entrées - liste des sorties)

- TYPE est le nom d'un module standard ou celui d'un bloc déjà stocké en bibliothèque.

- NOM est un identificateur (facultatif) individualisant le module dans le circuit (uniquement dans le cas des blocs bibliothèques). Si le nom est omis (forme B.1), un nom correspondant au numéro de module sera généré.

- Liste des entrées et liste des sorties : suite d'identificateurs de connexions séparés par des virgules. L'ordre relatif à l'intérieur d'une liste est sans signification pour les opérateurs combinatoires, mais est représentatif du rôle particulier des entrées ou des sorties pour les éléments séquentiels.
  Dans le cas des éléments séquentiels et des blocs bibliothèques, les nombres d'entrée-sortie sont fixés.
  Il est toutefois possible de ne pas utiliser une ou plusieurs sorties, à condition :
  . qu'une au moins des sorties du module existe
  . que l'emplacement de la (ou des) sortie(s) manquante(s) soit repéré par deux séparateurs consécutifs.
  Ex : BNAND (A,B-C); BNAND (A,B-,D)

Liste des modules standards :

Opérateurs combinatoires :

	Nombre d'entrées	Nombre de sorties
ET	≥ 1	1
ETNON	≥ 1	1
OU	≥ 1	1
OUNON	≥ 1	1
OUEX (disjonction)	2	1
EGAL (conjonction)	2	1
RAMIF (ramification)	1	≥ 1

Eléments séquentiels standards :

	Nombre d'entrées	Nombre de sorties
BNAND	2	2
BNOR	2	2
BINTER	3	1
BPNOR	3	2
BPINTER	5	1
BRSNOR	4	2
BMINT	2	1

B.3 - Il est possible d'utiliser une description itérative d'une suite de modules de même fonction (module standard ou bloc). Les instructions précédentes deviennent :

B.3.1 - TYPE/n,m/ (liste des entrées - liste des sorties).

B.3.2 - NOM/n,m/ - TYPE (liste des entrées - liste des sorties)

NOM est limité à 2 caractères. Ce sera l'identificateur du bloc itératif.

n et m sont les bornes de variation de l'indice d'itération.

n peut être omis, il sera alors pris égal à 1.

Les listes d'entrées et de sorties peuvent comporter :

. soit des identificateurs de connexions habituels (cas où la connexion entre dans tous les modules de l'itération)

. soit des identificateurs répétitifs, sous la forme

NOM ## I

ou NOM ## I + K

I symbolise l'indice de répétition

K est un nombre entier positif, à ajouter à l'indice.

Les identiticateurs itératifs sont générés et peuvent être utilisés dans une autre partie de la description.

B.4 - Une carte FIN termine la description des modules.

C) EMPLOI : cette instruction (facultative) indique le début de l'introduction des restrictions d'emploi.

Si l'instruction EMPLOI est absente, aucune restriction à l'emploi du circuit n'est introduite, autre que celles internes aux blocs appelés, qui sont automatiquement recopiées.

Si l'instruction EMPLOI est présente, elle est suivie d'une suite d'instructions donnant la condition logique correspondant à chacune des restrictions.

Ces conditions sont données sous forme d'une expression booléenne, en somme de produits éventuellement complémentée, dont les termes sont des identificateurs de connexions, à maintenir à la valeur 1 en permanence.

Symboles : + : ou logique
. : et logique
- : complémentation : ne porte que sur l'identificateur suivant ou sur l'ensemble de l'expression qui est alors précédée d'un point
$: à considérer en valeur initiale : ne porte que sur l'identificateur suivant et ne peut être appliqué qu'à une sortie d'élément séquentiel standard.

D) Une carte FIN termine la description topologique.

3. DESCRIPTION DES DEFAUTS

Activée par l'instruction
* DEFAUTS
suivie obligatoirement de

A) NOMCIRCUIT : nom du circuit pour lequel on décrit les défauts. La description topologique de NOMCIRCUIT doit obligatoirement être stockée en bibliothèque.

B) On retrouve ensuite une ou plusieurs fois une séquence d'instructions définissant des ensembles de défauts. S'il n'y a qu'un ensemble défini et si l'on effectue une commande d'ENCHAINEMENTS sur le circuit, la table TDD ne pourra qu'être vidée ou identique à TDR.

B.1 - ENSEMBLE : cette instruction indique la définition d'un nouvel ensemble de défauts. (Les différents ensembles seront ultérieurement répérés par leur numéro relativement à l'ordre dans lequel ils ont été définis).

B.2 - COLLAGES à 0

puis liste d'identificateurs de connexions.

B.3 - COLLAGES à 1

puis liste d'identificateurs de connexions

B.4 - Les instructions COLLAGES à 0 et COLLAGES à 1 peuvent être remplacées par une instruction COLLAGES suivie d'une liste. COLLAGES est équivalente aux deux instructions COLLAGES à 0 et COLLAGES à 1 relatives à cette liste.

B.5 - C/C ET

puis une liste de couples d'identificateurs de connexions sous la forme A1, A2 - A3, A4, définissant les courts-circuits de type ET (0 dominant).

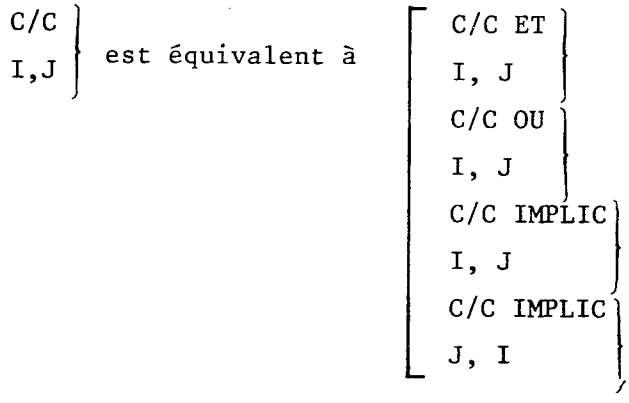
B.6 - C/C OU

puis une liste de couples d'identificateurs de connexions (même forme que ci-dessus) définissant les courts-circuits de type OU (1 dominant).

B.7 - C/C IMPLIC

puis une liste de couples d'identificateurs de connexions (même forme que ci-dessus) définissant les courts-circuits de type implication. Dans ce cas, l'ordre des connexions est significatif : la première connexion force sa valeur sur la seconde.

B.8 - Les instructions C/C ET, C/C OU et C/C IMPLIC peuvent être remplacées par une seule instruction C/C. Dans ce cas, pour chaque couple de connexions de la liste qui suit C/C, quatre défauts de court - circuit seront définis.



B.9 - BLOCS : cette instruction (facultative) permet de recopier, dans l'ensemble de défauts en cours de définition, certains des ensembles de défauts relatifs aux blocs qui ont été incorporés.

La carte BLOC est suivie d'une liste de noms de modules (donnés ou générés) indicés par le numéro de l'ensemble de défauts à considérer.

Ainsi, si la description topologique de NOMCIRCUIT a utilisé l'instruction NOM-TYPE (liste 1 - liste 2) signifiant l'insertion d'un module dont le nom bibliothèque est TYPE et identifié par NOM à l'intérieur de NOMCIRCUIT, NOM (1) signifie : pour le module NOM, incorporation du I^e ensemble de défauts de TYPE.

C) Une carte FIN termine la description des défauts.

3. GESTION DE LA BIBLIOTHEQUE

Le module de gestion de bibliothèque est activé par la commande :

** BIBLIOTHEQUE

Les commandes sont

1. * CREATION

   Initialise une bibliothèque vide (efface toute l'ancienne bibliothèque si celle-ci existait). Cette commande doit avoir été utilisée avant la première description.

2. * EDITE

   Permet l'impression de la description des circuits de la bibliothèque.

3. * REORGANISER

   Permet de retasser la bibliothèque.

4. * EFFACER

   Suivie d'une liste de noms de circuits.

   Permet de supprimer la description de certains circuits de la bibliothèque. Le nom des circuits effacés est alors disponible.

   Une carte * FIN termine la sous-commande de * EFFACER.

4. VALIDATION DE LA DESCRIPTION TOPOLOGIQUE

La validation de la description topologique est une simulation préalable, sans propagation de liste de défauts. Son emploi est facultatif. Le résultat de la validation est sans effet sur la suite du déroulement d'un programme.

Le module de validation est activé par :

** VALIDATION

suivie de

A) NOMCIRCUIT (obligatoire)

nom sous lequel le circuit à valider est stocké en bibliothèque.

B) Liste de valeurs

On donne ainsi la grille des entrées-sorties du circuit : la simulation sera faite en appliquant les entrées et une vérification des sorties calculées par rapport aux sorties attendues est faite, provoquant l'impression du résultat.

DESCRIPTION VALIDE

ou DESCRIPTION NON VALIDE

La grille d'entrées-sorties est donnée par une suite d'instructions (une par affichage).

Chaque instruction est une suite de valeurs logiques 0, 1 ou X séparées par des virgules, autant que le nombre total de pattes du circuit : en tête les entrées, puis les sorties, dans l'ordre de la description topologique (première instruction).

Une valeur X en entrée ou en sortie n'a pas la même signification :

- En entrée, X est traitée comme une véritable valeur logique.
- En sortie, X signifie que la valeur n'est pas à prendre en compte pour la validation .

5. ENCHAINEMENT DES COMMANDES DE TEST

1. INTRODUCTION

Le module d'enchaînement est activé par :

** ENCHAINEMENTS

suivi de

A) NOMCIRCUIT

puis

B) New

LONGUEUR = K (facultatif)

ou OLD

NOMCIRCUIT est le nom sous lequel le circuit étudié est stocké en bibliothèque. Les deux inscriptions (topologie et défauts) doivent être présentes.

La carte NEW initialise les fichiers de REPRISE et de RESULTATS : elle indique donc le début de l'étude du test d'un circuit.

Si LONGUEUR est spécifié, le test sera automatiquement découpé en séquences de K affichages.

La carte OLD indique que les fichiers de REPRISE et de RESULTATS existent déjà et sont à conserver. Il s'agit donc de la continuation d'une étude précédente. L'état de ces fichiers correspond à la dernière opération signalée par un message d'exécution au cours du passage précédent (voir ci-dessous).

2. CONSTRUCTION DE LA TABLE DES DEFAUTS A DETECTER

Est activée par la commande

* DAD

suivie d'une liste de numéros d'ensembles de défauts

La table des défauts à détecter (TDD) est alors constituée par la réunion de tous les défauts présents dans ces ensembles.

Cette table est sauvegardée en même temps que les fichiers REPRISE et RESULTATS. Elle est donc inchangée tant que l'instruction NEW ou une nouvelle commande * DAD n'est pas rencontrée.

Option : TOUS signifie que la table TDD contient tous les défauts (identique à TDR).

L'exécution de la commande *DAD provoque l'impression de la liste des défauts de la nouvelle table TDD en indiquant pour chacun d'eux s'il a déjà ou non été détecté.

3. ANALYSE D'UNE SOUS-SEQUENCE

Activée par la commande
* ANALYSE
suivie de

A) REPRISE
ou NOREPRISE
ou BLOC = K

REPRISE signifie que la sous-séquence à analyser doit être accolée à la fin de la séquence précédente : l'état initial du circuit est pris sur le fichier REPRISE.

NOREPRISE signifie le début d'une nouvelle séquence : l'état initial du circuit est totalement indéterminé.

B) Les options REPRISE et NOREPRISE doivent être suivies d'un ensemble d'instructions précisant, sous forme d'une liste de 0 et de 1, les états logiques appliqués aux entrées (une par affichage). Cette liste peut éventuellement être précédée d'un indicateur NT, qui signifie que la mesure des sorties ne sera pas effectuée pour cet affichage. Aucun défaut ne sera donc considéré comme détecté par un affichage comportant l'indicateur NT.

Un bloc est l'ensemble des affichages générés pour une même table de défauts à détecter. (Il peut contenir plusieurs séquences).
BLOC = K signifie que l'on désire connaître les défauts détectés sur une nouvelle table de défauts TDD par la ou les séquences générées précédemment pour une autre table. K est le numéro d'occurence de la sous-commande * DAD correspondante.

4. GENERATION ALEATOIRE

Activée par la commande

* GEAL

suivie de

A) REPRISE — Même signification
ou NOREPRISE — que ci-dessus

B) TAUX =...

Indique le pourcentage de couverture des défauts à atteindre, relativement à la table TDD actuelle. Le pourcentage correspond au cumul des défauts détectés depuis le début de l'étude (instruction NEW), il inclut donc les défauts éventuellement déjà détectés avant * GEAL.

C) LIM =...

Indique le nombre maximum de répétitions de la sous-séquence (provoque l'arrêt de GEAL même si le taux n'est pas atteint). (Instruction facultative).

D) Une suite d'instructions, donnant l'état logique à appliquer aux entrées sous forme d'une liste de valeurs 0, 1 ou A, définit la sous-séquence.

A signifie que la valeur (0 ou 1) doit être tirée aléatoirement.
Chaque affichage peut être précédé de NT comme précédemment.

5. SYNTHESE - ANALYSE

Activée par la commande

* SYAN

suivie de

A) REPRISE — Comme
ou NOREPRISE — précédemment

B) Temps =...

Indique le temps CPU maximum (en secondes) alloué à chaque synthèse (instruction facultative).

C) LIM = ...

Indique un nombre maximum de combinaisons d'entrées autorisé pour chacune des sous-séquences synthétisées. (Instruction facultative).

D) CONSTANT (instruction facultative)

Si l'instruction est présente, elle est suivie de l'une au moins des instructions

ETAT 0

puis liste des entrées primaires à maintenir constamment à 0

ETAT 1

puis liste des entrées primaires à maintenir constamment à 1.

E) RETESTER (instruction facultative)

suivi de : liste de numéros de défauts

Cet ordre permet de redemander la synthèse du test de certains défauts difficiles non détectés lors d'un premier passage. (On peut alors donner à la synthèse des paramètres limites plus grands).

EXEMPLE :

```
** ENCHAINEMENTS
   NOMCIRCUIT
   NEW
 * DAD
   1
 * SYAN
   NOREPRISE        Ce premier passage laisse par exemple les
   TEMPS = 5        défauts 15 et 20 non détectés.
** FIN

** ENCHAINEMENT
   NOMCIRCUIT
   OLD
 * SYAN
   REPRISE
   TEMPS = 10       On accorde dans ce deuxième passage plus de
   RETESTER         temps à la synthèse pour les deux défauts
   20, 15           difficiles.
** FIN
```

6. EDITION DES RESULTATS

1. INTRODUCTION

Le module d'édition est activé par

** EDITION

suivi de

NOMCIRCUIT

Les commandes décrites dans les paragraphes suivants concernent uniquement l'impression des divers résultats.

Vis-à-vis du module d'édition, les résultats sont organisés par séquences, qui sont des parties de la séquence de test pouvant être isolées. La manière dont une séquence a été obtenue (types de commandes utilisées, en un ou plusieurs passages dans ENCHAINEMENTS) est ignorée.

On rappelle qu'un bloc est l'ensemble des affichages générés pour une même table de défauts à détecter (TDD) : la définition d'un nouveau bloc correspond à la sous-commande * DAD. Le numéro d'un bloc est le numéro d'occurence de sa commande * DAD correspondante.

Dans toutes les sous-commandes suivantes, lorsqu'un numéro de bloc n'est pas spécifié, il s'agit du dernier généré.

2. COMMANDES

* GRILLE E/S

BLOC = K (facultatif)

. impression des connexions d'entrée sortie

. impression pour le bloc considéré des affichages avec la valeur des sorties.

* DIAGNOSTIC

BLOC = K (facultatif)

. impression pour le bloc considéré, pour chaque affichage et pour chaque sortie, des défauts détectés et de ceux provoquant une indétermination.

* LISTER

BLOC = K (facultatif)

. impression de la table des défauts à détecter du bloc considéré

. impression pour le bloc considéré, et pour chaque défaut, de l'historique du test :

- défauts détectés ou non
- défauts étudiés ou non
- défauts échoués en synthèse (échec total, échec dû au temps, échec dû au nombre de vecteurs).

* FUSIONNER

BLOC = K, L

ou

BLOC = K

Cette commande n'est utilisable que si deux blocs contiennent les mêmes affichages pour des ensembles de défauts différents. On a alors création d'un nouveau bloc par fusion pour chaque affichage et chaque sortie des listes de défauts correspondantes. Le résultat est donc le même que si le nouveau bloc avait été généré sur la réunion des deux ensembles de défauts.

ANNEXE II

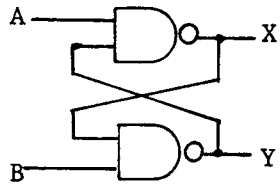
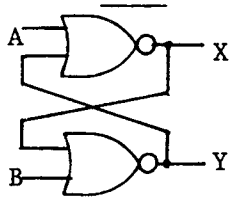
CONDITIONS DE BLOCAGE D'UN CYCLE PAR UN OPERATEUR

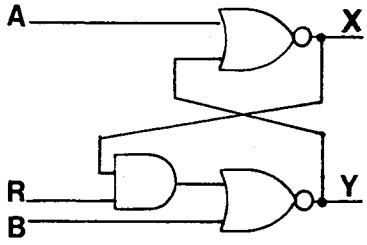
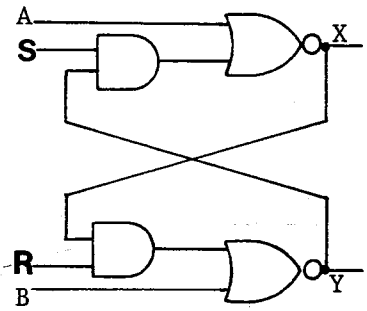
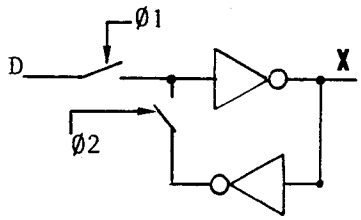
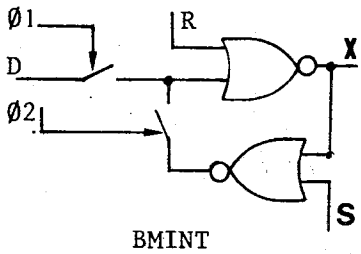
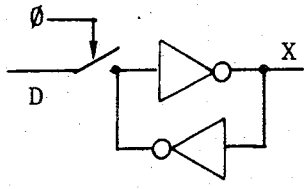
Pour les portes élémentaires, les conditions de blocage sont triviales : Nous considérons la condition de blocage entre l'entrée A et la sortie S d'une porte à n entrées A, B, C...

Porte	Condition de blocage
ET, ET NON	$\overline{B} + \overline{C} + \ldots = 1$
OU, OU NON	$B + C + \ldots = 1$

Il n'y a pas de blocage possible sur les opérateurs de disjonction et de conjonction ainsi que sur les opérateurs à une seule entrée.

Pour les opérateurs standards séquentiels, les conditions de blocage sont plus sophistiquées.
Nous noterons X^* la valeur initiale d'une variable interne.

Opérateur	Blocage entre l'entrée	et la sortie	Condition de blocage
BNAND 	A	X	$B\ X^* = 1$
	A	Y	$\overline{B} + \overline{Y^*} = 1$
	B	X	$\overline{A} + \overline{X^*} = 1$
	B	Y	$A\ Y^* = 1$
BNOR 	A	X	$\overline{B}.\overline{X^*} = 1$
	A	Y	$B + Y^* = 1$
	B	X	$A + X^* = 1$
	B	Y	$\overline{A}.\overline{Y^*} = 1$

Opérateur	Blocage entre l'entrée	et la sortie	Condition de blocage
BPNOR	A	X	$\overline{B}.(\overline{R}+\overline{X^*}) = 1$
	A	Y	$B + Y^* + \overline{R} = 1$
	B	X	$A + R.X^* = 1$
	B	Y	$\overline{A}.R.\overline{Y^*} = 1$
	R	X	$A + B + \overline{X^*} = 1$
	R	Y	$A + B + Y^* = 1$
BRSNOR	A	X	$S.\overline{B}.(\overline{R} + \overline{X^*} = 1$
	A	Y	$B + S.Y^* + \overline{R} = 1$
	B	X	$A + R.X^* + \overline{S} = 1$
	B	Y	$R.\overline{A}.(\overline{S} + \overline{Y^*}) = 1$
	R	X	$A + B + \overline{S} + \overline{X^*} = 1$
	R	Y	$A + B + S.Y^* = 1$
	S	X	$A + B + R.X^* = 1$
	S	Y	$A + B + \overline{R} + \overline{Y^*} = 1$
BINTER	D	X	$Ø2.\overline{Ø1} = 1$
	Ø1	X	$D.\overline{X^*} + \overline{D}\ X^* = 1$
	Ø2	X	$D.\overline{X^*} + \overline{D}\ X^* = 1$
BPINTER	D	X	$R + Ø2.\overline{Ø1} = 1$
	Ø1	X	$R + S.\overline{D} + \overline{D}\ \overline{X^*} + \overline{D}\ X^* = 1$
	Ø2	X	$R + S.\overline{D} + \overline{D}\ X^* + \overline{D}\ \overline{X^*} = 1$
	R	X	$Ø1.\overline{Ø2}\ D + Ø1\ Ø2(S + X^*)=1$
	S	X	$R + Ø1\ Ø2\ X^* = 1$
BMINT	D	X	$\overline{Ø} = 1$
	Ø	×	$D\ \overline{X^*} = \overline{D}\ X^* = 1$

BIBLIOGRAPHIE

1) AKERS S.B. Jr : "A Logic System for Fault Test Generation", International Symposium on Fault Tolerant Computing, pp.37-42, may 1975 et IEEE Trans. Comp. Vol C25 n° 6, pp.613-620, june 1976.

2) ARIMA T. et all : "A New Heuristic Test Generation Algorithm for Sequential Circuits", 11th Design Automation Workshop, pp.169-176, (1974).

3) BENNETS G., LEWIN D.W. : "Fault Diagnosis of Digital Systems - A Review", COMPUTER, pp. 12-20, Jul/Aug 1971.

4) BOUTE R., MC CLUSKEY E.J. : "Fault Equivalence in Sequential Machines", COMPUTER and Automata pp.483-507, Vol. C21, New York, April 1971.

5) BOURICIUS W.G. et all : "Algorithms for Detection of Faults in Logic Circuits", IEEE Trans. Comp. Vol. C20 n° 11, pp.1258-1264, Nov 1971 et IBM Thomas J. Watson Research Center, Oct. 1970.

6) BREUER M.A : "The Effect of Races, Delays and Delay - Faults on Test Generation", IEEE Trans. Comp. Vol. C23 n° 10, pp.1078-1092, Oct. 1974.

7) BREUER M.A. : "Generation of Fault Detection Tests for Sequential Circuits", International Symposium on Fault-Tolerant Computing, pp. 18-21, March 1971.

8) BREUER M.A. et all : "Identification of Multiple Stuck Type Faults in Combinational Networks", IEEE Trans. Comp. Vol. C25, n° 1, pp. 44-54, Jan. 1976.

9) BREUER M.A. : "Testing for Intermittent Faults in Digital Circuits", IEEE Trans. Comp., Vol C22 n° 3, pp. 241-246, March 1973.

10) CHANG H.Y. et all : "LAMP System Description", B.S.T.J. Vol. 53 n° 8, pp. 1431-1449, Oct. 1974.

11) CHANG H.Y. : "Comparison of Parallel and Deductive Fault Simulation Methode", IEEE Trans. Comp., Vol. C23 n° 11, pp.1132-1138, Nov.1974.

12) CHAPELL S.G. : "Automatic Test Generation for Asynchronous Digital Circuits", B.S.T.J., pp. 1417-1503, Oct. 1974.

13) FRIEDMAN A.D. : "Diagnosis of Short-Circuit Faults in Combinational Circuits", IEEE Trans. Comp. Vol. C23 n° 7, July 1974.

14) FRIEDMAN A.D., MENON P.R. : "Fault Detection in Digital Circuits", Prentice Hall 1971.

15) FUNATSU S. et all : "Test Generation in Japan", 12th Design Automation Workshop, pp. 114-122 (1975).

16) KAMAL S. : "An Approach to the Diagnosis of Intermittent Faults", IEEE Trans. Comp. Vol. C24 n° 5, May 1976.

17) KARIBSKII V.V. : "Construction of an Input Sequence that Detects a Given Failure of aDiscrete Device", Automation and Remote Control Vol. 33 n° 5, pp. 843-851, May 1972.

18) KUBO H. : "A Procedure for Generating Test Sequences to Detect Sequential Failures", NEC Research and Development, n° 12, pp. 69-78, Oct. 1968.

19) KREUWELS WG.J. :"Structural Testing of Digital Circuits", Philips Technical Review, Vol. 35 n° 10, pp. 261-270 (1975).

20) PREISS R.J. : "Fault Test Generation", Chapter 7 Design Automation of Digital Systems Prentice Hall, M.A. BREUER Ed. (1972).

21) MANGE D. : "Modèles Asynchrones des Bascules Bistables", Cahiers de la C.S.L, Ecole Polytechnique de Lausanne n° 5, pp. 256-286, Oct. 1973.

22) MEY K.C.Y. : "Bridging and Stuck at Faults", International Symposium on Fault Tolerant Computing, pp. 91-94, (1973).

23) MUTH P. : "A Nine Valued Circuit Model to Generate tests for Sequential Circuits", International Symposium on Fault Tolerant Computing, pp. 43-49, (1975) et "A Nine Valued Circuit Model for Test Generation", IEEE Trans. Comp. Vol. C25 n° 6, pp. 630-636, june 1976.

24) PIGNAL P. et all : "Une Méthode de Test Automatique pour les Ensembles Logiques", l'Onde Electrique, Vol. 48 n° 500, pp. 997-1003, Nov. 1968 et n° 501, pp. 1081-1088, Dec. 1968.

25) PLITMAN A.D. : "Design of a Sequential Test for a Multiple Fault in Asynchronous Circuits", Automation and Remote Control, pp. 1123-1131, July 1974.

26) PUTZOLU G.R., ROTH J.P. : "A Heuristic Algorithm for the Testing of Asynchronous Circuits", IEEE Trans. Comp. Vol. C20, June 1971.

27) RAULT J.C. : "La Détection et la Localisation des Défauts dans les Circuits Logiques", International Symposium on Fault Tolerant Computing, pp. 17-23 (1975).

28) RAULT J.C. : "Bibliographie sur la Détection et la Localisation des Défauts dans les Circuits Logiques", Rapport Thomson CSF, 33, rue de Vouillé 75015 PARIS.

29) ROBISON J.M. : "Applications of Logic Simulation in Design Automation at Texas Instrument", 9th Design Automation Workshop, pp. 138-143 (1972).

30) ROTH J.P. : "Programmed Algorithm to Compute Test to Detect and Distinguish Between Failures in Logic Circuits", IEEE Trans. Comp. Vol. EC 16 n° 5, pp. 567-580, Oct. 1967.

31) SUSSKIND A.M. : "Diagnosis for Logic Networks", IEEE Spectrum, pp.40-47 Oct. 1973.

32) SZYGENDA S.A. : "Tegas 2 : Anatomy of a General Purpose Test Generation and Simulation System for Digital Logic", 9th Design Automation Workshop, pp. 116-127 (1972).

33) TABARLY J.H. : Contribution à la Synhtèse de Tests Complets de Circuits Digitaux par Observation - Commande - Programme DEDALE", Thèse EEA, Univ. P. Sabatier, Toulouse, juillet 1974.

34) THOMAS J.J. "Automated Diagnostic Test Programs for Digital Networks", Computer Design, Vol. 10 n° 8, pp. 63-67, Aug. 1971.

35) THUEL J. : "Etude de l'Influence des Pannes Multiples dans les Réseaux Logiques et Application à leur Détection", Thèse Docteur Ingénieur USMG et INPG à Grenoble, janvier 1974.

36) TOTH A., CHIP HOLT. : "Automated Data Base Driven Digital Testing", Computer, pp. 13-19, Jan. 1974.

37) TULLOUE R., ZIRPHILE J. :"Simulation Déductive Etendue à trois Valeurs Logiques", Note Interne à paraître, Thomson-CSF/DIS, 33, rue de Vouillé, 75015 PARIS.

38) TYURIN A.V. : "Design of Diagnostic Test for Synchronous Sequential Circuits", Automation and Remote Control, Vol. C35 n° 7, pp. 1132-1140, Dec. 1974.

39) VERDILLON A. : "Pannes dans les Réseaux Acycliques", Thèse de 3e Cycle USMG, Décembre 1972.

40) VERDILLON A. : "Pannes dans les Réseaux sans Circuits", RAIRO, pp.85-99 Juillet 1974.

41) VERNA J.P. et all : "Automatic Test Generation and Test Verification of Digital Systems", 11th Design Automation Workshop, pp. 149-158, (1974)

42) YAU S.S., YANG S.C. : "Multiple Fault Detection for Combinational Logic Circuits", IEEE Trans. Comp. Vol. C24 n° 3, pp. 233-242, March 1975.

43) CAILLAT J., TULLOUE R., ZIRPHILE J. : "TAU-1 : Un Outil pour le Test des Circuits Intégrés Logiques", Revue Technique THOMSON-CSF à paraître (1976).

-o-o-o-o-