Contribution à la détection et à la localisation des pannes dans les sytèmes séquentiels

Marc Courvoisier

▶ **To cite this version:**

Marc Courvoisier. Contribution à la détection et à la localisation des pannes dans les sytèmes séquentiels. Automatique / Robotique. Université Paul Sabatier - Toulouse III, 1971. Français. NNT : . tel-00176335

HAL Id: tel-00176335
https://theses.hal.science/tel-00176335

Submitted on 3 Oct 2007

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Année 1971

THÈSE

présentée

A L'UNIVERSITÉ PAUL SABATIER DE TOULOUSE

en vue de l'obtention

du titre de Docteur de 3e Cycle

Spécialité : AUTOMATIQUE

par

Marc COURVOISIER

Maître ès Sciences

CONTRIBUTION A LA DÉTECTION ET A LA LOCALISATION DES PANNES DANS LES SYSTÈMES SÉQUENTIELS

Soutenue le 14 Janvier 1971 devant la Commission d'Examen

MM.	J. LAGASSE	Président
	Y. SEVELY	Examinateurs
	G. GRATELOUP	
	C. DURANTE	
	M. CHAUSSARD	
	G. PIEL	

- A V A N T - P R O P O S -

Nous ne pouvons commencer l'exposé de notre mémoire, sans remercier tous ceux qui ont contribué à sa réalisation.

Monsieur le Professeur LAGASSE, Directeur du LABORATOIRE D'AUTOMATIQUE et de ses Applications Spatiales, en nous accueillant dans son établissement, nous a permis d'apprendre ce qu'était le métier de chercheur.

Qu'il préside aujourd'hui notre Jury de Thèse, nous honore profondément. Nous l'en remercions.

Par sa présence à ce Jury et les tâches d'enseignement qu'il a bien voulu nous donner le moyen d'exercer, Monsieur le Professeur SEVELY nous a accordé doublement sa confiance. Nous ne saurions lui exprimer notre gratitude.

Nous remercions vivement Monsieur le Professeur GRATELOUP, Directeur du Département de Génie Electrique de l'Institut National des Sciences Appliquées de Toulouse, d'avoir prété une bienveillante attention à ce mémoire et accepté de siéger à notre Jury de Thèse.

Monsieur DURANTE, Maître de Conférences à la Faculté des Sciences de Montpellier, a été tout au long de notre étude un guide et un conseiller attentif. Maître d'oeuvre des travaux abordés au L.A.A.S. sur le test des systèmes logiques, il a, par ses connaissances, permis la rédaction de ce mémoire. Nous tenons à lui exprimer ici notre profonde reconnaissance.

Il nous est agréable de remercier Monsieur CHAUSSARD, Chef du Département "Automatique des Moyens de Production" à la Direction des Etudes et Recherches de l'E.D.F., pour l'intérêt qu'il a porté à notre travail, et l'accueil qu'il nous a réservé lorsque nous lui avons présenté notre mémoire.

Monsieur PIEL, Ingénieur en Chef de la Direction des Etudes et Développements de la Compagnie Internationale pour l'Informatique, par ses travaux, est à l'origine d'une méthode de test que nous allons présenter. Nous lui en sommes reconnaissant et nous le remercions chaleureusement de participer à notre Jury de Thèse.

Nous remercions également ici, Messieurs M. DIAZ, JC GEFFROY, J. RICHARD qui, par les fréquentes conversations que nous avons eues, nous ont évité moult erreurs et faux pas.

Nous n'oublierons pas, enfin, de remercier tout le service de documentation et plus particulièrement Madame DUFOUR grâce à laquelle ces quelques lignes pourront être portées à la connaissance de tous ceux qui en exprimeront le désir.

-:-:-:-:-

- INTRODUCTION -

Dès que l'homme a été capable d'utiliser son cerveau et ses mains pour réaliser des objets de toutes sortes, il s'est attaché à vérifier que le fruit de ses efforts remplissait correctement la tâche à laquelle il le destinait.

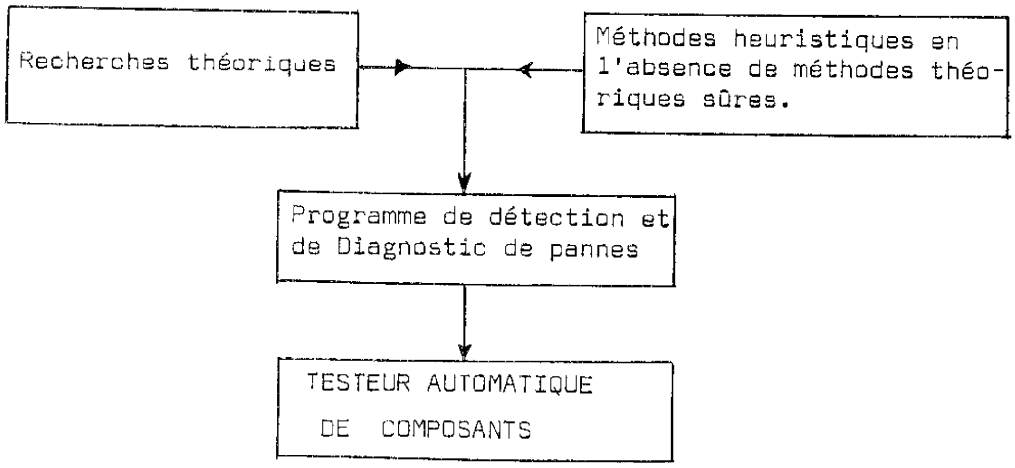
Rien n'échappe à cette règle et les systèmes séquentiels objet de notre étude en sont un exemple très démonstratif.

Si l'on veut vérifier le comportement d'un système, la moindre des choses est de connaître son fonctionnement de façon très précise. Dans le cas où cela n'est pas (manque d'informations) il faut entreprendre l'analyse du système afin d'en observer toutes les facettes. Fort de ces enseignements, on peut alors soumettre le système identifié à une procédure de vérification, c'est-à-dire le tester. A ce moment, plusieurs possibilités s'offrent à nous :

- réaliser un test exhaustif en obligeant le système à fonctionner selon tous les cas prévus.
- ou bien profiter de ses caractéristiques pour bâtir un test adapté, donc plus court, mais capable de détecter toute défaillance. Si le système, soumis à une procédure de test, dévoile un défaut, un autre problème se pose, plus délicat encore : connaître la cause du mauvais fonctionnement, soit puisqu'il s'agit ici de circuits logiques, localiser la panne avec le plus de précision possible.

Ces problèmes que nous venons d'évoquer vont constituer notre sujet de préoccupations tout au long de ce mémoire. Nous essayerons d'apporter, grâce à la puissance que constitue la simulation et le traitement par ordinateur, des méthodes algorithmiques permettant de mieux cerner les innombrables difficultés qui apparaissent lorsqu'on s'achemine vers l'élaboration de solutions acceptables. Celles que nous proposons ont été appliquées à des systèmes séquentiels de petite taille (du type Bascule RS, D, JK, compteurs,...) car notre but n'a pas été de créer un programme opérationnel destiné au test des gros ensembles logiques, mais plutôt d'aborder le problème du test des systèmes séquentiels en essayant d'en voir différents aspects.

Ce travail s'est effectué au sein du groupe Systèmes Logiques dirigé par Monsieur DURANTE. Il constitue une partie d'un ensemble décrit par le schéma ci-dessous.



Les recherches théoriques (18) qui ont porté jusqu'à présent sur les systèmes combinatoires, et les méthodes algorithmiques de test des systèmes séquentiels, ont conduit, d'une part à l'écriture d'un programme de localisation de pannes dans les systèmes combinatoires (20), et d'autre part, à la mise au point de plusieurs programmes de simulation et de test des systèmes séquentiels, qui seront décrits dans ce mémoire.

Le test physique des composants intégrés commerciaux a été également un thème de travail abordé : il s'est traduit par la conception et la réalisation d'un testeur automatique de composants (21).

Au chapitre I, sont donnés quelques rappels très simples sur les systèmes séquentiels, suivis de deux exemples, ainsi que quelques définitions relatives au test de ces mêmes systèmes. Les principales méthodes jusqu'à présent employées se trouvent résumées au chapitre II, la synthèse des résultats obtenus en fonction des hypothèses avancées permettant de tirer quelques conclusions.

Le chapitre III expose une méthode classique d'analyse des systèmes séquentiels ainsi que les algorithmes qui ont conduit à l'écriture d'un programme général. Il donne toutes les caractéristiques logiques d'un circuit (en travaillant sur des équations booléennes) par la seule connaissance de son schéma et constitue l'outil de base indispensable aux programmes de test qui

suivent.

Le problème du test fonctionnel d'un système séquentiel, abordé au chapitre IV a donné lieu à une méthode issue du dépouillement d'une série de spécifications. Elle consiste à tester chaque transition du graphe de fluence du système en recherchant parmi tous les vecteurs qui participent à une transition s'il en existe un ou plusieurs ayant la propriété par leur test de la vérifier de façon sûre. Si cela est possible, c'est que le test de ces vecteurs recouvre et évite le test des autres vecteurs de la transition. Le but de la méthode est donc de minimiser le nombre de vecteurs à tester pour s'assurer du bon fonctionnement d'un système. Ceci fait l'objet d'un premier programme. Le deuxième, à partir des résultats obtenus, traite l'enchaînement en séquence des vecteurs à tester et fournit, pour un circuit donné, une séquence de détection de pannes aussi courte que possible. Toute la première partie de cette étude est menée à l'aide des équations logiques (variables secondaires) déterminées par l'analyse. La recherche d'une séquence emploie la matrice des transitions du système où sont notés les vecteurs à tester.

La localisation des pannes, c'est-à-dire le diagnostic des systèmes séquentiels, étudié au chapitre V, s'est traduit également par l'écriture de deux programmes. En effectuant d'abord (premier programme) la simulation d'un ensemble connu de pannes, on a supposé que le circuit fonctionnait mal. Ceci permet de connaître les caractéristiques d'un système lorsqu'il est affecté d'une des pannes prévues grâce aux équations et tables qui résultent de l'analyse. La comparaison des tables des excitations secondaires et des sorties (deuxième programme) transformées par une panne avec celles du circuit correct, met en évidence les transitions défectueuses. L'emploi de séquences de synchronisation qui rend possible la méconnaissance de l'état initial du système, associé à la recherche précédente permet d'établir des séquences de test capables de localiser une ou plusieurs pannes parmi d'autres.

C H A P I T R E I

RAPPELS SUR LES SYSTEMES SEQUENTIELS

SIGNIFICATION ET BUT DU TEST

RAPPELS SUR LES SYSTEMES SEQUENTIELS
SIGNIFICATION ET BUT DU TEST

Nous ne pouvons aborder l'étude du test des systèmes séquentiels sans avoir au préalable défini les deux mots que nous venons d'employer :

- Systèmes séquentiels
- Test

Nous nous efforcerons de le faire au cours de ce premier chapitre, au moyen d'exemples, et nous en profiterons pour y inclure la terminologie qui nous sera indispensable par la suite.

I-1 RAPPELS SUR LES SYSTEMES SEQUENTIELS

I.1.1 Module

Elément physique réalisant des opérations booléennes élémentaires :

- complémentation : PAS
- intersection : ET
- réunion : OU
- complément de la réunion : NI (NOR)
- complément de l'intersection : ON (NAND)
- somme modulo 2 : OU EXCLUSIF

I.1.2 Système combinatoire

Assemblage de modules tel que l'information se propage des entrées vers les sorties. Il est caractérisé par le fait qu'à une combinaison des entrées correspond toujours la même combinaison des sorties. En d'autres termes, l'état futur du système est connu dès l'affichage d'un vecteur d'entrée quel que soit son état antérieur.

I.1.3 Systèmes séquentiels

I.1.3.1. Définitions

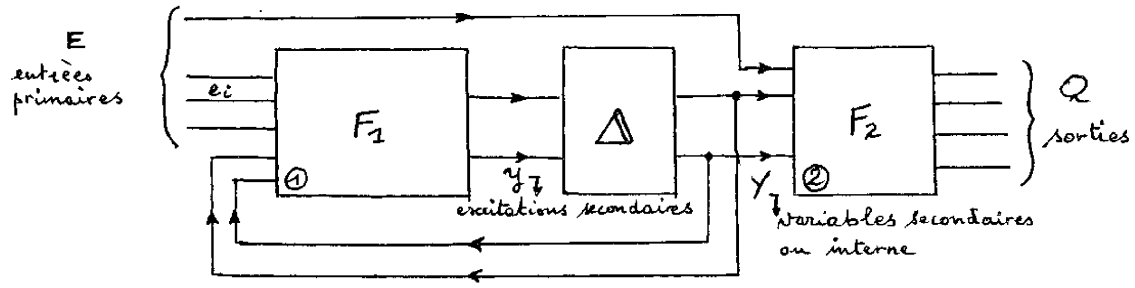
Un système est séquentiel lorsque le vecteur de sortie dépend non seulement du vecteur d'entrée appliqué mais aussi de l'état (niveaux logiques de tous les points du réseau) dans lequel il se trouve au moment où des informations différentes apparaissent en entrée.

Pour qu'un système agisse en fonction de son état antérieur (fonction mémoire) il est nécessaire qu'existent en certains points du réseau des liaisons prélevant les niveaux qui s'y trouvent pour les réinjecter en d'autres points où ils viendront contribuer à l'état futur au même titre que les entrées. On pourrait pour cette raison appeler ces boucles de réinjection, des entrées internes. A un moment, à un vecteur entrée primaire donné et à un vecteur interne entrée donné, correspond toujours le même état du système.

Le temps intervient de façon explicite par la présence des bouclages. En effet, les signaux se trouvent retardés à la traversée de chacun des modules, et les boucles envoient de nouvelles informations dans le circuit dès qu'elles sont sensibilisées. Le système réagit ainsi jusqu'à ce que toutes les sorties de modules gardent une valeur constante. Dans ce cas, on dit que le système s'est stabilisé. Il peut arriver qu'il y ait oscillation incessante. Nous verrons à quoi cela est dû.

Pour que l'étude des systèmes séquentiels puisse se faire autrement que pas à pas, nous allons échantillonner, ou quantifier le temps et supposer qu'à des endroits bien définis, la propagation est retardée en bloc, alors qu'ailleurs elle s'effectue instantanément. On est conduit à inclure des retards dans les branches de contre réaction, puisque ce sont elles qui donnent au système sa nature séquentielle et que tout le retard créé par les chaines de modules que traverse l'information, se "retrouve" à cet endroit.

Ceci conduit à représenter un système séquentiel par le schéma bloc suivant :



Un réseau combinatoire ① est soumis à un certain nombre d'entrées primaires et à d'autres entrées, non accessibles de l'extérieur, appelées variables secondaires ou variables internes. Les sorties de ce réseau sont les excitations secondaires qui vont constituer les variables secondaires après avoir été retardées de Δ , et former avec des entrées primaires éventuelles, les entrées d'un second réseau combinatoire ② , dont les sorties sont les sorties du système.

On peut représenter le fonctionnement de ces machines par les deux équations suivantes :

$$Y(T+\Delta) = F1\ (Y(T), E(T))$$

$$Q(T) = F2\ (Y(T), E(T))$$

Pour une configuration d'entrée et un état donné (variables secondaires et sorties), l'évolution du système va avoir lieu jusqu'à ce que les variables secondaires soient égales aux excitations secondaires; on dira qu'un nouvel état stable a été atteint. Si les variables secondaires ne sont jamais égales aux excitations secondaires, il y a oscillation.

Cette configuration représente le type le plus général de machine séquentielle. Les machines s'identifiant à cette structure sont nommées machines de MEALY.

Si les sorties Q ne dépendent que des variables secondaires, on obtient une catégorie de machines appelées machines de MOORE, qui satisfont aux équations suivantes :

$$Y(T-\Delta) = F1\ (\ Y(T),\ E(T)\ )$$

$$Q(T) = F2\ (\ Y(T)\ )$$

Elles montrent que les sorties sont liées aux états de la machine.

Pour avoir une interprétation immédiate du fonctionnement d'un ... uit on représente ses équations sous forme de tables. Il existe de nombreuses variantes. Nous utiliserons la table des excitations secondaires, la matrice des sorties et la matrice des transitions.

I.1.3.2. Table des excitations secondaires

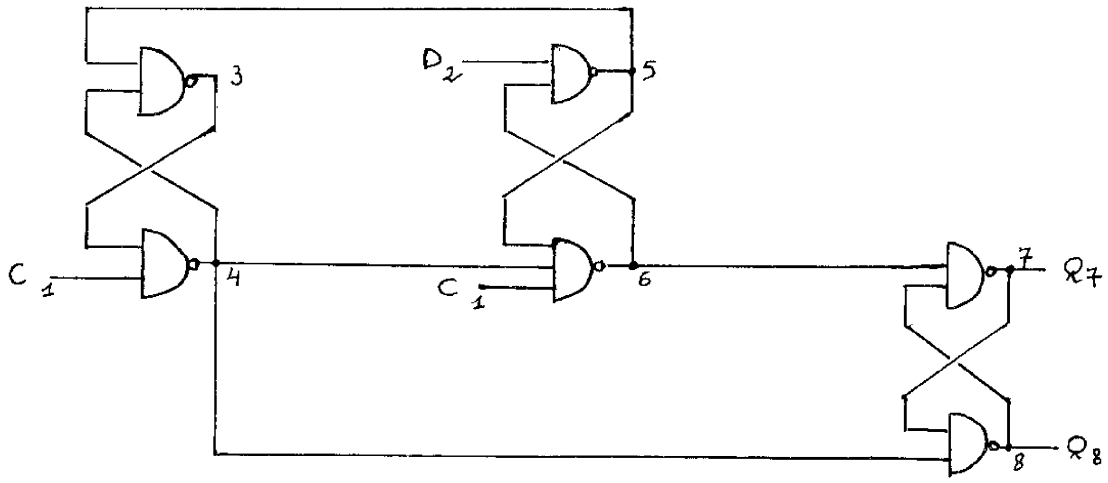
Elle est constituée de la manière suivante :

Dans la première ligne se trouvent les combinaisons binaires des entrées du circuit, chacune de ces combinaisons est une tête de colonne. Suivent autant de lignes que de combinaisons des variables secondaires. En tête de ces lignes, sont placées, à raison d'une par ligne, les combinaisons binaires des variables secondaires à l'instant t.

L'intersection d'une ligne et d'une colonne définit une case, à l'intérieur de laquelle figurent les variables secondaires à l'instant t+1, également sous forme de combinaisons binaires.

Suivre le fonctionnement d'un système soumis à une séquence d'entrée est très simple :

Connaissant l'état de départ (ligne), l'application d'un vecteur d'entrée (colonne) indique l'état suivant du système. S'il est égal au numéro qui figure en tête de ligne, l'évolution est stoppée, un état stable étant atteint. On entoure par un cercle les cases qui dans une ligne contiennent le même numéro que celui placé en tête de la ligne. Ce sont les états stables. Si ce numéro est différent, on se reporte dans la ligne correspondante, et le processus recommence. Tant que les entrées restent identiques à elles-mêmes, les déplacements se font dans une colonne. Si dans ce cas, il n'y a jamais



$$\begin{cases} Y_4(\tau+1) = \left(\overline{D} + \overline{Y_6(\tau)}\right) . Y_4(\tau) + \overline{C} \\ Y_6(\tau+1) = D . Y_6(\tau) + \overline{C} + \overline{Y_4}(\tau) \\ Y_8(\tau+1) = Y_6(\tau) . Y_8(\tau) + \overline{Y_4}(\tau) \end{cases}$$

$$\begin{cases} Q_8(\tau) = Y_8(\tau) \\ Q_7(\tau) = \overline{Y_6}(\tau) + \overline{Y_8}(\tau) \end{cases}$$

MATRICE DES EXCITATIONS

C	0		1	
D / Y_4 Y_6 Y_8	0	1	0	1
0 0 0	111	111	011	011
1 0 0	110	110	100	100
0 1 0	111	111	011	011
1 1 0	110	110	100	010
0 0 1	111	111	011	011
1 0 1	110	110	100	100
0 1 1	111	111	011	011
1 1 1	111	111	101	011

↖ Poids les plus forts à droite

MATRICE DES SORTIES

C	0		1	
D / Y_4 Y_6 Y_8	0	1	0	1
0 0 0	11	11	11	11
1 0 0	01	01	01	01
0 1 0	11	11	11	11
1 1 0	01	01	01	01
0 0 1	11	11	11	11
1 0 1	01	01	01	01
0 1 1	10	10	10	10
1 1 1	10	10	10	10

BASCULE D SN 7474

figure I.1

Matrice des transitions

CODAGE

Entrées

	D	C
1	0	0
2	1	0
3	0	1
4	1	1

Variables secondaires

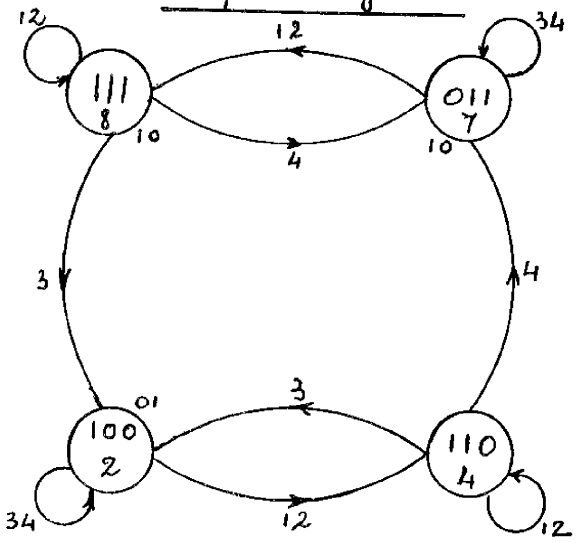
	Y_4	Y_6	Y_8
2	1	0	0
4	1	1	0
7	0	1	1
8	1	1	1

↳ Etats stables

Etats de départ ↑	2	4	7	8 ← Etats d'arrivée
2	$(3,4)_{01}$	$(1,2)_{01}$		
4	$(3)_{01}$	$(1,2)_{01}$	$(4)_{10}$	
7			$(3,4)_{10}$	$(1,2)_{10}$
8	$(3)_{01}$		$(4)_{01}$	$(1,2)_{0,1}$ ← sorties

↑ vecteurs d'entrée

Graphe de fluence



Bascule D 7474

figure I.2

passage dans un numéro cerclé, on dit que le système présente un cycle. Nous ne nous attarderons pas sur les conséquences qui en découlent, l'ouvrage de Messieurs PERRIN, DACLIN et DENOUETTE (2) développe largement ce sujet.

I.1.3.3. Matrice des sorties

Sa constitution est identique à celle de la table des excitations secondaires. Dans les cases sont placées les combinaisons binaires des sorties.

Cette matrice permet de voir immédiatement à quelle catégorie appartient la machine considérée. Celle de la figure I.1 montre que le circuit qu'elle représente est une machine de MOORE puisque les sorties gardent une valeur constante à chaque ligne. Ces deux tables décrivent complètement le fonctionnement d'un système séquentiel.

I.1.3.4. Matrice des transitions; graphe de fluence

Nous la définirons comme représentative du graphe de fluence du système. Elle indique les transitions entre états stables, et par là même contient moins d'informations que les deux matrices précédentes.

La première ligne donne les états d'arrivée, la première colonne ceux de départ. A l'intersection d'une ligne et d'une colonne se trouvent les vecteurs d'entrée permettant le passage de l'état qui figure en tête de ligne à celui inscrit en tête de colonne, ainsi que la valeur des sorties dans l'état d'arrivée.

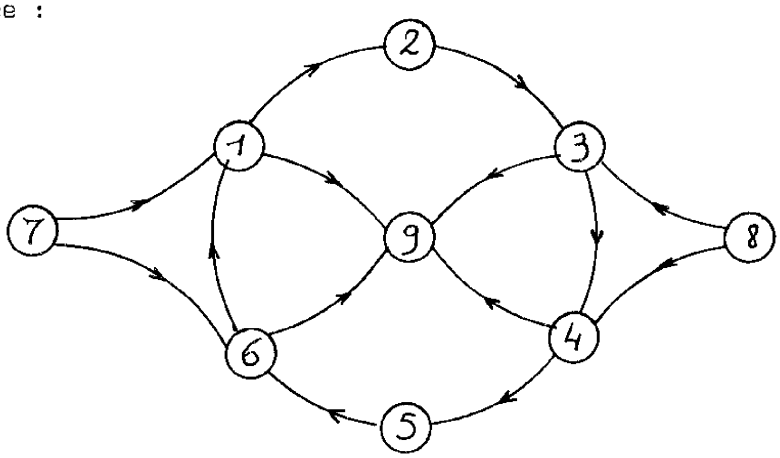
Si à l'intersection d'une ligne et d'une colonne, n'existe pas de vecteur, c'est qu'aucune transition ne lie les deux états correspondants. Sur la figure I.2 est représentée la matrice des transitions de l'exemple de la figure I.1, sous forme codée.

I.1.3.5. Circuits fortement connectés

Un circuit est fortement connecté (ou lié) si de n'importe quel état on peut atteindre n'importe quel autre. Le graphe de fluence de la figure I.2

donne l'exemple d'un circuit fortement connecté.

Par contre le diagramme ci-dessous est celui d'une machine non fortement liée :



En effet, si la machine est dans l'état 9, elle y reste bloquée. On dira que 9 est un état puits.

. Définition : un état puits est un état à partir duquel il est impossible d'atteindre tout autre.

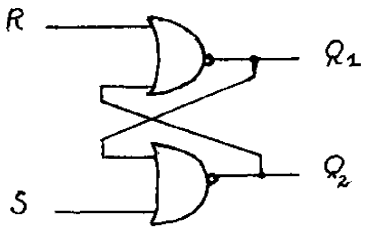
D'autre part, si la machine se trouve dans l'un des états suivants : 1 ..... 6, d'aucune façon elle ne pourra accéder aux états 7 ou 8. Ceux-ci sont appelés états sources.

. Définition : un état source est un état auquel il est impossible d'accéder à partir de tout autre.

I.1.4 Exemples

Nous allons maintenant donner deux exemples de machines séquentielles très largement utilisées de par leurs caractéristiques pour la constitution de compteurs, registres à décalages, et dont nous nous préoccuperons tout au long de ce mémoire.

I.1.4.1. Bascule RS



Equations :

$Y(T+1) = \overline{R + \overline{S+Y(T)}} = \overline{R}\ Y(T) + \overline{R}\ S$

$Q1\ (T) = Y\ (T)$

$Q2\ (T) = \overline{S + Y\ (T)}$

Tables :

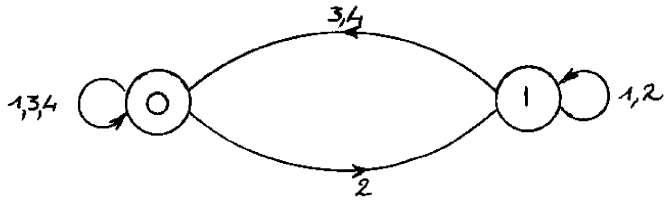
Y \ RS	00	01	11	10
0	(0)	1	(0)	(0)
1	(1)	(1)	0	0

Y (T+1)

Y \ RS	00	01	11	10
0	01	10	00	01
1	10	10	00	00

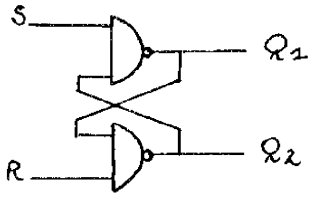
Q1, Q2

Graphe :



Cet élément est aussi appelé "mémoire RS". On voit en effet que pour la combinaison d'entrée 00, les deux états Y = 0 et Y = 1 sont stables. Leur valeur précédente est donc mémorisée. On peut également constater que $Q1 = \overline{Q2}$ dans les états stables sauf pour la combinaison 11, ce qui classe ce circuit parmi les machines de Mealy. Cependant lors d'un usage normal, il est souhaitable que la relation $Q1 = \overline{Q2}$ soit vérifiée, d'où l'interdiction de la combinaison d'entrée 11.

On obtient un système duel de celui-ci en interconnectant de la même manière deux modules ON :



Equations :

$Y(T+1) = \overline{S \cdot \overline{R\,Y(T)}} = \overline{S} + R\,Y(T)$

$Q1(T) = Y(T)$

$Q2(T) = \overline{R \cdot Y(T)}$

Tables :

Y \ SR	00	01	11	10
0	1	1	(0)	(0)
1	(1)	(1)	(1)	0

Y (T+1)

Y \ SR	00	01	11	10
0	11	11	01	01
1	11	10	10	01

Q1, Q2

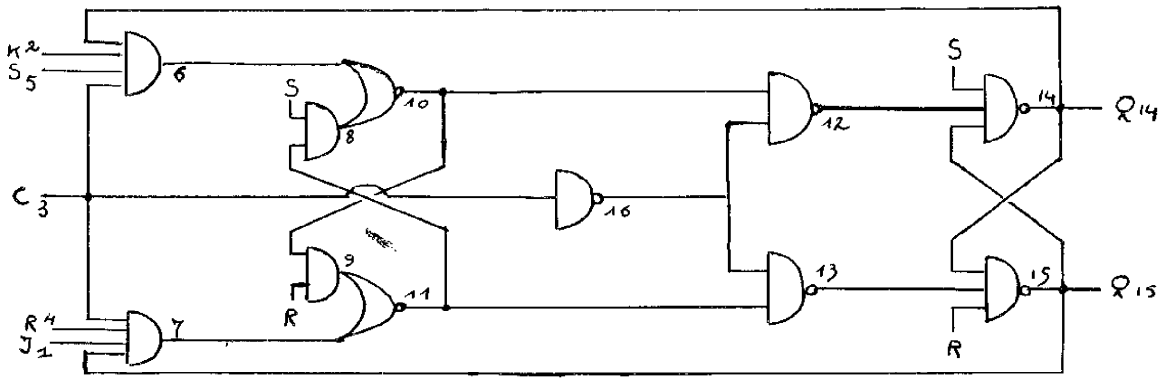
C'est sur la combinaison 11 qu'a lieu l'état mémoire, la combinaison 00 étant ici interdite (Q1 = Q2 = 1)

Pour palier l'inconvénient que provoque une combinaison interdite des entrées, a été conçue la bascule JK. Nous allons plus particulièrement nous intéresser aux bascules JK Maitre Esclave.

I.1.4.2. Bascule JK maitre esclave (figure I.3)

L'exemple type est la bascule SN 7472 (TEXAS INSTRUMENTS). Elle comporte cinq entrées primaires :

- sur J et K sont appliqués les signaux de commande
- l'horloge sert à valider ces deux entrées et à synchroniser le comportement du circuit.



$$\begin{cases} Y_{10}(T+1) = K'RY_{10}(T) + C'RY_{10}(T) + S' + RY_{10}(T)\,Y_{15}(T) + JCRY_{15}(T) \\ Y_{15}(T+1) = R' + S\,Y'_{10}(T)\,Y_{15}(T) + C'Y'_{10}(T) + CSY_{15}(T) \\ Q_{15}(T) = Y_{15}(T) \\ Q_{14}(T) = S' + C'Y_{10}(T) + Y_{15}(T) \end{cases}$$

	C		0				1			
	JK		00	10	01	11	00	10	01	11
	Y_{10}-Y_{15}									
1	0	0	(00)	(00)	(00)	(00)	(00)	10	(00)	10
2	1	0	01	01	01	01	(10)	(10)	(10)	(10)
3	0	1	(01)	(01)	(01)	(01)	(01)	(01)	11	11
4	1	1	00	00	00	00	(11)	(11)	(11)	(11)
			1	2	3	4	5	6	7	8

avec $R = S = 1$

$Y_{10}(T+1)$, $Y_{15}(T+1)$

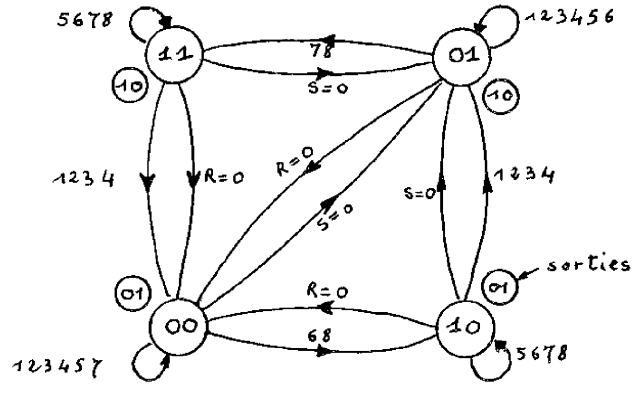


figure I.3

- R et S sont deux entrées de positionnement à 0 et à 1 agissant sur le niveau bas, et prioritaires sur les autres. Il ne faut pas les rendre agissantes en même temps (ce qui semble logique) sous peine de trouver Q14 = Q15 (cas des bascules RS)

Son fonctionnement est donné par les tables des excitations et des sorties, et le graphe de fluence de la figure I.3.

Sur le niveau haut de l'horloge, la première bascule appelée MAITRE prend en compte les informations qui apparaissent sur J et K, alors que la deuxième bascule (ESCLAVE) se trouve isolée dans un état mémoire. C'est l'inscription dans le maitre. Lorsque l'horloge redescend, la bascule Maitre est à son tour isolée (toute variation des entrées J et K ne crée sur elle aucun effet) et la bascule Esclave "recopie" l'état du Maître : transfert du Maître dans l'Esclave. Il y a donc un découplage parfait du système, d'où impossibilité d'oscillation et grande sécurité de fonctionnement.

De plus, toutes les combinaisons d'entrées peuvent être appliquées. On peut résumer leur rôle par la table de vérité simplifiée suivante :

J	K	QN+1
0	0	QN
0	1	0
1	1	$\overline{QN}$
1	0	1

I-2 SIGNIFICATION ET BUT DU TEST

TEST : Epreuve consistant en une tâche déterminée, destinée à apprécier ou à mesurer telle ou telle caractéristique d'un sujet.

Cette définition générale (Larousse 3vol. 1965) résume admirablement les quelques lignes ci-dessous où nous allons essayer d'exposer ce que doit représenter le test des systèmes séquentiels.

On peut en distinguer trois catégories :

- Test statique
- Test fonctionnel
- Test dynamique

I.2.1 Tests statiques

Ils consistent à vérifier les caractéristiques électriques d'un circuit, c'est-à-dire à s'assurer, que les tensions et courants d'entrée et de sortie restent à l'intérieur des normes données par le constructeur, ainsi que de l'associabilité de ce circuit avec d'autres (entrance et sortance, ou fan-in et fan-out).

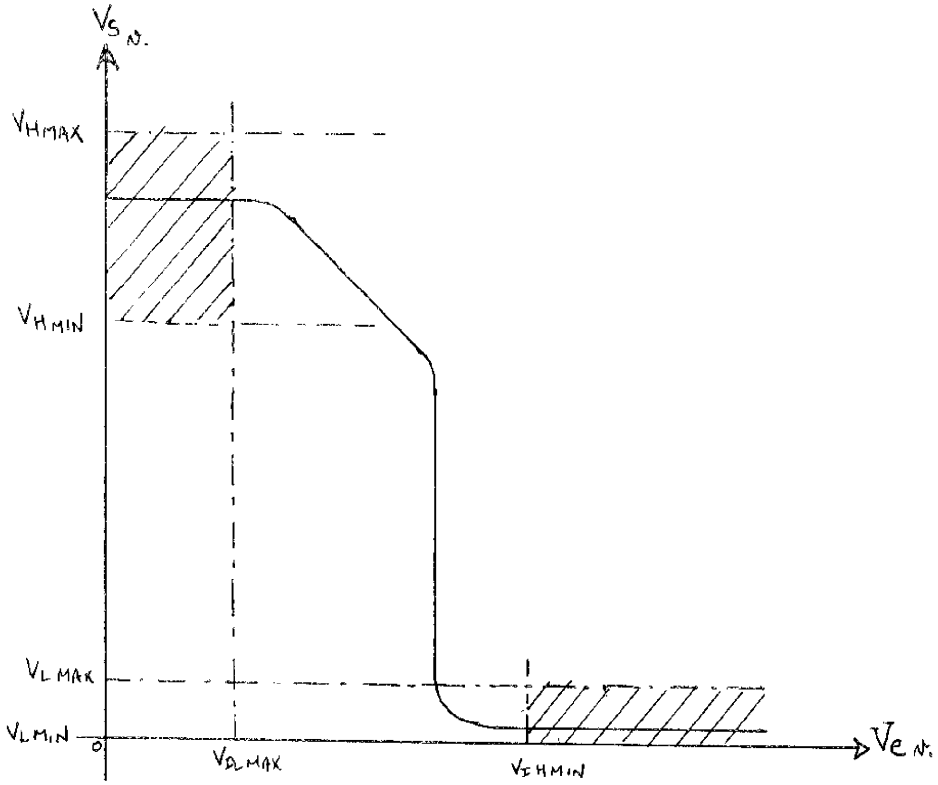
Examinons par exemple la figure I.4 qui représente la caractéristique VS (Ve) d'une porte NAND TEXAS SN 7400 (figure I.5). Il existe des seuils de tension tels que s'ils sont respectés, le fonctionnement du circuit est garanti. Ainsi sur la figure I.4, le point de fonctionnement doit se trouver à l'intérieur d'une des deux parties hachurées. Ceci conduit à l'élaboration de toute une série de tests sur la mesure des courants absorbés ou injectés par le circuit pour les tensions limites définies sur la figure I.4.

Cette partie constitue l'un des thèmes de travail sur l'analyse des circuits logiques. Elle a été développée à L.A.A.S. et a conduit à la conception et à la réalisation d'un testeur automatique de composants logiques intégrés (21).

Donnons ici un exemple qui nous permettra de tirer quelques conclusions :

(Test du courant débité par une entrée soumise à une tension V_{ILMAX} (0,4 V)
(soit $I_{I\ (LMAX)}$

Cette mesure correspond au schéma de la figure I.5. Il faut placer sur les entrées non testées un niveau de tension. Le cas le plus défavorable consiste à les soumettre à la tension haute maximum admissible par le circuit : 5,5 volts, entraînant ainsi une augmentation du courant débité par l'entrée sous test. On vérifie alors que le courant mesuré ne dépasse pas une certaine limite donnée.



Entrée : $\begin{cases} V_{ILMAX} : 0,8\,V \\ V_{IHMIN} : 2\,V \end{cases}$

Sortie $\begin{cases} V_{LMIN} : 0\,V. \\ V_{LMAX} : 0,4\,V. \end{cases}$ $\begin{cases} V_{HMIN} : 2,4\,V \\ V_{HMAX} : 5,5\,V. \end{cases}$

Figure I.4 : Caractéristique de transfert Porte NAND TTL SN 7400

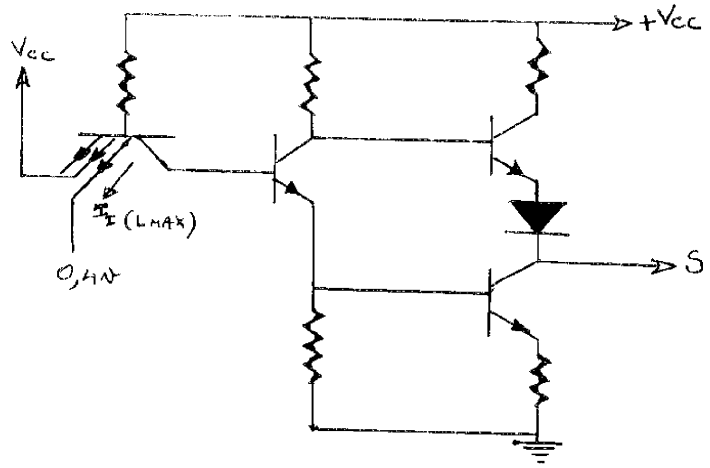


figure I.5 : Porte NAND TTL 7400

On voit apparaître d'une part ce que l'on peut appeler la notion de "Pire cas" qui donne au test son efficacité maximum, et d'autre part que le test statique que nous venons de définir a fait appel implicitement au test fonctionnel. En effet, on peut remarquer que le test sur une porte NAND (ON) d'une entrée à zéro n'est observable que si les autres entrées sont à un. (Cet exemple d'un circuit combinatoire a été choisi pour sa simplicité).

I.2.2 Tests dynamiques

Leur but est de vérifier le comportement temporel d'un circuit, c'est-à-dire d'étudier la transformation d'une impulsion étalon lors de son passage à travers un système logique (temps de montée, de descente, retard).

I.2.3 Tests fonctionnels

Cette catégorie de tests est destinée à vérifier le bon fonctionnement logique d'un circuit (combinatoire ou séquentiel) du point de vue des niveaux apparaissant sur les sorties, sans se préoccuper des temps de réponse (on les appelle aussi tests logiques statiques).

La première partie d'un test de bon fonctionnement consistera à s'assurer qu'un circuit réalise correctement la fonction logique prévue; c'est ce que nous appellerons par la suite le test Fonctionnel. Si un mauvais fonctionnement est détecté une deuxième partie fera appel à des tests de DIAGNOSTIC ayant pour tâche de localiser avec le plus de précision possible d'où vient la panne et quelle est sa cause.

Ce deuxième test s'il peut sembler superflu pour un utilisateur manipulant des "boites noires", devient très important, pour le fabricant s'il se rend compte que sur une chaîne de production une grande quantité de circuits ne fonctionne pas, et pour l'utilisateur qui ne veut pas jeter toute carte incorrecte.

Hypothèse fondamentale : nous dirons qu'un circuit est en panne si la fonction logique qu'il réalise est fausse, et que cette faute existe en permanence.

Le fait qu'un circuit soit en panne peut résulter d'une multitude de causes : court circuit entre interconnexions, connexions court-circuitées à la masse où à l'alimentation, coupées, mauvaises soudures, etc... De plus on conçoit qu'en règle générale il sera pratiquement impensable de réaliser le test exhaustif d'un circuit qui pour un circuit combinatoire à n entrées demande d'appliquer 2^n vecteurs d'entrée et pour un circuit séquentiel à n entrées et m états, nécessite $m.2^n$ affichages, auxquels il faut ajouter des séquences de positionnement dans les états initiaux.

Par conséquent on est amené à faire des hypothèses sur la nature des pannes susceptibles d'intervenir le plus souvent, et le moyen de rendre compte de leur effet, sous forme logique.

Le test des systèmes combinatoires donne lieu à des études théoriques développées en particulier au L.A.A.S. (18), ces dernières ayant permis la mise au point d'un programme de tests de détection et de localisation des pannes (20).

Nous examinerons au chapitre II de quelles façons ceci a été abordé, en ce qui concerne les systèmes séquentiels et nous pourrons alors résumer les principes généralement adoptés.

Donnons ci-dessous, pour compléter cette brève présentation, quelques définitions supplémentaires que nous retrouverons par la suite :

- Panne détectable : telle que la modification qu'elle impose au circuit est observable sur les sorties.
- Panne indétectable : l'inverse
- Redondance : méthode destinée à protéger le circuit contre la défaillance des modules qui le composent,en utilisant plusieurs modules à la place d'un seul.
- Circuit redondant : circuit utilisant la méthode précédente et pouvant donner lieu à des pannes indétectables.
- Panne simple : panne qui n'affecte qu'un seul élément d'un circuit.

En général les méthodes de test sont conçues de façon à détecter les pannes simples. En effet, si un test détecte toutes les pannes simples d'un

circuit il est peu probable qu'une panne multiple ne le soit pas; cependant le problème du masquage d'une panne par une autre risque de mettre ceci en défaut.

- Collage à 0 et à 1 d'une connexion (ca 0, ca 1) : cette dénomination correspond à la présente permanente sur une connexion d'une tension proche ou égale à celle de la masse (ca 0), ou d'une tension proche ou égale à celle de la source d'alimentation (ca 1). Ce type de panne se traduit en termes logiques par des niveaux 0 ou 1, et recouvre un nombre important de pannes physiques : cours circuits à la masse où à l'alimentation, connexions coupées ou manquantes ... Pour cette raison c'est souvent ce type de panne que les méthodes de test envisagent, et de plus c'est la seule hypothèse qui permet de traiter le problème.

C H A P I T R E II

PRINCIPALES METHODES DE TEST

PRINCIPALES METHODES DE TEST

Nous allons examiner, parmi celles que nous connaissons, les études qui nous paraissent les plus marquantes et les solutions proposées pour mener à bien le test des systèmes séquentiels.

Etant donnée la complexité du problème, chacune des méthodes exposées nécessite des hypothèses plus ou moins restrictives quant à la nature des systèmes qu'on peut lui soumettre, ou des pannes susceptibles d'exister. En général, les circuits auxquels ces méthodes s'appliqueront sont synchrones.

On peut classer les méthodes de test des systèmes séquentiels en deux catégories :

- celles qui reprennent les procédés utilisés pour le test des systèmes combinatoires, et les étendent :
  - . méthodes de sensibilisation des itinéraires
  - . méthodes de partitionnement de la table des pannes.
- celles utilisant les caractéristiques propres d'un système séquentiel.
  - . méthodes mettant en jeu les tables de fluence ou des phases.

II-1 METHODES DE SENSIBILISATION DES ITINERAIRES

II.1.1 Méthode du circuit séquentiel développé (4)

1. Hypothèses

- Les pannes possibles sont de type logique : collage à zéro ou à un des sorties de portes ou Flip-Flop , et sont connues à l'avance.
- Chaque circuit possède au plus une panne.
- Il est possible de positionner l'état mémoire du circuit correct momentanément dans un état particulier connu.

- Le circuit sans panne et chacun des circuits affectés d'une panne sont distinguables en leur appliquant une séquence d'entrée de longueur n partant des mêmes états mémoires, où n est connu et limité (au plus 10, p.ex.).

Une panne transforme un circuit en un autre différent. Par la première hypothèse, l'ensemble des différentes machines est connu. Donc, si N pannes sont supposées dans un circuit, celui qui va être testé est l'un des N+1 circuits (M_0, M_1 ..., M_N), en comptant le circuit correct M_0. Le but de cette méthode de test est de distinguer M_0 des autres.

2. Modèle de représentation des circuits séquentiels

Un système séquentiel est considéré ici comme une extension des circuits combinatoires. La figure II.1 en donne une représentation. Mais par rapport aux systèmes combinatoires apparaissent les difficultés suivantes :

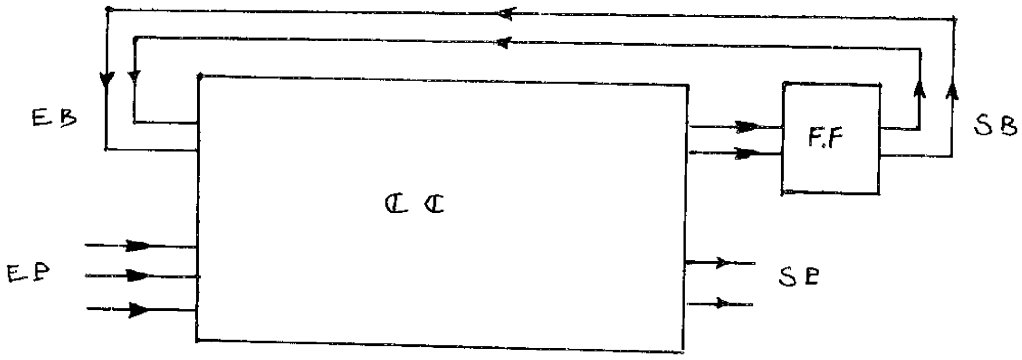
- impossibilité d'observer la partie SB des sorties. Par conséquent, l'effet d'une panne pourra s'observer sur les sorties primaires après plusieurs bouclages.

- Il est difficile de positionner les EB comme on le désire. Donc, même si on génère des entrées primaires qui détectent une panne, celles-ci peuvent ne pas être applicables avec succès si les EB n'ont pas la valeur adéquate.

Un circuit séquentiel est traité comme l'interconnexion d'une cascade de circuits combinatoires; la combinaison initiale des EB (EB_1) est connue et seul le comportement pendant les n tops d'horloge est considéré. Ceci est appelé un circuit séquentiel développé (figure II.2).

A la première impulsion d'horloge, l'état initial (EP_1, EB_1) étant connu, les sorties de CC_1 sont SB_1 et SP_1. A la seconde impulsion EB_2 prend les valeurs de SB_1 et ainsi de suite. Le circuit dans son ensemble est considéré comme combinatoire, de telle sorte que les techniques combinatoires peuvent s'appliquer, mis à part que EB_1 doit être donné.

La technique de sensibilisation des chemins est ici le D -Algorithme (17),(5),moyennant quelques modifications en vue de traiter le cas CC_1 (EB_1).



CC : circuit combinatoire
FF : Flips - Flops (éléments synchrones)
EP : Entrées primaires
SP : Sorties primaires
EB : Entrées de bouclage
SB : Sorties de bouclage

figure II.1

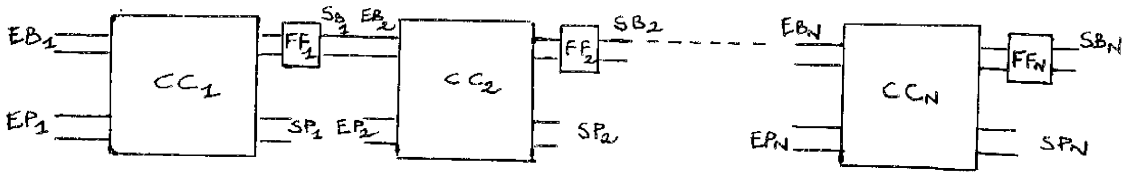


figure II.2

3. Procédure générale :

Elle utilise principalement trois sous-programmes :

. Le sous-programme A : cherche un chemin sensible se propageant à travers l'un des circuits CCI. Il essaie dans le cas où plusieurs chemins sensibles existant, d'en déterminer un permettant l'accès à une des sorties primaires. Si c'est impossible, il faudra allonger le circuit séquentiel développé et par là même la séquence.

. Le sous-programme B : propage un chemin sensible dans le circuit séquentiel développé en utilisant le sous-programme A pour chacun des CCi. Il permet de générer un test pour une panne donnée.

. Le sous-programme C : génère une séquence de transition (bridge séquence) faisant passer le système d'un état à un autre.

- Procédure générale

1. Le système est dans un état de départ donné (EB_1). Par convention $SB_1 = EB_1$. Aucune séquence n'ayant été générée, soit i = 0.

2. Si toutes les pannes ont été détectées (table des pannes vide), la procédure prend fin.

3. i = i +1. En partant de l'état présent du circuit ($SB_{(i-1)}$) on construit une séquence Ti détectant la première panne figurant dans la table.

4. Ti peut permettre de détecter d'autres pannes non encore traitées. Ceci est fait en utilisant un simulateur parallèle qui permet de connaître le fonctionnement du circuit soumis à chacune des pannes restantes, lorsqu'il est dans l'état EB_i et qu'on lui applique Ti.

Simulateur parallèle : étant donné un système séquentiel donnant naissance à (N+1) machines par la présence possible de n pannes connues, le simulateur parallèle consiste à donner en tout point du circuit, la valeur du signal pour chacune des pannes prévues.

5. Après Ti, l'état final est SB_i. S'il est identique pour chacune des pannes non encore détectées, on retourne en 2.

6. Si non i = i+1, et on construit une séquence Ti permettant de détecter la première de ces pannes. On revient en 4.

4. Résultats

Cette méthode programmée sur un calculateur NEAC - L3 de 8 K mots mémoire peut traiter des circuits comportant jusqu'à 250 interconnexions et 35 pannes (le nombre de pannes simulables sur un calculateur parallèle est égal au nombre de bits que comporte un mot machine) sélectionnées aléatoirement.

II.1.2. Logitest (15)

1. Hypothèses

- Les pannes envisagées sont du type ca 0 ou ca 1, simples.
- L'état initial du circuit peut être inconnu.

2. Procédure

a) Construction de la liste des tests élémentaires

Celle-ci est élaborée automatiquement à partir d'un algorithme semblable à celui des D cubes (17).

Les éléments composant le circuit sont des portes logiques (ET, OU, PAS, NI, ON) ou des bascules RS. Tout élément plus complexe sera décomposé de façon à retrouver des structures connues. Le test d'une bascule RS consiste à vérifier qu'elle peut conserver son état, ou changer d'état entre deux tops d'horloge. Les bascules jouent alors le rôle de mémoire pour le résultat d'un test.

b) Le test est constitué d'une succession d'affichages; entre chacun d'eux on ne conserve en mémoire que l'état des variables internes du circuit.

En un temps ti où tous les test n'ont pas été réalisés, les opérations principales se déroulent comme suit :

"α - Essayer de propager vers une sortie réelle (en priorité), ou une autre bascule, le signal de l'une des sorties pour toutes les bascules qui "couvrent" des tests initiaux.

β - Prendre un test initial, non encore réalisé, et essayer de le propager vers une sortie réelle (en priorité), ou une bascule.

γ - Si ce test est réalisable, le mettre dans la liste des tests "en instance", et dans la liste des tests couverts par la bascule utilisée pour la propagation.

δ - Revenir en β jusqu'à épuisement des tests initiaux" (extrait de 15).

Cette procédure se poursuit à chaque séquence t_1 ..... t_n jusqu'à ce que tous les tests soient réalisés.

Le programme Logitest 4 occupe 15500 mots dont 5000 pour les instructions du programme. Il est écrit en FORTRAN IV et permet de traiter des circuits comprenant jusqu'à 3000 opérateurs combinatoires et 100 bascules simples.

II-2 METHODE DE PARTITIONNEMENT DE LA TABLE DES PANNES (12-13)

Cette méthode a pour but la localisation des pannes dans les systèmes séquentiels asynchrones.

II.2.1. Hypothèses

- Panne unique
- La table des pannes est supposée connue
- Il est toujours possible de conduire le circuit dans un état interne connu quelle que soit la panne qui l'affecte.

II.2.2. Principe de la méthode

La table des pannes est constituée de n pannes P_1 ... P_n, représentées par n machines M_1 ... M_n, M_0 désignant le circuit correct.

Pour chaque vecteur d'entrée affiché, le circuit va donner une réponse pouvant être différente selon qu'il est représentable par telle ou telle machine M_0 ... M_n, et permettre d'effectuer un partitionnement des n+1 machines.

La méthode consiste alors à construire un arbre permettant de partitionner au mieux l'ensemble des machines c'est-à-dire de les distinguer les unes des autres.

A chaque sommet i du graphe sont associés :

- un ensemble E de machines (M_{i_1} ... M_{i_T})
- un vecteur d'entrée.

En simulant en parallèle le comportement des machines de l'ensemble E, on peut déterminer l'affichage qui procure la meilleure partition de cet ensemble.

Entre deux affichages successifs, une seule variable d'entrée peut changer. Cette restriction est due au fait que la méthode est destinée à effectuer le diagnostic des systèmes séquentiels asynchrones.

Le programme, écrit en langage machine sur calculateur CDC 1604 comporte 10000 instructions.

Les circuits traités peuvent posséder :

- 300 opérateurs
- 96 entrées primaires
- 96 sorties primaires
- 1000 pannes.

II.2.3. Améliorations (19)

La méthode précédente a été améliorée moyennant les hypothèses suivantes :

HYPOTHESES :

- Seuls sont considérés les circuits synchrones, et le comportement du circuit est supposé toujours parfaitement défini. (Hypothèse II.2.1)

- Les pannes prévues sont des collages à 0 ou à 1 de connexions simples.

- On suppose qu'il est possible d'amener le circuit dans un état interne connu quelle que soit la panne, par une combinaison des valeurs d'entrées primaires.

- Les tests élaborés ne concernent que la détection des pannes.

STRUCTURE GENERALE DU PROGRAMME :

A. Elimination des pannes indétectables

B. Recherche des combinaisons d'entrées permettant d'imposer un état connu au système et des pannes qu'elles détectent.

C. Utilisation de la méthode de partitionnement de la table des pannes en tenant compte de l'influence des pannes sur l'état interne du circuit.

D. Recherche aléatoire utilisant le tableau d'états.

E. Utilisation du tableau d'états pour réduire la longueur de la séquence obtenue en D.

II - 3 METHODE UTILISANT LA TABLE DE FLUENCE D'UN SYSTEME SEQUENTIEL (6-7-8)

La structure du circuit n'est pas supposé connue. La méthode consiste à vérifier que le circuit opère en accord avec la table de fluence qui le décrit.

Une table de fluence représente à la fois les matrices d'excitations et de sortie d'un système. Sa constitution est identique mais dans les cases, on indique le numéro de l'état atteint (équivalent décimal de la combinaison binaire des variables internes) et la valeur des sorties. La méthode générale (6) fait appel aux définitions suivantes :

Définitions :

1. une séquence de synchronisation est une séquence d'entrée dont l'application garantit de conduire le circuit dans un certain état final sans connaître l'état initial du circuit.

2. Une séquence propre (Homing séquence) est telle que son application rend possible la détermination de l'état final du circuit par l'observation de la séquence de sortie résultante.

Tout circuit séquentiel réduit possède une séquence propre.

3. Une séquence de distinction est une séquence d'entrée dont l'application permet de connaître l'état initial du circuit par l'observation de la séquence de sortie qu'il produit.

Nous citerons un cas particulier de cette méthode [8] qui nécessite les hypothèses ci-dessous :

HYPOTHESES

1. La table de fluence est fortement connexe
2. Elle est réduite (pas d'états équivalents)
3. On connait une séquence de distinction
4. Toutes les pannes affectent le circuit de façon permanente
5. Aucune d'entre elles n'augmente le nombre d'états du circuit.

Nous ne décrirons pas de façon précise cette méthode, sachons seulement qu'elle a pour but de vérifier CHACUNE des cases de la table de fluence, par une sous séquence, la séquence complète de test étant un assemblage de ces sous séquences.

Cette méthode est intéressante en ce sens qu'elle utilise directement une table caractéristique des systèmes séquentiels, sans chercher à adapter des méthodes conçues pour le combinatoire.

De plus, aucune restriction n'a été faite sur la nature des pannes traitables.

Cependant, les séquences de test obtenues sont assez longues et beaucoup de circuits ne peuvent satisfaire à toutes les hypothèses données.

II-4 METHODE DONNANT DES SEQUENCES DE TEST OPTIMALES POUR LES SYSTEMES SEQUENTIELS (11)

Elle utilise aussi la table de fluence d'une machine, mais constitue une extension de méthodes élaborées pour les systèmes combinatoires (9 - 10)

II.4.1. Définitions et hypothèses

II.4.1.1. Classes de machines

- Chaque machine possède une seule sortie binaire (0 ou 1)
- Chaque machine peut être dans n'importe lequel d'un nombre fixé d'états internes : q_λ , q_2 ... q_r, $1 \leqslant r \leqslant 2^P$

P : nombre de mémoires binaires dans le circuit

- Il existe une entrée de remise à zéro (RESET) qui force la machine dans un état unique q_λ .
- Chaque machine est déterministe : l'état interne présent et le symbole d'entrée présent sont suffisants pour déterminer de façon unique l'état interne suivant et la sortie présente.
- Les opérations s'effectuent de façon synchrone. Les entrées apparaissent à des instants discrets dans le temps.

II.4.1.2. Types de pannes

On considère seulement un ensemble limité de pannes permanentes.

Soit To la table de fluence décrivant la machine correcte, Mo. Une panne permanente transforme Mo en la machine Mi, décrite par une table Ti. Si To et T_i ne sont pas équivalentes, la faute, cause de la transformation, est détectable. L'hypothèse d'une panne permanente implique que T_i continue à décrire la machine défectueuse Mi pendant toute la durée du test.

II.4.1.3. Hypothèses

- Le signal RESET force TOUTES les m+1 machines Mo ... Mm dans le même état initial q_λ ,supposition restrictive puisque elle impose qu'aucune

faute n'existe sur le circuit RESET.

- Le nombre d'états r est borné par 2^p. Si une faute i transforme Mo en Mi, on suppose que 2^p est aussi une borne supérieure du nombre d'états de Mi.

II.4.1.4. Nature des tests

Ils consistent en une séquence de symboles d'entrée précédés d'un signal RESET. Une fois celui-ci appliqué, la machine sous test doit être dans l'état q_λ . La réponse normale de la séquence de test est alors connue. Si la séquence de sortie observée est différente de la réponse normale, on sait qu'une faute existe.

II.4.2. Tests de détection de pannes

Donnons ci-dessous la procédure utilisée :

1. Le circuit est analysé pour obtenir (m+1) tables de fluence To ... Tm, caractéristiques des machines Mo ... Mm.

2. Les ensembles de séquences qui détectent chaque faute sont déterminés en comparant les tables T_1 ... T_m, avec To. Cette comparaison s'effectue en formant le produit des tables T_1 .... T_m avec To, par l'emploi des règles de calcul des matrices booléennes (11). La construction de ces m ensembles est analogue à la construction des colonnes d'une table de fautes dans les circuits combinatoires (9 - 10).

3. On applique ensuite l'équivalent de la simplification par colonne dominante (9 - 10) en supprimant ceux des ensembles de séquences qui contiennent (ou dominent les autres).

4. Une séquence optimale est détectée comme étant la plus courte contenue dans tous les ensembles restants.

Cette méthode possède l'inconvénient de supposer que toutes les machines en panne peuvent être positionnées dans un état de départ unique. L'ensem-

ble des pannes possibles est également limité et si l'établissement des tables de fluence se fait à partir des équations logiques du circuit, transformées par la présence de la panne pour les tables T_1 ... T_m, ce sont des collages à 0 et à 1 de connexions que l'on étudiera.

II-5 - CONCLUSIONS

Si l'on reprend pour chacune des méthodes citées, les hypothèses et les résultats obtenus, on peut en déduire quelques constatations.

En règle générale :

- l'hypothèse de la panne unique permanente est admise
- les circuits testables sont de nature synchrone (sauf 12 -13)
- l'état de départ est supposé connu (sauf 15)
- de plus toutes ces méthodes (sauf 6 - 7 - 8) supposent au départ que le circuit est en panne, puisqu'elles consistent à rendre sensible un itinéraire aux effets d'une panne, ou à donner les tables transformées.

Les méthodes de sensibilisation des itinéraires supposent que les pannes résultant de collage à 0 ou à 1 d'interconnexions. Les autres, par contre, de par l'emploi de tables, peuvent admettre des types de pannes différents.

Les méthodes utilisant la sensibilisation des chemins ont jusqu'à présent reçu le plus d'applications du point de vue de leur programmation. Elles conviennent bien pour tester des systèmes contenant de grands blocs combinatoires. Les méthodes de partionnement de la table des pannes traitant mieux les circuits comportant de nombreux bouclages.

On peut remarquer enfin que la plupart de ces méthodes effectuent la détection de panne et non le diagnostic (sauf 12 - 13)

Les méthodes de sensibilisation des itinéraires, pour aussi intéressantes qu'elles se montrent dans leurs résultats ne semblent pas donner un moyen d'ouverture vers des études théoriques générales. Il en est de même des autres méthodes décrites qui sont avant tout destinées à trouver une application immédiate. Cette remarque ne constitue en aucun cas une critique mais elle justifie

le fait que dans ce mémoire nous n'avons pas voulu reprendre ces méthodes pour essayer de les adapter à tel ou tel problème, mais plutôt cherché à avancer d'autres idées et dans la mesure du possible à les appliquer et les justifier sur des exemples concrets, c'est-à-dire sur des circuits qu'utilise le technologue.

C H A P I T R E III

ANALYSE DES SYSTEMES SEQUENTIELS

ANALYSE DES SYSTEMES SEQUENTIELS

Nous pouvons tirer un enseignement supplémentaire des méthodes exposées au chapitre II.

Les unes nécessitent seulement la connaissance de la topologie (du schéma logique) du système. Parmi les autres, utilisant des tables de fluence, certaines peuvent même ignorer le schéma du circuit à condition que soit connu son fonctionnement.

En celà, les spécifications habituelles données par les constructeurs, bien que satisfaisantes pour l'utilisation normale, se révèlent parfois insuffisantes lorsqu'il s'agit de saisir le fonctionnement intime de l'élément à étudier. On ne trouve en effet pratiquement jamais les équations logiques (variables secondaires et sorties) ou les tables (excitation et sortie, table de fluence, matrice des transitions) seules capables de le représenter finement.

Dans la mesure où l'on veut utiliser ou essayer de proposer des méthodes nécessitant la connaissance de ces données, on s'aperçoit qu'il existe là une lacune.

Pour cette raison, il nous a semblé indispensable, avant toute étude relative au test des systèmes séquentiels, de créer un outil capable de les analyser et de nous fournir toutes les données désirées pour une investigation future. Cet outil, s'appuyant sur une méthode classique d'analyse constitue un programme qui nécessite la connaissance du schéma logique du circuit à étudier.

Ce chapitre présentera donc successivement :

- la méthode employée
- les algorithmes constitutifs du programme
- la façon d'utiliser ce programme et les résultats qu'on peut en attendre.

III-1 METHODE D'ANALYSE

Nous avons défini la nature séquentielle d'une machine par l'existence de branches de contre-réaction, ce qui lui permet d'agir non seulement en fonction des informations qui apparaissent à ses entrées mais aussi en se "rappelant" l'état antérieur dans lequel elle se trouvait.

L'analyse d'un tel système consiste, à partir de sa description topologique (nature et interconnexion des modules), à trouver un ensemble d'équations booléennes (variables secondaires et sorties) capable de décrire son comportement réel.

$$Y\ (T+1) = f_1\ (Y(T),\ E\ (T)) \qquad (1)$$

$$Q\ (T) \quad = f_2\ (Y(T),\ E\ (T)) \qquad (2)$$

L'équation (1) peut se décomposer en autant d'équations qu'il y a de variables secondaires :

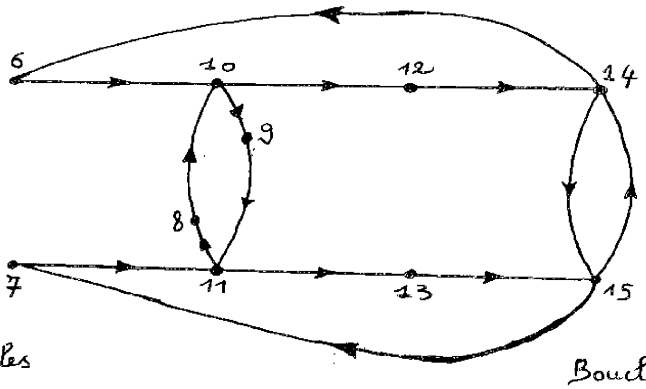
$$y_i\ (T+1) = f_1 \left(\sum_j y_j\ (T), \sum_k e_k (T) \right)$$

Cette dernière est représentable par un réseau combinatoire ayant pour entrées les $y_j(T)$ et les $e_k(T)$ et pour sortie $y_i(T+1)$.

On va donc chercher à découper une machine séquentielle en un certain nombre de circuits combinatoires, et pour cela à supprimer les bouclages. L'équation (2) est purement combinatoire puisqu'aucune variable secondaire ne se retrouve de part et d'autre de l'égalité.

III.1.1 Recherche des boucles et des boucles fondamentales

Reprenons le schéma logique de la figure I.3 en remplaçant les modules par des noeuds et les interconnexions par des branches orientées. Supprimons également les entrées primaires qui n'apportent aucune information quant à la caractéristique séquentielle du circuit. Nous obtenons un graphe où apparaissent très clairement les bouclages (fig. III.1.a)



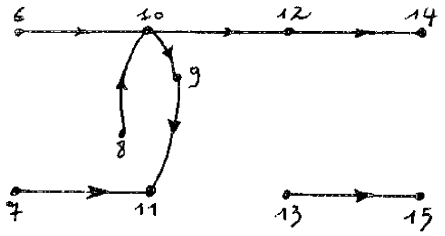
Boucles

6 14 12 10
6 14 15 13 11 9 10
7 15 13 11
7 15 14 12 10 8 11
8 11 9 10
14 15

Boucles fondamentales

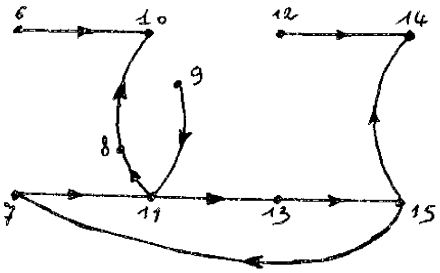
6 14 12 10
7 15 13 11
8 11 9 10
14 15

(a)



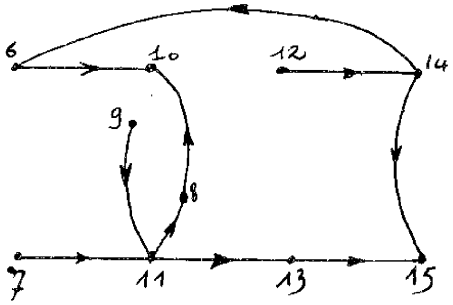
(b)

(11, 14, 15)



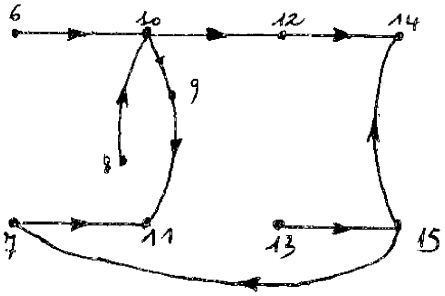
(c)

(10, 14)



(d)

(10, 15)



(e)

(11, 14)

figure III.1

Définition :

Une boucle est un chemin fermé passant par plusieurs noeuds.

La méthode consiste en un premier temps à dénombrer toutes les boucles du graphe.

Nous en obtenons ici six. Cependant on peut remarquer que deux d'entre elles :

$$\begin{cases} 6 & 14 & 15 & 13 & 11 & 9 & 10 \\ 7 & 15 & 14 & 12 & 10 & 8 & 11 \end{cases}$$

en contiennent une autre : 14, 15, plus petite.

On dira qu'elles ne sont pas fondamentales, en s'appuyant sur la définition suivante :

Définition :

Une boucle est fondamentale si aucun sous ensemble de ses opérateurs ne constitue lui même une boucle.

Il y a autant de boucles de contre réaction à ouvrir que de boucles fondamentales (une par boucle) pour rendre le circuit combinatoire, puisque l'ouverture des boucles fondamentales implique l'ouverture de celles qui ne le sont pas. Ceci constitue la deuxième partie de cette méthode d'analyse.

On obtient ici quatre boucles fondamentales (figure III.1. ⓐ)

III.1.2. Recherche des variables secondaires en nombre minimum

Si l'on se place maintenant en un ou plusieurs noeuds du graphe, on peut à un instant donné (puisque les signaux en ces noeuds sont connus par l'observation qu'on effectue sur eux), ouvrir les branches qui partent de ces noeuds en assignant aux noeuds sur lesquels elles arrivent, la valeur logique des signaux au moment de l'ouverture.

A partir de cette constatation, nous pouvons déterminer certains ensembles de noeuds tels que leur observation garantit l'ouverture de toutes les boucles fondamentales du circuit et "fige" le système en le rendant combinatoire entre deux changements successifs du vecteur d'entrée. Il existera de multiples solutions pour un graphe donné et il importe bien sûr d'effectuer un choix judicieux en cherchant celles qui nécessitent l'observation simultanée du minimum de noeuds; nous obtiendrons ainsi un nombre minimum de variables secondaires.

Si, sur le graphe de la figure III.1 (a), on observe les noeuds (11, 14, 15), le système devient effectivement combinatoire, mais les boucles 14 15 et 7 11 13 15 ont disparu (figure III.1. (b)), alors qu'il suffisait de les ouvrir. Il est donc possible qu'existe un ensemble plus petit. L'ensemble (10, 14) (figure III.1 (c)) laisse une boucle fondamentale fermée - solution non satisfaisante. Par contre, en observant les noeuds (10, 15) ou (11, 14) (fig. III.1. (d) et (e)), toutes les boucles fondamentales sont ouvertes et ces deux solutions constituent les ensembles minimaux.

La détermination de ces ensembles minimaux consiste à traiter un problème de recouvrement identique à la méthode de QUINE MC. CLUSKEY. Il suffit de former un tableau où sont portés, dans la première ligne, les numéros des noeuds, en tête des lignes suivantes, les boucles fondamentales, et dans le reste de chacune des lignes, les numéros des noeuds qui forment les boucles, au dessous des numéros correspondants de la première ligne. A partir de ce tableau (fig. III.2) on détermine facilement les ensembles de longueur minimum recouvrant toutes les boucles.

	6	7	8	9	10	11	12	13	14	15
6 14 12 10	X				(X)		X		[X]	
7 15 13 11		X				[X		X		(X)
8 11 9 10			X	X	(X)	X]				
14 15									[X]	(X)

Fig. III.2

on retrouve les deux ensembles minimaux : 10 15 ○
11 14 □

On peut aussi résumer ceci par l'équation booléenne suivante :

③ $(6+14+12+10) \cdot (7+15+13+11) \cdot (8+11+9+10) \cdot (14+15) = 1$

qui signifie : l'observation d'un des modules de la première boucle, ET, d'un des modules de la deuxième boucle, ET, ..., permet d'ouvrir toutes les boucles.

Parmi tous les ensembles obtenus, on ne retiendra que ceux mportant le minimum d'opérateurs.

On obtient ainsi les noeuds, ou, en revenant au circuit logique initial, les numéros de modules dont les sorties représenteront les variables secondaires du circuit.

III.1.3. Etablissement des équations

Nous abordons la dernière phase de l'analyse : reconstituer les circuits combinatoires satisfaisant aux équations ① et ② , c'est-à-dire, en partant des sorties des modules définies comme étant les variables secondaires, remonter les interconnexions jusqu'à ne rencontrer que des variables secondaires ou des entrées (fig. III.3.ⓐ et ⓑ).

Une fois les équations établies, nous connaissons parfaitement le fonctionnement du circuit et pouvons tracer les tables qui en résultent (fig. I.3)

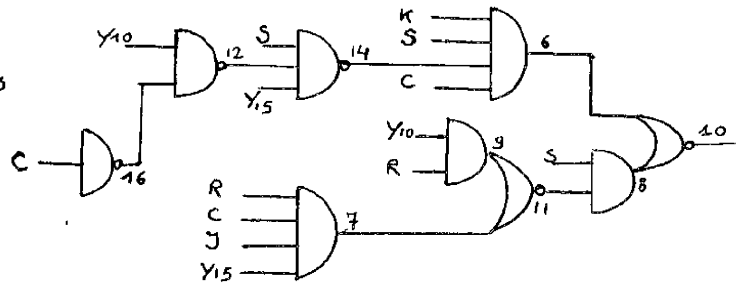
Ici se posent cependant deux problèmes :

1. Validité de la recherche du nombre minimum de variables secondaires,

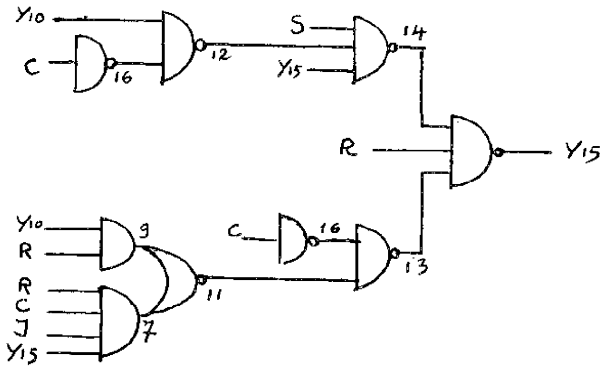
2. Problème du choix des variables secondaires.

1. Il peut arriver en effet que la synthèse d'un circuit ait été effectuée avec un certain nombre de variables secondaires et que l'Analyse par cette méthode conduise à le représenter par un ensemble de variables internes plus petit. Ceci n'est pas gênant en soi, mais ce qui l'est par contre, c'est que l'on aboutisse à un fonctionnement différent du circuit.

(a) variables secondaires

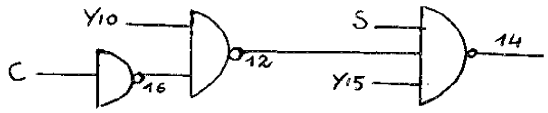


$$Y_{10(T+1)} = \left(\overline{K}R + \overline{C}R + RY_{15(T)}\right).Y_{10(T)} + \overline{S} + JCR\,Y_{15(T)}$$



$$Y_{15(T+1)} = \left(S.\overline{Y}_{10(T)} + CS\right).Y_{15(T)} + \overline{R} + \overline{C}.\overline{Y}_{10(T)}$$

(b) Sorties



$$Q_{14(T)} = \overline{S} + \overline{Y}_{15(T)} + \overline{C}.Y_{10(T)}$$

$$Q_{15(T)} = Y_{15(T)}$$

figure III.3

Cette remarque nous a été faite par Monsieur DAVID (Laboratoire d'Automatique de Grenoble) et l'exemple de la figure III.A qu'il nous a indiqué, illustre bien ce fait.

2. La deuxième remarque amène à la même conclusion : le fait de choisir un ensemble de variables secondaires plutôt qu'un autre, peut avoir pour conséquence que le fonctionnement ainsi obtenu ne soit pas identique au fonctionnement réel du système.

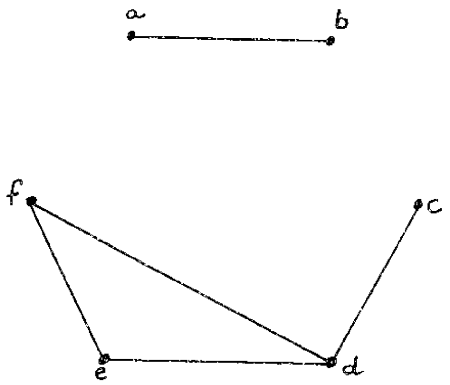
L'explication de ce phénomène est la suivante : choisir par l'analyse des variables secondaires différentes de celles utilisées pour sa synthèse, revient à imposer au circuit des retards en des endroits différents de ceux où ils existent réellement. Il peut en résulter l'apparition d'aléas de propagation ayant pour effet l'obtention d'un comportement ne correspondant pas à celui observé physiquement ((2) - Tome I page 382-386).

En ce qui nous concerne, nous allons utiliser constamment des systèmes composés d'éléments mémoire RS. Chacun d'eux, formera dans le graphe général une boucle fondamentale, et à coup sûr, un des deux noeuds qui lui sont propres, une variable secondaire, ce qui calquera bien le comportement physique du système.

Dans le cas général, si aucune information relative au fonctionnement du circuit ou à la façon dont a été effectuée la synthèse n'est connue, il devient très difficile d'assurer que tel ou tel ensemble de variables secondaires représente de façon satisfaisante le circuit.

- Tableau donné

e_1 e_2	00	01	11	10	S_1	S_2
	(a)	b	-	d	0	0
	a	(b)	c	-	0	0
	e	b	(c)	d	0	1
	e	-	c	(d)	0	1
	(e)	f	-	d	1	1
	e	(f)	c	-	1	1



- Synthèse

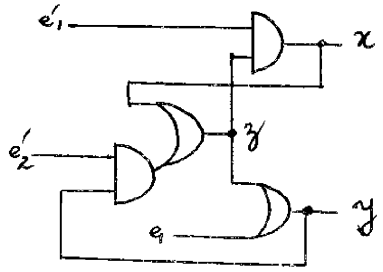
e_1e_2

XY	00	01	11	10	00	01	11	10	00	01	11	10	00	01	11	10
00	(a)	(b)	c	d	(00)	(00)	01	01	0	0	0	0	0	0	1	1
01	e	b	(c)	(d)	11	00	(01)	(01)	1	0	0	0	1	0	1	1
11	(e)	(f)	c	d	(11)	(11)	01	01	1	1	0	0	1	1	1	1
10	-	-	-	-	--	--	--	--	-	-	-	-	-	-	-	-
					xy				x				y			

variables internes

$$\begin{cases} x = Xe'_1 + e'_1e'_2Y = (X + e'_2Y)\cdot e'_1 \\ y = e_1 + X + Ye'_2 = (X + e'_2Y) + e_1 \end{cases}$$

- Circuit



- Analyse : si on ouvre en z : $z = Ze'_1 + (Z + e_1)e'_2$

Z \ e_1e_2	00	01	11	10	00	01	11	10
0	(0)	(0)	(0)	1	(a)	(b)	(c)	d
1	(1)	(1)	0	(1)	(e)	(f)	c	(d)

z

Depuis l'état (c), le passage de e_1 et e_2 à 0 amène dans l'état (a) Les sorties passent de 01 à 00 - Sur le circuit réel, les sorties passent de 01 à 11.

Figure III.A

III-2 ALGORITHMES

Avant d'expliquer les algorithmes mis en oeuvre dans le but de programmer la méthode d'analyse décrite ci-dessus, nous devons nous pencher un instant sur un programme élaboré au L.A.A.S., destiné à calculer les fonctions booléennes de sortie d'un système combinatoire, par l'emploi de sous programmes effectuant toutes les opérations de l'Algèbre de Boole : réunion, intersection, complémentation, simplification (programme inclus dans [20]).

III.2.1. Programme d'analyse des systèmes combinatoires et de calcul booléen

Le circuit, composé de modules logiques classiques (ET, OU, PAS, NI, ON, OU EXCLUSIF), est décrit sous la forme d'une matrice topologique (fig. III.4.ⓑ). On numérote les sorties de modules, après avoir fait de même avec les entrées, en allant vers la sortie.

La première colonne contient les numéros des sorties des modules.

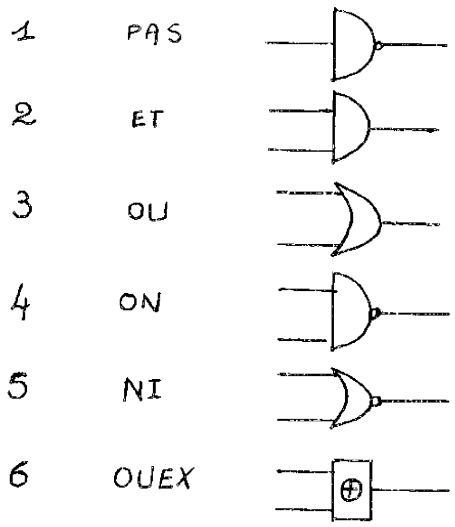
La deuxième colonne de la matrice indique la nature des modules à l'aide du codage donné par la fig. III.4.ⓐ .

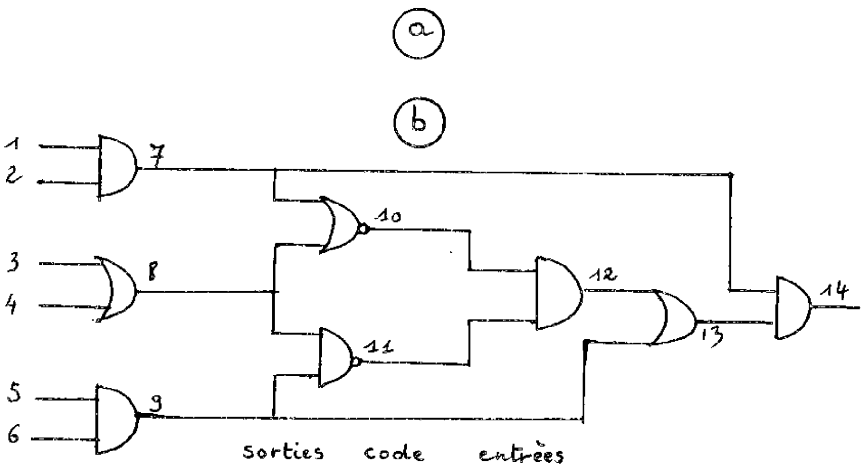
Dans les colonnes suivantes, sont inscrites les entrées de chacun des modules.

III.2.1.1. Algorithme

Le calcul s'effectue successivement sur chacune des lignes de la matrice, en commençant par la première. Selon le numéro de code, le sous programme correspondant est appelé. Il calcule la fonction booléenne à la sortie du module, compte tenu de ses entrées, en appelant à son tour des programmes de simplification dont le but essentiel est de ne pas stocker des expressions trop importantes en mémoire (compromis rapidité-taille). Après avoir enregistré le résultat en mémoire avec repérage du début et de la fin de l'enregistrement, le calcul continue en séquence sur une autre ligne.

CODAGE





sorties	code	entrées	
7	2	1	2
8	3	3	4
9	2	5	6
10	5	7	8
11	4	8	9
12	2	10	11
13	3	9	12
14	2	7	13

figure III.4

Lorsque les numéros figurant les entrées d'un module sont différents de ceux des entrées primaires, il faut que les fonctions relatives à ces numéros aient déjà été traitées et se trouvent stockées en mémoire, raison pour laquelle la numérotation du circuit doit se faire par ordre croissant des entrées vers la sortie. Il est ainsi possible de connaître la fonction logique réalisée en n'importe quel point du circuit.

III.2.1.2. Codage

Comme nous désirons obtenir des équations logiques et non des valeurs 0 ou 1 en certains points du circuit, nous ne pouvons pas utiliser de table de vérité et il devient nécessaire de choisir un codage particulier qui simule une expression booléenne par un nombre entier. En utilisant exclusivement la forme réunion de monômes, un mot machine peut représenter un monôme, et une liste de mots (une matrice), une réunion de monômes.

Par souci d'économie en place mémoire, il est intéressant qu'un seul mot machine puisse figurer un monôme comportant le plus grand nombre possible de variables, celles-ci seront repérées dans le mot par leur position et leur valeur en codant respectivement :

- La variable sous forme normale par 1 (sans présumer de
- La variable sous forme complémentée par 0 leur valeur effective)
- L'absence d'une variable par 3 (Ø)

Ainsi, 1 3 3 0 1 3 0 3 1 signifie $x_1 \overline{x}_4 x_5 \overline{x}_7 x_9$

Pour une machine dont le mot mémoire comporte 36 bits (IBM 7044) le plus grand entier (ayant éventuellement un 3 comme premier chiffre) est un nombre de 10 chiffres. 32 bits (C.I.I. 10070) donnent un entier de 9 chiffres, soit N = 9 variables dans un mot.

Si la fonction met en jeu n variables, il faudra $\frac{n}{N} + 1$ mots pour chaque monôme.

Exemple :

$x_3\ x_{12}\ \overline{x_{27}}$		3 3 1 3 3 3 3 3 3	3 3 1 3 3 3 3 3 3	3 3 3 3 3 3 3 3 0
$+\ \overline{x_1}\ \overline{x_9}\ x_{14}$	$\equiv$	1 3 3 3 3 3 3 3 0	3 3 3 3 1 3 3 3 3	3 3 3 3 3 3 3 3 3
$+\ x_{13}\ \overline{x_{18}}$		3 3 3 3 3 3 3 3 3	3 3 3 1 3 3 3 3 0	3 3 3 3 3 3 3 3 3

une fonction booléenne sera donc représentée par une matrice à deux indices M (I, J)

$I = 1,\ N$: nombre de monômes

$J = 1,\ NMAX = \frac{n}{N} + 1$

III.2.1.3. Sous programmes élémentaires

a) Intersection entre 2 monômes

Elle se réduit à une somme entre deux nombres arithmétiques suivie d'un décodage.

Ecrivons toutes les combinaisons que l'on peut obtenir :

$\bar{x}\ \emptyset$		$x.\bar{x}$		$x.\ x$		$\bar{x}.\bar{x}$		$x\ \emptyset$		$\emptyset.\emptyset$	
↓	0 +3 = 3	↓	0 +1 = 1	↓	1 +1 = 2	↓	0 +0 = 0	↓	1 +3 = 4	↓	3 +3 = 6
$\bar{x}$	0	$\cap$ nulle		x	1	$\bar{x}$	0	x	1	$\emptyset$	3

Décodage : $\begin{cases}3\\0\end{cases} \longrightarrow 0$ $\quad \begin{cases}4\\2\end{cases} \longrightarrow 1$ $\quad 6 \longrightarrow 3$ $\quad 1 \longrightarrow$ intersection nulle.

Exemples :

$$\begin{array}{ll} & \overline{x_2}\ x_3 \\ + & \overline{x_1}\ x_3 \\ \hline & \overline{x_1}\ \overline{x_2}\ x_3 \end{array} \qquad \begin{array}{lr} & 3\ 0\ 1 \\ + & 0\ 3\ 1 \\ \hline & 3\ 3\ 2 \\ \downarrow & 0\ 0\ 1 \end{array}$$

$$\begin{array}{ll} & \overline{x_2}\ x_3 \\ + & x_2 \\ \hline & \text{intersection nulle} \end{array} \qquad \begin{array}{lr} & 3\ 0\ 1 \\ & 3\ 1\ 3 \\ \hline & 6\ 1\ 4 \\ \downarrow & 3\ \times\ 1 \end{array}$$

b) Complémentation

La complémentation d'un monôme produit autant de mots que de variables qui y apparaissent, en inversant leur valeur.

Ex. : $\overline{x}_2\ x_4\ x_5$ (30 3 1 1) ⟶ (3 1 3 3 3 / 3 3 3 0 3 / 3 3 3 3 0) : $x_2 + \overline{x}_4 + \overline{x}_5$

c) Réunion

Il suffit de placer les deux listes à réunir l'une après l'autre. A l'aide de ces trois sous programmes élémentaires, peuvent être construits tous les autres, ET, OU, NI, ON, ... entre plusieurs réunions de monômes.

Nous utiliserons tous ces sous programmes dans le programme d'analyse dont nous allons expliquer les algorithmes.

III.2.2. Algorithmes

La planche III.1 représente les différentes parties du programme d'analyse. Reprenons-les une à une afin d'expliciter leur rôle.

III.2.2.1. La description du circuit à analyser (fig. III.5. ⓐ) s'effectue également par une matrice topologique en utilisant les mêmes codages. Seule différence : sur une ligne, les entrées d'un module peuvent se traduire par des numéros supérieurs au numéro du module lui-même, mettant en évidence l'existence d'un bouclage (fig. III.5. ⓒ).

III.2.2.2. Passage du circuit au graphe

Cette matrice voit sa deuxième colonne supprimée ainsi que dans chaque ligne les numéros correspondant aux entrées primaires. On obtient alors une matrice (fig. III.5. ⓓ) représentative du graphe du circuit (fig. III.5. ⓑ), à partir de laquelle va s'effectuer la recherche des boucles.

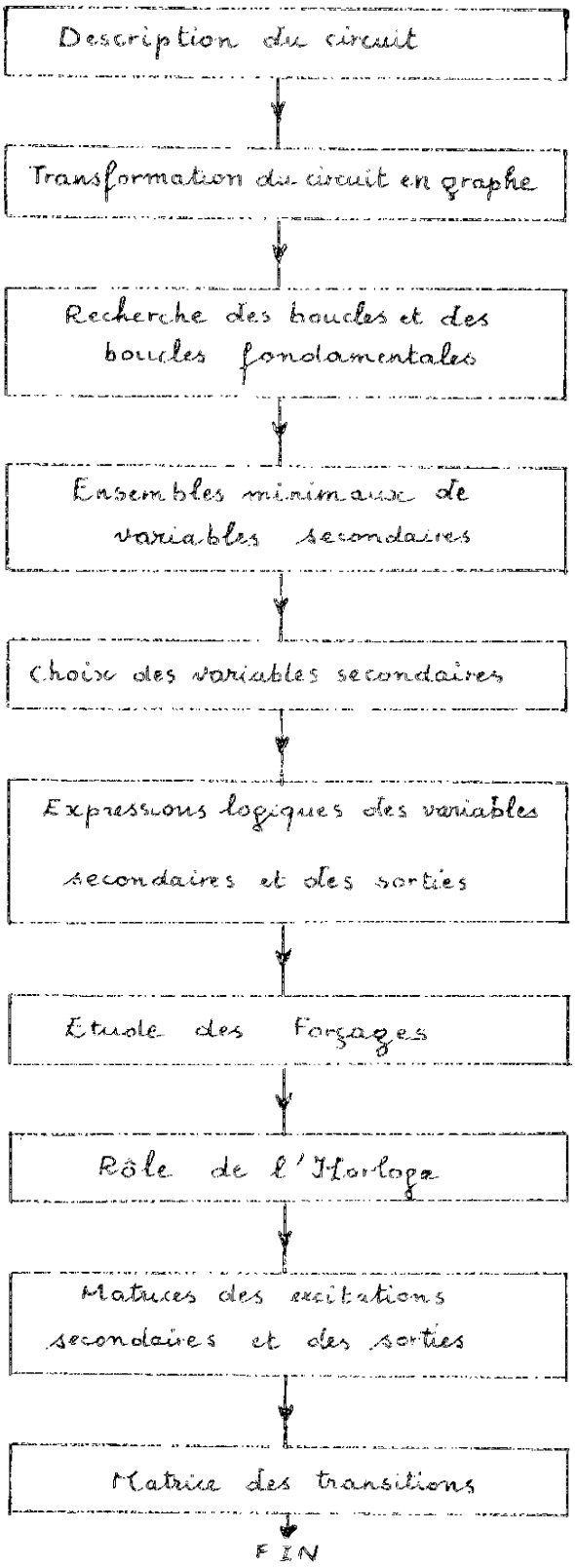
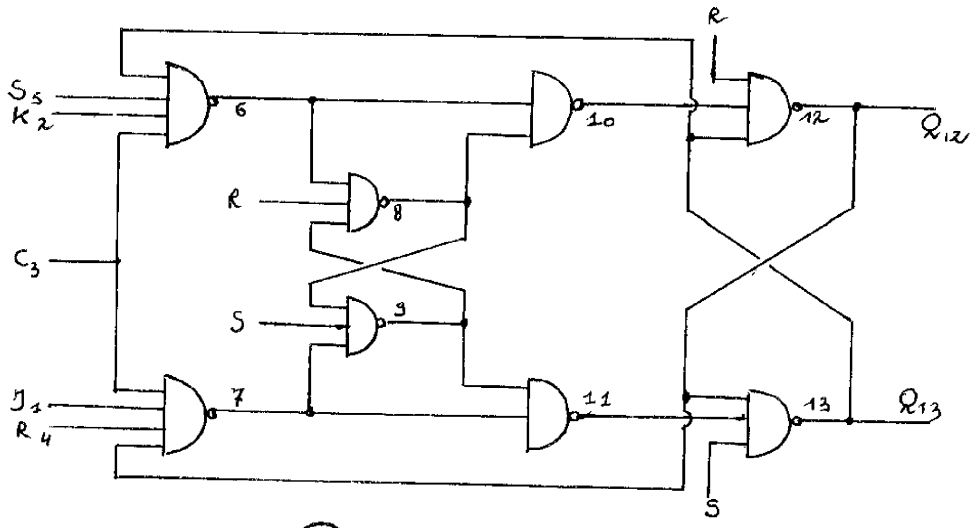
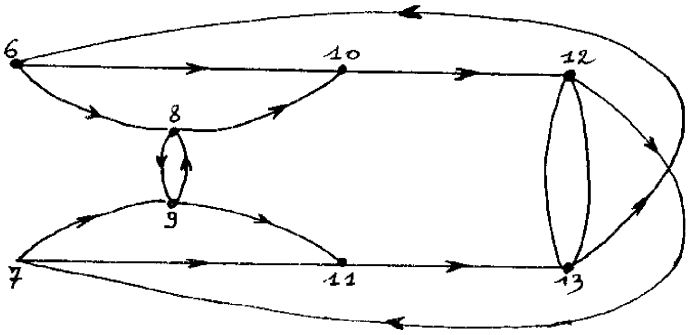


Planche III . 1



(a) circuit



(b) graphe

(c.) matrice topologique

6	4	2	3	5	13
7	4	1	3	4	12
8	4	4	6	9	0
9	4	5	7	8	0
10	4	6	8	0	0
11	4	7	9	0	0
12	4	4	10	13	0
13	4	5	11	12	0

(d) matrice du graphe

6	13	
7	12	
8	6	9
9	7	8
10	6	8
11	7	9
12	10	13
13	11	12

BASCULE JK-MS SN 74H72 (Texas Instruments)

figure III.5

III.2.2.3. Recherche des boucles et des boucles fondementales

BOUCLES :

Nous allons systématiquement partir de chacun des noeuds du graphe et détecter toutes les boucles passant par ces noeuds en supprimant à chaque fois celles déjà trouvées.

Aidons-nous de l'exemple de la fig. III.5. La première colonne de la matrice représentative du graphe contient tous les noeuds de ce graphe. Dans la première ligne figure le noeud numéro 6. Sur lui arrive une branche provenant du noeud numéro 13. Nous allons remonter cette branche en notant qu'aucune autre n'arrive en 6.

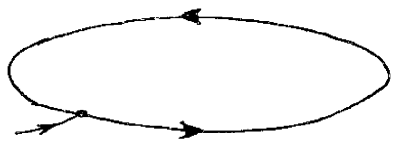
On poursuit le même cheminement depuis le noeud 13, en construisant ainsi un arbre (figure III.6). Il n'est suivi qu'un seul chemin à la fois mais à chaque bifurcation en un noeud donné, il faut noter que toutes les branches qui y arrivent n'ont pas été explorées.

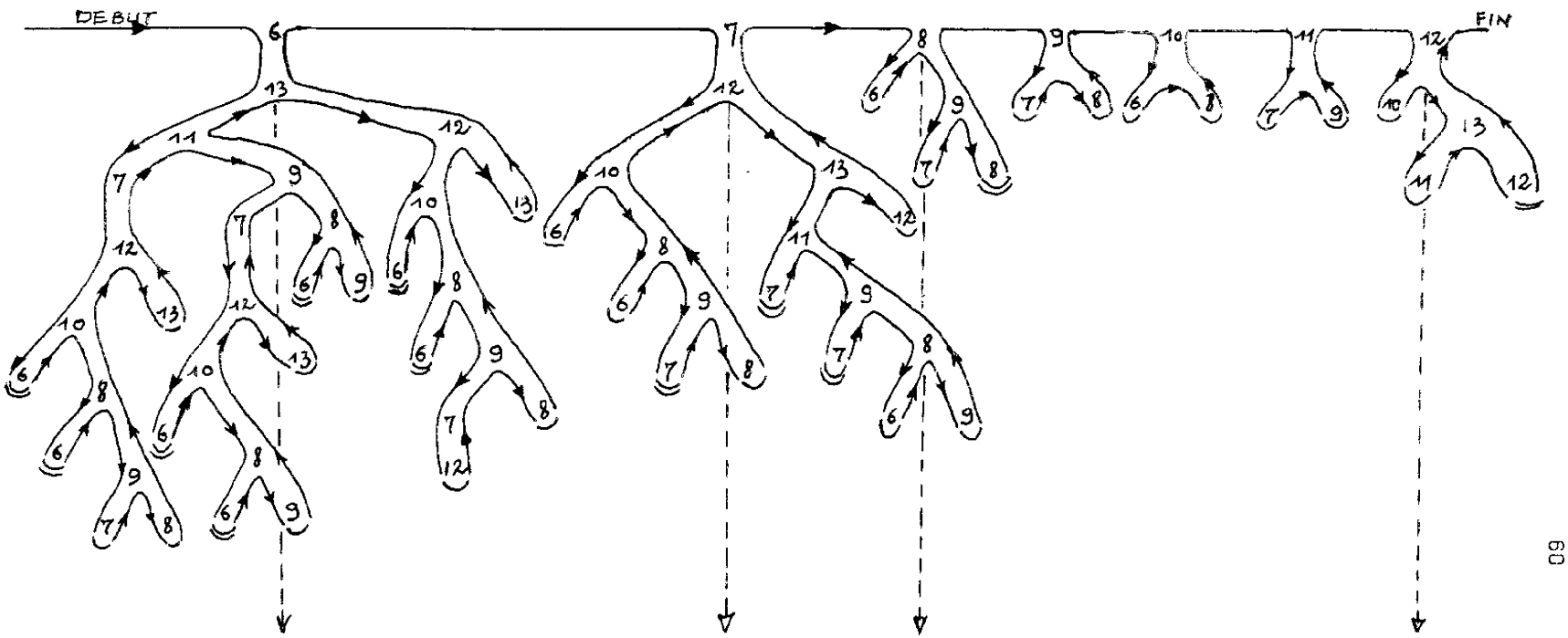
PREMIER TEST D'ARRET :

Un chemin prend fin et constitue une boucle lorsque le noeud atteint est celui de départ (6 - 13 - 11 - 7 - 12 - 10 - (6)).

On revient alors au noeud précédent en regardant s'il existe une branche non parcourue, et dans l'affirmative, un nouveau chemin se forme. La procédure se poursuit jusqu'à ce que toutes les branches de l'arbre crée depuis le noeud n° 6 soient examinées, en définissant auparavant un deuxième test d'arrêt :

Lorsque par le chemin 6 - 13 - 11 - 7 - 12 - 10 - 8, la première branche (de la bifurcation au noeud 8) venant du noeud 6, a montré l'existence d'une boucle (6- 13- 11 - 7 - 12 - 10 - 8), un autre chemin est emprunté conduisant au noeud 9, puis 7. Or, 7 a déjà été rencontré sur ce chemin, c'est-à-dire que l'on est entré (conformément au schéma ci-dessous) à l'intérieur d'une boucle qui sera trouvée plus tard.





6 13 11 7 12 10
6 13 11 7 12 10 8
6 13 11 9 7 12 10
6 13 11 9 7 12 10 8
6 13 11 9 8
6 13 12 10
6 13 12 10 8

7 12 10 8 9
7 12 13 11
7 12 13 11 9

8 9

12 13

Arbre de recherche des boucles (JK 7472)

figure III.6

DEUXIEME TEST D'ARRET :

Si, en parcourant un chemin, on trouve le numéro d'un noeud qui y figure déjà, ce chemin est abandonné.

Après avoir épuisé toutes les boucles de l'arbre formé depuis le noeud 6, on passe à la ligne suivante de la matrice du graphe en recommençant le même algorithme. Il faut ajouter puisqu'on a quitté le premier noeud du graphe (dont le n° est le plus petit), un troisième test d'arrêt.

En effet, la branche 7 - 12 - 10 amène ensuite sur le noeud 6. Comme toutes les branches passant par 6 sont maintenant connues, il est inutile de poursuivre la recherche sur ce chemin, sous peine de retrouver les boucles déjà déterminées.

TROISIEME TEST D'ARRET :

Si, en parcourant un chemin, on trouve un numéro de noeud inférieur à celui de départ, ce chemin est abandonné.

A l'aide de ces trois tests d'arrêt, l'algorithme peut se poursuivre en séquence jusqu'à la $(n-1)$: ième ligne de la matrice du graphe. Point n'est besoin de partir du dernier noeud puisqu'en entrées ne figurent que des numéros inférieurs au sien.

Après avoir décrit l'arbre complet de la fig. III.6, on est assuré de connaître toutes les boucles du graphe.

BOUCLES FONDAMENTALES :

Il faut distinguer dans la liste suivante :

1	6	13	11	7	12	10		
2	6	13	11	7	12	10	8	
3	6	13	11	9	7	12	10	
4	6	13	11	9	7	12	10	8
5	6	13	11	9	8			
6	6	13	12	10				
7	6	13	12	10	8			
8	7	12	10	8	9			
9	7	12	13	11				
10	7	12	13	11	9			
11	8	9						
12	12	13						

les boucles fondamentales de celles qui ne le sont pas, en se rappelant la définition donnée au § III.1.1.

Il suffit de comparer systématiquement chacune des boucles à celles qui les suivent .

Pour rendre l'algorithme plus rapide, on tiendra compte de deux remarques :

- Si deux boucles ont la même longueur, il est inutile de les comparer, l'une ne pouvant contenir dans l'autre.
- Pour deux boucles de longueur différente, les numéros constituant la plus courte seront comparés à la plus longue.

De plus, si un numéro appartenant à une boucle ne se retrouve pas dans une autre boucle, la comparaison ne doit pas se poursuivre sur les autres numéros (compte tenu des deux remarques précédentes).

La figure III.7 donne le déroulement des opérations qui s'effectuent sur la liste ci-dessus, mettant en évidence les deux boucles fondamentales

8 9

et 12 13

1 6 13 11 7 12 10 2 6 13 11 9 7 12 10 4 6 13 11 9 7 12 10 8 5 6 13 11 9 8 6 6 13 12 10 7 6 13 12 10 8 8 7 12 10 8 9 9 7 12 13 11 10 7 12 13 11 9 11 8 9 12 12 13 1 supprime 2	3 6 13 11 9 7 12 10 4 6 13 11 9 7 12 10 8 5 6 13 11 9 8 6 6 13 12 10 7 6 13 12 10 8 8 7 12 10 8 9 9 7 12 13 11 10 7 12 13 11 9 11 8 9 12 12 13 6 supprime 1	3 6 13 11 9 7 12 10 5 6 13 11 9 8 6 6 13 12 10 7 6 13 12 10 8 8 7 12 10 8 9 9 7 12 13 11 10 7 12 13 11 9 11 8 9 12 12 13 3 supprime 4
5 6 13 11 9 8 6 6 13 12 10 7 6 13 12 10 8 8 7 12 10 8 9 9 7 12 13 11 10 7 12 13 11 9 11 8 9 12 12 13 6 supprime 3	6 6 13 12 10 7 6 13 12 10 8 8 7 12 10 8 9 9 7 12 13 11 10 7 12 13 11 9 11 8 9 12 12 13 11 supprime 5	6 6 13 12 10 8 7 12 10 8 9 9 7 12 13 11 10 7 12 13 11 9 11 8 9 12 12 13 6 supprime 7
8 7 12 10 8 9 9 7 12 13 11 10 7 12 13 11 9 11 8 9 12 12 13 12 supprime 6	9 7 12 13 11 10 7 12 13 11 9 11 8 9 12 12 13 11 supprime 8	9 7 12 13 11 11 8 9 12 12 13 9 supprime 10
	11 8 9 12 12 13 12 supprime 9	

Algorithme de recherche des boucles fondamentales - Bascule 74H72

figure III.7

III.2.2.4. Détermination des variables secondaires en nombre minimum

Deux solutions s'offrent à nous :

- l'exploitation directe du tableau des boucles fondamentales,
- l'utilisation de l'équation (3)

Nous avons choisi d'abord la deuxième solution, ayant à notre disposition des programmes d'intersection et de simplification des fonctions booléennes (§ III.2.1.3.). Il ne reste qu'à transformer, par le codage indiqué au § III.2.1.2 une boucle fondamentale constituée d'un certain nombre d'opérateurs, en une fonction où ces équations jouent le rôle de variables à raison d'une par monôme.

Ex : 8 9 donne $x_8 + x_9$

et sous forme codée :

3 3 3 3 3 3 3 1 3

3 3 3 3 3 3 3 3 1

L'algorithme complet devient alors extrèmement simple :

Transformer les deux première boucles en fonctions, effectuer leur intersection, puis l'intersection du résultat avec la troisième boucle codée à son tour, et ainsi de suite jusqu'à épuisement de la liste des boucles fondamentales. Dans le cas particulier d'un circuit formé d'une seule boucle fondamentale, chacun de ses opérateurs représente une solution.

Cette méthode, bien que séduisante, possède un gros désavantage qui n'apparait pas sur l'exemple que nous traitons.

Si l'on veut être sûr de posséder à la fin toutes les solutions minimales, il faut conserver après chaque intersection tous les monômes et ce n'est qu'après la dernière que l'on pourra détecter les ensembles les plus courts. Mais si le graphe comporte un nombre assez grand de boucles fondamentales constituées elles-mêmes de beaucoup d'opérateurs, la liste définitive risque d'avoir une longueur très importante puisqu'au cours d'une intersection seule la simplification $Ax+x = x$ interviendra. Ceci limite donc la taille des circuits susceptibles d'être analysés.

Pour atténuer cet effet, nous avons transformé l'algorithme en ne gardant après chaque intersection que les termes de longueur minimale. On peut ainsi, sans occuper trop de place mémoire, déterminer les ensembles minimaux

de graphes où s'entrelacent une centaine de boucles.

Malheureusement, cet artifice peut amener, selon la topologie du graphe, une perte d'information.

Examinons la liste de la figure III.8.a.

Si, après l'intersection des deux première boucles, on garde les ensembles les plus petits, il reste : 6 , 13, 11, 10. L'intersection de ces ensembles avec la troisième donne : 6, 10 et la solution finale 10 - 16 , unique. On a caché la solution 11 - 15 qui convient également par l'élimination du terme 11 - 15 au cours de l'intersection de la troisième boucle avec les deux précédentes.

Cette perte d'information que nous constatons ici à un premier degré devient critique si toutes les solutions minimales se trouvent supprimées par les simplifications successives. C'est le cas de l'exemple de la fig. III.8.b qui montre la non validité de ce procédé en général.

Malgré ceci, cet algorithme s'applique, avec succès, à de nombreux circuits, et nous l'avons largement utilisé.

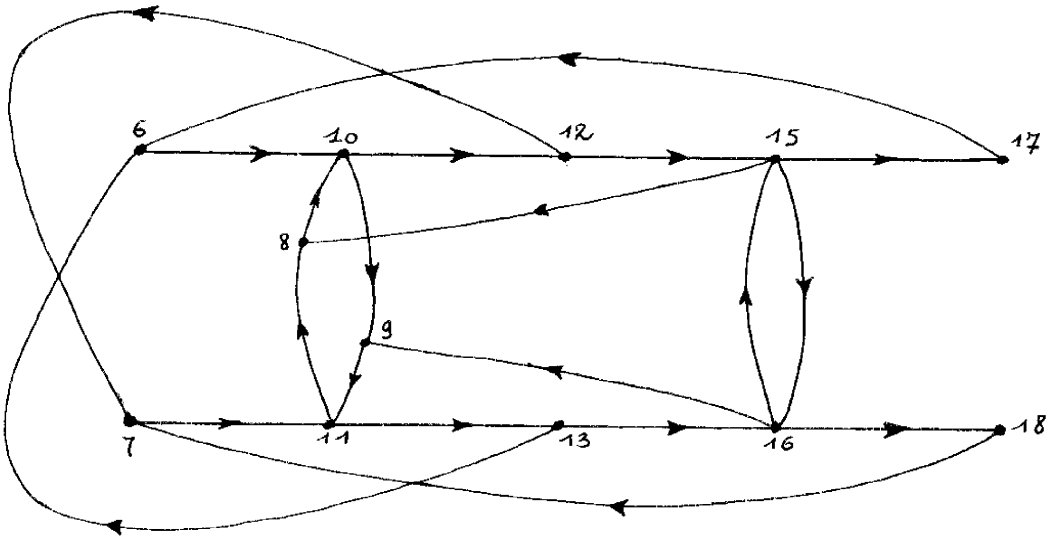
Cependant, si l'on considère les trois critères suivants :

- rapidité
- économie en place mémoire
- sûreté

on voit que cette méthode ne satisfait bien que le premier, pas du tout le second, et pas toujours le troisième.

Nous allons examiner d'autres solutions afin de mieux répondre à ces trois exigences. Elles sont basées sur l'exploitation directe du tableau des boucles fondamentales.

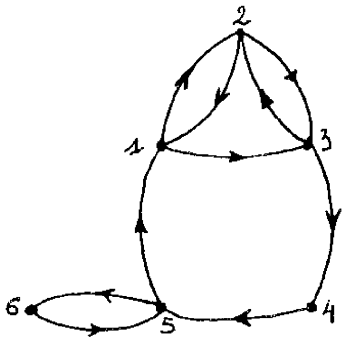
1 - Soit n le nombre de modules du circuit. On forme pour p variant de 1 à n, toutes les combinaisons de ces n numéros et on regarde à chaque valeur de p si une au moins des C_n^p combinaisons recouvre le tableau des boucles fondamentales. Lorsque cela est, la valeur de p correspondante indique le nombre minimum de variables secondaires, et il suffit d'essayer les C_n^p combinaisons pour avoir toutes



Boucles fondamentales

6	13	11	7	12	10
6	13	11	9	10	
6	17	15	12	10	
7	12	10	8	11	
7	18	16	13	11	
8	11	9	10		
8	15	12	10		
9	16	13	11		
15	16				

ⓐ



Boucles fondamentales

1 2	
1 3 4 5	
2 3	
5 6	

$$(1+2)\cdot(1+3+4+5)\cdot(2+3)\cdot(5+6) = 1$$

En simplifiant

1 2 5	
1 2 6	
1 3 5	
1 3 6	

Sans simplifier

2 5

ⓑ

figure III .8

les solutions minimales. Cet algorithme est très intéressant à appliquer lorsqu'un circuit comporte un très petit nombre de variables secondaires. Dans le cas contraire, le temps calcul devient rapidement prohibitif et le premier critère n'est pas du tout satisfait.

2 - En conservant le même principe, nous pouvons accélérer cette recherche, en nous contentant d'une seule solution (la première trouvée), et en déterminant au préalable une borne minimale du nombre de variables secondaires. Pour cela il suffit de classer les modules par ordre de poids décroissant (le poids d'un module étant le nombre de fois qu'il apparait dans le tableau des boucles fondamentales), le plus petit ensemble de modules dont la somme des poids est supérieure ou égale au nombre de boucles, indiquant la borne minimale. Cette méthode, beaucoup plus rapide ne satisfait quand même pas assez bien au premier critère.

3 - On voit que le critère rapidité est très important, ainsi que l'économie. La solution que nous proposons maintenant y satisfait très bien, mais par contre ne donnera pas, dans le cas général, une solution minimale.

On classe également les modules par ordre décroissant de poids. On considère le premier module et on note les boucles qu'il recouvre, en diminuant d'un le poids des autres modules qui les constituent. On reclasse la liste par ordre décroissant de poids, et on reprend l'algorithme sur le module qui suit. De cette façon, dès que le tableau des boucles est recouvert, l'ensemble des modules qui ont permis cette couverture, constitue la solution.

Sur l'exemple de la fig. III.8.b on trouverait 1, 2, 5, solution non minimale, mais dans le cas de circuits composés de bascules RS, la solution proposée a toujours été minimale, sur les exemples traités.

Cette méthode est, à l'heure actuelle, en essai, nous indiquerons les résultats obtenus dans l'annexe I.

III.2.2.5. Choix des variables secondaires

Nous avons déjà évoqué ce problème au § III.1.3. et à cause des difficultés qu'il soulève, nous avons conçu un algorithme laissant la responsabilité du choix à l'utilisateur.

Celui-ci peut, à priori, penser que telle ou telle sortie de module constitue une variable secondaire du circuit, surtout s'il est composé de bascules RS élémentaires (§ I.1.4.1.).

Déroulement de l'algorithme :

1 - Le circuit comporte des bascules RS.

On indique en donnée leur nombre, puis on inscrit sur une "carte de choix" dans un ordre préférentiel les deux numéros de sortie de chacune des bascules, et ensuite d'autres numéros du circuit car la présence des bouclages fait que le nombre de variables secondaires peut être supérieur au nombre de bascules RS.

En prenant le circuit de la fig. I.3, la carte de choix se présentera par exemple comme suit :

10 11 / 14 15 / 12 13 ...

Le premier chiffre est essayé. Les solutions contenant ce chiffre sont conservées au détriment des autres. S'il n'en existe pas, le deuxième est essayé à son tour et, en cas d'échec, il est indiqué à l'utilisateur qu'une erreur a été commise sur la carte de choix, la comparaison continuant toutefois sur les autres données.

Dans l'hypothèse où le premier chiffre convient, on saute au troisième et le procédé recommence. Si après avoir comparé autant de groupes de deux chiffres que le circuit comporte de bascules, il y a encore plusieurs solutions possibles, l'examen des autres données s'effectue, mais maintenant chiffre par chiffre. L'algorithme s'arrête dès qu'une seule solution reste. Dans le cas contraire où la liste des données est épuisée, il reste plusieurs solutions, la première est prise en considération.

2 - Le circuit ne comporte pas de bascules RS

Le même algorithme de choix sera quand même applicable. On indique qu'il existe une bascule RS, et sur la carte de choix, on placera en première donnée, un chiffre ne pouvant en aucun cas constituer une variable secon-

Ensembles minimaux

1	7	11	17	21	28	32
2	6	12	17	21	28	32
3	7	11	16	22	28	32
4	6	12	16	22	28	32
5	7	11	17	21	27	33
6	6	12	17	21	27	33
7	7	11	16	22	27	33
8	6	12	16	22	27	33

Carte de choix

6 7 / 11 12 / 16 17 / 21 22 / 27 28 / 32 33 / 0

1)- 6 →
- 6 12 17 21 28 32
- 6 12 16 22 28 32
- 6 12 17 21 27 33
- 6 12 16 22 27 33

7 sauté

2)- 11 → impossible

12 →
- 6 12 17 21 28 32
- 6 12 16 22 28 32
- 6 12 17 21 27 33
- 6 12 16 22 27 33

3)- 16 →
- 6 12 16 22 28 32
- 6 12 16 22 27 33

17 sauté

4)- 21 → impossible

22 →
- 6 12 16 22 28 32
- 6 12 16 22 27 33

5)- 27 →
- 6 12 16 22 27 33

STOP

Choix des variables secondaires - Ex : compteur binaire 3 bits (fig III.12)

figure III.9

daire ; par exemple une entrée primaire, puis toutes les autres variables secondaires prévues.

Le premier chiffre sera alors essayé sans succès, ensuite le second, et les suivants, un par un.

Exemple : compteur binaire 3 bits (fig. III.12)

choix des variables secondaires (fig. III.9)

III.2.2.6. Expression logique des variables secondaires et des sorties

Nous connaissons les modules dont les sorties représentent les variables secondaires du circuit. La détermination des expressions logiques nécessite, pour chaque variable secondaire et chaque sortie, d'établir un circuit combinatoire associé satisfaisant aux équations (1) et (2) . A partir du circuit initial, depuis le module désigné comme variable secondaire, il suffit de remonter les connexions jusqu'aux modules qui n'ont pour entrées que des variables secondaires ou des entrées primaires (fig. III.3 (a) et (b)) .

En termes d'algorithme, on construit en s'aidant de la matrice topologique du circuit séquentiel, pour chaque variable secondaire et chaque sortie, une nouvelle matrice topologique.

La première ligne est celle du module de départ. Les entrées qui ne sont ni variables secondaires, ni entrées primaires forment d'autres lignes de la matrice, et ainsi de suite.

C'est à ce niveau que nous utilisons tous les programmes de calcul (§ III.2.2.), mais en repartant des entrées, c'est-à-dire en lisant la matrice que nous venons d'établir, à l'envers.

L'équation ainsi calculée et simplifiée est donnée sous forme $\sum\prod$ (réunion de monômes) et un sous programme de décodage la traduit directement en clair en fonction du nom que l'on a attribué aux entrées, et du numéro des variables secondaires.

L'analyse pourrait s'achever ici, mais nous avons pensé qu'il était intéressant d'élaborer des algorithmes qui exploitent les expressions logiques ci-dessus établies et donnent directement tous les renseignements que l'on peut désirer.

III.2.2.7. Etude des forçages

Certaines entrées d'un système séquentiel jouent un rôle bien particulier et possèdent de par leur situation dans le circuit, le pouvoir de le figer dans un état déterminé lorsqu'on les rend agissantes, d'où leur nom de forçage. Il est donc intéressant d'étudier leur effet sur les variables secondaires et les sorties.

Nous avons prévu qu'un circuit puisse posséder une remise à zéro et une remise à un, ou seulement l'une des deux. On les distinguera ainsi par un codage particulier (§ III.3.1.) dans la mesure bien entendu où on le désire, si non elles seront considérées comme des entrées normales, au même titre que les autres.

Avec le codage que nous utilisons (§ III.2.1.2.) il n'est pas possible de rendre de façon explicite une variable égale à zéro ou un. Nous allons employer la méthode suivante.

. pour faire $x = 1$ dans une fonction $F(x, X)$, il suffit de former l'expression $F(x, X).x$ puis de remplacer x partout où il subsiste par 1.

. de même $x = 0$, s'obtient en formant $F(x, X).\bar{x}$ et en remplaçant partout $\bar{x}$ par 1.

Examinons l'effet de $R = 0$, $S = 1$ sur l'expression ci-dessous :

$$Y_{15}(\tau+1) = \bar{R} + S\bar{Y}_{10}(\tau)\,Y_{15}(\tau) + \bar{C}\bar{Y}_{10}(\tau) + CSY_{15}(\tau)$$

qui s'écrit sous forme codée :

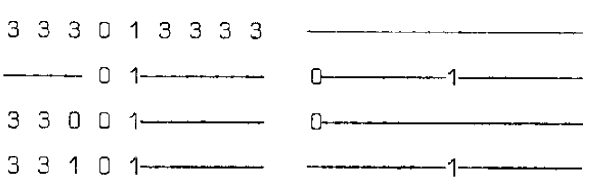
	3 3 3 0 3 3 3 3 3	3 3 3 3 3 3 3 3 3
	———————1———————	0———————1———————
	3 3 0 ———————————	0———————————————
	———1———1———————	———————————1———

Effectuons l'intersection de ces termes avec $\bar{R}S$:

3 3 3 0 1———————	———————————————

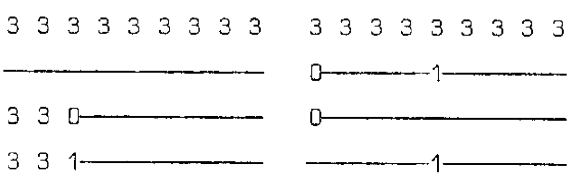
On obtient :

$$\overline{RS} + \overline{R}S\,Y_{10(T)}\,Y_{15(T)} + \overline{R}S\,\overline{C}\,\overline{Y}_{10(T)} + \overline{R}S\,C\,Y_{15(T)}$$



Remplacer $\overline{R}$ et S par "1" logique consiste avec ce codage à mettre un 3 aux places respectives de R et S, donc à rajouter : 3 - valeur inscrite, ce qui donne :

$$1 + \overline{Y}_{10(T)}\,Y_{15(T)} + \overline{C}\,Y_{10(T)} + C\,Y_{15(T)}$$



$$= 1 \qquad =$$

Le terme 3 3 ———— 3 représentant le "1" logique.

Par ce moyen, il est désormais possible d'inscrire la valeur d'une ou plusieurs variables dans une expression logique, et, dans le cas présent, de regarder l'effet des différentes combinaisons des forçages sur le circuit.

L'algorithme permet de détecter la combinaison qui laisse au circuit sa liberté de fonctionnement, et place en mémoire les équations logiques simplifiées qui en résultent. Celles-ci seront utilisées dans la suite du programme.

III.2.2.8. Rôle de l'horloge

L'horloge possède également dans un circuit un rôle particulier, celui de valider ou non les niveaux apparaissant sur les entrées primaires, et bloquer à des instants déterminés l'évolution des variables secondaires (systèmes asynchrones synchronisés).

Cet algorithme met en évidence de façon très claire, les isolements et les transferts dans le cas de systèmes composés de bascules Maître-Esclave, résultant des niveaux appliqués sur l'entrée d'Horloge.

Il procède de la même manière que celui décrit précédemment à condition toutefois qu'une des entrées ait été signalée comme étant une horloge par un codage particulier.

On applique alors successivement les niveaux 0 et 1 sur l'entrée désignée et on regarde comment se transforment les équations des variables secondaires.

III.2.2.9. Matrices des excitations secondaires et des sorties

Si le circuit possède des entrées de forçage, ce sont les expressions logiques simplifiées que l'on va transcrire sous forme de tables. Celles-ci contiendront à l'intérieur de chaque case les valeurs binaires des variables secondaires (0, 1).

Le calcul se déroule comme suit :

On génère une base binaire correspondant aux variables secondaires, et, pour chaque combinaison de cette base (figurant en tête de ligne), est générée une autre base binaire correspondant aux entrées. De cette façon, le calcul des matrices s'effectue ligne par ligne. A une combinaison donnée dans chacune des bases correspond une case de la matrice. Pour déterminer les valeurs des variables secondaires qu'il faut y inscrire, on introduit dans un monôme tout comme au § III.2.2.7., les deux combinaisons des entrées et des variables secondaires dont l'intersection constitue la case à remplir.

On pourrait obtenir les valeurs des variables secondaires à l'instant T + 1 en effectuant l'intersection du monôme d'assignement des variables à l'instant t avec les expressions logiques. Si l'intersection est nulle, la valeur de la variable secondaire à (T+1) est 0 si non elle est 1 puisque toutes les variables sont assignées.

Nous utilisons ici un autre procédé, plus rapide car faisant appel à un sous programme plus simple que le sous programme d'intersection.

Appliquons la méthode de simplification de première espèce (Ax C x) entre le monôme d'assignement et les expressions logiques des variables secondaires.

- Si le monôme d'assignement est inclus dans un des monômes de la fonction auquel on le compare, celle-ci conserve le même nombre de termes et sa valeur est "1".

- S'il n'est pas inclus, la fonction résultante possède un terme de plus et sa valeur est "0".

De toute façon, le monôme d'assignement ne peut inclure un des monômes de la fonction puisqu'il est complètement défini et représente un sommet de l'hypercube à n + p dimensions.

(n = nombre d'entrées, p = nombre de variables secondaires)

EXEMPLE :

Valeur de $Y_{10\ (T+1)} = (\overline{K} + \overline{C} + Y_{15(T)}) \cdot Y_{10(T)} + JC\ Y_{15(T)}$

1) Pour J = K = C = 0, $Y_{10(T)} = Y_{15(T)} = 1$

On compare le minterm $\overline{J}\ \overline{K}\ \overline{C}\ Y_{10}\ Y_{15}$ à la liste. Il est inclus dans le monôme $\overline{K}\ Y_{10}$ donc la fonction vaut 1.

2) Pour J = K = C = $Y_{10(T)} = 0$, $Y_{15(T)} = 1$

Le minterm $\overline{J}\ \overline{K}\ \overline{C}\ \overline{Y}_{10}\ Y_{15}$ n'est inclus dans aucun des monômes de la fonction. L'équation résultante comporte un terme de plus, et la fonction $Y_{10(T+1)} = 0$.

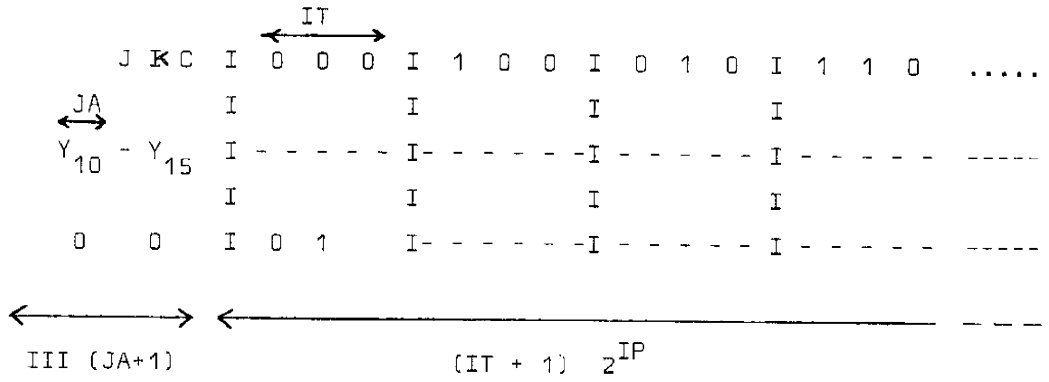
Le procédé se poursuit ainsi sur une ligne. Après épuisement de la base binaire des entrées, la ligne est écrite puis effacée avant que soit généré un autre terme de la base des variables secondaires, et de nouveau successivement, tous les termes de la base des entrées. Ceci évite un encombrement inutile des mémoires et permet de tracer des matrices assez importantes.

Le tracé de la matrice des sorties s'effectue de façon strictement identique en utilisant les équations booléennes des sorties.

Il se pose un problème au niveau de la présentation des résultats. Comme ces algorithmes peuvent traiter des circuits dont le nombre de variables secondaires n'est pas limité, on ne peut fixer a priori la largeur d'une colonne sous peine soit de perdre de la place soit de ne point en avoir assez.

Il est donc nécessaire que la largeur d'une colonne s'adapte à tous les cas, et résulte d'un calcul.

Une matrice se présentera de la façon suivante :



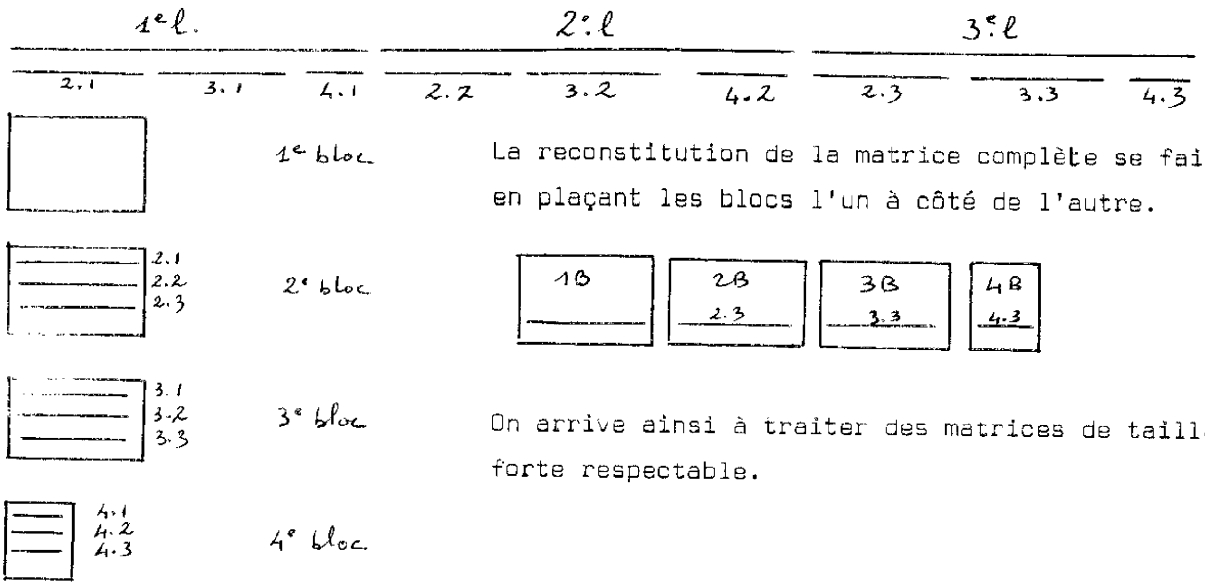
La largeur d'une colonne doit être égale au plus grand des deux nombres :

- nombre d'entrées
- nombre de variables secondaires

En appelant JA : nombre de caractères utilisés pour écrire YN.
III : nombre de variables secondaires
IP : nombre d'entrées
IT : nombre de caractères inscriptibles dans une colonne
(IT = IP si IP $>$ III, IT=III si III $>$ IP)

si l'expression III (JA+1) + (IT + 1) 2^{IP} $>$ 131 (nombre de caractères utilisables sur la largeur d'une page), la matrice ne contiendra pas dans une page de résultat. Dans ce cas, la premier partie(les 131 premiers caractères) de la ligne est écrite directement, et le reste stocké sur bande magnétique.

Lorsque toute la matrice a été calculée, on relit la bande, et les lignes sont écrites par fragments de 131 caractères, ceci jusqu'à épuisement de l'enregistrement.



La reconstitution de la matrice complète se fait en plaçant les blocs l'un à côté de l'autre.

On arrive ainsi à traiter des matrices de taille forte respectable.

III.2.2.10 Matrice des transitions

Nous lui donnons pour rôle d'indiquer les transitions qui s'effectuent entre états stables; elle diffère en cela de ce qu'il est convenu habituellement d'appeler une matrice des transitions ((2) p. 302)

Ce souci de différenciation est doublement justifiable :

- d'abord par le fait qu'avec les circuits que nous étudions toute transition aboutit à un état stable (les états instables intermédiaires n'existant que pour respecter les adjacences entre variables secondaires)
- ensuite parce que le dépouillement des résultats s'en trouve simplifié et permet le tracé immédiat d'un graphe de fluence donnant suffisamment de renseignements à l'utilisateur.

Ce manque d'information peut par contre être gênant lorsqu'il s'agit de localiser une panne dans un circuit, raison pour laquelle le moment venu, nous n'utiliserons pas cette matrice (chap. V).

L'algorithme n'en est pas pour autant plus simple, bien au contraire puisque dans un premier temps nous allons devoir chercher les états stables. Comme au § III.2.2.8., les calculs sont menés à l'aide des expressions logiques des variables secondaires.

Recherche des états stables :

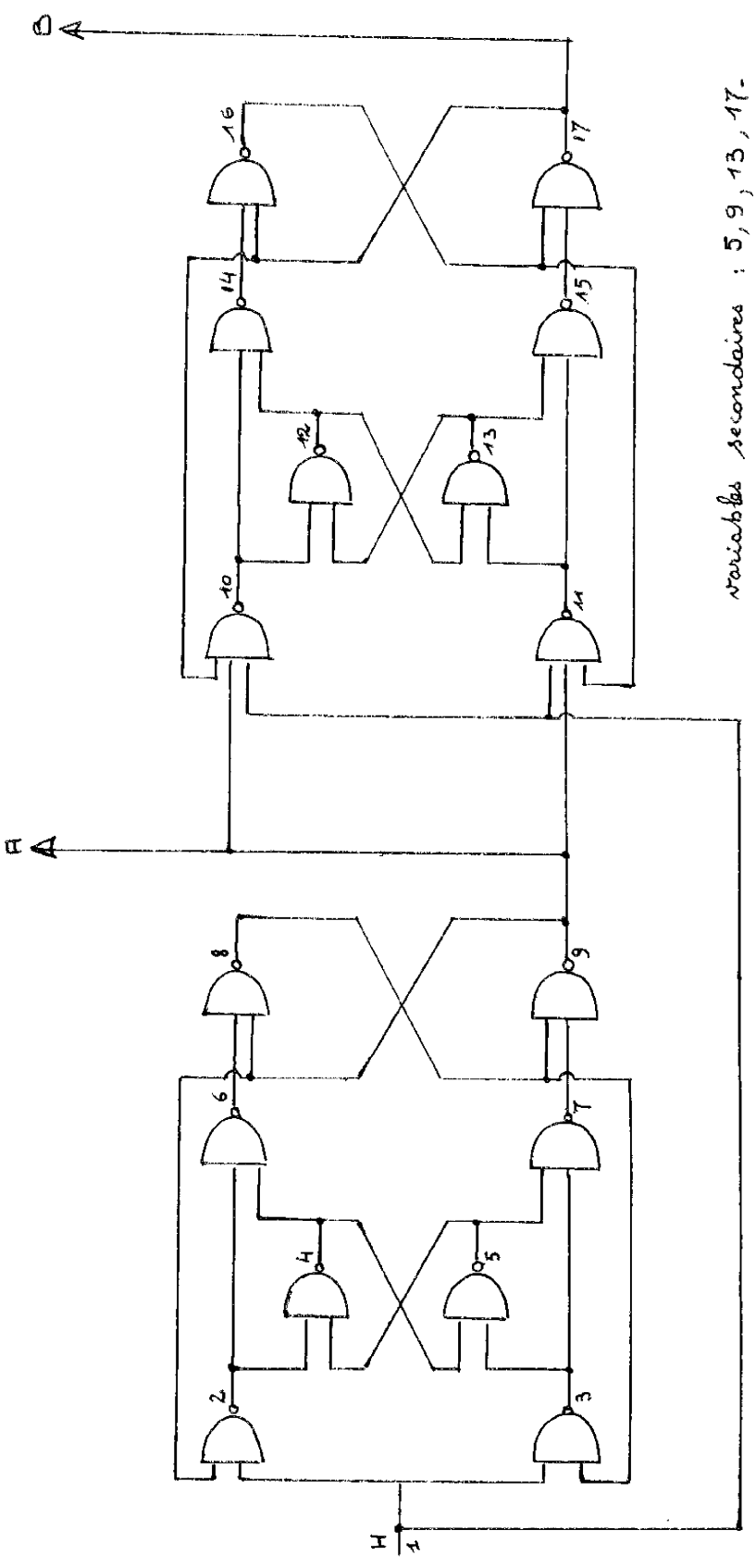
On génère deux bases : - une pour les variables secondaires
- une pour les entrées.

Pour chaque terme de la base des variables secondaires, commence une génération complète de la base des entrées. On crée également un monôme d'assignement et son intersection avec les équations booléennes des variables secondaires donne les valeurs des variables secondaires (T+1), successivement. Chacune est comparée au fur et à mesure au terme courant de la base des variables secondaires. Si elle est différente, on génère immédiatement un autre terme de la base des entrées (pas d'état stable dans cette case), si non on continue sur une autre variable secondaire. Dans le cas où chaque VS (T) = VS (T+1) l'état représenté par le terme courant de la base des VS est stable. Par contre, si toute la base des entrées a été épuisée, l'état examiné est instable. Celarevient à se déplacer dans une ligne de la matrice des excitations secondaires, en regardant si dans une case au moins, la combinaison binaire est égale à celle placée en tête de ligne. On obtient de cette façon la liste des états stables que l'on code par leur équivalent décimal +1 (l'état 1 représentant la combiniaon binaire 0 0 0 ... 0). Le même codage est appliqué aux entrées du circuit (vecteur n° 1 : 0 0 0 .... 0).

Etablissement de la matrice des transitions entre états stables :

On part d'un état stable, et pour chaque vecteur d'entrée, on cherche l'état atteint par la même méthode que ci-dessus. S'il est stable, le processus s'arrête, si non la valeur obtenue est donnée comme état initial et la recherche reprend. Il peut arriver que le système ne se stabilise jamais, ce qui correspond à l'existence d'un cycle que l'on détecte et signale à l'utilisateur.

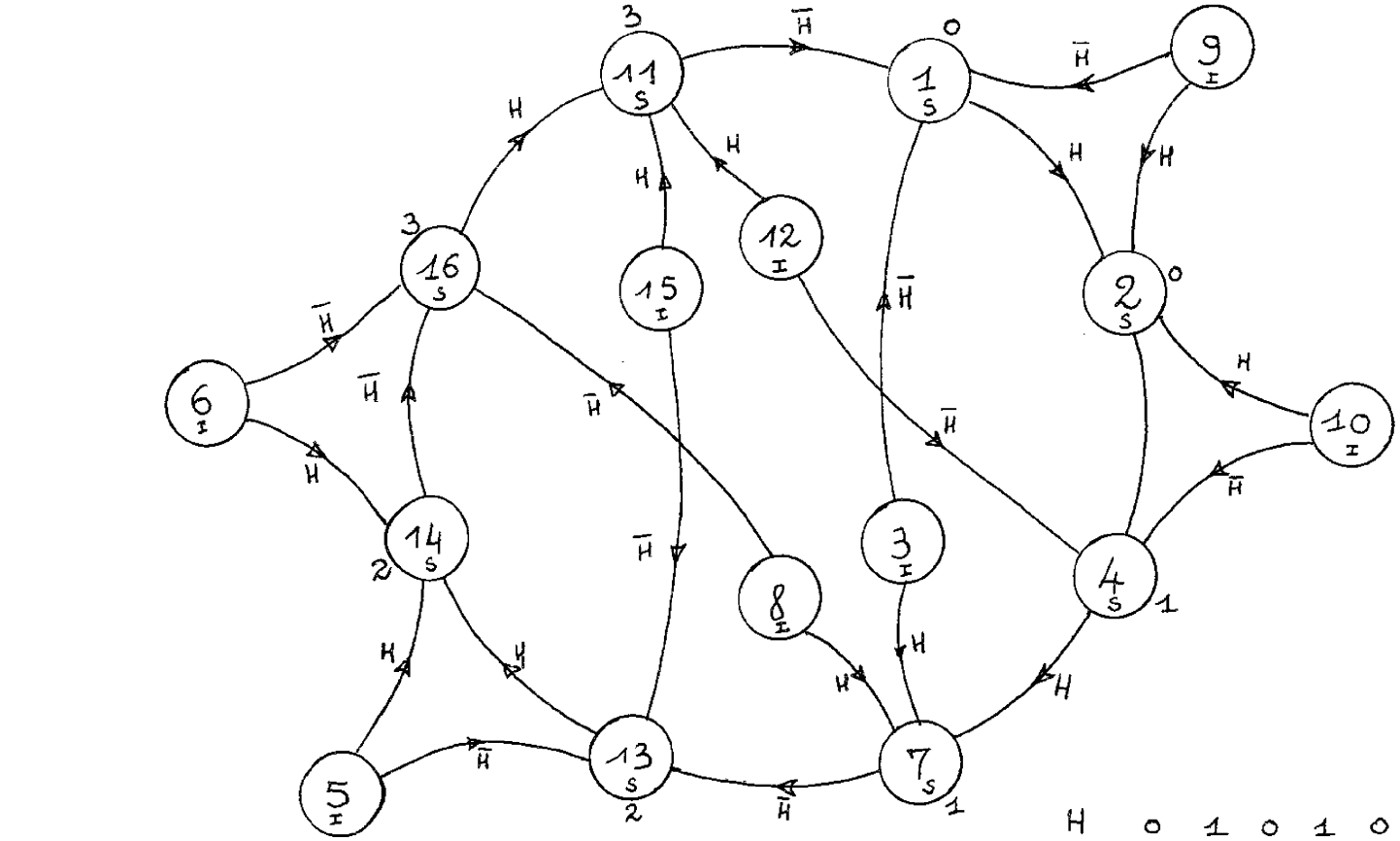
Ceci représente sur la matrice des excitations, le déplacement dans la colonne du vecteur d'entrée appliqué, à partir d'un état stable. La mise en page se traduit comme indiqué fig. I.2, l'espacement entre les colonnes étant fixe. La valeur des sorties n'est pas marquée, seuls figurent les vecteurs assurant les transitions.



COMPTEUR BINAIRE 2 BITS

(2 x JK 74H72)

figure III.10



COMPTEUR BINAIRE 2 BITS

figure III.11

Ici, aussi, le calcul s'effectue ligne par ligne pour éviter l'encombrement mémoire, et dans le cas d'un dépassement de format, la mise sur bande et la technique du tracé par blocs sont employées

La matrice des transitions peut être mise en mémoire en vue d'une utilisation ultérieure (§ III.2.2.11).

III.2.2.11 Analyse des compteurs

Elle utilise les mêmes algorithmes. En dépouillant les résultats d'analyse nous avons remarqué qu'en règle générale les compteurs appartiennent à la classe des systèmes non fortement connectés, c'est-à-dire qu'il existe des états sources et des états stables autres que ceux appartenant à la séquence normale de comptage.

Pour connaître exactement le graphe de fonctionnement de ces éléments et aider au dépouillement de la matrice des excitations secondaires, nous avons élaboré un algorithme qui soumet l'élément à une séquence de comptage à partir de tous les états stables et instables.

Les figures III.11 et III.13 représentent respectivement les graphes complets d'un compteur 2 bits (2 x JK 74 H 72) et d'un compteur 3 bits (3 x JK 7473) binaires dont les schémas sont donnés par les figures III.10 et III.12.

Cette analyse supplémentaire procure une connaissance extrèmement précise des caractéristiques d'un compteur. Elle permet en particulier de savoir s'il existe des séquences de comptage autres que celle prévue, ou des cycles entre états instables dans lesquels le système pourrait être conduit lors de la mise sous tension par exemple. Elle apporte donc une aide fort intéressante à la synthèse car elle évite le passage à la réalisation physique, destiné à la vérification.

Dans le cas particulier des deux compteurs, que nous avons étudié ici, on voit que même si juste après la mise sous tension ces circuits se trouvent dans un état instable ou dans un état stable n'appartenant pas à la séquence normale, après un top d'horloge, au maximum, on se retrouvera sur la séquence de comptage

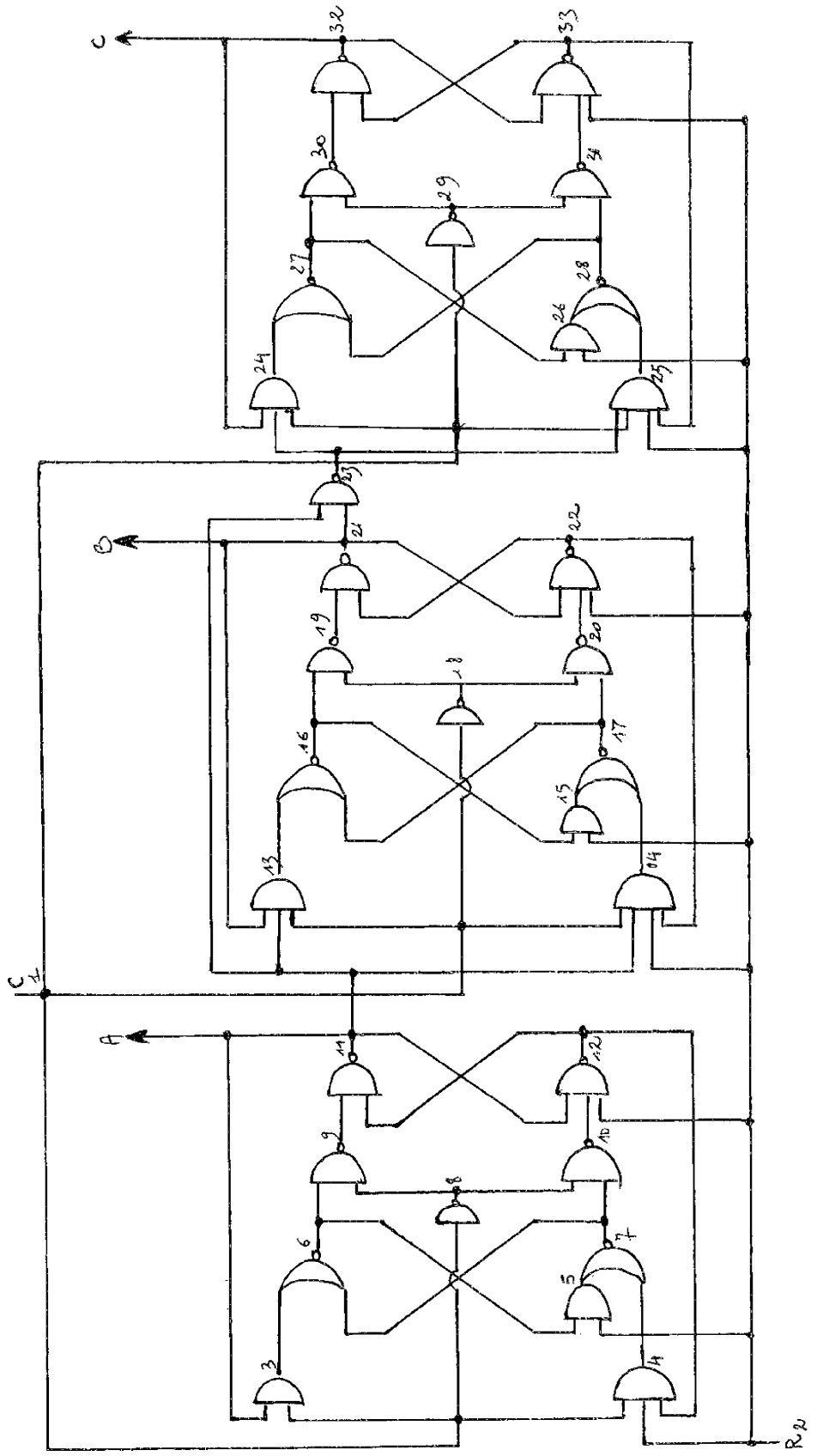
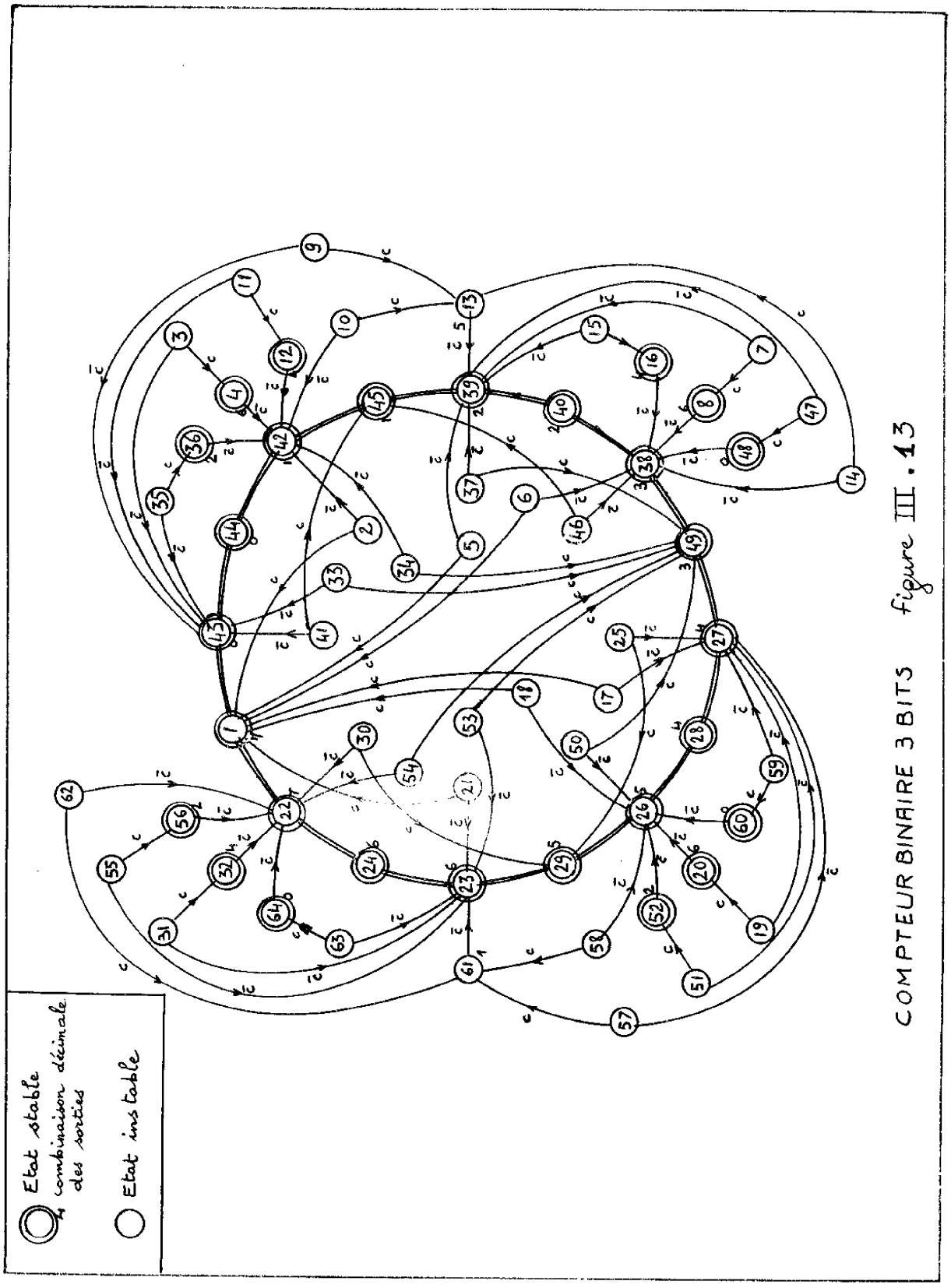


figure III.12



COMPTEUR BINAIRE 3 BITS figure III.13

même si l'entrée de remise à zéro n'est pas utilisée (les transitions qu'elle implique n'ont pas été représentées pour ne pas alourdir davantage les graphes). Il n'existe donc pas de cycle, ou de séquence parasite et on peut en conclure que la synthèse de ces circuits est satisfaisante.

Cet algorithme sera très utile par la suite (§ V.2.3.). Il nécessite la mise en mémoire de la matrice des transitions en autant de mots mémoires que de transitions entre états stables. On code en effet dans un mot l'état de départ, le vecteur transition, et l'état d'arrivée, comme suit :

0 1 2 0 7 0 0 8

ce qui signifie : le vecteur 7 amène de l'état 12 à l'état 8. On voit qu'il est prévu place pour 999 états (9 VS ⟶ 512), et 99 vecteurs d'entrée (6 entrées ⟶ 64).

III-3 UTILISATION - RESULTATS - POSSIBILITES - AMELIORATIONS

III.3.1. Utilisation du programme

Nous allons indiquer succinctement les données à fournir au programme afin qu'il puisse correctement se dérouler :

1. Nombre d'exemples à traiter
2. Nom ou type du circuit
3. Nombre de modules composant le circuit (NBMOD)
   Nombre d'entrées primaires (y compris forçages) (NEP)
   Nombre d'entrées maximum sur un module (NEMAX)
   Nombre de bascules que comporte le circuit (NBASC)
   Nombre de sorties (NBSOR)
   Nombre de choix effectués sur les variables secondaires (NCHOIX)
4. Numéro des sorties
5. Matrice topologique du circuit
   Elle possède NBMOD lignes et (NEMAX + 2) colonnes
6. Choix des variables secondaires.

Il est préférable que la dernière donnée concernant ce choix soit un zéro, indiquant ainsi la fin du choix.

7. Nom des entrées primaires.

Elles peuvent être repérées par n'importe quel chiffre ou lettre mais dans la limite d'un caractère alphanumérique (J, K, C ...)

Leur position sur la carte de donnée doit correspondre à leur numérotation dans le circuit. Cependant si celui-ci comporte une horloge que l'on veut considérer comme telle, on la placera après les autres entrées primaires. Les forçages seront inscrits après l'horloge.

Ex : J K H R S ces entrées ayant dans le circuit respectivement les numéros 1, 2, 3, 4, 5.

8. Codage des entrées primaires.

Cette donnée sert à distinguer les entrées particulières :

code Horloge : 80
R à zéro : 90
R à 1 : 91

Ainsi pour l'exemple précédent on indiquera 1.2.8090.91.

Ce programme, conçu au départ comme un seul bloc, a été ensuite découpé en sous programmes, chacun traitant une partie du § III.2. Dans ces conditions il devient possible, en composant un programme d'appel extrêmement simple, de commencer l'analyse du circuit à n'importe quel niveau, selon la connaissance préalable que l'on en a, gagnant ainsi sur le temps calcul. On utilise de plus la technique du dimensionnement variable des matrices dans les appels de sous-programmes ce qui permet de traiter des exemples de complexité très diverse en n'ayant qu'à adapter les dimensions des matrices de calcul dans le programme principal d'assemblage.

III.3.2 Résultats

Temps calcul :

L'analyse d'un circuit comportant beaucoup de modules, mais peu

de boucles et de variables secondaires sera plus rapidement traitée que celle d'un circuit modeste en nombre de modules mais dont le graphe est très enchevêtré et le nombre de variables secondaires important. (cas des circuits composés de bascules).

Pour une bascule Maître-Esclave (10 à 15 modules, 5 entrées, 2 VS) le temps de calcul avoisine la minute (IBM 7044).

Pour un compteur synchrone par 10 (4 JKMS interconnectées) il est de l'ordre de 15' (CII 10070). Cela peut sembler grand mais il faut penser que ce circuit comporte une cinquantaine de modules où 70 boucles sont enchevêtrées, parmi lesquelles 16 boucles fondamentales nécessitant 8 variables secondaires. De plus le tracé de la matrice d'excitations nécessite $2^8 \times 8 \times 2 = 4096$ appels du sous programme de simplification de première espèce, celui de la matrice des sorties $2^8 \times 2 \times$ nombre de sorties appels de ce même sous programme, et celui de la matrice des transitions environ un millier de fois l'appel du sous programme d'intersection (qui lui-même en appelle deux).

III.3.3. Possibilités et limites

La limite sur le nombre de variables secondaires que peut posséder un circuit est fixée par la "taille" du calculateur utilisé. Elle dépend

- de la complexité des expressions logiques obtenues qui, à mesure qu'elle croît, entraîne le même accroissement de la place mémoire occupée,
- et du nombre de boucles du graphe.

Nous avons limité par contre le nombre d'entrées, en imposant qu'elles figurent dans le premier mot mémoire parmi ceux qui constituent un monôme, dans le seul but de faciliter le décodage.

Ainsi, un mot de 32 bits (CII 10070) permet 9 entrées,
un mot de 36 bits (IBM 7044) , 10 entrées.

Le tracé des matrices d'excitations, des sorties et des transitions n'est absolument pas limité, car il suffit d'adapter la dimension de la ligne de calcul à la largeur maximum que l'on désire. Notons toutefois qu'à partir d'une certaine limite, le tracé de ces tables, pour aussi spectaculaire qu'il puisse être

devient inexploitable et augmente assez considérablement le temps de calcul.

Parmi les neuf ou dix entrées possibles, peuvent être particularisées une horloge, une remise à zéro, une remise à un. Si le circuit comporte plusieurs horloges, on ne peut en distinguer qu'une, les autres étant considérées comme des entrées normales, ce qui n'est pas génant puisque l'analyse est traitée de façon asynchrone par l'étude des niveaux apparaissant sur les entrées primaires.

III.3.4. Améliorations - Possibilités futures

III.3.4.1. Améliorations

Les améliorations que nous proposons ci-dessous ont pour but de rendre possible l'analyse de circuits plus importants. Elles ne modifient en rien les codages employés et demandent seulement une légère retouche des algorithmes décrits au § III.3.

1. Augmentation du nombre d'entrées

En n'imposant pas aux entrées de forçage de figurer seulement dans le premier mot mémoire représentant un monôme, la limitation de leur nombre disparait. Cela implique de modifier le sous programme de décodage et les algorithmes des § III.2.2. (7.8.9.10).

2. Augmentation du nombre de variables secondaires

Au cours de l'établissement des équations logiques un gain de mémoires non négligeable peut être obtenu en procédant de la façon suivante :

Nous avons vu au § III.2.1.2. qu'une variable est repérée dans un mot par sa position. Cette méthode, bien adaptée au combinatoire (systèmes à grand nombre d'entrées), crée ici un gaspillage de mémoires surtout lorsque les circuits analysés possèdent peu d'entrées, beaucoup de modules, et dont les variables secondaires sont représentées par des modules au numéro élevé.

Par exemple l'expression $\overline{A}\ B\ Y_{25}.\ Y_{47}.\ \overline{Y_{57}}$ d'un circuit à deux entrées A et B, trois variables secondaires, (25, 47, 57) et 57 modules, nécessite 7 mots mémoires de 9 chiffres.

Si par contre, au lieu de repérer les variables secondaires par leur place, on les repère par leur ordre, il devient possible de les "caler" juste après les entrées primaires et dans le cas présent, un seul mot mémoire suffit (gain de 7 en place mémoire).

Ainsi toute limitation sur la taille d'un circuit disparait pratiquement. Cela demande de réécrire parmi les programmes de calcul booléen, ceux qui, après la lecture de la matrice topologique, effectuent les codages, et de refaire un programme de décodage qui restitue leur numéro aux variables secondaires. Ce travail ne crée pas de difficulté importante et doit être mené à bien en un temps relativement bref (cf. annexe I).

III.3.4.2. Possibilités futures

Une première possibilité, qui est en définitive une application de ce programme, a pour objet l'aide à la conception.

En effet, les résultats qu'il fournit peuvent être exploités de différentes manières et l'une d'elles, par l'emploi des équations logiques ou des tables mises en mémoire (chap. V) consiste à donner la réponse d'un circuit soumis à une séquence d'entrée, permettant ainsi de vérifier que le système conçu fonctionne comme prévu. Il suffit pour cela d'écrire un programme d'appel générant la séquence d'entrée désirée. Nous avons déjà abordé ce sujet au § III.2.2.11.

La deuxième possibilité, fort importante, est l'extension de ce programme à des ensembles plus volumineux constitués de bascules, registres, compteurs... Il n'est alors plus question de décomposer le circuit au niveau du module élémentaire. Une solution consiste à effectuer l'analyse systématique par le programme présent des éléments classiques cités ci-dessus, et à stocker les résultats (équations booléennes, tables) à l'aide d'un codage judicieux, sous forme de sous programmes. La confection d'un programme d'assemblage, qui à partir de la

description de l'interconnexion des éléments (simples ou complexes) figurant dans la bibliothèque de sous programmes disponibles calculerait les équations du système complet, permettrait l'analyse à grande échelle.

Ce programme tel qu'il est actuellement (2000 instructions FORTRAN, 34 sous programmes), représente pour nous un outil destiné à nous fournir tous les éléments nécessaires au test des systèmes séquentiels, but principal de notre étude.

CHAPITRE IV

TESTS DE DETECTION

IV-1 GENERALITES

Au cours de ce chapitre nous allons proposer une méthode permettant de s'assurer qu'un système séquentiel fonctionne conformément aux données du constructeur ou aux résultats de l'analyse préalable qu'on a pu effectuer sur lui (chap. III), sans aborder le problème de la localisation des pannes (chap. V).

Ce sujet a été traité selon deux optiques très différentes (chap. II) :

- utiliser les algorithmes mis au point en vue de tester les systèmes combinatoires, en les adaptant au séquentiel.

- élaborer des algorithmes en employant des moyens propres aux systèmes séquentiels (ex. : tables de fluence).

A partir de la première solution, de nombreuses recherches et applications ont été et continuent à être développées (16) de par les résultats satisfaisants obtenus. Mais si l'on cherche à établir des méthodes algorithmiques, la conséquence est que celles-ci consistent en une adaptation de l'algorithme des D cubes. Par contre, l'extension au séquentiel de méthodes théoriques créées pour le combinatoire (18) ouvre un vaste champ d'investigation et peut laisser espérer des solutions efficaces et élégantes.

Cependant, traiter ce problème au moyen des outils propres aux systèmes séquentiels semble à la fois logique et raisonnable. Jusqu'à présent, les méthodes proposées utilisent la table de fluence dont l'inconvénient, lorsqu'on arrive au stade de la programmation, est d'occuper une place importante en mémoire et de limiter la taille des circuits testables. Or la façon la plus condensée de représenter un système séquentiel est encore de faire appel à ses équations (variables secondaires, sorties), solution que nous avons adoptée.

La méthode que nous exposerons dans les pages qui suivent résulte de l'interprétation de spécifications (3) sur une série de circuits intégrés logiques. Elle est systématique et a trouvé son illustration par l'écriture d'un programme en FORTRAN IV.

IV-2 DEPOUILLEMENT ET INTERPRETATION DE SPECIFICATIONS SUR UNE FAMILLE DE CIRCUITS INTEGRES LOGIQUES (3)

Examinons les tests relatifs à une bascule JK Maître-Esclave type 7473 (fig. IV.3) dont le schéma logique est donné fig. IV.1 et le graphe de fonctionnement fig. IV.2. Sur ce dernier figurent deux axes BM et BE ayant la signification suivante :

Un déplacement dans le graphe par une transition, selon l'axe BM correspond à une inscription dans le maître des informations apparaissant sur J et K, alors qu'un déplacement selon BE résulte d'un transfert des valeurs du maître dans l'esclave. On peut donc d'ores et déjà définir plusieurs sortes de tests :

- isolement - du maître
  - de l'esclave
- inscription dans le maître et transfert dans l'esclave.
  (insc. 0, transf. 0 - Inscript. 1, Transf. 1)

Ce circuit est muni d'une entrée de remise à zéro agissant sur le niveau bas et notée pour cela $\overline{RZ}$: il existera aussi des tests de forçage du maître et de l'esclave à zéro.

Penchons nous par exemple sur les tests d'isolement du maître.

La bascule maître, constituée de deux portes NI satisfait à la table d'excitation ci-dessous :

QM \ AB	0 0	1 0	0 1	1 1
0	(0)	(0)	1	(0)
1	(1)	0	(1)	0

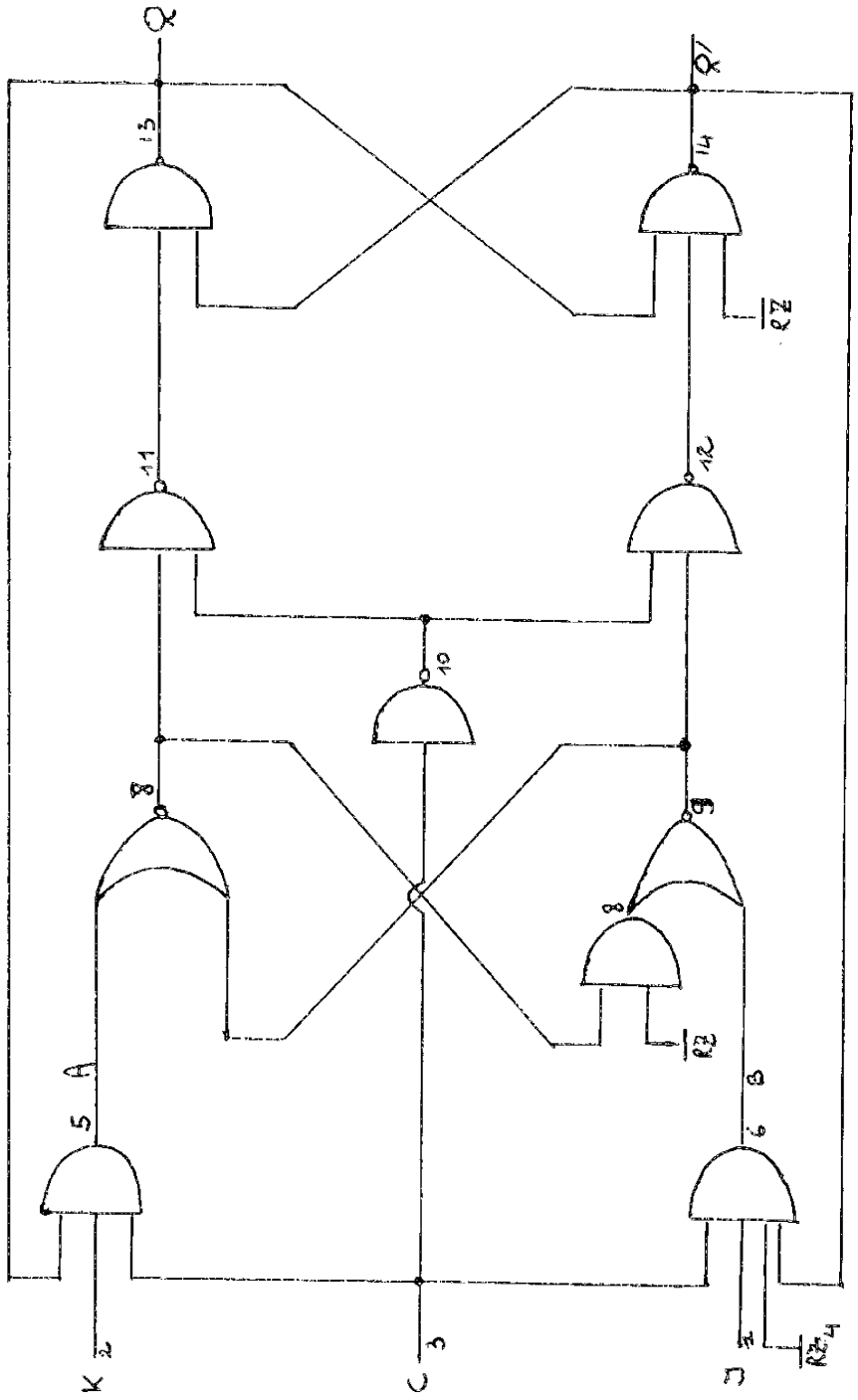
L'entrée de forçage n'étant pas agissante : $\overline{RZ} = 1$

avec $A = C\,K\,Q$

$B = C\,J\,\overline{Q}$

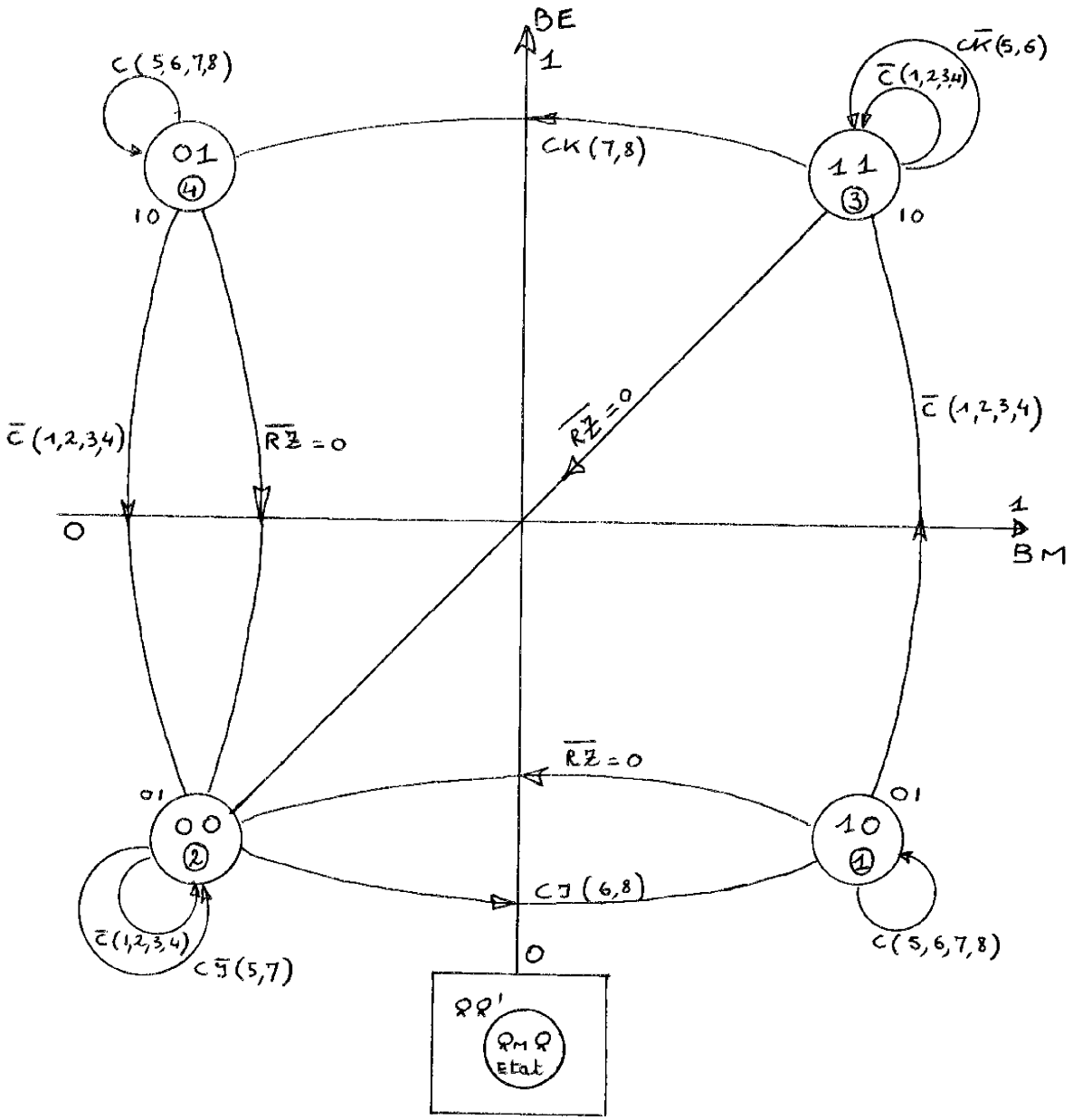
Son état d'isolement est obtenu pour A = B = 0

Appliquons cette condition dans chacun des quatre états du



Bascule JK SN 7473

Figure IV.1



Codage :

	J	K	C
1	0	0	0
2	1	0	0
3	0	1	0
4	1	1	0
5	0	0	1
6	1	0	1
7	0	1	1
8	1	1	1

Bascule SN 7473 - Graphe de fluence

figure IV.2

OBJET	variantes	Fin de test vérifier Q	Fin de test vérifier $\overline{Q}$	ETAT INITIAL Etats supposés (BM)	Etats supposés (BE)	Connexions C	Connexions $\overline{RZ}$	Connexions J	Connexions K	1e transition à	1e transition sur	2e transition à	2e transition sur
Isolement de BE à l'état 0		L	H	1	0	2	V_H	X	0				
Isolement de BE à l'état 1		H	L	0	1	"	"	0	X				
Inscription 0	a	L	H	1	1	0,8	V_H	X	2	2	C	0,8	C
Transfert 0	b	L	H	1	1	2	V_H	X	0	2	K	"	"
Inscription 1	a	H	L	0	0	2	0	2	X	V_H	$\overline{RZ}$	0,8	C
	b	"	"	"	"	0,8	V_H	"	"	2	C	"	"
Transfert 1	c	"	"	"	"	2	"	0	"	2	J	"	"
Isolement BM par Q et C		L	H	0	1	V_H	5,5	5,5	0	0,8	C		
Isolement BM par $\overline{Q}$ et C		H	L	1	0	V_H	2	0	5,5	"	"		
Isolement J		L	H	0	0	5,5	5,5	0,8	0	0	C		
BM par K		H	L	1	1	5,5	2	0	0,8	"	"		
Forçage à 0 de BE		L	H	X	1	V_H	V_H	X	X	0,8	$\overline{RZ}$		
Forçage à 0 de BM		H	L	1	X	V_H	V_H	X	0	0,8	$\overline{RZ}$		

X : indifférent - $2,4 < V_H < 5,5$ - L : niveau bas - H : niveau haut

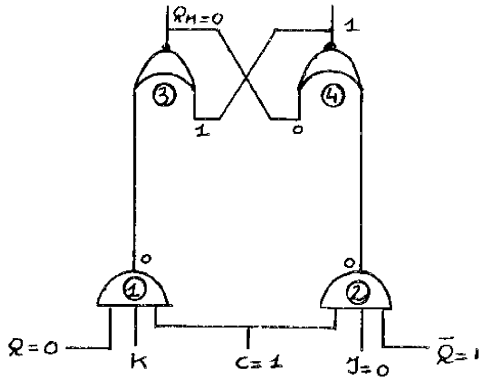
Test Fonctionnel JK7473 (Extrait de [3])

figure IV.3

graphe de fluence.

Etat 0 0 (2)

Examinons les deux vecteurs 5 et 7 ($C\bar{J}\bar{K}$ et $C\bar{J}K$) qui réalisent la condition d'isolement A = B = 0, en reprenant le schéma logique.



J = 0 est obligatoire puisque les deux autres entrées du ET n° 2 valent 1.
Sur le ET n° 1 par contre, la valeur de K paraît indifférente, Q assurant le niveau 0 sur la sortie de ce module.
Cependant si on applique K = 1, et que le fil de liaison ramenant la sortie Q sur le ET n° 1 soit coupé, sa sortie va passer à 1 et "renforcer" sur le NI n° 3 le 1 logique qui existait déjà sur l'autre entrée, pouvant ainsi cacher une panne éventuelle sur celle-ci (par ex. : court circuit).

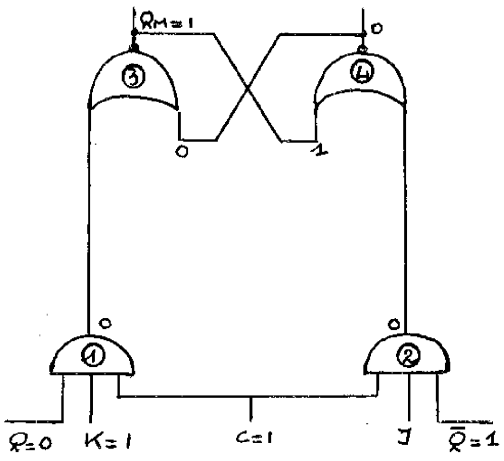
Par contre, en imposant K = 0, on assure le niveau 0 à la sortie du ET n° 1 ce qui ne favorise plus le niveau 0 à la sortie du NI n° 3. Restent les vecteurs 1, 2, 3, 4 ($\bar{C}$). C = 0 impose le niveau 0 à la sortie des ET n° 1 et 2 indépendamment de toute valeur des entrées directes et de bouclage. En particulier, une coupure sur le fil J qui imposait un niveau 1 constant se trouverait masquée par cette valeur de C et détectable par aucune autre transition. Une coupure sur l'entrée K, non observable dans cet état, le sera dans l'état (3), ainsi qu'une coupure sur l'horloge C, par la transition (4)→(2) où l'isolement du maître ne pourrait être effectif.

Le test du vecteur 5 ($C\ \bar{J}\ \bar{K} = 1$) appelé dans ces spécifications isolement de BM par J, étant le plus restrictif recouvre les autres vecteurs et suffit pour s'assurer de l'isolement de BM dans l'état (00).

Comme seules les sorties de l'esclave sont observables, il faut en plus pour vérifier que l'isolement est effectif, transférer la valeur du maître dans l'esclave par 1 ($\bar{C}\ \bar{J}\ \bar{K} = 1$).

Physiquement, on doit appliquer des tensions sur les entrées C,J et K, et pour rendre le test encore plus restrictif on choisira C = 5,5 v, J = 0,8 v, K = 0 (fig. II.1)

Etat 1 0 (1)



Sur le ET n° 1, Q = 0 assure l'isolement. On placera alors toutes les autres entrées de ce module à 1 pour qu'une panne sur la connexion de bouclage ne soit pas masquée. Sur le ET n° 2 C = 1, $\overline{Q}$ = 1. Il faut mettre J = 0 afin que la sortie de ce module n'impose pas l'état du NI n° 4 cachant ainsi une panne éventuelle sur la bascule maître. On en déduit que le test du vecteur 7 (C $\overline{J}$ K = 1) donne la condition la plus restrictive sur l'état (10) (isolement de BM par $\overline{Q}$ et C). Il faut lui ajouter un transfert du maitre dans l'esclave : 3 ($\overline{C}$ $\overline{J}$ K = 1) qui le rend observable sur les sorties.

Physiquement, J = 0v, K = 5,5 v, et comme C s'applique aux deux ET d'entrée, il faudrait d'une part (ET 1) : C = 5,5 v, et d'autre part (ET 2) C = 2, 4v pour ne pas contrarier J = 0. Dans ce cas on affichera une tension comprise entre 2,4 v et 5,5 v sur C.

Si l'élément répond correctement au vecteur d'entrée appliqué 7 (C $\overline{J}$ K = 1) on est sûr qu'il fonctionnera bien pour tous les autres vecteurs de cet état.

En raisonnant de même, on déterminera les vecteurs 5 (C $\overline{J}$ $\overline{K}$ = 1) et 6 (C J $\overline{K}$ = 1) dans les états respectifs (3) et (4)

De l'interprétation de ces spécifications, présentée ci-dessus sur un exemple résultent quelques constatations.

On peut dans chacun des états déterminer le ou les vecteurs qu'il est nécessaire et suffisant de tester, et savoir pour un vecteur donné quels rôles jouent les entrées qui le composent. Ceci permet de définir les tests très précisément.

D'autre part, dans chaque état, on cherche à placer l'élément dans les conditions les plus défavorables de fonctionnement, conditions telles que toute panne ne soit pas masquée par un affichage des entrées. Si on peut ainsi déterminer un vecteur complètement spécifié (une des combinaisons binaires des entrées), un seul affichage suffit. Si une des entrées est indifférente (par ex. C X K) il faudra deux affichages (C J K, C $\bar{J}$ K) pour que le test dans l'état considéré soit complet.

On conçoit alors qu'au lieu de tester chacune des cases de la table de fluence (§ II.3), on ne testera que quelques unes d'entre elles et que le nombre de tests total sur un élément va diminuer dans des proportions appréciables.

Cette recherche d'un PIRE CAS de fonctionnement, s'effectue ici à deux stades :

- pire cas logique
- pire cas électronique.

En effet, après avoir déterminé les niveaux à donner aux entrées d'un système, connaissant les caractéristiques électriques,on rend le test encore plus "affligeant" en imposant les limites de tensions admissibles.

Cela est facilité si on a à sa disposition des tests statiques du circuit. Cette remarque constitue la réciproque de ce que nous avions noté au chapitre I (§ I.2.1) et montre qu'il existe une liaison entre les tests statiques et fonctionnels (21).

IV-3 SYSTEMATISATION DE LA DEFINITION DES VECTEURS DE TOUTES LES TRANSITIONS

On peut représenter l'exemple précédent par le schéma de la fig. IV.4 a. Pour tester complètement ce circuit, il faut parcourir toutes les transitions :

- transitions entre états
- transitions rebouclées sur un état.

On en revient à établir des tests d'inscription-transfert et d'isolement.

Les équations $A = C\ K\ Q$ $\qquad D = \overline{Q_M . \bar{C}}$

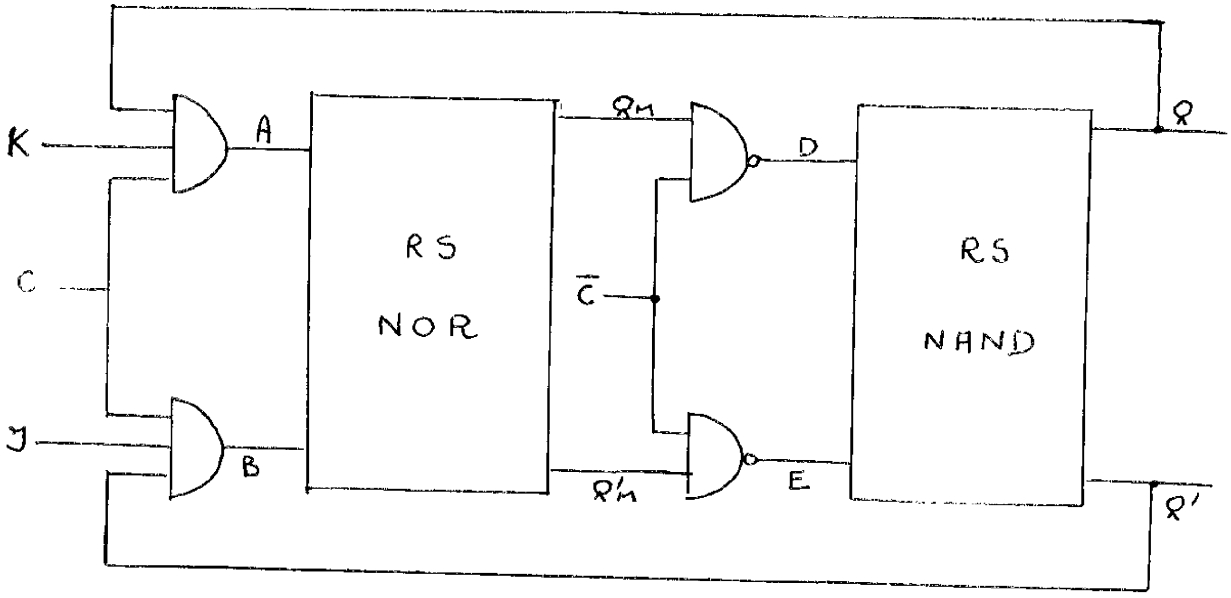
$B = C\ J\ Q'$ $\qquad E = \overline{Q'_m . \bar{C}}$

permettent de définir le comportement des variables A, B et D, E pour toutes les combinaisons possibles des commandes J, K, C, Q_M, Q'_M, Q, Q', et par conséquent de mettre en évidence quels sont les vecteurs commandes qui sont à l'origine de tel ou tel isolement ou inscription (ceux-ci étant définis par les tables de la fig. IV.4.b).

On commence par construire un tableau qui comporte autant de lignes que de combinaisons binaires des commandes. L'analyse de l'exemple choisi montre qu'il possède quatre états stables (fig. IV.2), pas d'état transitoire instable et que toujours les conditions $Q'_M = \bar{Q}_M$ et $Q' = \bar{Q}$ se trouvent respectées, dans l'hypothèse bien sûr où le système fonctionne correctement. Profitons de cette remarque pour noter qu'en matière de test de détection, on raisonnera toujours sur le circuit correct.

Il est alors possible de supprimer Q'_M et Q' ce qui porte à 32 le nombre de lignes du tableau résultant des cinq commandes : C, J, K, Q_M, Q (fig. IV.5, colonnes 1 à 5). En colonne 6 sont inscrits les vecteurs d'entrée sous forme codée (codage fig. IV.2).

Les quatre colonnes suivantes représentent les valeurs de A, B, D, E, pour chacune des combinaisons binaires de C, J, K, Q_M, Q.



(a)

Q_M \ AB	00	01	11	10
0	(0)	1	(0)	(0)
1	(1)	(1)	0	0

Q \ DE	00	01	11	10
0	1	1	(0)	(0)
1	(1)	(1)	(1)	0

(b)

figure IV.4

A ce niveau, on peut noter les lignes correspondant à un état stable (11ème colonne, chiffres cerclés). Dans les colonnes 12 et 13 sont inscrits pour chaque ligne les vecteurs adjacents à celui de la ligne considérée et les lignes correspondantes où on les retrouve. A partir du graphe de la fig. IV.2 il est possible de repérer les vecteurs de transition entre états; on les caractérise dans la colonne 11 en plaçant le numéro de l'état de départ suivi de la lettre I. Ceci permet d'achever le remplissage des colonnes 11, 12 et 13 et de passer à l'interprétation du tableau.

Interprétation :

Pour une ligne donnée les valeurs binaires situées dans les colonnes A, B et D, E donnent, à l'aide des deux tables de la fig. IV.4. b la signification du vecteur d'entrée, c'est-à-dire montrent son effet sur les bascules maître et esclave. On indique ceci sur le tableau en ajoutant deux colonnes.

La première concerne le maître. Comme deux éventualités peuvent se produire sur lui, on le subdivise en deux : isolement - inscription. La deuxième a trait à l'esclave et se trouve également divisée en deux : isolement - transfert.

Etats stables :

Examinons par exemple la ligne n° 9 :

On trouve A = B = 0 ⟶ isolement du maître

$A = C\ K\ Q$ avec C = 1, K = 0, Q = 0

$B = C\ J\ \overline{Q}$ avec C = 1, J = 0, $\overline{Q}$ = 1

L'isolement s'effectue par K = Q = J = 0

D = E = 1 ⟶ isolement de l'esclave

$D = \overline{Q_M \cdot \overline{C}}$ avec Q_M = 0 C = 1

$E = \overline{\overline{Q}_M \cdot \overline{C}}$ avec Q_M = 0 C = 1

⇒ isolement par C = 1

Dans tous les états stables, la procédure est identique.

ligne	Q_M	C	J	K	Q	transition par	CkR A	$CJ\bar{R}$ B	$\overline{Q_M}\bar{C}$ D	$\overline{\bar{Q}_M}\bar{C}$ E	Etats	Etats initiaux vecteurs	Etats initiaux lignes
1	0	0	0	0	0	1	0	0	1	0	②	2 3 5	5 3 9
2	0	0	0	0	1	1	0	0	1	0	4I	5	10
3	0	0	0	1	0	3	0	0	1	0	②	1 4 7	1 7 11
4	0	0	0	1	1	3	0	0	1	0	4I	7	12
5	0	0	1	0	0	2	0	0	1	0	②	1 4	1 7
6	0	0	1	0	1	2	0	0	1	0	4I	6	14
7	0	0	1	1	0	4	0	0	1	0	②	2 3	5 3
8	0	0	1	1	1	4	0	0	1	0	4I	8	16
9	0	1	0	0	0	5	0	0	1	1	②	1 7	1 11
10	0	1	0	0	1	5	0	0	1	1	④	6 7	14 12
11	0	1	0	1	0	7	0	0	1	1	②	3 5	3 9
12	0	1	0	1	1	7	1	0	1	1	④	5 8	10 16
13	0	1	1	0	0	6	0	1	1	1	2I	2 5	5 9
14	0	1	1	0	1	6	0	0	1	1	④	5 8	10 16
15	0	1	1	1	0	8	0	1	1	1	2I	4 7	7 11
16	0	1	1	1	1	8	1	0	1	1	④	6 7	14 12
17	1	0	0	0	0	1	0	0	0	1	1I	5	25
18	1	0	0	0	1	1	0	0	0	1	③	2 3 5	22 20 26
19	1	0	0	1	0	3	0	0	0	1	1I	7	27
20	1	0	0	1	1	3	0	0	0	1	③	1 4	18 24
21	1	0	1	0	0	2	0	0	0	1	1I	6	29
22	1	0	1	0	1	2	0	0	0	1	③	1 4 6	18 24 30
23	1	0	1	1	0	4	0	0	0	1	1I	8	31
24	1	0	1	1	1	4	0	0	0	1	③	2 3	22 20
25	1	1	0	0	0	5	0	0	1	1	①	6 7	29 27
26	1	1	0	0	1	5	0	0	1	1	③	1 6	18 30
27	1	1	0	1	0	7	0	0	1	1	①	5 8	25 31
28	1	1	0	1	1	7	1	0	1	1	3I	3 5	20 26
29	1	1	1	0	0	6	0	1	1	1	①	5 8	25 31
30	1	1	1	0	1	6	0	0	1	1	③	2 5	22 26
31	1	1	1	1	0	8	0	1	1	1	①	6 7	29 27
32	1	1	1	1	1	8	1	0	1	1	3I	4 6	24 30
colonne	1	2	3	4	5	6	7	8	9	10	11	12	13

Bascule MAITRE		conduit à la	Bascule ESCLAVE		conduit à la
isolement par	inscription par	ligne	isolement par	transfert de par	ligne
C = 0				$Q_M = 0$ inopérant	
C = 0				$Q_M = 0$ C=0	1
C = 0				$Q_M = 0$ inopérant	
C = 0				$Q_M = 0$ C=0	3
C = 0				$Q_M = 0$ inopérant	
C = 0				$Q_M = 0$ C=0	5
C = 0				$Q_M = 0$ inopérant	
C = 0				$Q_M = 0$ C=0	7
J = Q = K = 0			C = 1		
J = K = $\bar{Q}$ = 0			C = 1		
J = Q = 0			C = 1		
	0 inopérante CKQ=1		C = 1		
	1 CJ$\bar{Q}$=1	29	C = 1		
K = $\bar{Q}$ = 0			C = 1		
	1 CJ$\bar{Q}$=1	31	C = 1		
	0 inopérante CKQ=1		C = 1		
C = 0				$Q_M = 1$ C=0	18
C = 0				$Q_M = 1$ inopérant	
C = 0				$Q_M = 1$ C=0	20
C = 0				$Q_M = 1$ inopérant	
C = 0				$Q_M = 1$ C=0	22
C = 0				$Q_M = 1$ inopérant	
C = 0				$Q_M = 1$ C=0	24
C = 0				$Q_M = 1$ inopérant	
J = K = Q = 0			C = 1		
J = K = $\bar{Q}$ = 0			C = 1		
J = Q = 0			C = 1		
	0 CKQ=1	12	C = 1		
	1 inopérante CJ$\bar{Q}$=1		C = 1		
K = $\bar{Q}$ = 0			C = 1		
	1 inopérante CJ$\bar{Q}$=1		C = 1		
	0 CKQ=1	16			

figure IV.5

Transitions : Elles résultent soit d'une inscription dans le maître
soit d'un transfert dans l'esclave

Ex : ligne 13

$Q_M = Q = 0$

$A = 0$, $B = 1 \longrightarrow$ inscription dans le maître.

$B = 1$ avec $C = 1$, $J = 1$, $\overline{Q} = 1$ amène l'état 2 à 1. On trace une colonne supplémentaire pour indiquer la ligne où amène la transition. La lecture du tableau dans ce cas, s'effectue comme suit :

de l'état (2) le vecteur 6 (ligne 13) adjacent aux vecteurs 2 et 5 (lignes 7 et 9) amène dans l'état (1) (ligne 29). L'inscription dans le maître est due à $C\ J\ \overline{Q} = 1$, l'isolement de l'esclave à $C = 1$.

La construction complète du tableau indique pour chaque vecteur de chaque état, les variables qui participent de façon active aux isolements et transferts, c'est-à-dire le fonctionnement précis du circuit.

Recherche des vecteurs à tester

Elle s'effectue également sur le tableau.

Exemple : 1 - Etat 2

Isolement : Les vecteurs 1, 2, 3, 4, 5, 7 assurent l'isolement. Mais les vecteurs les plus fragiles sont 5 et 7 puisque 1, 2, 3, 4 isolent le maître de façon inconditionnelle grâce à $C = 0$. On testera 5 et 7. Il n'est pas possible ici de faire un choix sur l'un de ces deux vecteurs ce qui montre que la décomposition d'un circuit en blocs élémentaires affecte la précision des résultats.

Inscription : les deux vecteurs inscription 6, 8 (lignes 13 et 15) doivent être testés.

2 - Etat 1

Isolement : Les vecteurs 6 et 8 correspondent à une inscription

Etat 2 $Q_M = 0$, $Q = 0$

6, 8 : inscription 1 dans BM par C J $\overline{Q} = 1$ (vérifié par 1 2 3 4 état 1)
isolement de BE à 0 par C = 1

5, 7 : isolement par J = Q = 0 de BM

1, 2, 3, 4, : transfert 0 inopérant dans BE (vérification du test de 5 et 7)

Etat 1 $Q_M = 1$, $Q = 0$

5, 7 : isolement par J= Q = 0 de BM

6, 8 : inscription 1 inopérante dans BM

1, 2, 3, 4 : transfert de $Q_M = 1$ dans BE par C = 0 (conduit dans l'état 3)
+ vérification de 5 et 7.

Etat 3 $Q_M = 1$, $Q = 1$

5, 6 : isolement de BM par K = $\overline{Q}$ = 0

7, 8 : inscription 0 dans BM par C K Q = 1 (vérifié par 1 2 3 4 état 4)
isolement de BE à 1 par C = 1

1, 2, 3, 4 : vérification de 5 et 6

Etat 4 $Q_M = 0$, $Q = 1$

7, 8 : inscription 0 inopérante dans BM

5, 6 : isolement de BM par K = $\overline{Q}$ = 0

1, 2, 3, 4 : transfert de $Q_M = 0$ dans BE par C = 0 (conduit dans l'état 2)
+ vérification de 5 et 6.

Bascule JK 7473 - Liste des vecteurs à tester ( )

- Figure IV.6 -

inopérante dans le maître. Ils ne sont pas à tester puisqu'ils l'ont déjà été lors de la transition (2)→(1)

Les vecteurs 5, 7 font l'objet d'un test d'isolement. Ici aussi il y a une perte de précision puisqu'on avait trouvé au paragraphe précédent qu'il suffisait de tester le vecteur 7.

Transfert : les vecteurs 1, 2, 3, 4 constituent le test complet de transfert de l'état (1) à (3) .

La figure IV.6 donne les résultats complets relatifs à cet exemple.

Cette méthode permet une très bonne compréhension du fonctionnement du système mais manque de précision en ce qui concerne la recherche des vecteurs critiques nécessaires et suffisants à tester. De plus, l'établissement du tableau est assez long et semble difficile à programmer. Elle illustre cependant de façon simple l'existence de vecteurs particuliers à tester, et associée à l'interprétation des spécifications présentées plus haut, elle nous a guidé dans la mise au point de la méthode que nous allons maintenant exposer.

IV-4 METHODE DU PIRE CAS

IV.4.1. Hypothèse

Sous sa forme la plus générale, l'équation booléenne d'une veriable secondaire peut s'écrire :

$$y_i(T+1) = A.\, y_i(T) + B + C\, \overline{y_i}(T) \qquad \text{IV.1}$$

avec

$$\begin{cases} A = f_1\,(ek,\ y_j\,(T)) \\ B = f_2\,(ek,\ y_j(T)) \\ C = f_3\,(ek,\ y_j\,(T)) \end{cases} \qquad \begin{matrix} e = \text{entrée} \\ j \neq i \end{matrix} \qquad \text{IV.2}$$

Or les circuits que nous étudions sont composés de bascules RS élémentaires dont une des sorties de chacune d'elles joue le rôle de variable secondaire. Leurs équations (§ I.1.4.1.) ne comportent pas le terme C. $\overline{y}$ (T). Mais de par la présence de bouclages dans le circuit, il peut apparaître d'autres variables secondaires non liées aux bascules élémentaires pour lesquelles on ne doit pas à priori exclure le terme C $\overline{y}$ (T).

Examinons quelle est sa contribution dans l'équation générale.

A	B	C	y (T+1)
0	0	0	0
0	0	1	$\overline{y}$ (T)
0	1	0	1
0	1	1	1
1	0	0	y (T)
1	0	1	y(T) + $\overline{y}$(T)
1	1	0	1
1	1	1	1

Lorsque C = 0, on a $\begin{cases} y(T+1) = 1 \text{ si } B = 1 \\ y(T+1) = 0 \text{ si } A = 0 \text{ et } B = 0 \\ y(T+1) = y(T) \text{ si } A = 1 \text{ et } B = 0 \end{cases}$

B réalise l'inscription 1, A l'isolement (A = 1) et le transfert 0 (A = 0) quand B vaut 0. Ces deux termes suffisent donc à décrire le fonctionnement normal d'un système.

Lorsque C = 1, on a : $y(T+1) = \overline{y}(T)$ si A = B = 0, qui est la caractéristique d'un cycle.

$y(T+1) = 1$ si B = 1. La contribution apportée ici par C$\overline{y}$ n'apparait pas puisque B = 1.

$y(T+1) = y(T) + \overline{y}$ (T) si A = 1 et B = 0.

Il y a ici un risque d'apparition d'aléas statiques, à moins que $B \supset AC$ ((2) p. 371) . Ceci est visible sur les équations lorsque celles-ci n'ont pas été minimisées par les méthodes classiques de simplification.

En effet, si les équations sont simplifiées, nécessairement les termes A et C seront disjoints (A. C = 0). On n'aura jamais A = C = 1 et par conséquent le cas $y(T+1) = y(T) + \overline{y}(T)$ n'interviendra pas.

Donc lorsqu'on effectue l'Analyse d'un circuit séquentiel, et qu'au moment de l'établissement des équations on utilise un algorithme de simplification, on ne tient plus compte de la présence éventuelle d'aléas statiques et il est de plus possible que ces équations ne représentent plus le circuit originel du point de vue de la sensibilité à une panne. Cette remarque mérite d'être approfondie lorsqu'on utilise des circuits où le terme $C\overline{y}$ apparait et des méthodes de test qui travaillent à partir des équations logiques.

Sur les exemples que nous avons traités, le terme $C\overline{y}$ n'est jamais apparu. Pour cette raison, au cours de la méthode que nous allons exposer, il n'a pas été assigné, ni programmé. Dans un cas plus général, il faudra prendre en compte cette remarque et y remédier, ce qui ne pose aucune difficulté de principe. De plus, il faudra fournir au programme d'établissement des tests fonctionnels, les équations non simplifiées du système. (ne pas oublier que le programme d'analyse décrit au chapitre III donne des équations minimisées).

Nous écrirons ici :

$$y(T+1) = A.\ y(T) + B \qquad \text{IV.3}$$

IV.4.2. Exemple

Reprenons le même exemple afin de comparer les résultats obtenus. Les sorties des modules 9 et 13 (fig. IV.1), constituent les variables secondaires du circuit. Dans les équations logiques ci-dessous, l'influence de la variable de remise à zéro : $\overline{RZ}$, a été supprimée ($\overline{RZ} = 1$) (pour simplifier).

$$\begin{cases} Y_9(T+1) = \left(\overline{J} + \overline{C} + Y_{13}(T)\right) \cdot Y_9(T) + K\ C\ Y_{13}(T) = Q_M, \\ Y_{13}(T+1) = \left(\overline{Y_9}(T) + C\right) \cdot Y_{13}(T) + \overline{C}\ \overline{Y_9}(T) = Q \end{cases}$$

A l'aide de ces seules équations, nous allons chercher s'il est possible pour chaque transition (bouclée et entre états) de distinguer des vecteurs particuliers assurant le test de cette transition.

ETAT 2 $\quad Q_M = 0,\ Q = 0 \quad$ ou $Y_9(T) = 1,\ Y_{13}(T) = 0$

La transition bouclée sur cet état se matérialise par six vecteurs et à priori on ne pourra dire avec certitude que cette transition s'effectue correctement qu'après les avoir testés successivement.

On a :

$$\begin{cases} Y_9(T+1) = A_1\, Y_9(T) + B_1 & \text{avec} \quad \begin{cases} A_1 = \overline{J} + \overline{C} + Y_{13}(T) \\ B_1 = K\, C\, Y_{13}(T) \end{cases} \\ Y_{13}(T+1) = A_2\, Y_{13}(T) + B_2 & \text{avec} \quad \begin{cases} A_2 = \overline{Y_9}(T) + C \\ B_2 = \overline{C}\, \overline{Y_9}(T) \end{cases} \end{cases}$$

Pour que les conditions $\begin{cases} y_9(T+1) = Y_9(T) = 1, \\ Y_{13}(T+1) = Y_{13}(T) = 0 \end{cases}$ se vérifient,

il faut obligatoirement $B_2 = 0$ et A_1 ou $B_1 = 1$. Cette seule contrainte suffit mais il est possible d'être beaucoup plus exigeant. En effet, si $B_1 = 1, \Rightarrow Y_9(T+1) = 1$ indépendamment de tout état antérieur. La conséquence est que toute panne dans la portion de circuit que représente Y_9 se trouvera masquée par la valeur de B_1. Par contre en plaçant $B_1 = 0$ et $A_1 = 1$, on sensibilise $Y_9(T+1)$ à son état antérieur. Il en est de même pour $Y_{13}(T+1)$. Mettre $A_2 = 0$ impose la valeur 0 alors que $A_2 = 1$ pourra dévoiler l'existence d'une panne ayant pour résultat $Y_{13}(T) = 1$. (Coupure du fil de bouclage, collage d'un module dont l'effet équivaut à $Y_{13}(T) = 1$ ..)

Il vient : $\begin{cases} A_1 = 1,\ B_1 = 0 \\ A_2 = 1,\ B_2 = 0 \end{cases}$ qui sont d'ailleurs les conditions d'un véritable isolement, toute autre configuration étant représentative d'une inscription inopérante.

Il est bien évident qu'une panne qui bloquerait B_1 à 1 ne serait pas détectable dans cet état, mais elle le deviendra dans un état où $Y_9(T) = 0$ et $Y_9(T+1) = 0$.

Par ces assignements donnés à A et B, on se rapproche des constatations faites au § IV.2 à savoir montrer l'existence dans toutes les transitions de vecteurs les plus restrictifs capables de sensibiliser le circuit par leur test à toutes les pannes possibles. Nous appellerons ceci : la recherche du pire cas.

Reprenons les assignements ci-dessus en essayant de les rendre encore plus restrictifs.

$A_1 = 1$ réalise l'isolement désiré. On va rendre cette condition la plus fragile possible en plaçant le minimum de monômes de A à la valeur 1 par un choix judicieux des variables d'entrée qui interviennent. On notera - $A_1 = 1-$

Pour la même raison, on rendra la condition $B_1 = 0$ la plus solide possible, soit - $B_1 = 0+$

Ainsi si à partir de l'état 2 la variable secondaire Y_9 soumise au vecteur que nous allons calculer garde la valeur 1 on a la certitude qu'elle satisfera aux autres vecteurs, moins éprouvants.

Un raisonnement identique conduit à écrire que le pire cas pour $Y_{13}(T+1) = Y_{13}(T) = 0$ s'obtient par le "meilleur 1" sur A_2, soit $A_2 = 1+$, et le "moins bon zéro" sur B_2 : $B_2 = 0-$.

Nous devons maintenant afficher des valeurs aux trois entrées J, K et C afin de satisfaire au mieux : $A_1 = 1-$, $B_1 = 0+$
$A_2 = 1+$, $B_2 = 0-$

- $A_1 = 1-$: comme $Y_{13}(T) = 0 \longrightarrow$ J = 0 ou C = 0 et si possible une seule de ces deux variables à 0.

- $B_1 = 0+$: $B_1 = 0$ est déjà satisfait par $Y_{13} = 0$. On renforce cette condition par K = 0 ou C = 0

- $A_2 = 1+$: $\overline{y_9} = 0 \longrightarrow$ C = 1

- $B_2 = 0-$: $\overline{y_9} = 0 \longrightarrow$ C = 0

On voit qu'après l'assignement des différentes variables d'entrée, se pose un problème de compatibilité. En effet C apparait sous les deux formes 0 et 1.

Nous donnerons au paragraphe suivant des critères systématiques de compatibilité. Raisonnons ici de façon intuitive :

J = 0 et K = 0 sont acquis sans ambiguité. Comme J = 0, la condition 1 - sur A_1 sera d'autant mieux réalisée que l'on mettra C = 1. Si l'on veut respecter la condition 1+ sur A_2, C = 1 est obligatoire, tandis que C = 0 n'est que souhaitable dans B_1 et B_2 puisque d'une part Y_{13} et K sont à 0 et d'autre part $\overline{Y}_9 = 0$.

On choisira donc le vecteur $\overline{J}\,\overline{K}\,C = 1$ (5), qui représente le pire cas de fonctionnement du circuit dans l'état 2, résultat en accord avec celui du § IV.2.

IV.4.3. Généralisation et systématisation : Méthode du pire cas

Nous essayerons, au cours de l'exposé qui suit, de caractériser les notions introduites et les algorithmes par des exemples aussi simples que possible. Dans l'annexe II, on trouvera un exemple d'application de cette méthode. Il ne permettra pas d'enviseger tous les cas possibles mais, espérons le, clarifiera la démarche suivie.

La première partie consiste à déterminer successivement pour chacune des équations logiques des variables secondaires, les assignements des entrées réalisant les conditions de pire cas, et à donner à ces assignements des priorités dont nous définirons les critères.

Ceci est indispensable par le fait que les assignements se font sur chaque variable secondaire sans tenir compte des autres. C'est dans la deuxième partie que l'on étudie la compatibilité de ces assignements entre eux. Par le jeu du codage choisi nous passerons des formes $\sum\pi$ à $\prod\sum$ afin que chaque variable d'entrée se situe dans une colonne de la liste des assignements. Il faudra

bien entendu que la variable n'y apparaisse que sous une seule forme. Dans le cas contraire, nous devons rendre compatible les assignements, compte tenu des critères de priorité choisis.

IV.4.3.1. Critères d'assignements et de priorités

Avant de donner les critères d'assignements des termes A et B, nous allons écrire leur forme générale, afin d'aboutir à un type d'équation sur laquelle s'effectuera l'assignement des entrées primaires, seules commandables.

1. Forme générale des termes A et B de $Y_i(T+1) = A\ Y_i(T) + B$.

A et B sont des fonctions des entrées et des variables secondaires autres que y_i

$$A = f_1\ (ek,\ y_j)$$
$$B = f_2\ (ek,\ y_j)$$

Trois types de termes apparaissent dans l'expression booléenne résultante :

termes comportant
- des combinaisons d'entrées : E
- des combinaisons de variables secondaires : Z
- des combinaisons de variables secondaires et d'entrées.

On écrira : $A = E_1 + Z_1 + E_2 . Z_2$

La forme la plus générale que peuvent prendre E_1 et E_2, en fonction de k entrées $(e_1 \ldots\ldots ek)$ est :

$$\begin{matrix} E_1 \\ E_2 \end{matrix} = \sum_{i=1,k} a_i e_i^{\alpha_i} + a_i e_i^{\alpha_i} . \sum_{j=i+1,k} a_j e_j^{\alpha_j} + a_i e_i^{\alpha_i} . a_j e_j^{\alpha_j} . \sum_{l=j+1,k} a_l e_l^{\alpha_l} \ldots\ldots$$

$\sum$ indiquant la somme booléenne (réunion, OU inclusif)

Les ai sont des coefficients valant 0 ou 1 selon que le terme correspondant ei existe ou non.

αi signifie pour $\alpha i = 0$, $a_i e_i^{\alpha_i} = a_i \bar{e_i}$

pour $\alpha i = 1$, $a_i e_i^{\alpha_i} = a_i e_i$

En posant :

$$\begin{cases} X_1 = \sum_{i=1,k} a_i e_i^{\alpha_i} \\ X_2 = \sum_{i=1,k} a_i e_i^{\alpha_i} \cdot a_j e_j^{\alpha_j} \quad (j = i+1, k) \\ - \\ - \\ - \end{cases}$$

On obtient une écriture plus condensée de E_1 et E_2.

$$\begin{matrix} E_1 \\ E_2 \end{matrix} = \sum_{m=1,k} X_m$$

De la même façon :

$$\begin{matrix} Z_1 \\ Z_2 \end{matrix} = \sum_{i=1,j} b_i y_i^{\alpha_i} + b_i y_i^{\alpha_i} \cdot \sum_{l=i+1,j} b_l y_l^{\alpha_l} + \cdots$$

$$= \sum_{m=1,j} Y_m \qquad \text{avec} \qquad \begin{cases} Y_1 = \sum_{i=1,j} b_i y_i^{\alpha_i} \\ Y_2 = \sum_{i=1,j} b_i y_i^{\alpha_i} \cdot \sum_{l=i+1,j} b_l y_l^{\alpha_l} \\ - \\ - \end{cases}$$

A l'aide de ces notations, A et B peuvent s'écrire :

$$\begin{matrix} A \\ B \end{matrix} = E_1 + \sum_{m=1,j} Y_m + \sum_{p=1,j} \sum_{l=1,k} X_l \cdot Y_p$$

Le nombre maximum de termes de E_1 et E_2 est :

$$N = C_n^1 + C_n^2 + \ldots + C_n^p + C_n^n \qquad \text{avec } n = k$$

soit $N = 2^k - 1$

Représentons E_1 et E_2 sous la forme intersection de monômes ($\prod \sum$), déduite de la forme $\sum \prod$ précédente.

$$\begin{matrix} E_1 \\ E_2 \end{matrix} = \prod_P \sum_{i=1,n} a_i \, e_i^{\alpha_i}$$

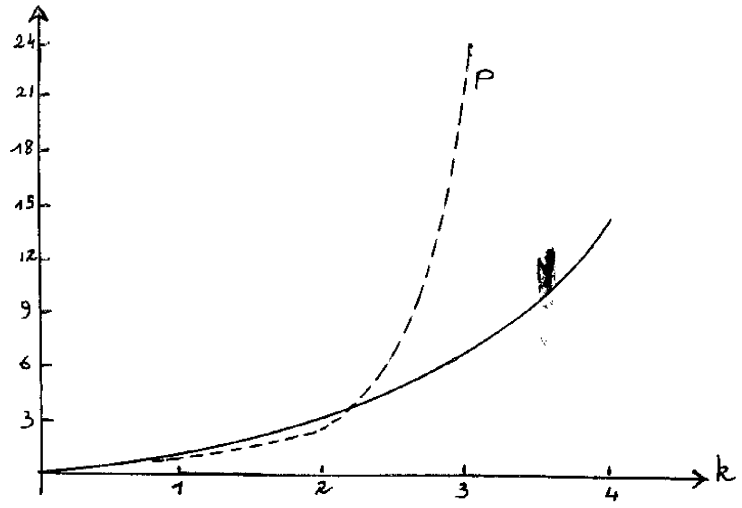
chaque $\sum a_i \, e_i^{\alpha_i}$ ayant au plus N variables ($n_{max} = N$)

Le nombre P de termes $\sum a_i \, e_i^{\alpha_i}$ est égal au produit du nombre de variables de chacun des monômes de la forme $\prod \sum$, soit :

$$P = 1^{C_n^1} \times 2^{C_n^2} \times 3^{C_n^3} \times \ldots\ldots (n-1)^{C_n^{n-1}} \times n$$

ici $n = k$ soit $P = k \times 2^{C_k^2} \times 3^{C_k^3} \times \ldots\ldots \times (k-1)^{C_k^{k-1}}$

K	N	P
1	1	1
2	3	2
3	7	24
4	15	20736



On voit qu'en passant sous la forme $\Pi\sum$ à partir de la forme $\sum\Pi$ le nombre de termes produits croît très vite avec le nombre d'entrées primaires. Ces valeurs N et P constituent des bornes supérieures.

Exemples :

1) $Y = e_1 + e_2 + e_1 e_2$ On a ici k = 2 , N = 3

en passant sous la forme $\Pi\sum$ on obtient :

$$Y = (e_1+e_1+e_2) \cdot (e_1+e_2+e_2) \qquad P = 2$$

2) $Y = A B + C D + E F$ avec N = 3 et k = 6

sous forme $\Pi\sum$:

$$Y=(A+C+E)\cdot(A+C+F)\cdot(A+D+E)\cdot(A+D+F)\cdot(B+C+E)\cdot(B+C+F)\cdot(B+D+E)\cdot(B+D+F).$$

On a P = 8, la borne supérieure est loin d'être atteinte.
Par contre, la borne supérieure du nombre de variables de chacun des facteurs est atteinte puisque 3 variables apparaissent dans chacun d'eux.

2. Assignements de A et B

Nous allons donner les valeurs à imposer à A et B afin de réaliser le pire cas. Celles-ci sont déduites d'une interprétation physique du

fonctionnement des circuits (voir § IV.4.2.) traduite sur les équations. Pour cette raison et afin d'éviter une étude longue sur un ou plusieurs exemples, nous indiquerons les résultats, sans les justifier davantage.

Cas des transitions bouclées (isolement)

. si y(Γ) = 0 $\begin{cases} A = 1+ \\ B = 0- \end{cases}$

. si y(Γ) = 1 $\begin{cases} A = 1- \\ B = 0+ \end{cases}$ réalisent le pire cas.

- Cas des transitions entre états (inscriptions, transferts)

. passage de y(Γ) de 0 à 1 : B = 1-, A quelconque

. passage de y(Γ) de 1 à 0 $\begin{cases} A = 0- \\ B = 0- \end{cases}$

Si une des variables secondaires ne change pas dans la transition, on est ramené aux conditions ci-dessus sur les transitions bouclées pour cette variable.

Nous verrons après avoir explicité les assignements de A et B, que dans certains cas où ils ne sont pas réalisables, on doit apporter des modifications.

3. Assignement des entrées

Reprenons l'expression générale des fonctions A et B.

$$\begin{matrix} A \\ B \end{matrix} = E_1 + \sum_{n=1,j} Y_n + \sum_{p=1,j} \sum_{\ell=1,k} X_\ell Y_p$$

$$= E_2 + \sum_{n=1,j} Y_n + \sum_{p=1,j} E_p Y_p$$

Condition 1+ :

On l'obtient en imposant $\begin{cases} E_1 = 1 \\ \text{tous les } Ep = 1 \end{cases}$ $\forall$ Y_n et Y_p

or : $E_1 = \sum_{m=1,k} X_m$ Il faut qu'au moins un des $X_m = 1$ ou sur la

E_p

forme $\Pi \sum$, que chacun des facteurs $\sum a_i e_i^{\alpha i}$ égale 1, condition réalisée si au moins un des $a_i e_i^{\alpha i} = 1$ pour chaque somme.

Par exemple si E = A B + C D = 1, il faut que chacun des termes entre parenthèses de E = (A+C) (A+D) (B+C) (B+D) = 1 soit égal à 1.

Condition 0+ :

1) Si au moins un des $Y_n = 1 \longrightarrow$ impossibilité

2) $\sum Y_n = 0$, $\forall$ $Y_p \longrightarrow$ $E_1 = 0$, tous les $E_p = 0$

Tous les X_m doivent être nuls. En prenant la forme complémentée en $\Pi \sum$ cela revient à dire que tous les $\sum$ doivent être égaux à 1.

Ex : A B + C D = 0 $\qquad$ $\underbrace{(\overline{A} + \overline{B})}_{1}$ $\underbrace{(\overline{C} + \overline{D})}_{1}$ = 1

Ce passage à la forme $\Pi \sum$ trouve son intérêt au niveau de l'algorithme (rangement des variables par colonne).

Condition 1- :

- Si l'équation IV.3 se réduit à Y (T+1) = Y(T) + B, cette condition devient impossible. Physiquement, cela correspond à un circuit mal conçu puisque dès que Y(T) = 1, Y(T+1) reste bloqué à cette valeur.

- $\sum Y_n = 0$

  - si $\sum Y_p = 0$ $\qquad$ $E_1 = 1$

    E_p quelconque (pas d'assignement)

- si $\sum Y_p = 1$ $\quad E_1 = 0$
  Pour ceux des Y_p qui valent 1 : $E_p = 1$
  Pour ceux des Y_p qui valent 0 : E_p non assignés

- $\sum Y_n = 1$
  - si $\sum Y_p = 0$ $\quad E_1 = 0$
    E_p non assignés
  - si $\sum Y_p = 1$ $\quad E_1 = 0$
    $E_p = 1$ si $Y_p = 1$
    E_p non assigné si $Y_p = 0$

  S'il n'y a pas de terme E_1 et que $\sum Y_h = \sum Y_p = 0 \rightarrow$ impossibilité.

Condition 0- :

Si $\sum Y_n = 1$: impossibilité

$\sum Y_n = 0$

$E_1 = 0$
si $\sum Y_p = 0$ chacun des $E_p = 1$
si $\sum Y_p = 1$ $\quad E_p = 0$ quand $Y_p = 1$
$\quad E_p = 1$ quand $Y_p = 0$

Cas impossibles :

Nous venons de voir, en détaillant les assignements sur A et B que pouvaient exister des cas d'impossibilité. Examinons ce qu'il en résulte sur la fonction $Y(T+1)$

Transitions bouclées :

. $y_i(T) = 0$. si B = 0- n'est pas respecté (B = 1 par $\sum Y_n = 1$) il s'ensuit que $y_i(T+1) = 1$. On en conclut que l'état auquel participe y_i n'est pas stable (mauvaise description du circuit)

. $y_i(T) = 1$. si on a A = 0 au lieu de A = 1- (pas de terme E_1 quand $\sum Y_n = \sum Y_p = 0$) il faut reporter la condition sur B = B=1-

. Si B = 1 il est inutile de placer A = 1-. On inverse les deux conditions en imposant $\begin{cases} A = 0+ \\ B = 1- \end{cases}$

Justification : si B = 1 la cause vient du terme $\sum Y_n$ qui vaut 1. On essayera donc de le rendre aussi fragile que possible (1-) et pour ne pas masquer une panne éventuelle sur les Y_n on affichera A = 0+ sensibilisant ainsi $Y_i(T+1)$ aux valeurs des autres variables secondaires.

Transitions entre états :

Les assignements imposés doivent impérativement être remplis. Cependant, si au cours de la transtion d'un état à un autre, plusieurs variables secondaires changent de valeur, cela peut nécessiter plusieurs étapes (passage par des états transitoires).

4. Critères de priorité.

A la fin du § IV.4.1. après que l'assignement des entrées ait été effectué nous nous sommes aperçus que des incompatibilités pouvaient apparaître en ce sens qu'une variable devait à la fois prendre les valeurs 0 et 1. Pour que cette méthode, destinée à trouver son application par l'écriture d'un programme, soit systématique, il faut définir des critères de priorité à partir desquels on bâtira l'étude de la compatibilité.

Revenons sur les assignements donnés à A et B.

Dans le cas des transitions bouclées, la condition 0- est absolument indispensable. Il en est de même en ce qui concerne la condition 1- si l'on veut réaliser le pire cas, mais nous avons vu que parfois elle pouvait se reporter sur un autre terme. Quant aux conditions 1+ et 0+ elles sont souhaitables dans le sens du pire cas mais une impossibilité ou une modification telle que ces assignements deviennent moins forts ne porte pas à conséquence.

Dans le cas des transitions entre états, 0- et 1- ne doivent pas être mises en défaut.

L'incompatibilité la plus génante consiste en un désaccord sur une même variable sollicitée simultanément aux conditions 0- et 1-. Conformément à ce que nous venons de remarquer, on ne touchera pas à l'assignement issu de 0- puisque cette condition doit impérativement être remplie, et on essayera de supprimer l'incompatibilité en rendant la condition 1- un peu moins forte. Nous détaillons la démarche suivie en expliquant l'algorithme. En conclusion, les priorités s'établiront comme suit :

- Transitions bouclées :

condition	:	0-	1-	$\begin{bmatrix}1+\\0+\end{bmatrix}$ dans un ordre
degré de priorité	:	3 ou 2	1	0 décroissant

- Transitions entre états

condition	:	$\begin{bmatrix}0-\\1-\end{bmatrix}$	$\begin{bmatrix}1+\\0+\end{bmatrix}$
degré de priorité	:	3 ou 2	0

Mais lorsqu'une condition impose une priorité, il ne faut pas l'appliquer à tous les termes de A et B.

0- Priorité aux t_q = 0 car c'est grâce à eux que 0-
t_p = 0 est respectée

Les t_q = 0 ont une priorité de degré 0. Ils ne sont en effet pas indispensables, mais améliorent le pire cas.

Aux termes de t_q et t_p ($\sum a_i e_i^{\alpha_i}$ de la forme générale,) ne faisant intervenir qu'une seule variable d'entrée, sera affectée une priorité de degré 3.

Aux termes produits d'intersections de variables d'entrées primaires : priorité de degré 2 (explications au § IV.4.3.2.)

1- Priorité de degré 1 sur
- E_p = 1 correspondant aux Y_p valent 1, car E_1 étant placé à 0 pour le pire cas, E_p devient discriminant.

- $E_1 = 1$ lorsque $\sum Y_p$ et $\sum Y_n = 0$. C'est en effet le seul terme qui peut assurer la condition 1-

Dans le cas des transitions entre états, priorité de degré 3 sur ces mêmes termes, puisque la condition 1- devient absolument indispensable (inscription 1)

1+
0-

Priorité de degré 0 car ces conditions ne sont souhaitables que pour améliorer le pire cas.

IV.4.3.2. Etude de la compatibilité

Elle s'effectue sur la base des critères d'assignements et de priorité définis ci-dessus. Elle a été séparée des critères de priorité car ceux-ci sont établis en même temps que les assignements. La liste des monômes obtenus, classée par ordre décroissant de priorité, est traitée colonne par colonne grâce à l'emploi des formes $\Pi\sum$.

C'est à ce moment que l'on tient compte des contradictions possibles sur une même variable, et que l'on va les corriger pour arriver à la définition correcte de chacune des variables. Il faut donc prévoir toutes les incompatibilités possibles en fonction des critères de priorité, le report de certaines conditions, le changement de priorité éventuel d'un terme, ceci afin d'améliorer encore, dans la mesure du possible, la recherche du pire cas en tenant compte de l'intéraction des variables secondaires, les unes sur les autres.

Cette partie est liée bien sûr aux notions de priorité mais nous l'avons présentée à part car elle remet constamment en cause les assignements et priorités définis à l'origine, jusqu'à ce que la dernière colonne (correspondant à la dernière variable d'entrée), comporte une seule valeur de la variable (monoforme). Il faut nécessairement arriver à une solution puisque la transition pour laquelle on cherche le vecteur (complètement spécifié ou non) à tester, existe.

Comme la méthode utilise un procédé algorithmique, nous en donnerons les détails au § IV.4.4.3 et noterons ici quelques points particuliers.

La distinction entre les priorités de degré 2 et 3 s'explique de la façon suivante :

Les termes de priorité 2 comportent plusieurs variables. Si un conflit apparait sur une variable appartenant à la fois à un terme de priorité 2 et à un terme de priorité 3, il sera réglé en supprimant l'assignement de cette variable dans le terme de priorité 2 puisqu'il reste d'autres variables capables d'assurer la condition 0-.

Exemple : $(e_1)_3 \quad (\bar{e}_1\ e_2\ \bar{e}_3)_2$

Ces termes devant valoir 0 dans la condition 0-, on affecte les valeurs des entrées en passant sous forme $\prod \sum$: $(\bar{e}_1)_3 = 1 \quad (e_1 + \bar{e}_2 + e_3) = 1$ et en notant $(0)_3$, $(1 + 0 + 1)_2$

En supprimant la première valeur dans le deuxième terme, on réalise la compatibilité. Les conditions 0- existent toujours et cette opération a rendu la deuxième plus forte (0 plus fragile) puisque moins de variables la satisfont.

Pour une raison identique, chaque fois qu'une variable apparaissant dans un terme caractérisé par une priorité 2 ou 1 ne présente pas d'incompatibilité (soit directement, soit après traitement), toutes les autres variables de ce terme verront leur valeur inversée dans le but d'améliorer encore le pire cas.

IV.4.4. Algorithmes

Afin de diminuer le nombre d'instructions du programme et par souci de clarté, les algorithmes ont été élaborés de façon modulaire. Nous essayerons de les expliquer en appliquant le même principe et en nous aidant d'organigrammes généraux. Il n'est en effet pas question de rentrer dans les détails des organigrammes à cause de leur complexité.

IV.4.4.1. Procédure générale (planche IV.1 ⓐ)

Elle se divise en deux grands blocs :

- assignements et priorités
- compatibilité

A partir de chacun des états stables du système définissant les valeurs initiales des variables secondaires cette procédure s'effectue successivement sur les transitions bouclées et sur les transitions entre états, celles-ci étant données par la matrice des transitions sous forme codée (§ III.2.2.11)

Si une transition entre états se fait au moyen d'un seul vecteur, ce dernier représente le pire cas puisqu'il n'en existe pas d'autre; il est donc inutile de le calculer.

Lorsque la procédure, à partir d'un état initial donné, a déterminé le vecteur à tester, celui-ci est mis en mémoire en pointant à l'aide d'un indicateur la ligne correspondante de la matrice des transitions.

Ex : (fig. IV.7.c),vecteur 3 à tester au cours de la transition entre les états 8 et 2. Après avoir repéré la ligne 8 03 002, on indique qu'un test est nécessaire en caractérisant cette ligne comme suit : 1 008 03 002. Après que tous les états du système aient été explorés, la matrice des transitions devient une matrice de test.

IV.4.4.2. Assignements et priorités

IV.4.4.2.1. Aiguillage des termes A et B sur les conditions de pire cas en fonction des états initiaux des variables secondaires

1. Cas des transitions bouclées $E_A = E_D$

Chacune des variables secondaires dont l'expression logique est donnée sous forme $\sum \prod$ est soumise à une procédure de factorisation afin de

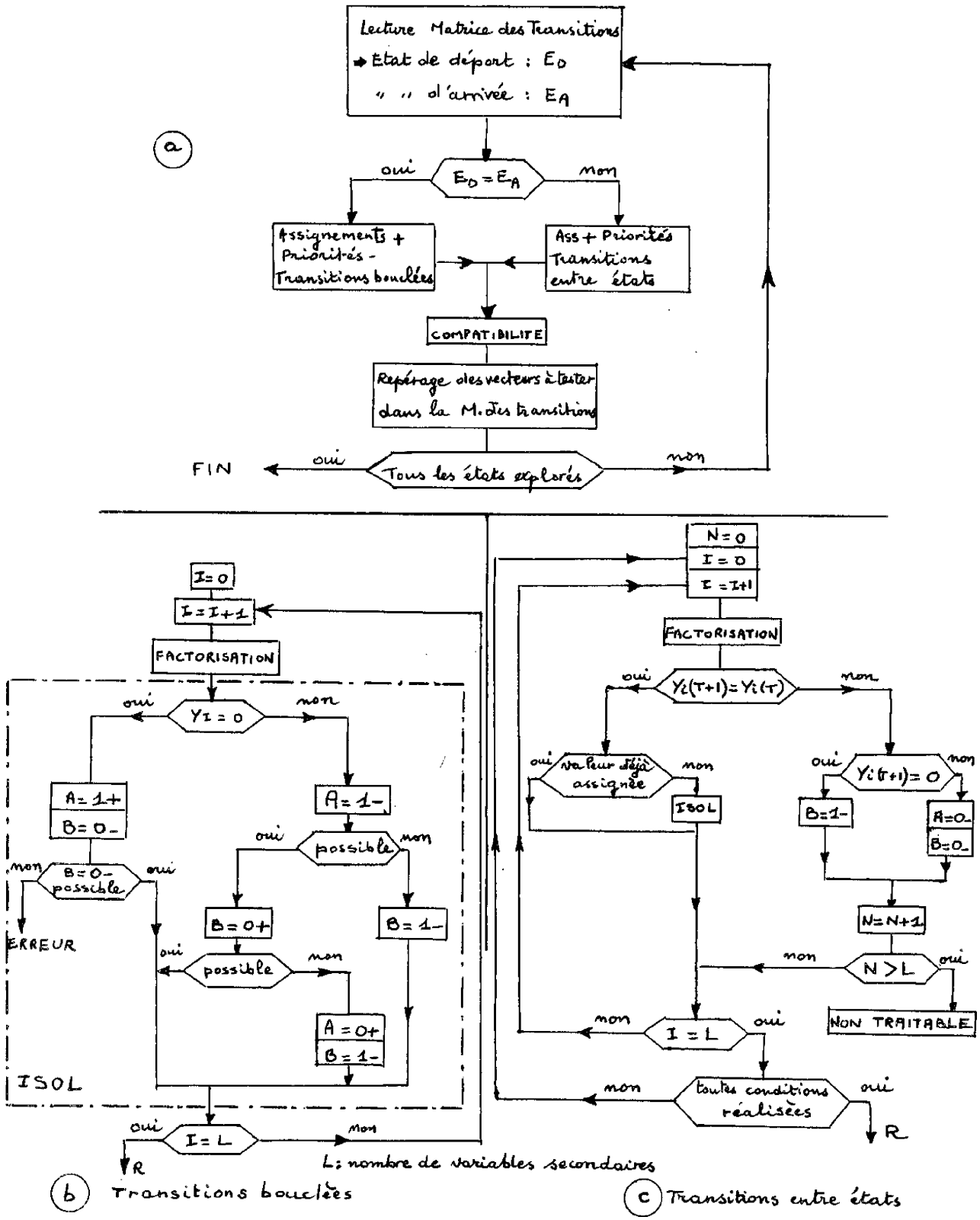

Lecture Matrice des Transitions
Etat de départ : E_D
" " d'arrivée : E_A
a
oui
E_D = E_A
non
Assignements + Priorités - Transitions bouclées
Ass + Priorités Transitions entre états
COMPATIBILITE
Repérage des vecteurs à tester dans la M. des transitions
FIN
oui
Tous les états explorés
non
I=0
I=I+1
FACTORISATION
oui
Y_I = 0
non
A=1+ B=0-
A=1-
oui
possible
non
non
B=0- possible
oui
ERREUR
B=0+
B=1-
oui
possible
non
A=0+ B=1-
ISOL
oui
I=L
non
R
b
Transitions bouclées
L: nombre de variables secondaires
N=0
I=0
I=I+1
FACTORISATION
oui
Y_i(T+1)=Y_i(T)
non
oui
Valeur déjà assignée
non
ISOL
oui
Y_i(T+1)=0
non
B=1-
A=0- B=0-
N=N+1
non
N>L
oui
NON TRAITABLE
non
I=L
oui
non
toutes conditions réalisées
oui
R
c
Transitions entre états

Planche IV.1

séparer les termes A et B. Ensuite, selon la valeur de $Y_i(T)$, A et B se voient affecter les conditions correspondantes (planche IV.1.b.)

2. Cas des transitions entre états : $E_A \neq E_D$

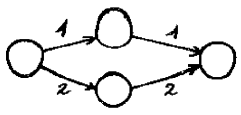
La procédure de factorisation existe ici aussi. Conformément à l'organigramme,s'effectue l'aiguillage des termes A et B. Lorsqu'au cours d'une transition une ou plusieurs variables secondaires gardent la même valeur dans l'état initial et dans l'état final, l'algorithme d'orientation de A et B est identique au cas précédent (bloc ISOL, planche IV.1.c).

Si une des conditions 0- s'avère impossible sur une ou plusieurs des variables secondaires, l'algorithme se reboucle au début avec pour nouvelles valeurs initiales des y celles qui ont été modifiées et celles qui n'ont pu être satisfaites.

Il existe cependant un cas non traitable : on a supposé que si la valeur d'une variable secondaire est identique dans l'état initial et dans l'état final, elle le reste aussi pendant les états transitoires. Or, il peut arriver qu'elle change puis revienne à son état antérieur.

Ex. : 010 ⟶ 110 ⟶ 111 ⟶ 001 (1e variable : 0 ⟶ 1 ⟶ 0)

On peut connaître tous les vecteurs qui assurent la transition. Mais par contre, on ne connait pas les états transitoires puisque seuls l'état initial et l'état final sont donnés par la matrice des transitions. Il faudrait donc rechercher sur les équations logiques l'évolution du système soumis successivement à chacun des vecteurs d'entrée, et garder en mémoire les états transitoires. Ensuite, connaissant les états initiaux, il devient possible d'oublier les vecteurs d'entrée et d'effectuer la recherche des vecteurs à tester, autant de fois qu'il y a de possibilités de passage d'un état à un autre :



Cet algorithme, s'il ne présente pas de difficulté majeure, va demander un temps calcul plus important et comme nous ne nous sommes pas heurté, à ce problème au cours des exemples traités, il ne nous a pas semblé indispensable de le préciser dans l'immédiat. Cependant pour le cas où cela arriverait au cours

d'un exemple, nous avons prévu l'algorithme suivant :

Imposer une valeur à une variable secondaire alors qu'elle devrait prendre la valeur opposée implique que les autres variables secondaires se verront dans l'impossibilité de réaliser les conditions voulues. Pour éviter que l'algorithme ne se reboucle indéfiniment, une transition de ce type est déclarée non traitable lorsque le nombre de fois que le bouclage a eu lieu devient supérieur au nombre de variables secondaires (sur l'organigramme, quand $N > L$).

IV.4.4.2.2. Conditions de pire cas (1+, 1-, 0+, 0-)

Elles utilisent la mise à 0 et à 1 des termes E_1 et E_p sous la forme $\prod \sum$.

1. Mise à 1 de E_1 (sous programme DECEP)

Les équations logiques en mémoire sont codées de la même façon que dans le programme d'analyse (§ III.2.1.2.) sous forme $\sum \prod$. L'algorithme de mise à 1 des E_1 va permettre à la fois l'affectation des valeurs des entrées et le passage en $\prod \sum$.

Soit $x_1 x_3 + x_2 \overline{x}_4 + x_3 \overline{x}_5 = 1$ ⟶

1 3 1 3 3	représentation
3 1 3 0 3	codée
3 3 1 3 0	

Rappelons que dans chaque nombre entier, 1 indique la présence d'une variable x sous forme directe, 0 sous forme complémentée, sans présumer de la valeur effective appliquée. Un nombre représente le ET entre des variables, l'opération OU étant figurée par une succession de nombres.

Le passage en $\prod \sum$ se traduit par l'inversion de ces opérations. La réunion sera obtenue en mettant les variables dans un même nombre, et l'intersection se traduira par un ensemble de nombres. Ainsi l'algorithme pour retrouver la fonction :

$(x_1+x_2+x_3)\cdot(x_1+x_2+\overline{x}_5)\cdot(x_1+\overline{x}_4+x_3)\cdot(x_1+\overline{x}_4+\overline{x}_5)\cdot(x_3+x_2+x_3)\cdot(x_3+x_2+\overline{x}_5)\cdot(x_3+\overline{x}_4+\overline{x}_5)\cdot(x_3+\overline{x}_4+x_3)$

et simultanément les valeurs des variables : ($x_1 = 1$, ou $x_2 = 1$, ou $x_3 = 1$) et

(....)) se déroule de la façon suivante :

1 3 1 3 3 ⟶ 1 3 3 3 3 x_1
3 3 1 3 3 x_3

3 1 3 0 3 ⟶ 3 1 3 3 3 x_2
3 3 3 0 3 $\bar{x}_4$

On fait l'intersection logique de 3 1 3 3 3 et de 3 3 3 0 3 avec les deux premiers termes :

1 1 3 3 3
3 1 1 3 3 ⟶ $(x_1+x_2)\ (x_2+x_3)\ (x_1+\bar{x}_4)\ (x_3+\bar{x}_4)$
1 3 3 0 3
3 3 1 0 3

3 3 1 3 0 ⟶ 3 3 1 3 3
3 3 3 3 0 l'intersection avec les termes précédents donne :

1 1 1 3 3
3 1 1 3 3
1 3 1 0 3
3 3 1 0 3
1 1 3 3 0
3 1 1 3 0
1 3 3 0 0
3 3 1 0 0

1 et 0 indiquent maintenant les valeurs effectives à placer sur les entrées pour que la fonction vaille 1.

1 1 1 3 3 par exemple signifie : $x_1 = 1$ ou $x_2 = 1$ ou $x_3 = 1$. On a donc conservé la même écriture, mais l'interprétation du contenu des mémoires et de leur association est différente.

2. Mise à 1 des E_p (sous programme D P R S)

Le même principe est utilisé mais il faut reconnaître les termes $\sum E_p\ Y_p$. Cela nécessite sur l'expression logique de A ou B, la recherche des E_p (c'est-à-dire des monômes contenant une ou plusieurs variables secondaires); comme chaque E_p peut comporter plusieurs monômes ($E_p = \sum X_m$), à mesure que les monômes

de A et B sont considérés, le sous programme calcule les formes $\Pi\Sigma$ par rapport à chacun des Y_p.

Exemple : $x_1 \bar{x}_2 Y_1 + x_3 x_5 Y_2 + x_4 Y_1 + x_1 Y_2$

Opérations : 1) $x_1 \cdot \bar{x}_2$; 2) $x_3 \cdot x_5$ } préparation de la forme $\Pi\Sigma$

3) $(x_1 \cdot \bar{x}_2) + x_4$ $\quad x_3 \cdot x_5$

$x_1 + x_4$; $\bar{x}_2 + x_4$; $x_3 \cdot x_5$ } Forme $\Pi\Sigma$ pour Y_1

4) $(x_1 + x_4) \cdot (\bar{x}_2 + x_4) \cdot (x_3 \cdot x_5) + x_2$

$x_1 + x_4 = 1$; $\bar{x}_2 + x_4 = 1$; $x_3 + x_2 = 1$; $x_5 + x_2 = 1$ } Forme $\Pi\Sigma$ pour Y_2

3. Mise à 0 (sous programme I V E P)

On travaille également sur la forme $\Pi\Sigma$, mais la valeur booléenne à afficher sur les entrées s'obtient en inversant les chiffres codés de la forme $\Sigma\Pi$. En effet, 1 3 0 3 $\leftarrow$ $x_1 \bar{x}_3 = 0$

donne sous forme $\Pi\Sigma$: $\bar{x}_1 + x_3 = 1$ soit 0 3 1 3

$x_1 = 0 + x_3 = 1$

4. Condition 1+

Le calcul s'effectue successivement sur chacun des monômes. S'il ne contient pas de variable secondaire : mise à un des entrées primaires par le S.P. DECEP, si non, entrée dans le S.P. DPRS qui classe ce monôme par rapport aux Y_p déjà rencontrés et le fait contribuer à la forme définitive.

Sur l'organigramme correspondant (pl. IV.2.a), on a séparé les termes E_1 et E_p pour plus de clarté mais en réalité, les monômes sont traités dans leur ordre d'apparition sans classement préalable dans la catégorie E_1 ou parmi un des E_p. Cette remarque s'applique également aux organigrammes de 0+, 1-, 0-.

Exemple :

$$Y_8(T+1) = \underbrace{\left(J'K + (J'+C)\, Y_{14} + KC + K\, Y_{14}' \right)}_{A} \cdot Y_8(T) + \underbrace{J'KC' + J'C'Y_{14} + KC\, Y_{14}'}_{B}$$

(Nous reprendrons cet exemple sur chacune des conditions).

Condition 1+ sur A :

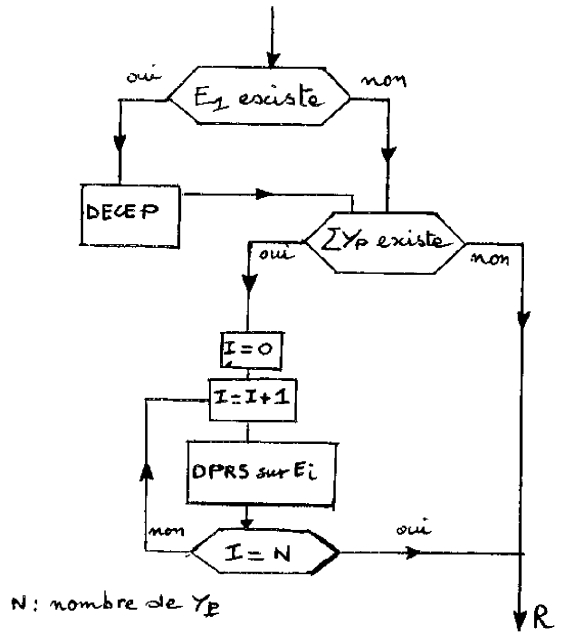
$$A = (J'+K)\cdot(J'+C)\cdot(K+C)\cdot K + (J'+C)\, Y_{14} + K\, Y_{14}'$$

$$= a_1 + a_2\, Y_{14} + a_3\, Y_{14}'$$

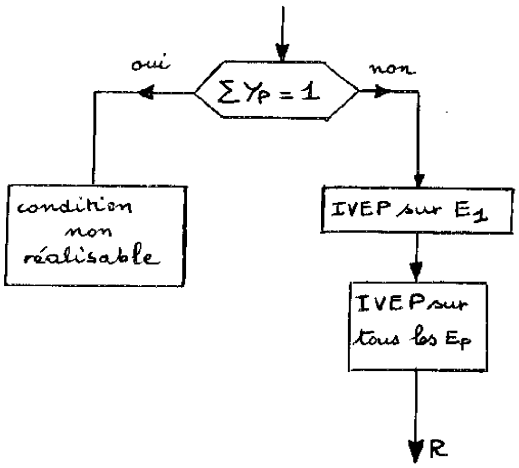
1) $a_1 = 1$ $\quad J' + K \cdot J' + C \cdot K + C \cdot K$ donne $\begin{cases} J = 0 \text{ ou } K = 1 \\ \text{et } J = 0 \text{ ou } C = 1 \\ \text{et } K = 1 \text{ ou } C = 1 \\ \text{et } K = 1 \end{cases}$ soit $\begin{matrix} 0\ 1\ 3 \\ 0\ 3\ 1 \\ 3\ 1\ 1 \\ 3\ 1\ 3 \end{matrix}$

2) $a_2 = 1$ $\quad J' + C$ donne $J = 0$ ou $C = 1$ soit $0\ 3\ 1$

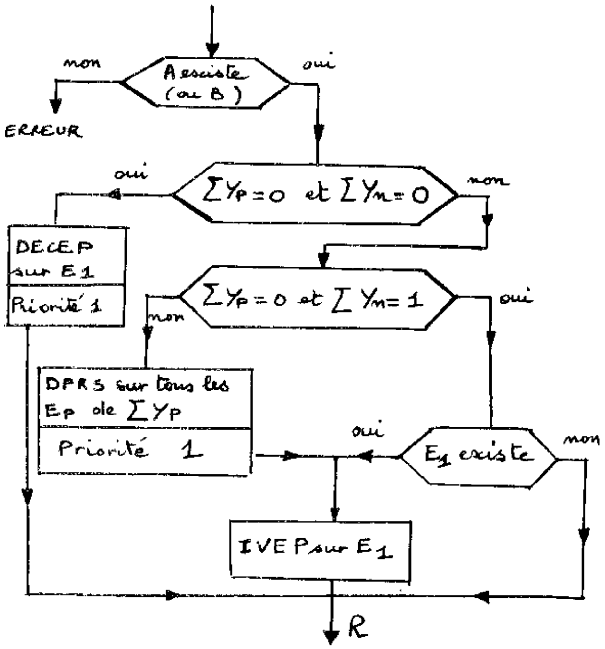
3) $a_3 = 1$ $\quad K$ donne $K = 1$ soit $3\ 1\ 3$



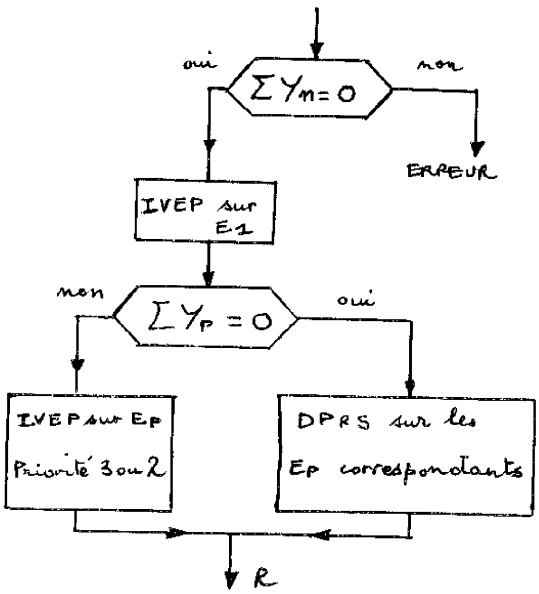
(a) CONDITION 1+



(b) CONDITION 0+



(c) CONDITION 1-



(d) CONDITION 0-

Assignements de A et B

Planche IV.2

Pour que le pire cas soit réalisé il faut que ces trois conditions soient simultanément considérées. Comme on est maintenant en $\Pi \Sigma$ avec affectation des valeurs, il suffit de regrouper les assignements dans une même liste :

0 1 3
0 3 1
3 1 1
3 1 3
0 3 1
3 1 3

5. Condition 0+ (Pl. IV.2.b)

Elle se traduit par l'inversion de toutes les entrées primaires (SP. IVEP).

Exemple : condition 0+ sur B

$$B = J' K C' + J' C' Y_{14} + K C Y_{14}'$$

$$= b_1 + b_2 Y_{14} + b_3 Y_{14}'$$

pour que $b_1 = 0$ il faut J = 1 ou K = 0 ou C = 1 soit 1 0 1
pour que $b_2 = 0$ il faut J = 1 ou C = 1 soit 1 3 1
pour que $b_3 = 0$ il faut K = 0 ou C = 0 soit 0 3 0

On obtient donc la liste

1 0 1
1 3 1
0 3 0

6. Condition 1- (Pl. IV.2.c)

On indique les termes de priorité 1 en plaçant le chiffre 1 devant le nombre entier correspondant :

Exemple : condition 1 - sur **A**.

On a ici $\sum Y_n = 0$

- si $Y_{14} = 0$ ceci implique $\sum Y_p = 1$ (terme K Y_{14}')

. donc $E_1 = 0$ (soit (J+K') (K'+C') = 1 — 1 0 3 / 3 0 0

ce qui revient à inverser la valeur des variables dans la représentation codée (en $\sum\pi$) de départ : 0 1 3 / 3 1 1

. et dans K Y_{14}', K = 1 soit 3 1 3

Seul ce terme va assurer l'isolement. Il est donc indispensable et nous devons lui affecter une priorité 1 : 1 3 1 3

d'où la liste { 1 3 1 3 ; 1 0 3 ; 3 0 0

- si $Y_{14} = 1$ on a aussi $\sum Y_p = 1$ (par J' C' Y_{14})

On fera donc de même $E_1 = 0$.

Mais $E_p = 1$ va jouer sur J' C' d'où J = 0 et C = 0 — 0 3 3 / 3 3 0

et la priorité 1 conduit à 1 0 3 3 / 1 3 3 0

7. Condition 0- (Pl. IV.2.d)

Les termes de priorité 2 et 3 sont repérés de la même manière par les chiffres 2 et 3.

Exemple : condition 0- sur B.

On a $\sum Y_n = 0$.

On fait $E_1 = 0$ soit J' K C' = 0 (J = 1 ou K = 0, ou C = 1).

Avec une priorité de degré 2, on obtient : 2 1 0 1.

. si $Y_{14} = 0$ $\quad \sum Y_p = 1 \quad (K\ C\ Y_{14}')$

Il faut donc K = 0 ou C = 0 ce qui donne 2 3 0 0

Pour le terme J' C' Y_{14} on assignera J = 0 et C = 0 (sensibilisation de Y_8 à la valeur de Y_{14}), sans priorité : 0 3 0.

L'assignement complet est le suivant : $\begin{bmatrix} 2\ 1\ 0\ 1 \\ 2\ 3\ 0\ 0 \\ 0\ 3\ 0 \end{bmatrix}$

. si $Y_{14} = 1$ $\quad \sum Y_p = 1\ (J'\ C'\ Y_{14})$

ceci implique J = 1 ou C = 1 d'où 2 1 3 1

et dans le terme K C Y_{14}' on placera K = 1 et C = 1 : 3 1 1

d'où la liste $\begin{bmatrix} 2\ 3\ 0\ 0 \\ 2\ 1\ 3\ 1 \\ 3\ 1\ 1. \end{bmatrix}$

IV.4.4.3. Compatibilité

Nous avons vu au paragraphe précédent comment s'organisait la liste des assignements. Dans un nombre figurent les valeurs (les niveaux logiques) à appliquer aux entrées, repérées par leur position, l'une ou l'autre de ces valeurs, suffisant à assurer la condition requise. Une succession de nombres représente autant de conditions à satisfaire simultanément. Donc traiter la compatibilité revient à examiner chaque colonne (qui définit une variable). Lorsqu'une variable est biforme il faut arriver à supprimer l'incompatibilité qui apparait. Les critères de priorité définis plus haut vont rendre cette tâche possible mais nous allons être amené à examiner tous les cas susceptibles d'intervenir entre deux degrés de prio-

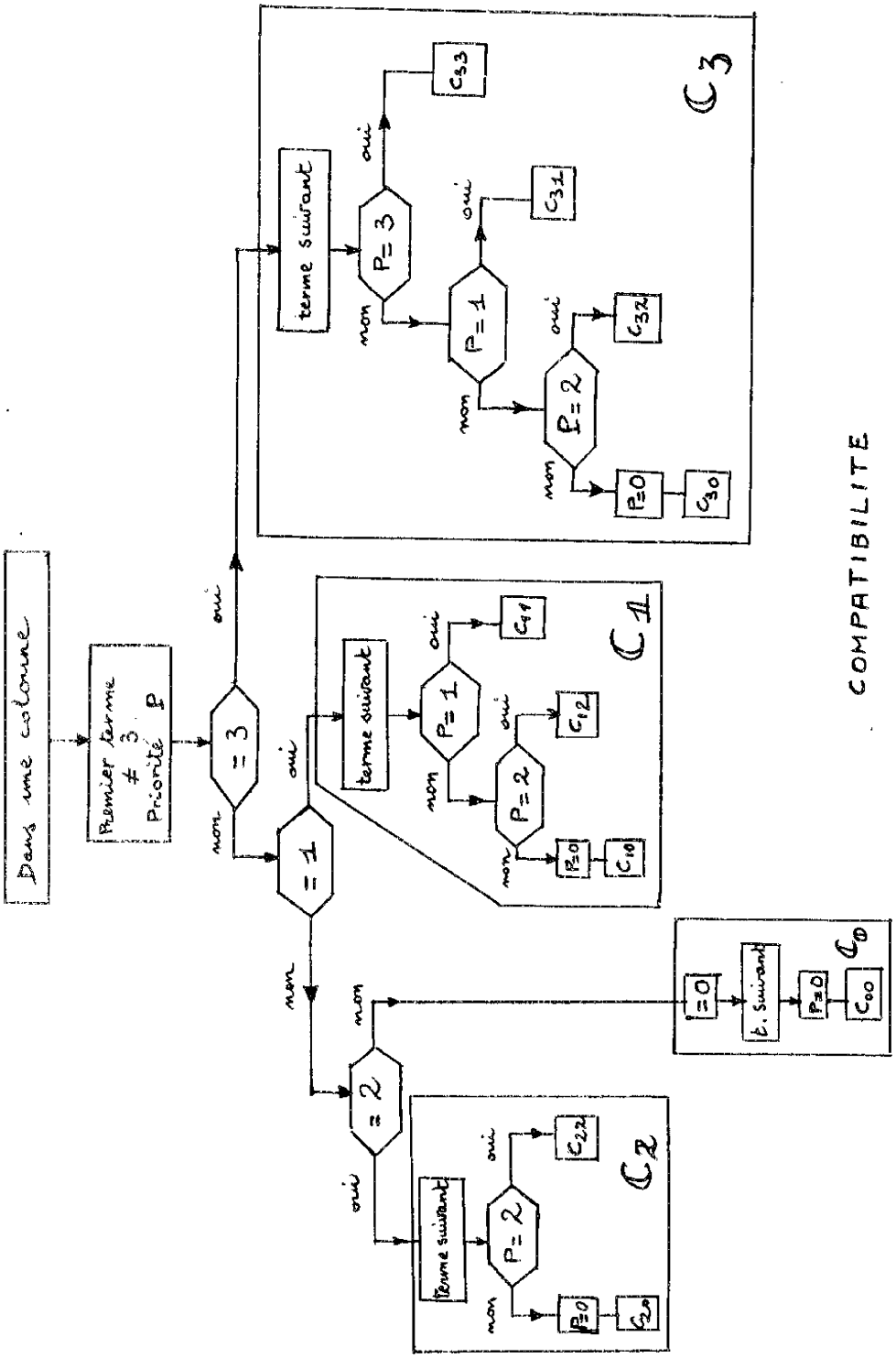

Dans une colonne
Premier terme ≠ 3 Priorité P
= 3
= 1
= 2
= 0
oui
non
terme suivant
P= 3
P= 1
P= 2
P=0
C3
C31
C32
C30
C1
C11
C12
C10
C2
C22
C20
C0
C00
COMPATIBILITE

Planche IV.3

rité et à construire les solutions capables de régler les litiges en présence (Planche IV.3).

Nous allons présenter les algorithmes de façon schématique, en indiquant l'essentiel de la procédure mise au point. Nous indiquerons, quand cela sera possible simplement, comment s'effectuent les transformations sur les expressions codées et classées. Nous ne traiterons pas dans chacun des cas, un exemple complet, car si cela éclaircirait incontestablement la méthode au début, on arriverait rapidement à se perdre dans le dédale des transformations, à répéter plusieurs fois le même processus, et à allonger considérablement l'exposé. (il y a en effet dix cas de compatibilité à observer, liés en certains points les uns aux autres).

Nous traiterons dans l'annexe II un exemple correspondant à un circuit séquentiel réel, que s'il ne fait pas intervenir tous les cas, permettra de bien saisir le déroulement et le sens des diverses opérations.

IV.4.4.3.1. Compatibilité par rapport à un terme de priorité 3 (C_3)

1. D'un terme de priorité 3 : $C_{3\,3}$

La valeur de la variable doit être identique. Le cas contraire correspond à une impossibilité physique : défaut d'isolement d'une variable secondaire à 0 dans une transition bouclée; mise à 0 non réalisable d'une variable secondaire dans une transition entre états.

2. D'un terme de priorité 1 : $C_{3\,1}$

a - S'il y a compatibilité (variable monoforme) la comparaison continue sur un autre terme.

b - Si non deux cas peuvent se présenter :

. Le monôme de priorité 1 comporte plusieurs variables.

On supprime l'assignement de la variable contradictoire :

Ex : 3 1 3 3 ⟶ 3 1 3 3
1 0 1 3 1 3 1 3

puisque la condition 1- est satisfaite si $x_1 = 0$ ou $x_2 = 1$

. Le monôme de priorité 1 est à une seule variable.

Il faut retrouver à quel terme A ou B appartient le monôme contenant cette variable puis essayer de reporter la condition 1- sur d'autres monômes de A ou B de façon qu'elle donne un assignement compatible avec le terme de priorité 3.

Cela doit être possible sur un terme B_i (cas du transfert $0 \rightarrow 1$ de y_i) mais peut ne pas l'être sur un terme A_i. Dans ce cas extrême, on reportera la condition 1- sur le B_i correspondant et la comparaison recommencera sur la première colonne de la liste, car le nouvel assignement obtenu peut introduire des variables différentes dans les colonnes déjà traitées.

3. D'un terme de priorité 2 : $C_{3\,2}$

a. compatibilité

Exemple : $x_1 x_2 x_4 + x_3 \overline{x}_5 = 0 \rightarrow (\overline{x}_1 + \overline{x}_2 + \overline{x}_4)(\overline{x}_3 + x_5) = 1$

	P	
sous forme codée	2	0 0 3 0 3
	2	3 3 0 3 1

Si d'autre part, la condition $x_1 = 0$ doit être impérativement respectée,

on a : 3 0 3 3 3 3 qui est compatible avec la liste ci-dessus.

Pour rendre l'assignement plus restrictif, on inverse les valeurs des variables x_2 et x_4 en souhaitant seulement qu'elles seront satisfaites.

Ainsi le terme 2 0 0 3 0 3 donne { 3 1 3 3 3 / 3 3 3 1 3 } sans priorité.

ce qui signifie : puisque $x_1 = 0$ suffit à mettre $x_1\ x_2\ x_4$ à 0, imposons $x_2 = 1$ et $x_4 = 1$ pour rendre la condition 0- sur $x_1\ x_2\ x_4$ la plus fragile. x_1 disparait puisqu'il est contenu dans le terme de priorité 3.

b - incompatibilité

On supprime la variable contradictoire dans le terme de priorité 2.

S'il ne reste alors qu'une seule variable dans ce terme, il faut lui donner la priorité 3 car elle devient essentielle.

Ex : 3 3 1 3 3 ⟶ 3 3 1 3 3
2 1 0 3 3 3 1 3 3 3

L'algorithme doit reprendre au début de la première colonne pour étudier l'effet de cette nouvelle priorité.

4. D'un terme de priorité 0 : $C_{3\ 0}$

En cas de conflit, l'assignement du terme de priorité 0 est supprimé.

IV.4.4.3.2. Compatibilité par rapport à un terme de priorité 1 ($C_{\ 1}$)

1. D'un terme de priorité 1 : $C_{1\ 1}$

a - Compatibilité :

Passage au terme suivant de la liste. Lorsque toute la liste est compatible avec le terme de référence, si celui-ci comporte plusieurs variables on inverse celles qui n'ont pas participé à la comparaison, comme en $C_{3\ 2}$, mais en conservant le terme de référence avec sa priorité.

b - Incompatibilité

On regarde si les deux monômes comportent une ou plusieurs variables :

- si dans l'un d'entre eux ou les deux figurent plusieurs variables celle qui provoque l'incompatibilité voit son assignement détruit dans le monôme à plusieurs variables (s'il n'y en a qu'un) ou les deux (s'ils sont tous deux à plusieurs variables).

Dans le premier cas l'exploration de la colonne se poursuit avec pour référence le terme dont l'assignement n'a pas été détruit (retour en (a)). La seconde solution demande la réexploration complète de la colonne pour redéfinir le terme prioritaire.

Si les deux monômes se réduisent à la seule variable sur laquelle s'effectue la comparaison, il faut essayer de reporter la condition 1- sur l'expression logique A de laquelle est issu le premier ou le deuxième monôme. En cas de succès la comparaison recommence au début, si non l'ultime possibilité consiste à reporter la condition 1- sur les termes B des variables secondaires génératrices des termes contradictoires.

2. D'un terme de priorité 2 : C_{12}

a - compatibilité :

L'algorithme est identique au cas a C_{32}, en ajoutant a C_{11}

b - incompatibilité :

Utilise b C_{32}. Si le terme reste de priorité 2 (plusieurs variables) on se reboucle en a C_{11}.

3. D'un terme de priorité 0 : C_{10}

Une contradiction se résoud en supprimant la variable sans priorité. Elle est suivie d'un retour en a C_{11}.

IV.4.4.3.3. Compatibilité par rapport à un terme de priorité 2 (C2)

1. D'un terme de priorité 2 : C_{22}

a - Compatibilité :

On continue sur les autres termes de la colonne. Si en fin de liste,la variable de référence n'a pas trouvé d'assignement de priorité 2 qui puisse créer un litige (b),les autres variables du monôme subissent une inversion (comme en a C_{32}),ce qui impose de garder la variable de référence avec une priorité de degré 3.

b - Incompatibilité

Dans ce cas,la variable est rendue indifférente dans les deux termes comparés. Si,après cette comparaison,il reste dans chacun au moins deux variables l'algorithme reprend au début de la colonne. Par contre si l'un d'eux ne possède qu'une seule variable on lui affecte une priorité 3 et on reprend la comparaison au début (1e colonne).

2. D'un terme de priorité 0 : C_{20}

Les termes non prioritaires incompatibles sont supprimés et on se retrouve dans le cas a C_{22}.

IV.4.4.3.4. Compatibilité par rapport à un terme de priorité 0 (C_0)

D'un terme de priorité 0 : C_{00}

Lorsque dans une colonne la comparaison met en jeu des termes sans priorité, ils proviennent de conditions souhaitables. L'algorithme ci-dessous permet de réaliser au mieux le pire cas par un choix très précis de la valeur de la variable.

On compte dans les termes comportant une seule variable située dans dans la colonne examinée, le nombre de fois que cette variable prend la valeur 0 et la valeur 1.

- Si l'une des deux formes apparait plus souvent, on supprime tous les monômes à une seule variable de valeur opposée, et dans les monômes incompatibles à plusieurs variables on rend indifférente la valeur de la variable dans la colonne examinée.

Ex :

3 0 3 3
3 1 3 3
3 0 0 3
3 1 3 3
3 1 0 3

↑ ↑
P = 0

⟶

3 1 3 3
3 3 0 3
3 1 3 3
3 1 **0** 3

↑

- Dans le cas où les deux formes sont représentées identiquement, le même processus s'exerce sur les monômes à plusieurs variables. Si,là aussi,la fréquence d'apparition est la même, il devient impossible de donner une valeur définitive. La suppression dans la colonne de tous les assignements indiquera l'indifférence de la variable d'entrée concernée.

Exemples : - 3 0 3 3 } t.à 1 seule variable
3 1 3 3 }

termes à plusieurs variables { 3 0 0 3 / 0 0 3 3 / 3 1 3 0 }

⟶

3 0 3 3
3 0 0 3
0 0 3 3

assignement possible grâce aux termes à plusieurs variables.

↑
↓

- 3 0 3 3 } 1 seule variable
3 1 3 3 }

3 0 0 3 } plusieurs variables
0 0 3 3 }
3 1 3 0 }
0 1 3 0 }

choix impossible ⟶

~~3 3 3 3~~
~~3 3 3 3~~
3 3 0 3
0 3 3 3
3 3 3 0
0 3 3 0

↑ colonne indifférente.

IV.4.4.4. Vecteurs à tester

Après que l'algorithme de compatibilité ait été appliqué à toutes les entrées du système, on obtient le vecteur à tester en contractant les termes qui restent (par une opération d'intersection) (cf. § III.2.1.3)

Ex :

1	3	3	3
3	0	1	3
1	3	3	1
3	3	1	3
1	0	1	1

1	3	3	
3	3	0	
1	3	0	
1	3	0	→

→ | 1 0 0 |
  | 1 1 0 |

Chaque fois qu'une variable est indifférente, cela double le nombre de vecteurs à tester pour recouvrir toutes les pannes possibles.

IV.4.5 Utilisation du programme - Possibilités

IV.4.5.1. Utilisation du programme

Ce programme est écrit en FORTRAN IV et comporte environ 1700 instructions.

Il est destiné à fonctionner avec le programme d'analyse qui lui fournit toutes les données nécessaires. Mais, la taille de l'ordinateur peut empêcher le passage simultané de ces deux programmes. La solution consiste alors, lorsqu'on veut connaître les vecteurs à tester d'un système séquentiel, à en effectuer l'analyse, puis, à l'aide des résultats obtenus à fournir au programme de test les équations des variables secondaires et la matrice des transitions codée, données principales dont il a besoin.

IV.4.5.2. Possibilités

. nombre d'entrées : comme dans le programme d'analyse, la limite est de 9 ou 10 selon que le mot mémoire comporte 32 ou 36 bits. Pour s'en

affranchir, il suffit de recou rir aux moyens donnés au § III.3.4.1..

. nombre de variables secondaires : non limité (limitation par le nombre de mémoires).

. temps calcul : il dépend de la complexité des expressions logiques des variables secondaires, (ceci intervient au niveau du passage de $\sum\prod$ à $\prod\sum$ cf. § IV.4.3.1.) de leur nombre et du nombre d'états du système. Ce programme a été mis au point à l'aide d'exemples de taille modeste, bascules du type JK.M.S, D, à 2 ou 3 entrées, 2 ou 3 v.s et 4 états stables,pour les raisons suivantes :

- mise au point plus rapide si le temps de calcul est assez court.

- vérification possible à la main.

Les temps obtenus varient selon l'exemple de 30 s à 2' (IBM 7044).

IV.4.6 Conclusions

Cette méthode n'a mis en jeu qu'une seule hypothèse : les variables secondaires doivent s'écrire sous la forme $Y_{T+1} = AY_T + B$, ce qui écarte les systèmes susceptibles de comporter des cycles ou des aléas statiques.

En particulier aucune hypothèse n'a été avancée quant aux types de pannes qui pouvaient intervenir.

Elle donne les vecteurs nécessaires à tester à partir de chacun des états stables du système, de la même façon que les spécifications présentées au § IV.1. Ceci demande la détermination d'une séquence de positionnement dans un état initial qui peut se faire à partir d'une entrée de forçage.

Cette manière d'envisager le test fonctionnel des systèmes séquentiels trouve son application dans certains testeurs de composants [21].

La deuxième solution possible,consiste à déterminer une séquence de détection. Nous donnerons au § IV.5 une méthode algorithmique,qui,à partir de la

liste des vecteurs à tester, génère une séquence de détection aussi courte que possible,en fonction des critères adoptés.

Ajoutons encore que les résultats obtenus sont extrêmement précis car la plupart du temps ils se traduisent par des vecteurs complètements assignés. Cela est dû à l'étude de la compatibilité qui pousse la comparaison très loin. Si l'utilisateur conteste cette précision et veut se contenter de résultats plus grossiers, la programmation sous forme modulaire permet de changer facilement tel ou tel critère.

IV-5 SEQUENCES DE DETECTION

IV.5.1. Buts - Hypothèses

La méthode du pire cas que nous venons d'exposer, donne pour chacun des états d'un système, les vecteurs dont le test assure le bon fonctionnement du circuit. Comme nous venons de l'indiquer, cela nécessite l'élaboration de séquences de positionnement dans ces états, soit par une remise à zéro ou à un, soit par une séquence de synchronisation, soit encore par une séquence propre (Homing séquence).

On sait que pour être sûr du bon fonctionnement d'un système, il faut en explorer tous les états, en effectuant toutes les transitions. On est conduit à l'élaboration d'une séquence de test qui doit satisfaire cette condition.

Ce principe est appliqué en (6, 8), mais le test systématique de toutes les cases de la table de fluence, se traduit par des séquences de longueur importante.

Nous allons également chercher à établir des séquences de test de détection mais en utilisant les résultats de la méthode précédente. Ceci présente un avantage par le fait que nous ne testerons pas toutes les cases de la matrice des excitations (c'est-à-dire tous les vecteurs de toutes les transitions) et que les séquences obtenues seront de longueur inférieure à ce que donneraient

(6) et (8).

Il est bien évident que cette méthode va nécessiter davantage de calculs et augmenter le coût de la recherche des tests. Cependant elle va permettre d'accélérer les tests réels, et par là même d'augmenter la cadence de production. On peut penser également (compte non tenu des limitations que nous nous sommes volontairement imposées) qu'il sera alors possible de tester des circuits plus complexes.

HYPOTHESES :

La méthode du pire cas n'impose pas que le système soit fortement lié puisqu'elle teste chacun des états séparément. Ici par contre nous ne pouvons traiter que les circuits fortement connectés car il s'agit de déterminer une séquence qui explore tous les états du système.

Nous supposerons de plus que l'état initial du circuit est donné :

- par une entrée de forçage

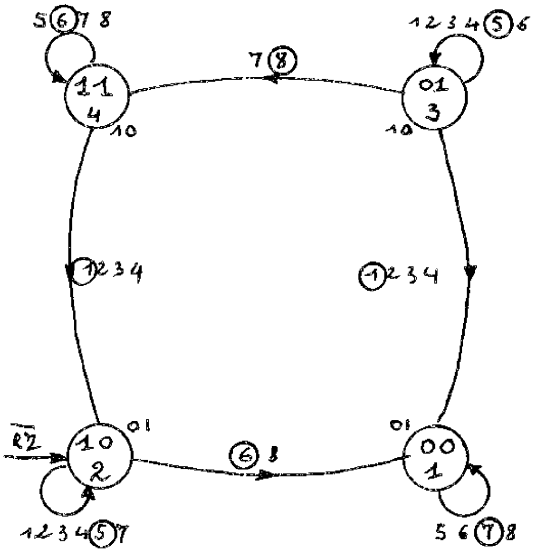
ou - par une séquence de synchronisation.

CONTRAINTE :

Nous imposerons qu'entre deux vecteurs voisins de la séquence, l'adjacence soit respectée, c'est-à-dire qu'une seule variable change pour passer d'un vecteur à un autre.

IV.5.2. Méthode et algorithmes

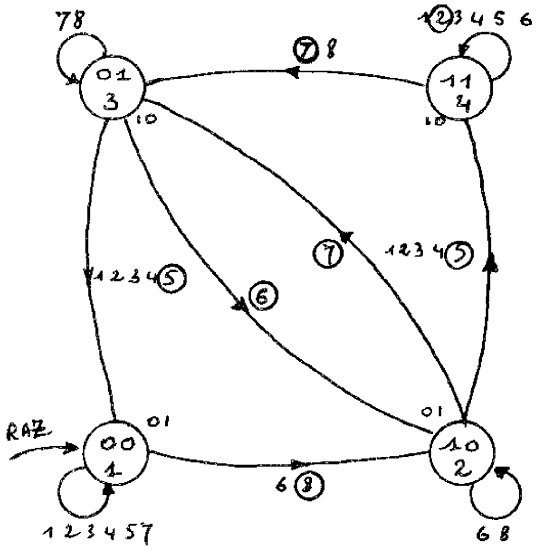
Nous emploierons au cours de ce paragraphe des sigles afin de ne pas répéter constamment les mêmes mots.



code :

	J	K	C
1	0	0	0
2	1	0	0
3	0	1	0
4	1	1	0
5	0	0	1
6	1	0	1
7	0	1	1
8	1	1	1

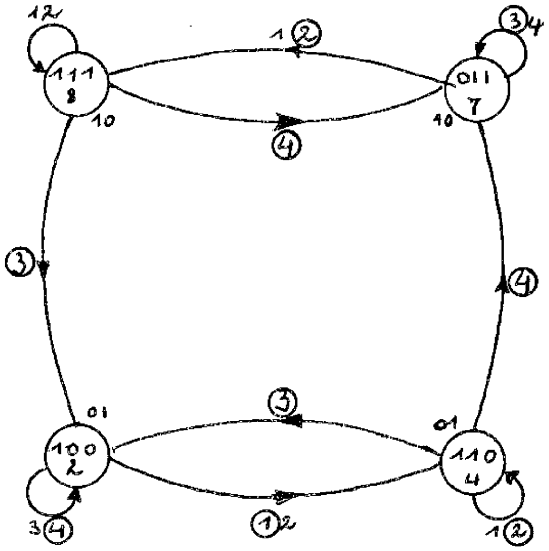
(a) JK 7473



(b) JK 74H72

code :

	D	C
1	0	0
2	1	0
3	0	1
4	1	1



(c) D 7474

○ vecteurs à tester

figure IV.7

IV.5.2.1. Définitions

Vecteur transfert (V.T.) vecteur provoquant le passage d'un état stable à un autre avec changement de la sortie.

ex : . fig. IV.7.a - depuis les états 1 et 4 : vecteurs 1, 2, 3, 4

. fig. IV.7.b - depuis l'état 2 : vecteurs 1, 2, 3, 4, 5, 7
- depuis l'état 3 : vecteurs 1, 2, 3, 4, 5, 6

. fig. IV.7.c - depuis l'état 4 : vecteur 4
- depuis l'état 8 : vecteur 3

Vecteur transfert inopérant (V.T.I.) quand un état stable ne possède pas de vecteur transfert, on dira qu'il a des vecteurs transfert inopérants si des vecteurs transfert d'un autre état stable conduisent à lui.

ex : . fig. IV.7.a - 2 et 3 n'ont pas de V.T. - 1, 2, 3, 4, V.T. de 1 et 4 seront des V.T.I. de 2 et 3.

. fig. IV.7.b - pour 1 et 4 : 1,2,3,4,5

. fig. IV.7.c - pour 2 : 3, pour 7 : 4

Vecteur inscription : tout vecteur faisant changer le système d'état. Un vecteur transfert est un vecteur inscription particulier.

Vecteur isolement : tel que son application laisse le système dans le même état. Un V.T.I. est un vecteur isolement particulier.

IV.5.2.2. Méthode

- Hypothèse : l'état initial du système est connu

- Critère : dans chaque état, on testera d'abord les vecteurs isolement qui font partie de la liste des vecteurs à tester, puis les vecteurs inscription.

Dès qu'un vecteur isolement ou inscription a été choisi, s'il n'est pas vecteur transfert (opérant ou inopérant), il faut lui adjoindre un V.T.

adjacent qui assure l'observabilité du test sur les sorties. (voir § IV.5.2.3.1.) Ce critère a été choisi afin de systématiser l'algorithme. La première partie de l'élaboration d'une séquence de détection ne tient pas compte de l'adjacence sauf en ce qui concerne le point précédent. Au fur et à mesure de l'établissement de la séquence, la liste des vecteurs à tester diminue. Il peut arriver que sur un état tous les vecteurs soient testés.

Ex : fig. IV.7.b et IV.8.a : après le test des vecteurs 8, 7,5 (branche 1), on se retrouve dans l'état (1) . Comme seul le vecteur 8 doit être testé dans cet état et qu'il l'a été effectivement, il faut passer à l'état le plus proche non complètement testé (état (2) , où il reste le vecteur 5 à tester) par une séquence adjacente à la fois au dernier vecteur qui a amené dans l'état (1) : 5, et au prochain vecteur qui sera testé dans l'état (2) : 5.

En général, on n'obtiendra pas une seule séquence. En effet, lorsqu'après avoir testé un vecteur inscription, on cherche à rendre ce test observable grâce à un V.T. adjacent, plusieurs vecteurs peuvent satisfaire cette condition et aucun n'est à éliminer a priori.

- par exemple : fig. IV.7.b après le vecteur 8 ((1) $\xrightarrow{8}$ (2)), les vecteurs 5 et 7 sont adjacents.

Le fait que plusieurs vecteurs doivent être testés pour une même transition (fig. IV.7.a : (4) $\xrightarrow{7,\ 8}$ (3)), implique également l'ouverture d'une branche en autant de branches que de vecteurs à tester.

La méthode consiste donc à construire un arbre de séquence et à déterminer la branche la plus courte, après que l'adjacence ait été réalisée sur chacune des branches.

IV.5.2.3. Algorithmes

IV.5.2.3.1. Détermination des V.T. et V.T.I. de chaque état

Elle s'effectue à l'aide de la matrice de test donnée par le programme de recherche des pires cas.

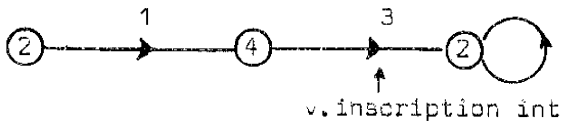
Il suffit de noter les transitions provoquant un changement de la sortie. Les états sources de ces transitions possèdent comme V.T. ceux qui les constituent. On regarde ensuite s'il est possible d'affecter aux états non munis de V.T., des V.T.I., en déterminant si les transitions qui amènent à ces états sont formées de V.T. des états antécédents.

Nous devons indiquer l'utilité des V.T.I. : lorsqu'on teste un vecteur d'entrée qui laisse un système dans un état inchangé, ceci revient à s'assurer d'un bon isolement. Or il est possible que cet isolement ne porte que sur une partie du circuit et ne se traduise pas de façon immédiate sur les sorties (par ex : fig. IV.7.a : Etat n° 2, vecteur 6 :isolement du maître).

On dira que le résultat du test n'est pas observable et on affichera un autre vecteur d'entrée (ou plusieurs si cela est nécessaire) pour transférer les informations contenues dans la portion de circuit sur laquelle portait le test, jusqu'aux sorties. Si l'isolement est effectivement bon le circuit reste dans le même état. On confie cette tâche aux V.T.I.. Leur nom se justifie par le fait qu'ils sont chargés de transférer, si tout s'est bien passé, des informations qui existent déjà. Ils réalisent cependant l'observabilité du test (ex. précédent : V.T.I. = 1)

Un autre cas peut intervenir de par la contrainte d'adjacence que nous nous sommes imposée : c'est le cas où il n'existe pas de V.T. adjacent à un vecteur inscription à tester. La fig. IV.7.c en donne un exemple. Après avoir testé le vecteur inscription amenant de l'état 2 à l'état 4, on ne peut appliquer le vecteur transfert 4 assurant l'observabilité à cause de la condition d'adjacence. On cherche alors si de l'état atteint par le vecteur inscription, existent d'autres vecteurs conduisant dans des états où la sortie est identique, en respectant l'adjacence. Si celà est possible, comme ces vecteurs ne sont pas des V.T. dans le sens où nous avons défini ce terme, il suffit de trouver sur le nouvel état atteint un V.T. adjacent pour réaliser l'observabilité du test initial. On a donc, pour tester un vecteur inscription, utilisé un autre vecteur inscription intermédiaire observable par un V.T. adjacent.

Sur l'exemple précédent ceci donne :



Le cas le plus défavorable apparait lorsqu'aucun vecteur autre que les V.T. non adjacents au vecteur inscription à tester ne permet de changer d'état. On utilise alors pour vecteur intermédiaire un vecteur isolement (ou un vecteur inscription inopérant) de l'état atteint dont le rôle est simplement de créer l'adjacence. Mais ceci fausse le test initial du vecteur inscription et lui ôte toute raison d'être, car le pire cas qu'il représentait se trouve détruit par un assignement différent. (Sur les exemples traités nous n'avons jamais recontré ce cas).

IV.5.2.3.2. Liste des vecteurs à tester dans chaque état

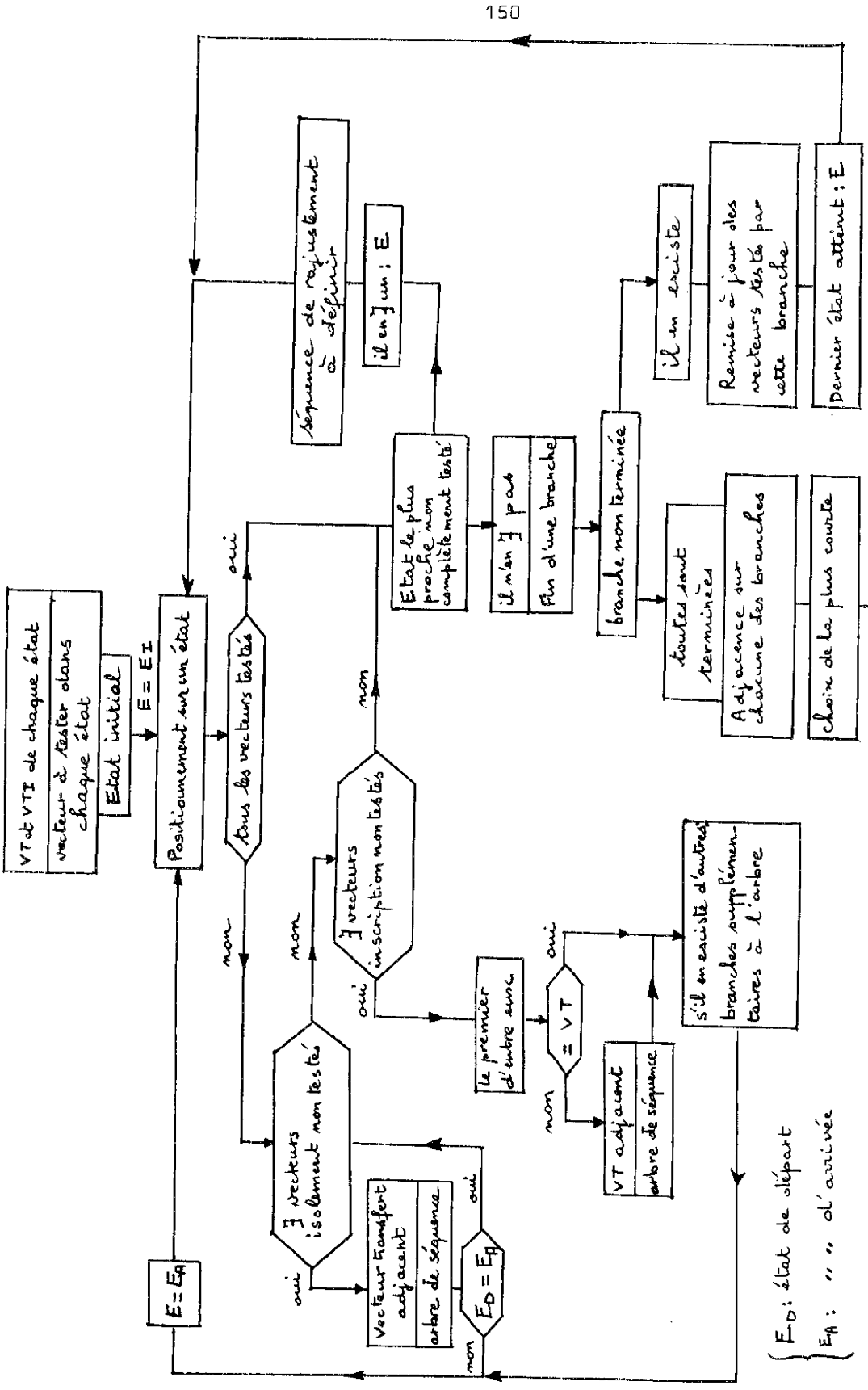
Ce n'est qu'une transcription de la matrice de test sous forme plus condensée. Ceci présente l'intérêt d'éviter une recherche dans la matrice des transitions à chaque fois.

Si d'un état on doit tester les vecteurs 5 et 8, dans la liste des vecteurs à tester, on placera le nombre 58, ceci limite le nombre d'entrées du circuit à trois car $2^3 < 10$. Cette limitation, génante, a été introduite pour deux raisons.

- elle nous a permis de mettre au point rapidement le programme de recherche des séquences de détection.
- les circuits que nous étudions (type Bascule JK, D) ne possèdent pas plus de 3 entrées (non compris les forçages).

De plus il est facile de s'en affranchir en réservant une place plus grande pour écrire les vecteurs : 05/08 porte le nombre d'entrées possibles à 6

005/008 porte le nombre d'entrées possibles à 9.



SEQUENCES DE DETECTION

Planche IV.4

Cela demande simplement de transformer le codage et le décodage des nombres composés des vecteurs d'entrée.

Le procédé est identique en ce qui concerne le stockage en mémoire des V.T. (cf. paragraphe précédent).

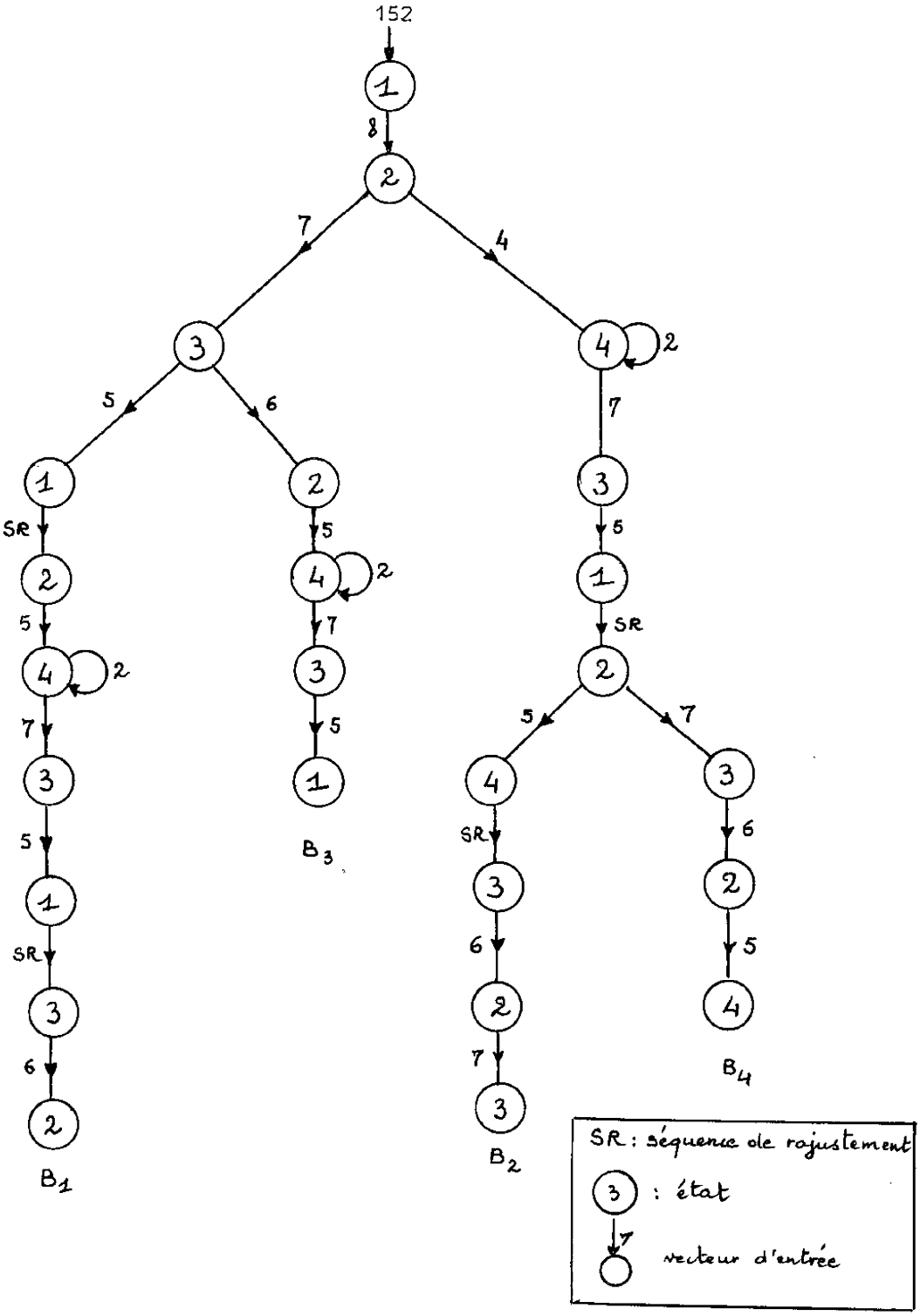
IV.5.2.3.3. Etablissement des séquences de détection sans l'adjacence

L'algorithme suit l'organigramme de la planche IV.4

Le positionnement s'effectue au départ sur l'état initial donné en entrée. S'il existe des vecteurs isolement à tester, ils le sont, conformément à leur ordre d'apparition dans la liste des vecteurs à tester. Après chacun d'eux,on doit adjoindre un V.T. (ou plusieurs) d'adjacence, sauf si le vecteur isolement en est un. Il y a ici une possibilité d'ouverture de l'arbre de séquence : s'il existe plusieurs V.T. adjacents,il faut essayer toutes les possibilités et créer autant de branches nouvelles que de vecteurs adjacents moins un, car on ne peut savoir à priori quelle branche sera la plus courte. On continue la recherche sur la première branche, mais les autres sont mises à jour jusqu'au point d'ouverture,en inscrivant les vecteurs transferts qui sont à l'origine des nouvelles branches (noté : arbre de séquence sur l'organigramme). Si le test d'un vecteur isolement suivi de la condition d'adjacence conduit dans un autre état, on suit la transition en laissant non testés, les vecteurs isolement de l'état de départ (s'il en reste). Il faudra donc revenir sur cet état pour finir le test. Le processus recommence sur le nouvel état atteint. Si le test des vecteurs isolement n'a pas modifié l'état du système, on réalise ensuite le test des vecteurs inscription. Ici aussi,peut apparaître l'ouverture de l'arbre à deux niveaux :

- s'il y a plusieurs vecteurs inscriptions à tester
- s'il y a plusieurs V.T. adjacents à un vecteur inscription donné.

En suivant toujours la première branche, la séquence s'allonge et l'algorithme se reboucle de la même façon jusqu'à ce que tous les vecteurs soient testés sur cette branche.



Arbre de séquence sans adjacence : Bascule JK 74 H 72

figure IV . 8 . a.

Ensuite, on recherche s'il existe des branches non terminées et dans l'affirmative, l'algorithme reprend sur la plus longue après avoir recensé les vecteurs testés par ce tronçon de branche non achevée. Celle-ci, au cours de son élaboration peut donner naissance à de nouvelles branches pour les mêmes raisons que ci-dessus.

La fig. IV.8.a représente l'arbre de séquence du circuit SN 74H72 dont le graphe est donné fig. IV.7.b

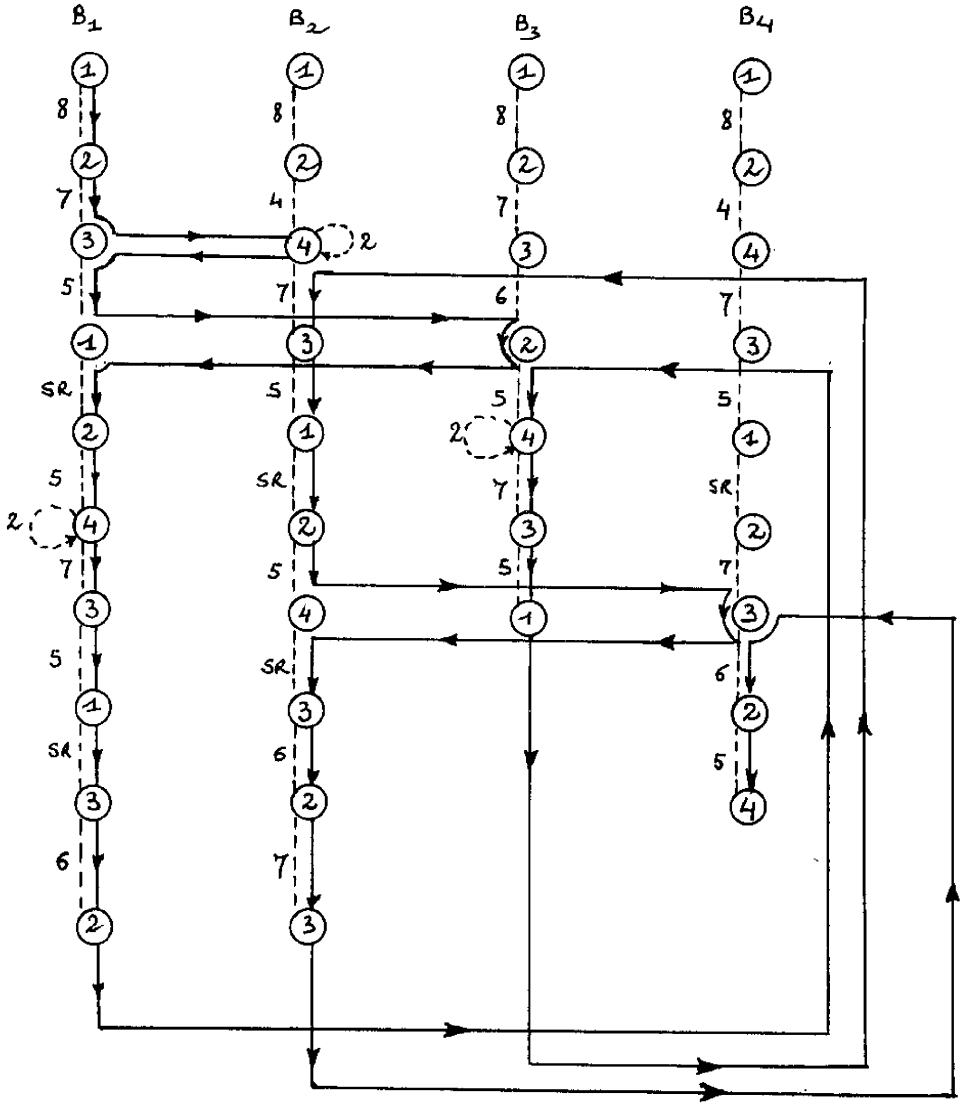
Nous allons examiner comment il se construit (fig. IV.8.b)

La liste des vecteurs à tester se présente de la façon suivante :

Etat ① : 8
Etat ② : 57
Etat ③ : 56
Etat ④ : 27

On cherche le premier vecteur à tester à partir de l'état ① (état initial) : 8 . Il conduit à l'état ②. La liste précédente devient : 0
57
56
27

Les sorties en ② et ① sont identiques. Il faut donc chercher un vecteur transfert adjacent à 8, capable d'assurer l'observabilité. Il y a deux possibilité : 7 et 4. On prend en compte la première (B_1) et on note qu'un branchement a été créé à ce niveau en inscrivant en mémoire la fraction de branche (① , ② , ④ : états, et 8, 4 : vecteurs) qu'il faudra exploiter par la suite (B_2). En revenant sur la première branche on note que le vecteur 7 est testé : 0 et qu'il conduit à l'état ③. Depuis cet état, deux vecteurs
50
56
27
sont à tester : 5 et 6. Il faut ici aussi créer une nouvelle branche B_3 (① , ② , ③ , ② : états, 8, 7, 6 : vecteurs d'entrée). On continue ensuite sur la



Construction de l'arbre de séquence: ——→

Séquences de détection (sans adjacence): - - - - -

figure IV.8.b.

première branche par le vecteur 5, (les vecteurs qui restent à tester sont alors : $\begin{pmatrix} 0 \\ 50 \\ 06 \\ 27 \end{pmatrix}$, jusqu'à ce que tous les vecteurs soient traités. Il n'y a pas sur cette branche d'autre ouverture. Lorsqu'elle est terminée, on remonte ligne par ligne jusqu'à rencontrer la branche la plus longue non terminée (B_3).

Après avoir reconstitué la liste des vecteurs qui restent à tester : $\begin{matrix} 0 \\ 50 \\ 50 \\ 27 \end{matrix}$, l'algorithme reprend sur cette branche. Le processus se poursuit ainsi conformément à la figure IV.8.b, jusqu'à ce que toutes les possibilités aient été envisagées.

On obtient pour cet exemple quatre séquences de test sur lesquelles on doit appliquer l'algorithme d'adjacence qui suit.

IV.5.2.3.4. Adjacence des séquences de détection

On doit considérer deux cas distincts

- adjacence de deux vecteurs d'un même état
- adjacence à réaliser à l'aide d'une séquence de rajustement (4) (bridge séquence)

1. Adjacence de deux vecteurs d'un même état

Elle met en jeu deux vecteurs isolement, ou un vecteur isolement et un vecteur inscription conduisant dans un autre état. Dans ces deux éventualités, il faut trouver un ensemble de vecteurs, le plus court possible qui constitue une séquence d'adjacence et laisse le système dans l'état où il se trouve.

Prenons par exemple la branche n° 3 de la fig. IV.8.a et représentons la façon dont sont stockées les informations en mémoire.

8	7	6	5	2	7	5
2	3	2	4	4	3	1

La première ligne indique les vecteurs de la séquence de détection établie sans la contrainte d'adjacence, la deuxième les états dans lesquels les vecteurs de la colonne correspondante amènent le système.

Après le test d'isolement du vecteur 2 dans l'état 4 doit s'effectuer le test du vecteur 7 qui assure la transition de l'état 4 à l'état 3.

L'adjacence entre les deux vecteurs 2 et 7 va se déduire de la construction d'un arbre. Le sommet de cet arbre est constitué par le vecteur 2. On place au dessous de 2 tous les vecteurs qui lui sont adjacents et on continue ainsi à partir de ceux-ci jusqu'à rencontrer le vecteur 7 (fig. IV.9.a). Simultanément s'élabore un deuxième arbre où figurent les états atteints par chacun des vecteurs.

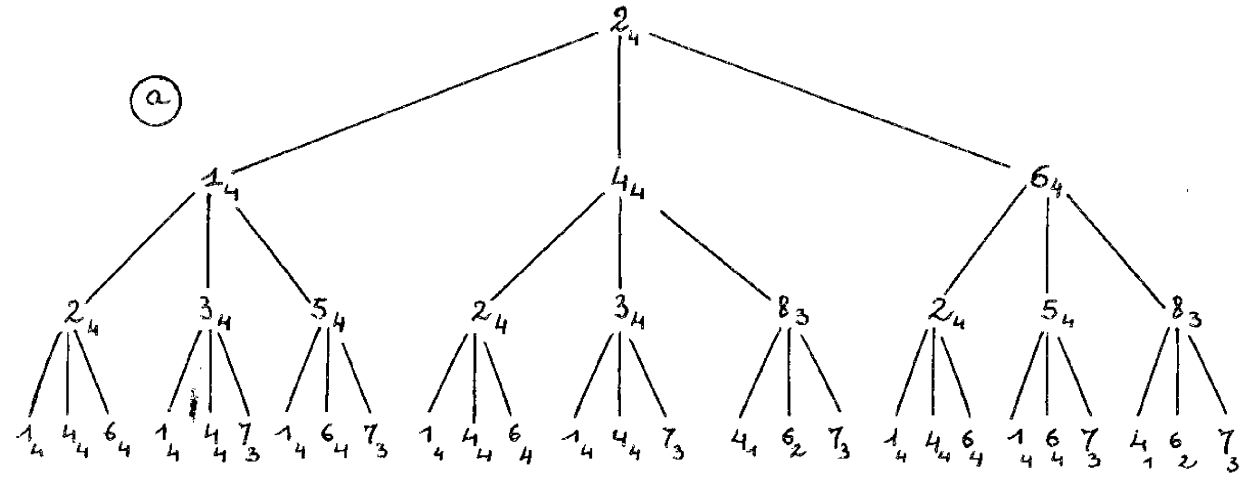
Les solutions suivantes permettent le passage du vecteur 2 au vecteur 7.

2	1	3	7
2	1	5	7
2	4	3	7
~~2~~	~~4~~	~~8~~	~~7~~
2	6	5	7
~~2~~	~~6~~	~~8~~	~~7~~

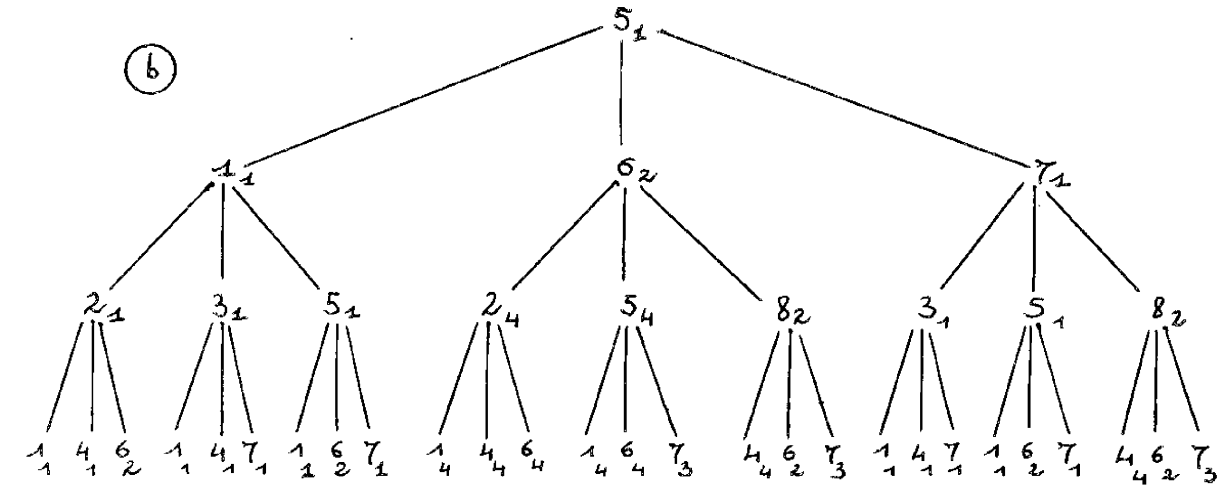
Les 4ème et 6ème solutions ne conviennent pas car le vecteur 8 provoque la transition de l'état 4 à l'état 3. Sur les quatre solutions qui restent, on choisit la première trouvée afin d'abréger la recherche et de diminuer le temps de calcul. On inclut donc la séquence d'adjacence 1 3 entre les deux vecteurs de la séquence primitive.

En appliquant ce principe à toutes les paires de vecteurs successives de cette séquence, on obtient une séquence de détection adjacente.

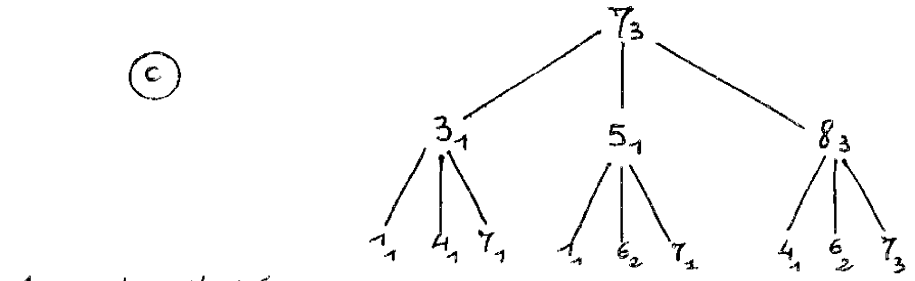
8	7	8	6	5	6	2	1	3	7	5
2	3	3	2	4	4	4	4	4	3	1



Arbre d'adjacence entre deux vecteurs d'un même état (3e branche)
codage : voir fig. IV



Arbre d'adjacence conduisant dans un état donné.



1 ← vecteur d'entrée
1 ← état

figure IV . 9

qui constitue une des quatre solutions de cet exemple.

2. Adjacence à réaliser à l'aide d'une séquence de rajustement.

A partir du même exemple (fig. IV.8.a) examinons la branche numérotée 1.

8	7	5	-	5	2	7	5	-	6
2	3	1	2	4	4	3	1	3	2

Il existe ici deux séquences à définir qui devront assurer les transitions entre états (l'état sur lequel elles conduiront est connu) en respectant l'adjacence. Le problème est plus complexe et nécessite deux étapes. Exposons les en déterminant la deuxième séquence d'adjacence à inclure dans la séquence ci-dessus.

- La première étape consiste à construire un arbre à partir de l'ensemble $\binom{5}{1}$ dans lequel on détecte la séquence la plus courte aboutissant à l'état 3 (fig. IV.9.b).

On obtient 3 solutions possibles :

5 6 5 7
5 6 8 7 et on choisit ici aussi la première combinaison afin de rendre
5 7 8 7 le calcul plus rapide.

- La deuxième étape revient à traiter le cas de l'adjacence de deux vecteurs d'un même état entre $\binom{7}{3}$ et $\binom{6}{2}$ (fig. IV.9.c).

La séquence à inclure est alors donnée par la réunion des deux sous séquences calculées

6	5	7	|	8
2	4	3	|	3

Ces deux algorithmes permettent de traiter tous les problèmes d'adjacence qui se posent dès que l'arbre de séquence est établi, et d'inclure des séquences de longueur minimale.

IV.5.2.3.5. Séquence de détection la plus courte

L'algorithme d'adjacence est appliqué à toutes les séquences de l'arbre, car on ne peut a priori connaître la séquence la plus courte, de par les vecteurs rajoutés capables de modifier le rapport des longueurs des séquences.

La séquence de détection la plus courte, que nous retenons pour solution se déduit alors de la simple comparaison des longueurs de chacune. Une fois qu'elle est connue, le calcul de la séquence de sortie correspondante donne la solution complète.

IV.5.2.3.6. Tracé de la séquence de détection

Cette solution sous forme codée est traduite de façon visuelle par un sous programme de tracé. Pour utiliser au mieux le format de la page, la hauteur des niveaux est fonction du nombre d'entrées - sorties du système, ce nombre pouvant varier dans des limites très larges (limite inférieure : 2 - limite supérieure : résultats lisibles).

Ainsi la séquence

Entrées	:	8	7	8	6	5	6	2	1	3	7	5
Sorties	:	2	1	1	2	1	1	1	1	1	1	2
Etats	:	2	3	3	2	4	4	4	4	4	3	1

solution de l'exemple étudié correspond à la figure IV.10.

IV.5.3. Utilisation du programme - Possibilités

Ce programme comporte environ 1000 instructions FORTRAN. Il peut être connecté à la suite du programme d'élaboration des pires cas (si la capacité du calculateur le permet) ou utilisé seul à partir des résultats de ce dernier. Il faut alors lui donner en entrées :

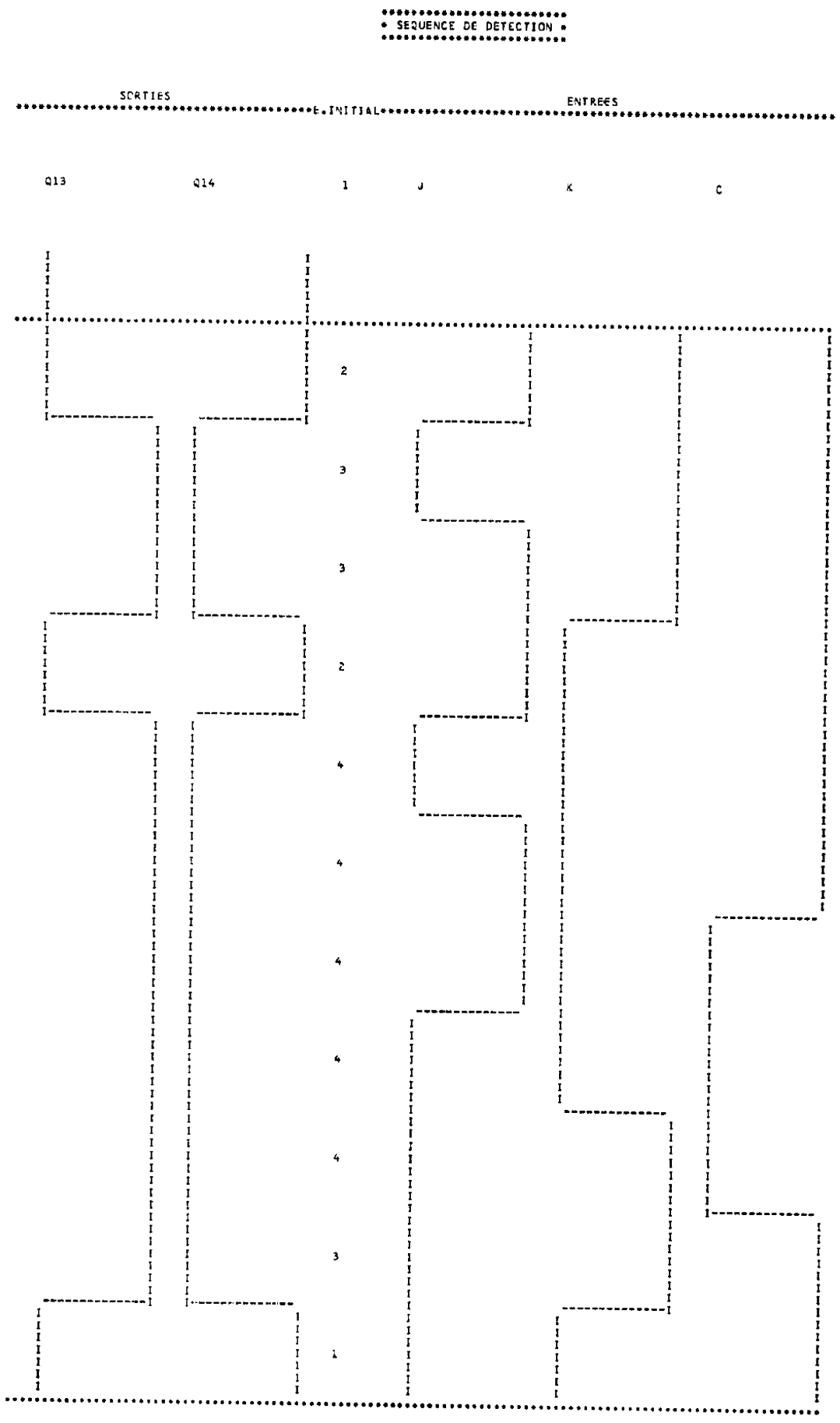


FIGURE IV 10

Séquence de détection-JK 74H72

- la matrice de test
- les entrées et sorties du système
- le nombre de variables secondaires
- les états stables et la valeur des sorties correspondantes (combinaison binaire des sorties).
- l'état initial du système.

POSSIBILITES :

A cause des codages employés (§ IV.5.2.3.2.) le nombre d'entrées, de variables secondaires et de sorties est pour l'instant limité à trois. Ce nombre peut sans grosses difficultés devenir plus grand,au détriment de la place mémoire.

TEMPS CALCUL :

Pour les circuits du type bascule D ou JKMS, le temps de calcul y compris le tracé des séquences varie entre 30" et 1' (IBM 7044).

CONCLUSION :

Nous avons vu que cette méthode de détermination des séquences de détection était applicable aux circuits fortement connectés. Toutefois de par la nature des méthodes d'analyse et de test fonctionnel exposées aucune hypothèse n'a été faite quant au caractère synchrone ou asynchrone d'un circuit, non plus qu'à la nature des pannes en mesure de l'affecter. La contrainte d'adjacence a été introduite ici pour la même raison. Elle accroît la longueur des séquences obtenues qui reste malgré tout fort raisonnable (74H72 : 11, 7472 : 36) si l'on pense à ce que donnerait le test systématique de tous les vecteurs de chaque état compte tenu de l'observabilité (par la bascule 74H72 : on obtiendrait une séquence de longueur 40 environ).

En règle générale cette méthode ne donne pas des séquences minimales à partir de la matrice de test. En effet, avoir imposé depuis chaque état le test des vecteurs isolement avant celui des vecteurs inscription (en vue de

systématiser l'algorithme) peut nécessiter l'adjonction de vecteurs d'adjacence qui n'interviendraient pas si les vecteurs isolement étaient testés "au passage" sur un état. Cet accroissement de longueur n'est cependant que très minime ce qui permet de dire que les séquences obtenues ont une longueur quasi optimale. (par exemple : pour la bascule 7472 ceci diminuerait d'un la longueur de la séquence totale).

C H A P I T R E V

TESTS DE DIAGNOSTIC

TESTS DE DIAGNOSTIC

V-1 INTRODUCTION

En résumant, au chapitre II les principales méthodes de test mises au point nous avons pu voir que le problème du diagnostic des systèmes séquentiels avait été très peu abordé.

SESHU ([12 - 13], § II.2) a fourni une solution algorithmique très intéressante par partitionnements successifs de la table des pannes.

Nous allons partir de la même idée : c'est-à-dire la simulation d'un ensemble de pannes choisies, mais en utilisant le programme d'analyse. On obtiendra ainsi les caractéristiques du circuit soumis à chaque panne. L'exploitation de ces résultats s'effectuera par une méthode faisant appel à la notion de séquence de synchronisation et dont le but est de donner des séquences de test permettant de distinguer une panne ou un ensemble de pannes parmi l'ensemble complet choisi.

V-2 SIMULATION D'UN ENSEMBLE DE PANNES ET ANALYSE DU SYSTEME SOUMIS A CHAQUE PANNE.

HYPOTHESES :

- Les pannes susceptibles d'affecter le fonctionnement du circuit se traduisent par des collages à 0 ou à 1 des sorties des modules élémentaires.

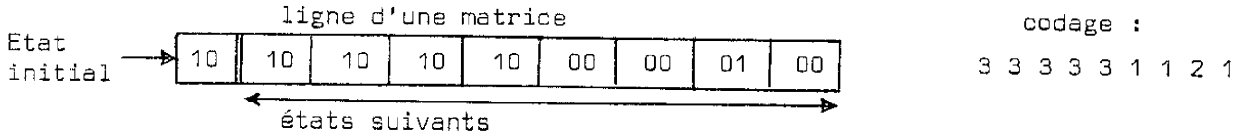
- Il n'apparait qu'une panne à la fois.

V-2.1. Algorithmes

1. En un premier temps, on effectue l'analyse du circuit sans panne (chap. III), les tables d'excitation et de sorties ne faisant pas intervenir les forçages. Nous verrons par la suite que la recherche des séquences de test met en jeu ces deux types de tables, et pour cette raison, nous avons modifié légèrement le programme d'analyse. Cette deuxième version permet la mise en mémoire des tables ainsi que le collage effectif à 0 ou 1 des variables secondaires et des sorties.

Il est également possible d'obtenir les résultats sur carte en vue d'une utilisation ultérieure.

La mise en mémoire nécessite un codage. Afin de ne pas employer trop de place, chaque ligne d'une matrice est codée de la façon suivante :



c'est-à-dire par un nombre entier dont chaque chiffre représente l'équivalent décimal + 1 de la combinaison des variables secondaires, ou l'équivalent décimal de la combinaison binaire des sorties. Ce codage s'il est économique réduit par contre le nombre d'entrées, de variables secondaires et de sorties à trois (comme dans le programme de recherche des séquences de détection) et présente l'inconvénient de nécessiter un décodage pour revenir au niveau d'une case de la matrice (compromis rapidité place).

2. Suppression des forçages

On peut supposer que les circuits de forçage ne sont pas affectés de pannes et ne pas tenir compte de leur effet. Cela revient à supprimer les entrées correspondantes et à apporter des transformations éventuelles à la topologie du circuit. En effet, si certains modules du circuit n'existent que pour permettre l'introduction des forçages (fig. IV.2, module n° 7), leur utilité disparait. L'algorithme doit donc prévoir tous les cas possibles de transformation en fonction de la nature des modules qui composent le circuit.

Cette distinction présente l'intérêt de voir si les entrées prioritaires de forçage apportent un supplément d'information en cas de panne, et permettent le cas échéant un diagnostic plus fin. Nous préciserons ce point plus loin.

3. Simulation des pannes.

Successivement, la sortie de chacun des modules est bloquée à zéro puis à un. Il faut alors déterminer l'incidence d'une panne sur la structure et le comportement du système. La planche V.1 résume tous les cas possibles. Selon la nature des modules qui ont pour entrée la sortie bloquée, la panne pourra ou non se propager dans le circuit. Ainsi l'entrée d'un ET ou d'un ON collée à 0 implique respectivement le collage à 0 ou à 1 de ces modules. De même, l'entrée d'un OU ou d'un NI collée à 1 leur impose le collage à 1 ou à 0 de leurs sorties. On conçoit dès à présent que certains ensembles de pannes ne seront pas discernables si une panne réagit en chaine sur plusieurs modules. Le but de l'algorithme suivant est de donner une nouvelle matrice topologique image du circuit transformé par une panne.

- collage de la sortie d'un module
- suppression de la ligne correspondante
- recherche des lignes ayant pour entrée la sortie du module collé.

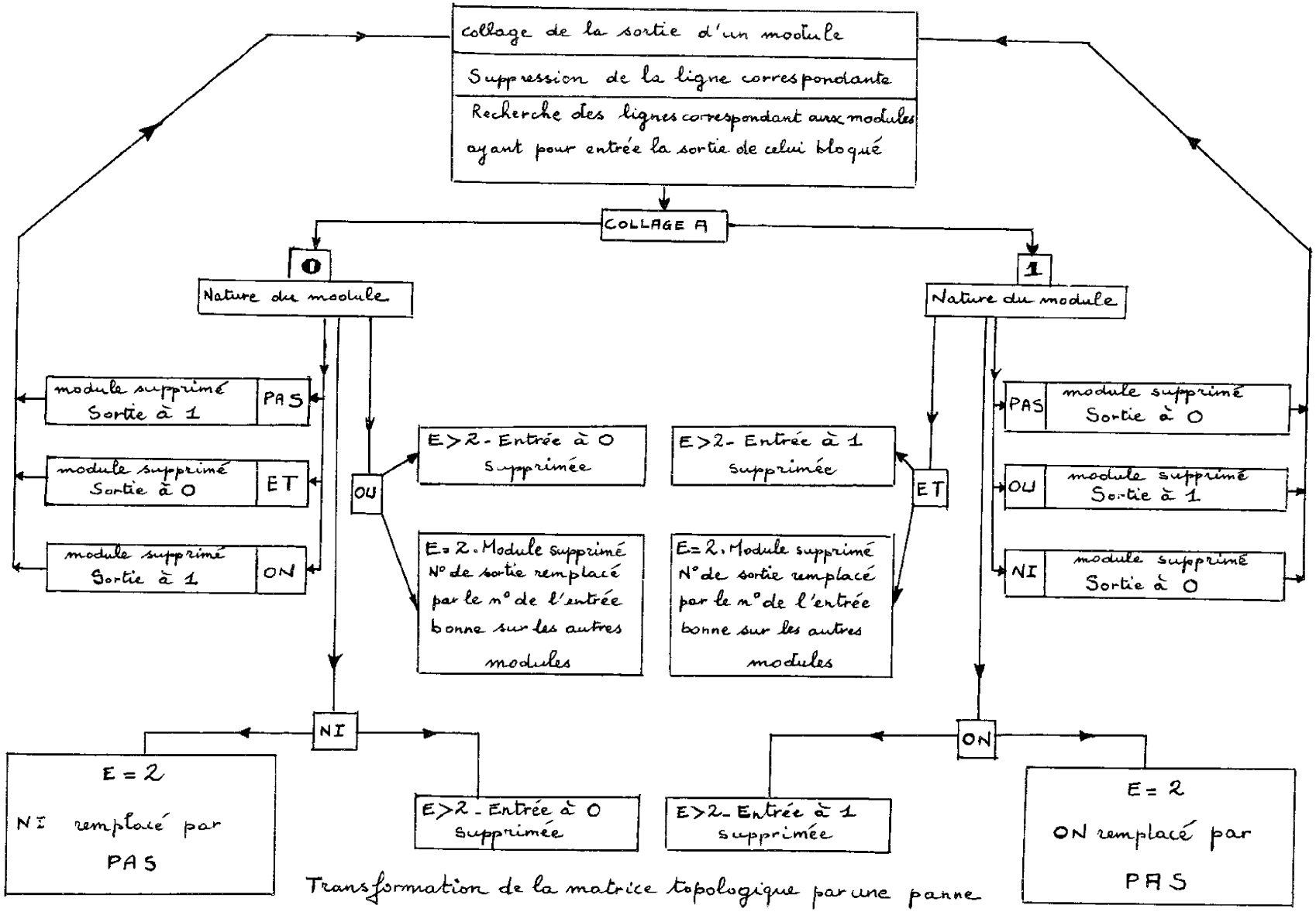
L'effet du collage d'une des entrées d'un module correspond à l'un des cas ci-dessous :

. Collage à 0 d'un module :

- PAS : ce module est supprimé et l'influence de la valeur 1 de sa sortie sera examinée par la suite.
- OU : Si le module possède plus de 2 entrées, on supprime l'entrée collée.

S'il a deux entrées, ce module devient inutile. Le numéro de sa sortie est remplacé par le numéro de l'entrée valide partout où il intervient.

- ET : suppression du module. L'effet sur le reste du circuit de sa sortie à 0 s'étudiera ensuite.



Transformation de la matrice topologique par une panne

Planche V.1

- NAND : idem sortie à 1

- NOR : - Si le nombre d'entrées dépasse 2 : suppression de l'entrée en panne.

- S'il est égal à 2, ce module est remplacé par un PAS

. Collage à 1 d'un module.

-PAS : supprimé sortie à 0

- OU : supprimé sortie à 1

- ET : 2 entrées : module inutile dont le n° est remplacé par celui de l'entrée bonne.

. plus de 2 entrées : suppression de l'entrée défectueuse.

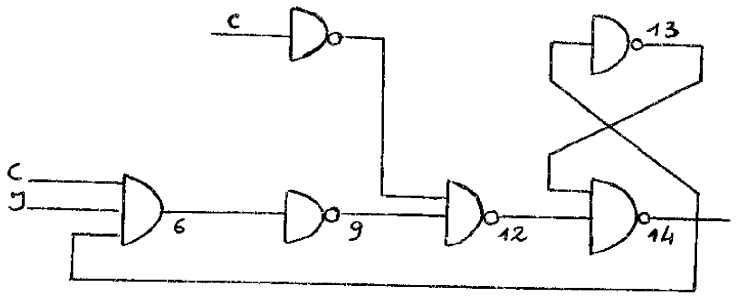
- NAND : 2 entrées : module remplacé par un PAS

. $>$ 2 : entrée supprimée

- NOR : supprimé - sortie à 0.

Après l'examen de l'influence du collage d'un module sur tous les modules dont sa sortie constitue une des entrées, l'algorithme recommence sur ceux qui se trouvent à leur tour figés... Chaque modification du circuit se traduit immédiatement sur la matrice topologique.

Illustrons ceci par un exemple : le collage à 1 du module 5 de la fig IV.1 transforme le circuit comme suit (l'entrée de forçage $\overline{RZ}$ et le module 7 étant enlevés) :



4. Equations et tables

Connaissant la matrice topologique du circuit soumis à une panne simple, on peut déterminer ses caractéristiques par l'analyse. On utilise les mêmes variables secondaires que celles du circuit sans panne afin de comparer les fonctionnements respectifs à partir de références identiques. Ainsi l'analyse se réduit à l'établissement des équations logiques et des tables. Le collage à 0 ou à 1 d'une variable secondaire est prévu.

5. Forçages considérés comme entrées normales

En transformant le codage des entrées primaires (§ III.3.1.) le programme d'analyse ne fera plus jouer un rôle particulier aux forçages et la simulation des pannes reprend en tenant compte des entrées ainsi ajoutées.

V-2.2 Interprétation des résultats

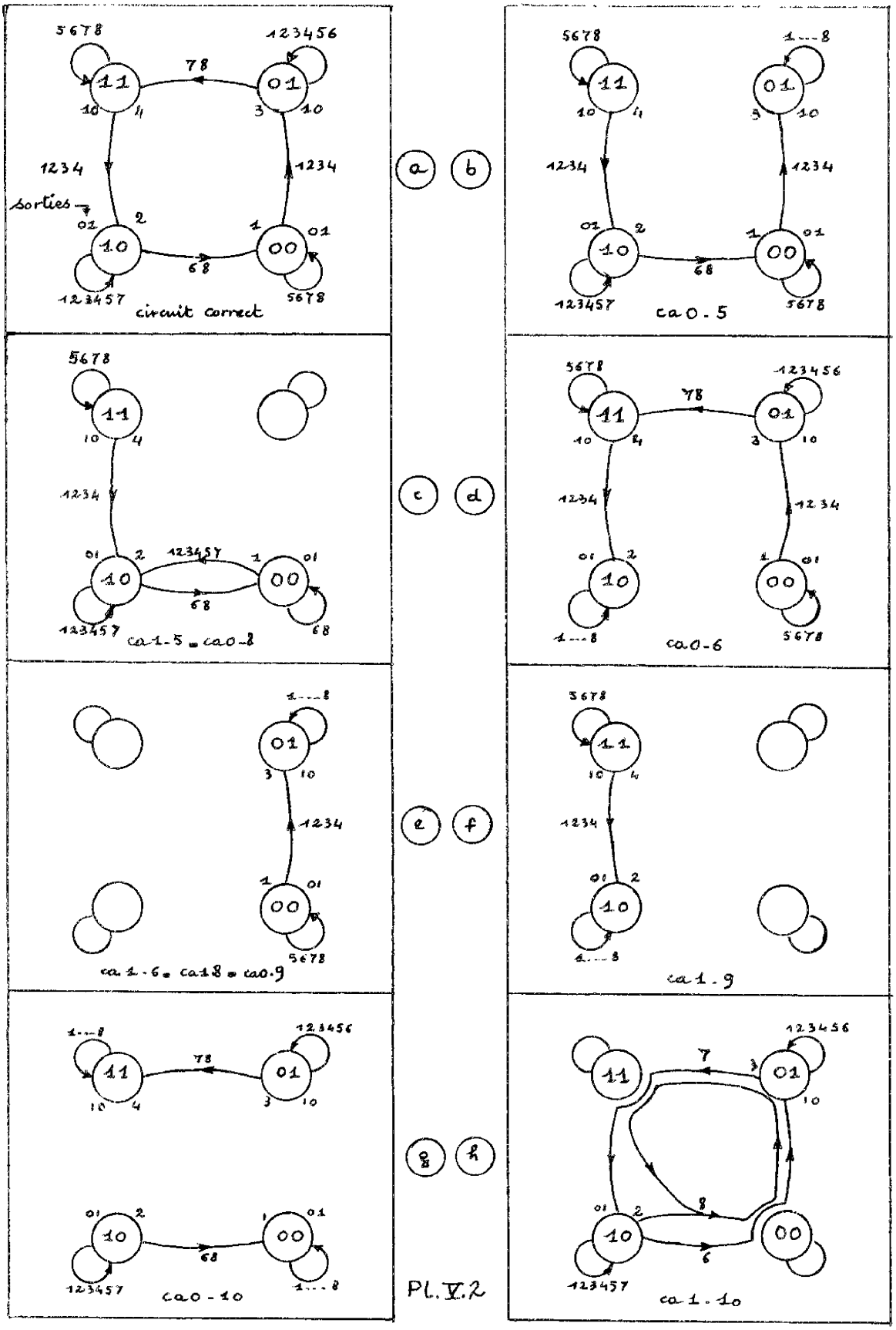
V.2.2.1. Sans forçage

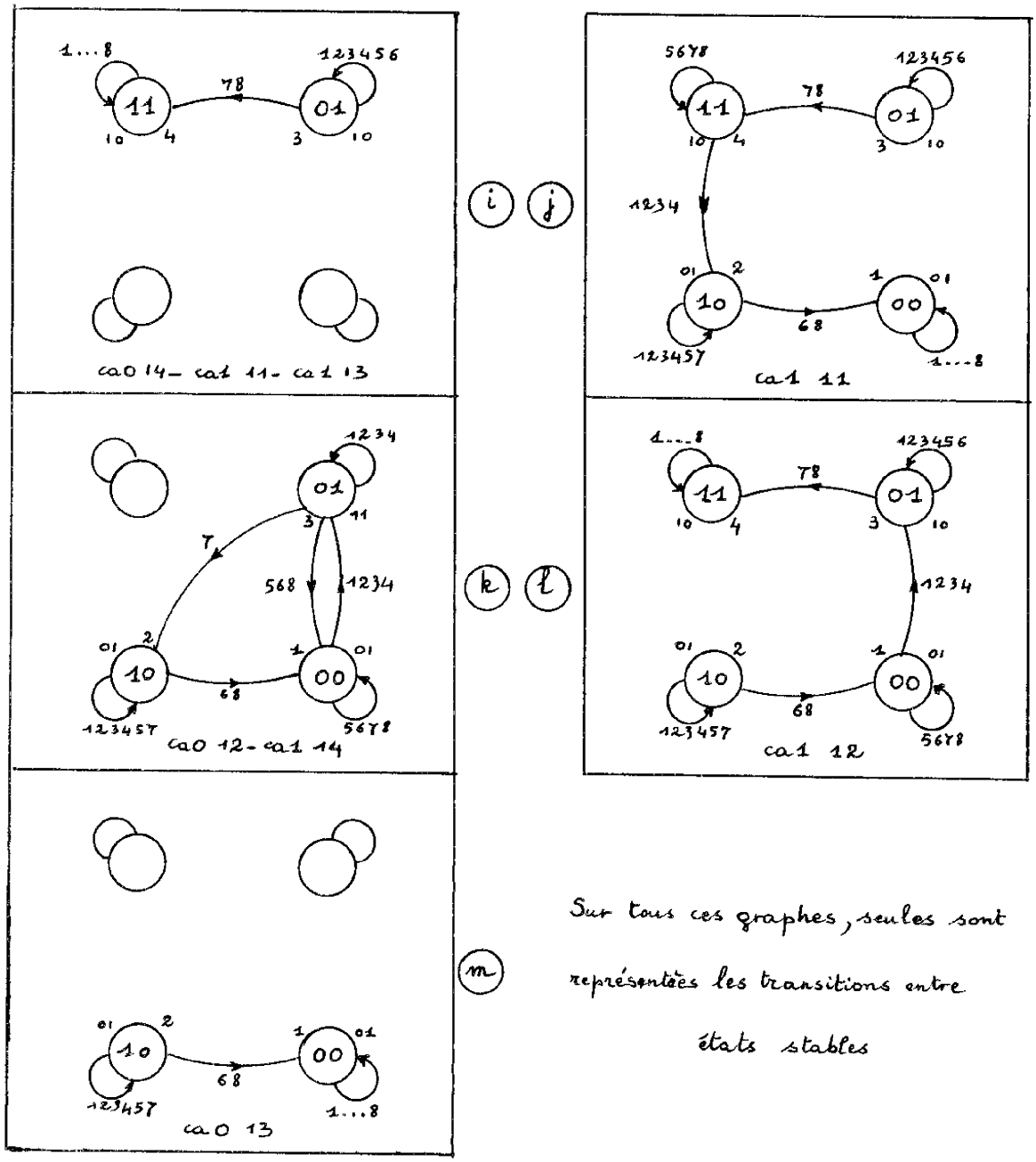
Les planches V-2 et V-2 bis représentent les graphes de fluence du circuit JK 7473 (fig. IV.1) soumis à l'algorithme de simulation de pannes (entrée de remise à zéro supprimée).

- On peut d'abord remarquer que certaines pannes ont un effet identique sur le circuit : mêmes équations logiques $\Longrightarrow$ mêmes graphes. Ce phénomène correspond à la réaction en chaine signalée au paragraphe précédent, dûe à la nature des modules successifs.

Nous appellerons ces pannes : pannes strictement identiques car aucune méthode ne pourra les différencier.

C'est le cas par exemple du collage à 1 de 5 et du collage à 0 de 8. En effet, si le module 5 est collé à 1, il impose au module 8 (NI), un niveau 0 constant sur sa sortie. Donc la panne ca 08 ne peut se distinguer du collage à 1 de 5.


a
b
c
d
e
f
g
h
circuit correct
ca0.5
ca1.5 = ca0.8
ca0.6
ca1.6 = ca18 = ca0.9
ca1.9
ca0.10
ca1.10
Pl. V.2



Sur tous ces graphes, seules sont représentées les transitions entre états stables

Bascule JK 7473

Planche V. 2. bis

La réciproque est vraie également car la sortie du module 5 conduit au module 8 et seulement à lui.

- En observant les sorties du système, deux pannes qui se traduisent par deux graphes différents peuvent ne pas être discernables. Sur la planche V.2 le graphe (d) montre le blocage du système dans l'état (10), alors que le graphe (c) traduit le mauvais bouclage d'une transition laissant une variable secondaire libre d'évoluer. Cependant, vue des sorties, cette évolution n'est pas observable et tout se passe comme si le système se figeait dans un état. Ces deux pannes pourtant différentes ne peuvent être discernées : nous les appellerons pannes indiscernables.

Lorsque le graphe s'ouvre en faisant apparaître un état puits, le système ira se bloquer dans cet état (ex. pl. V.2.b)

Essayons d'interpréter comment se traduisent les effets des pannes simulées sur les équations logiques du circuit.

Reprenons les équations des variables secondaires sous la forme

$$y_{(T+1)} = A.\ y_{(T)} + B$$

. si $B = 0$ et $y_{(T)} = 0$, on a $y_{(T+1)} = 0$

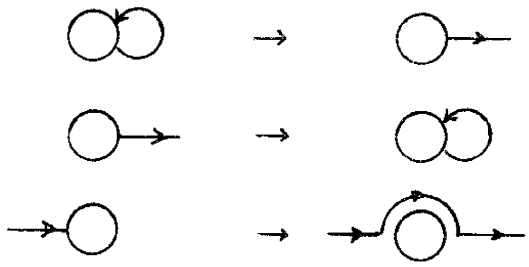
Cette condition nécessaire de blocage de $y = 0$ n'est en général pas suffisante pour que tout le système se bloque. Elle le devient si elle se vérifie sur tous les y.

. si $A = 1$ et $y_{(T)} = 1 \Rightarrow y_{(T+1)} = 1$ (remarque identique)

. si $A = 1$, $B = 0 \Rightarrow y_{(T+1)} = y_{(T)}$. La variable secondaire se bloque dans l'une des deux valeurs 0 ou 1. Ceci est à l'origine de l'ouverture de plusieurs transitions d'un graphe et de sa transformation en plusieurs sous graphes (g).

. si $A = 0$, toute possibilité d'isolement de la variable secondaire disparait puisque $y_{(T)} = B$, ce qui peut se traduire par le fait qu'un état devient instable pour une transition (h, k) (cycles, états d'arrivée différents).

. Lorsque A et B sont modifiés mais existent toujours ou que les conditions ci-dessus s'appliquent différemment sur chaque variable secondaire (par ex. A = 0 dans l'une, B = 0 dans l'autre) on peut trouver un mélange des trois types suivants de transformation d'une transition (c)



A la vue de cette brève analyse, nous pensons qu'il serait intéressant, dans l'avenir, d'exploiter plus profondément cette étude : il sera peut être possible de déterminer directement avec les équations d'un circuit, la façon dont se transforme son graphe de fonctionnement et par là même, de caractériser une panne parmi un ensemble donné.

Examinons un peu plus en détail les graphes de la planche V.2

- en (b) : une seule transition a disparu. Le circuit s'il n'est pas dans l'état (3) au départ, évolue normalement, mais dès qu'il atteint cet état, la variable secondaire se bloque à 0. C'est donc le maitre qui subit l'influence du collage à 0 de 5, l'esclave n'étant pas du tout affecté. En effet, puisque la sortie du module 5 se trouve bloquée à 0, dès que la sortie du module 9 passe à 0, le module 8 voit sa sortie imposée à 1, ce qui fige le module 9 à 1. Dès ce moment, 8 et 0 ne peuvent plus être modifiés et le maître se bloque.

- en (g) : lorsque le module 10 est collé à 0, aucun transfert n'est possible du maître dans l'esclave, puisque les portes 11 et 12 se trouvent bloquées à 1 et isolent l'esclave. Le graphe se coupe donc en deux par disparition des deux transitions de transfert 0 et 1 dans l'esclave.

- en (h) : le module 10 étant collé à 1, tout isolement de l'esclave disparait si bien que les informations inscrites dans le maître sont immédiatement transférées dans l'esclave et par le jeu des bouclages, réinjectées dans le maître. Il s'ensuit, selon le vecteur d'entrée affiché, l'apparition de cycles et la transformation de deux états stables en états instables. Ces trois exemples montrent que le tracé des graphes correspondant aux pannes simulées traduit bien la modification physique qui en résulte.

V.2.2.2. Avec les entrées de forçage

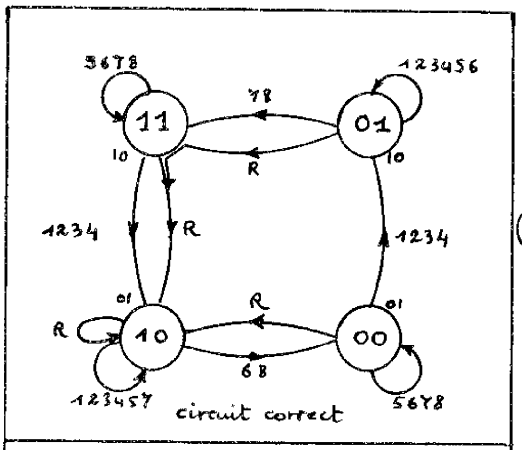
En reprenant la même procédure, on obtient les graphes de fluence représentés sur les planches V.3 et V.3 bis. Ceux-ci vont nous permettre à partir d'un cas particulier de noter quelques remarques importantes. Sur les graphes la lettre R signifie entrée de forçage agissante (R = 0).

- Si les états dans lesquels la remise à zéro amenait le système, disparaissent, celle-ci n'agit plus et ne peut que donner 1 en sortie (Pl. V.3. bis (d)) cela montre que l'hypothèse avancée en (1 1) (§ II.4) :

- la remise à zéro amène le circuit dans un état particulier, quelle que soit la panne qui l'affecte -, ne repose sur aucun fondement valable.

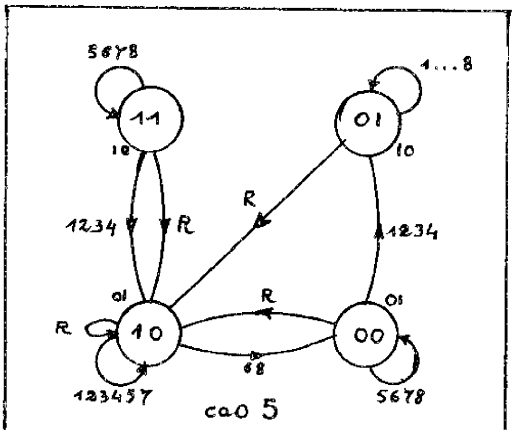
- Dans le cas où il existe des états pour lesquels la sortie vaut 0 et 1 (pl. V.2 (e)) on peut aussi remarquer que la remise à 0, place le système à 0 ou à 1 selon la configuration des entrées primaires (pl. V.3 (e));ceci renforce la remarque précédente et implique que l'hypothèse d'un état de départ connu ne peut être raisonnablement envisagée.

Par contre, lorsqu'elle est agissante, cette entrée de forçage peut rendre stables des états qui ne le seraient pas sans elle (pl. V.3 (f) et pl. V.2 (e)) et permettre de discerner une panne à l'intérieur d'un ensemble. (ex : ca 1.8 pl. V.3 (f) qui était confondu avec ca 1 -6, ca 0 9)

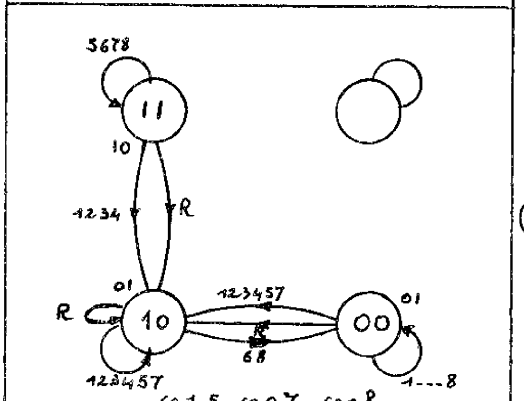


circuit correct

a b

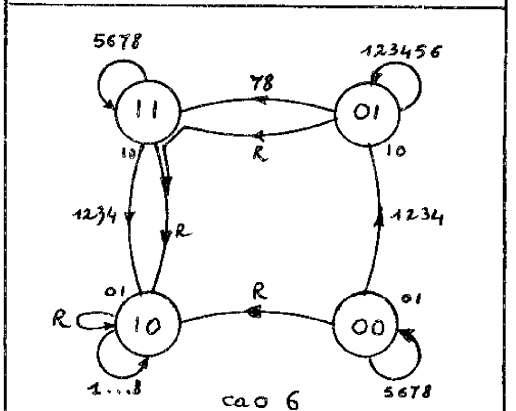


cas 5

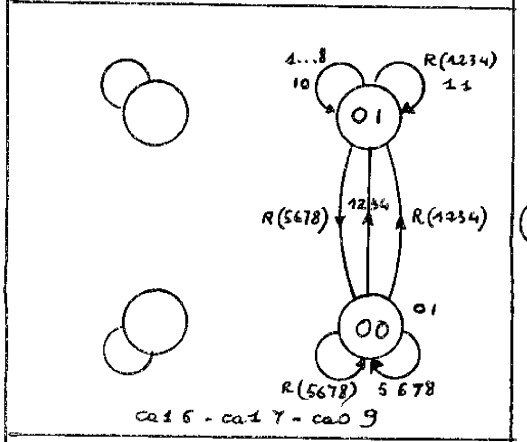


cas 5 - cas 7 - cas 8

c d

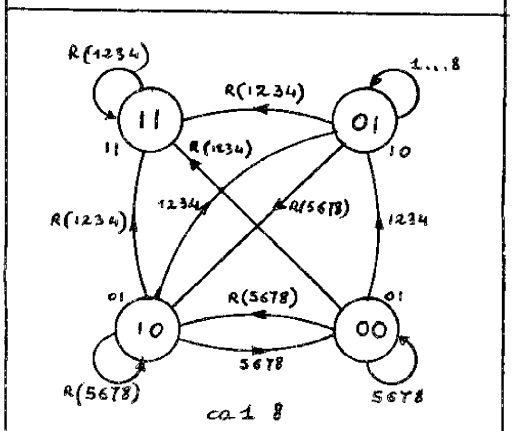


cas 6

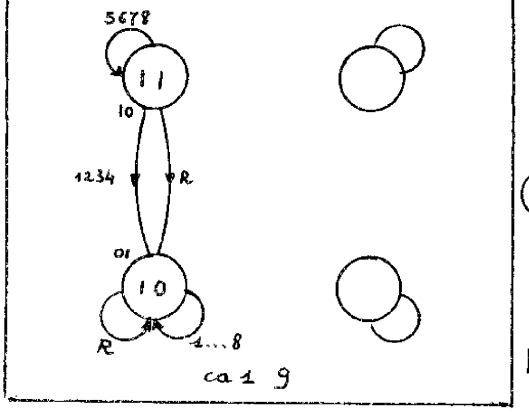


cas 6 - cas 7 - cas 9

e f



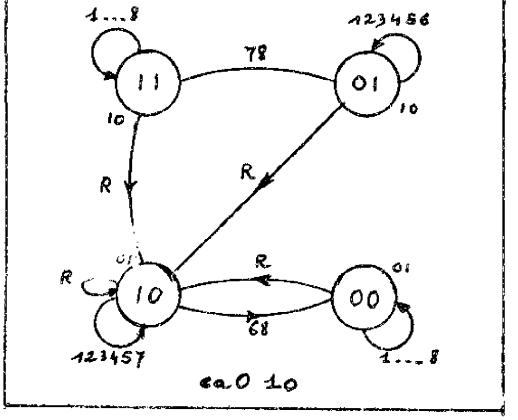
cas 8



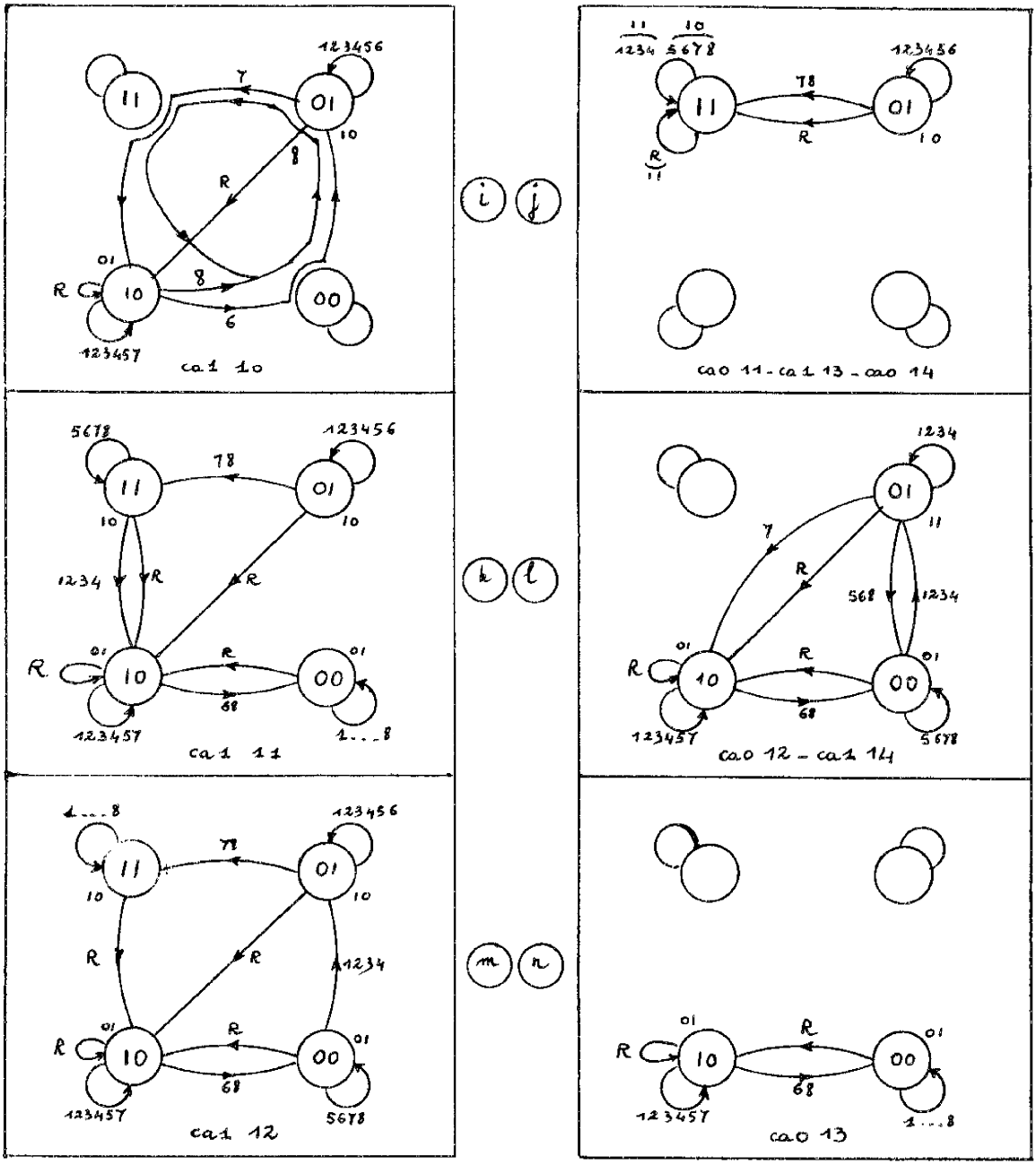
cas 9

g h

Pl V.3



cas 10



JK 7473 avec entrée

de remise à zéro

Planche V.3.bis

Elle peut aussi refermer un graphe qui avait été ouvert par une transition mauvaise (pl. V.3 (m) et pl. V.2 (l)), supprimer des états puits (pl. V.3 (f) au lieu de pl. V. 2 (i)), et connecter des sous graphes disjoints par une panne (pl. V.3 (h) au lieu de pl. V.2 (g)).

On conclut de ceci que même si l'entrée de forçage n'accomplit plus correctement sa tâche, elle apporte quand même un supplément d'information (en assurant des transitions que les vecteurs d'entrée primaires ne peuvent plus satisfaire), qui doit se retrouver au moment de l'établissement des séquences de test.

Ceci est dû au fait que dans le circuit, cette entrée possède une position particulière qui la rend prioritaire. Elle est en effet capable de figer des sorties de modules et par conséquent de contrebalancer l'effet d'une panne. C'est le cas par exemple du collage à 1 du module 12. Si l'on ne fait pas intervenir cette remise à 0, lorsque le module 13 voit sa sortie passer à1, le module 14 (ON) ayant ses deux entrées à 1, passe à 0 ce qui a pour résultat de bloquer définitivement le module 13 à 1. Si par contre on utilise la remise à zéro, en plaçant un 0 sur cette entrée, elle oblige le module 14 à passer à 1 ce qui permet le déblocage du module 13 et referme le graphe de fonctionnement du circuit (pl. V.3 bis (m))

V.2.3 Cas particulier : Analyse et Diagnostic des compteurs

Nous examinons ces éléments comme des cas particuliers à cause de leur entrée unique. La seule séquence de test applicable est une séquence de comptage. La simulation de collage à 0 et à 1 de tous les modules dans laquelle on inclut le tracé de la matrice des transitions suivi de l'algorithme du § III.2.2.11 donne une analyse fine et complète du comportement d'un compteur en cas de panne. On peut à l'aide de ces résultats tracer le graphe de fonctionnement du circuit soumis à chacune des pannes. Si le compteur comporte n modules, on obtient 2 n+1 graphes ce qui demande un travail de dépouillement assez considérable et impensable lorsque le système devient important. Nous l'avons effectué sur un exemple simple : celui d'un compteur deux bits (2 JK 74H72 - fig. III.10 : schéma, fig. III.11 : graphe du circuit correct).

On observe des cycles de comptage différents (pl. V.4, V.5, V.6, V.7) selon la panne présente et la comparaison des graphes permet un regroupement des pannes en ensembles de pannes indiscernables (tableau V.1)

Deux pannes seront dites indiscernables si la séquence d'Entrée-Sortie est identique. C'est le cas par exemple du ca 0 de 2 et du ca 1 de 2 dont le graphe de fonctionnement diffère mais donne lieu à la même séquence de comptage.

Une panne peut aussi appartenir à plusieursgroupes :

- ca 0 3 qui selon l'état initial bloque le circuit à la valeur de sortie 0 ou 2, ou l'entraine dans un cycle de comptage.

- ca 0 11 qui permet au système de fonctionner selon deux cycles différents, toujours en fonction de l'état initial.

La localisation des pannes se fait ici de façon assez grossière mais cela est dû au fait que l'on ne peut commander le compteur que par une seule entrée,supprimant ainsi la possibilité de l'orienter dans un état particulier.

L'utilisation de la remise à zéro fournit le moyen de rendre le partitionnement des pannes plus précis grâce aux propriétés énoncées au paragraphe précédent.

V.2.4. Conclusions

Cet exemple qui représente un des circuits les plus difficiles à tester (le pire cas étant le registre à entrée et sortie série), a été développé ici dans le seul but de montrer les possibilités qu'offrent les programmes d'analyse et de simulation de pannes par la précision de leurs résultats.

De plus si l'algorithme de simulation de pannes a été prévu pour étudier les pannes simples de collage à 0 et à 1 des sorties de modules, par l'écriture d'un programme adéquat, il est tout à fait possible de se pencher sur le problème des pannes multiples, ou de tout autre type de panne (tel le court

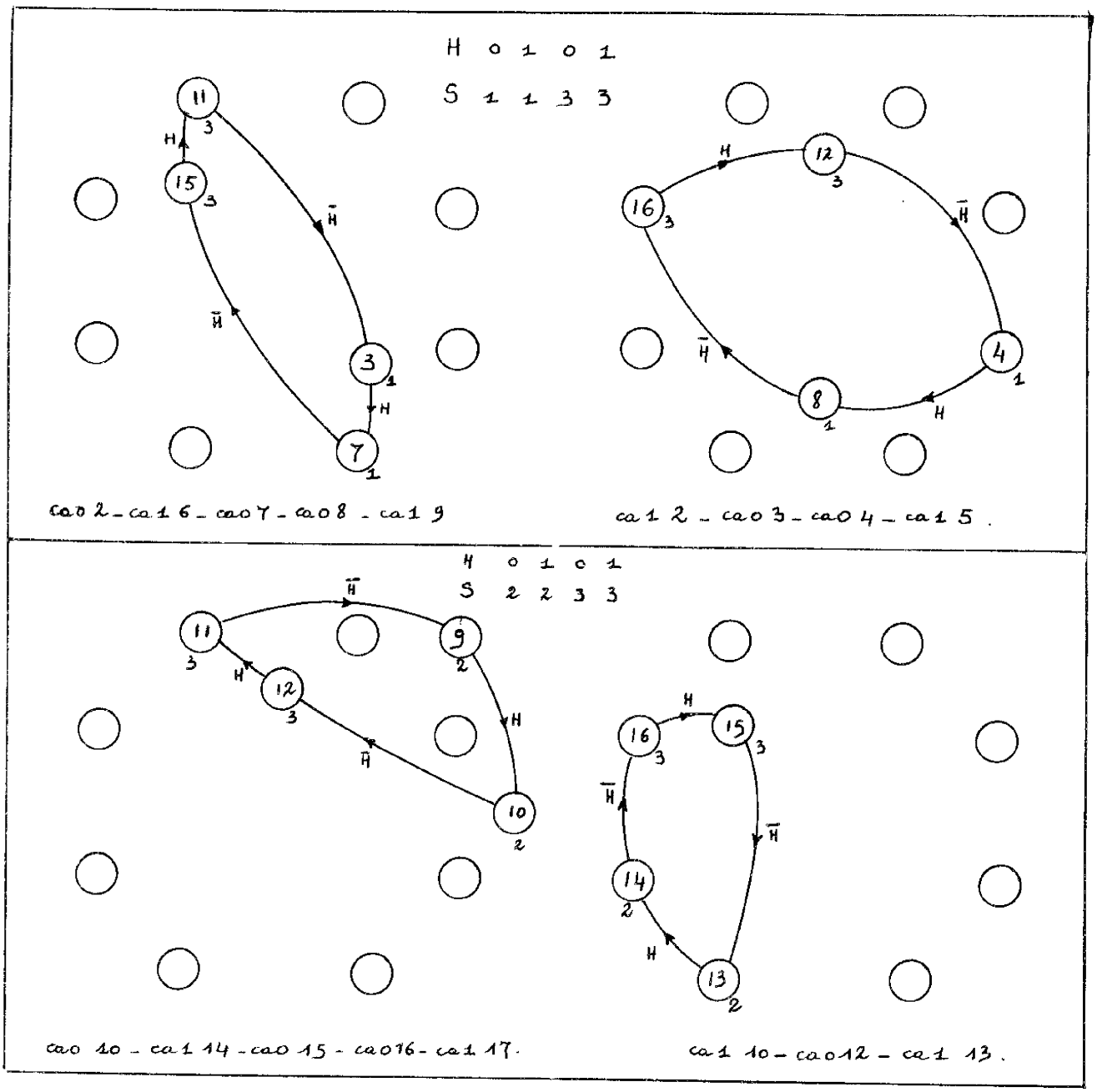

H 0 1 0 1
S 1 1 3 3
11
15
3
7
H
$\bar{H}$
ca0 2 - ca1 6 - ca0 7 - ca0 8 - ca1 9
12
16
4
8
ca1 2 - ca0 3 - ca0 4 - ca1 5 .
H 0 1 0 1
S 2 2 3 3
11
9
12
10
ca0 10 - ca1 14 - ca0 15 - ca0 16 - ca1 17.
16
15
14
13
ca1 10 - ca0 12 - ca1 13 .

Planche V 4

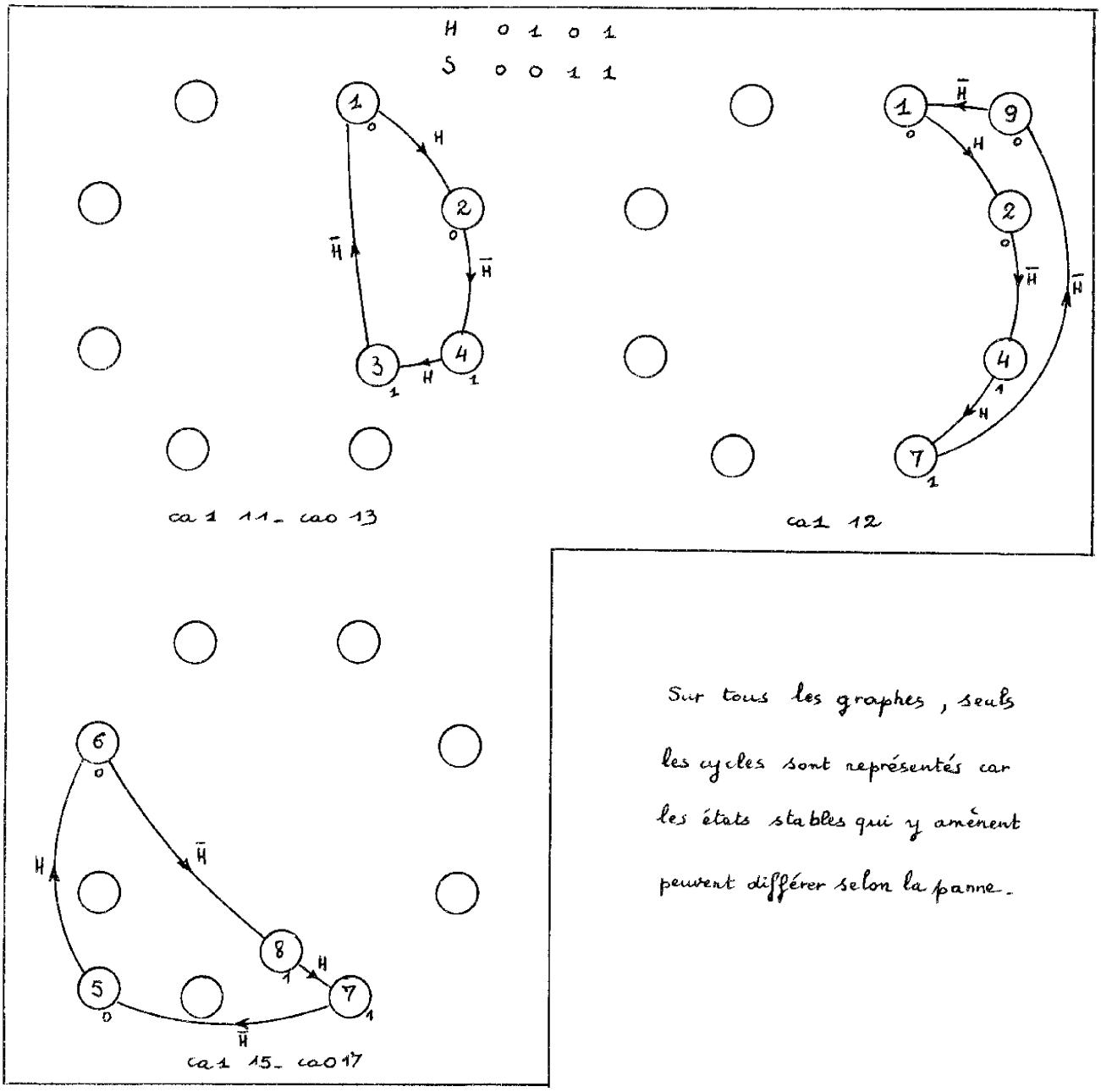

H 0 1 0 1
S 0 0 1 1
1
2
3
4
ca 1 11 - cao 13
1
9
2
4
7
ca1 12
6
8
5
7
ca 1 15 - ca0 17
Sur tous les graphes, seuls les cycles sont représentés car les états stables qui y aménent peuvent différer selon la panne.

Planche V 5

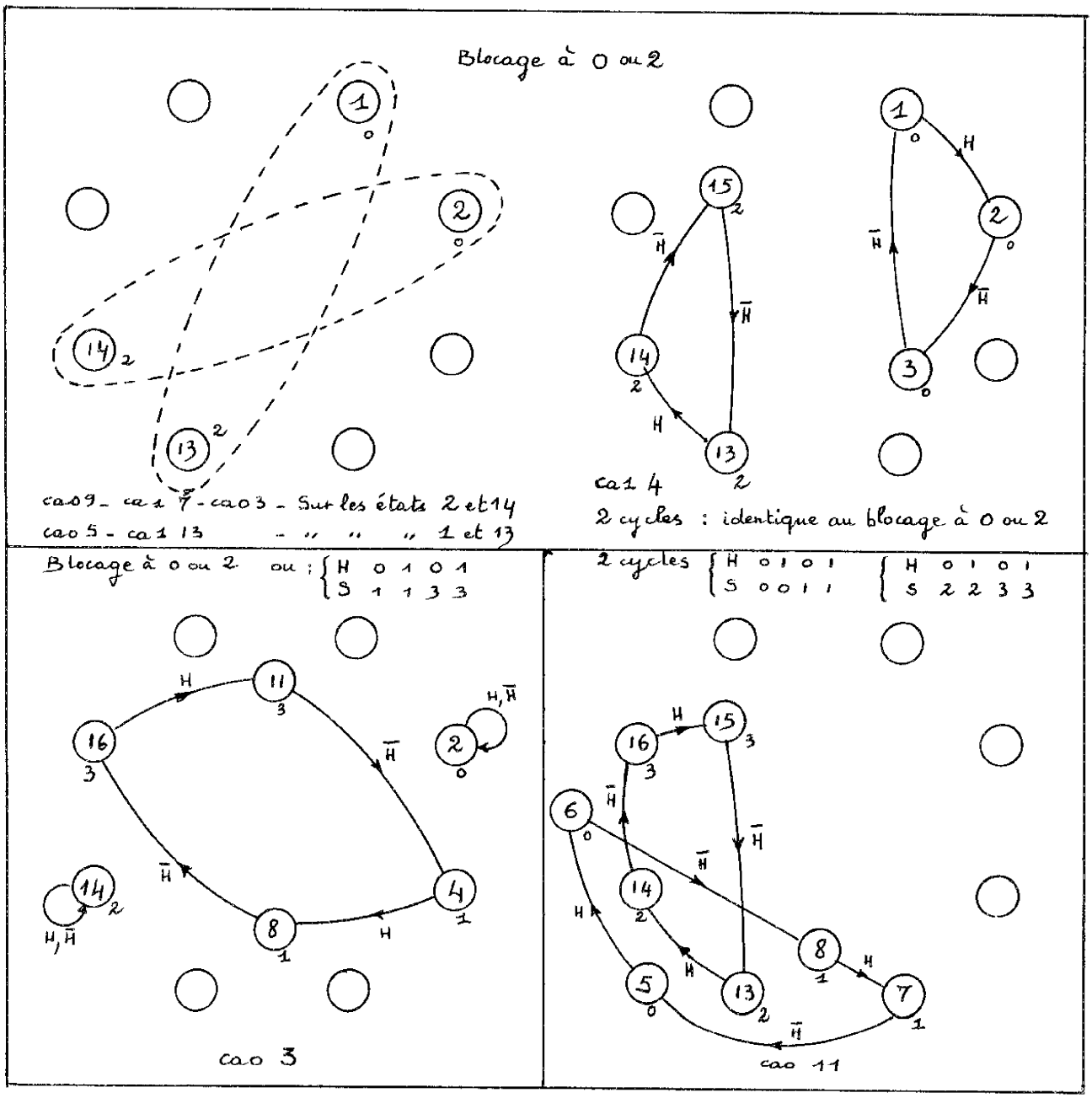

Blocage à 0 ou 2
cas 9 - cas 7 - cas 3 - Sur les états 2 et 14
cas 5 - cas 13 - " " " 1 et 13
cas 4
2 cycles : identique au blocage à 0 ou 2
Blocage à 0 ou 2 ou : H 0 1 0 1 / S 1 1 3 3
cas 3
2 cycles H 0 1 0 1 / S 0 0 1 1 | H 0 1 0 1 / S 2 2 3 3
cas 11

Planche V 6

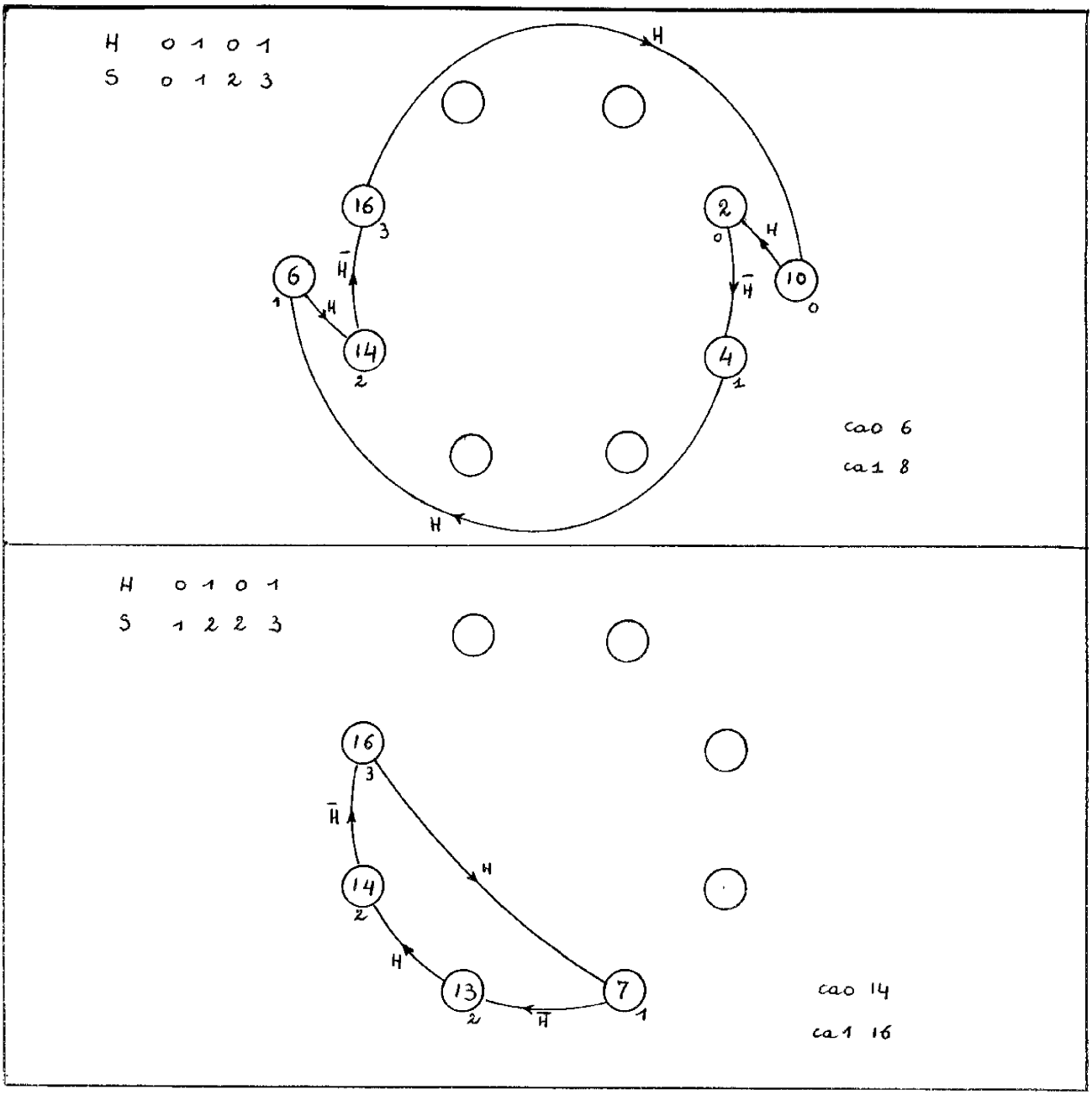

H 0 1 0 1
S 0 1 2 3
16
3
2
0
6
1
10
0
14
2
4
1
cao 6
ca1 8
H 0 1 0 1
S 1 2 2 3
16
3
14
2
13
2
7
1
cao 14
ca1 16

Planche V 7

SEQUENCES

ENTREE	0	1	0	1
SORTIE	1	1	3	3

	PANNE COLLAGE A	DU MODULE
	0	2
	1	2
*	0	3
	0	4
	1	5
	1	6
	0	7
	0	8
	1	9

SEQUENCES

ENTREE	0	1	0	1
SORTIE	2	2	3	3

	PANNE COLLAGE A	DU MODULE
	0	10
	1	10
*	0	11
	0	12
	1	13
	1	14
	0	15
	0	16
	1	17

SEQUENCES

ENTREE	0	1	0	1
SORTIE	0	0	1	1

	PANNE COLLAGE A	DU MODULE
*	0	11
	1	11
	1	12
	0	13
	1	15
	0	17

BLOCAGE A 0 OU 2

	PANNE COLLAGE A	DU MODULE
*	0	3
	1	3
	1	4
	0	5
	1	7
	0	9

SEQUENCES

ENTREE	0	1	0	1
SORTIE	0	1	2	3

	PANNE COLLAGE A	DU MODULE
	0	6
	1	8

SEQUENCES

ENTREE	0	1	0	1
SORTIE	2	2	3	1

	PANNE COLLAGE A	DU MODULE
	0	14
	1	16

Compteur binaire 2 bits - Regroupement des pannes indiscernables

Tableau V 1

circuit de deux métallisations voisines dans un circuit intégré, modifiant le schéma logique du système) qui permettent de serrer de plus près la réalité.

V-3 SEQUENCES DE DIAGNOSTIC

A la vue des résultats présentés concernant deux exemples simples, on conçoit qu'en matière de pannes dans les systèmes séquentiels, tout peut arriver et qu'une méthode générale de localisation de pannes, si tant est que l'on en trouve une, n'est pas envisageable à court terme étant donnée la complexité du problème.

Nous nous limiterons donc à l'étude de systèmes de petite taille, répondant aux hyptohèses ci-dessous qui s'ajoutent à celles données au début du § V.2. (Nous utiliserons jusqu'à la fin de ce chapitre, des sigles pour éviter les répétitions continuelles).

HYPOTHESES :

- les circuits doivent être fortement connectés
- ils doivent posséder au moins une séquence de synchronisation

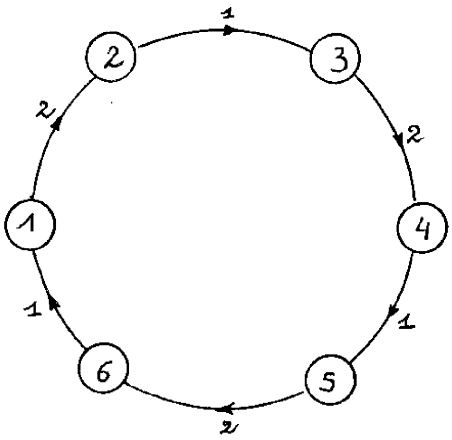
V.3.1. Propriétés des séquences de synchronisation

Rappelons la définition donnée au chapitre II.

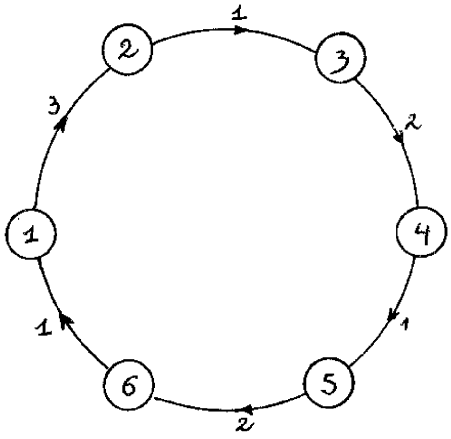
DEFINITION :

Une séquence de synchronisation (S.S.) est une séquence d'entrée dont l'application garantit de conduire le circuit dans un certain état final, sans que soit connu l'état initial dans lequel il se trouvait.

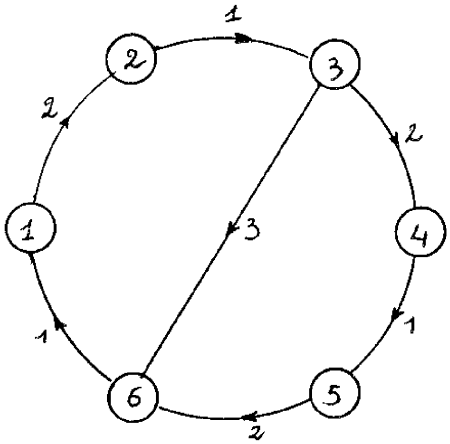
L'intérêt des S.S. réside dans le fait qu'elles permettent d'opérer à partir de n'importe quel état et de se passer d'entrées prioritaires de



a



b



c

Une ss amenant en (1) :

1 2 1 2 1

Deux ss :

1 3 2 1 3 1 conduit en (1)

1 3 2 1 2 1 3 2 " " " (6)

figure V 1

forçage qui, on l'a vu, ne présentent pas toutes les garanties en cas de panne.

Quelles sont les conditions que doit remplir un système pour avoir une S.S.?

THEOREME : un système séquentiel aura une S.S. conduisant dans l'état J, si depuis tout état I ≠ J, il existe au moins une séquence Si amenant de I à J et telle que appliquée depuis J, elle laisse ou ramène le système dans l'état J.

La démonstration de ce théorème est immédiate : en effet si les conditions énoncées ci-dessus se trouvent vérifiées on arrivera toujours à déterminer un assemblage des séquences Si tel que appliqué à n'importe quel état il conduise sur l'état J.

Sans aborder une classification rigoureuse, on peut remarquer que les systèmes qui ne possèdent pas de S.S. présentent un graphe de fluence absolument symétrique sur les états et sur les transitions (fig. V.1.a).

La moindre dissymétrie sur une des transitions entraine l'apparition d'une ou plusieurs S.S. (fig. V.1. b et c).

V.3.2. Méthode

Nous allons exposer brièvement les points essentiels que nous reprendrons plus en détail au paragraphe suivant.

La méthode consiste à comparer les matrices d'excitation et de sortie du circuit en panne à celles du circuit correct.

- en premier lieu on détermine toutes les transitions défectueuses observables (sur les sorties).
- ensuite on essaie de trouver une ou plusieurs séquences capables d'amener le circuit bon et le circuit en panne sur l'état à partir duquel une transition est mauvaise, quel que soit l'état initial. Plus on déterminera de séquences et plus la panne étudiée aura des chances de pouvoir se distinguer des autres.

Une séquence complète de test sera formée par l'assemblage d'une séquence de positionnement et de la séquence de test de la transition défectueuse considérée. Le même procédé s'applique à toutes les pannes simulées et celles pour lesquelles les mêmes séquences d'entrée-sortie ont été calculées se trouvent regroupées en un ensemble de pannes indiscernables.

V.3.3 Algorithmes

V.3.3.1. Procédure générale (planche V.8)

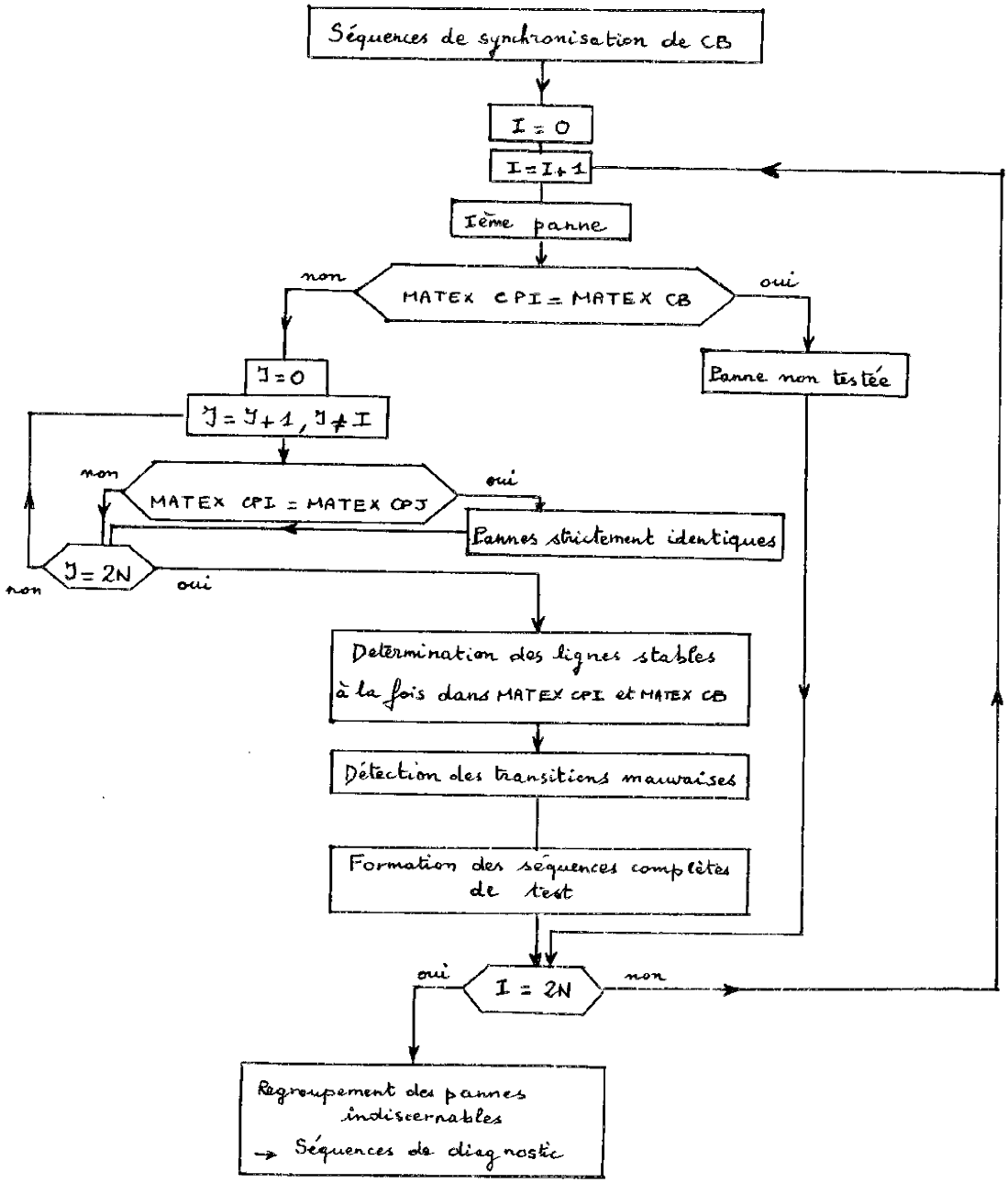
Si le circuit possède n modules on a en mémoire les (2 n+1) matrices d'excitation et les (2 n+1) matrices des sorties calculées par le programme de simulation et d'analyse des pannes.

- En premier lieu s'effectue la recherche des S.S. du circuit correct. Si celui-ci n'en comporte pas, il n'est pas testable par cette méthode. (En vue d'une utilisation ultérieure de ce programme, conjointement avec le programme de simulation de pannes, cette recherche devra s'effectuer dès le stade de l'analyse afin d'éviter un calcul éventuellement inutile).

- Ensuite l'effet de chaque panne est examiné successivement. Si la matrice d'excitation du circuit soumis à la ième panne se révèle identique à celle du circuit correct, le collage à 0 ou 1 correspondant n'affecte pas le fonctionnement logique de la partie séquentielle du système. Il peut en effet arriver qu'un module ait pour tâche d'assurer un bon fonctionnement dynamique (suppression d'aléas) en dehors de toute considération d'ordre logique.

Ceci nous permet de préciser que seuls les modules appartenant à la partie séquentielle d'un circuit et participant effectivement au fonctionnement logique seront testés.

On compare également toutes les autres matrices d'Excitation à celles de la ième panne afin de mettre en évidence les pannes strictement identiques



Procédure générale

Planche V 8

et de ne pas reprendre les calculs sur celles-ci,dans un souci d'économie de temps.

Après ces comparaisons préalables commence l'algorithme d'élaboration des séquences. On recherche des lignes stables à la fois dans la matrice des Excitations du circuit soumis à la ième panne (MATEX CPI), et sur celle du circuit bon (MATEX CB),ce qui revient à supposer qu'il existe au moins un état stable conservé malgré la panne (dans le cas de pannes simples, cette hypothèse est peu contraignante car il semble fort peu probable qu'une panne simple puisse affecter un circuit au point de transformer tous les états stables en états instables et inversement). Cette condition non remplie implique que toutes les transitions de CPI sont mauvaises, limite que nous ne prévoirons pas. Placer les deux circuits dans le même état stable constitue le seul moyen systématique pour déterminer les transitions mauvaises. Cet algorithme, appliqué simultanément dans CB et CPI donne toutes les transitions défectueuses caractérisant CPI. A l'aide de ces résultats s'effectue ensuite la formation des séquences complètes de test. La comparaison des séquences établies pour chaque panne permet enfin de les regrouper en ensemble de pannes indiscernables. Le diagnostic sera d'autant plus fin qu'il y aura beaucoup d'ensembles composés de peu d'éléments.

V.3.3.2. Séquences de synchronisation (14)

On les détermine à l'aide de la matrice des excitations secondaires en formant un arbre.

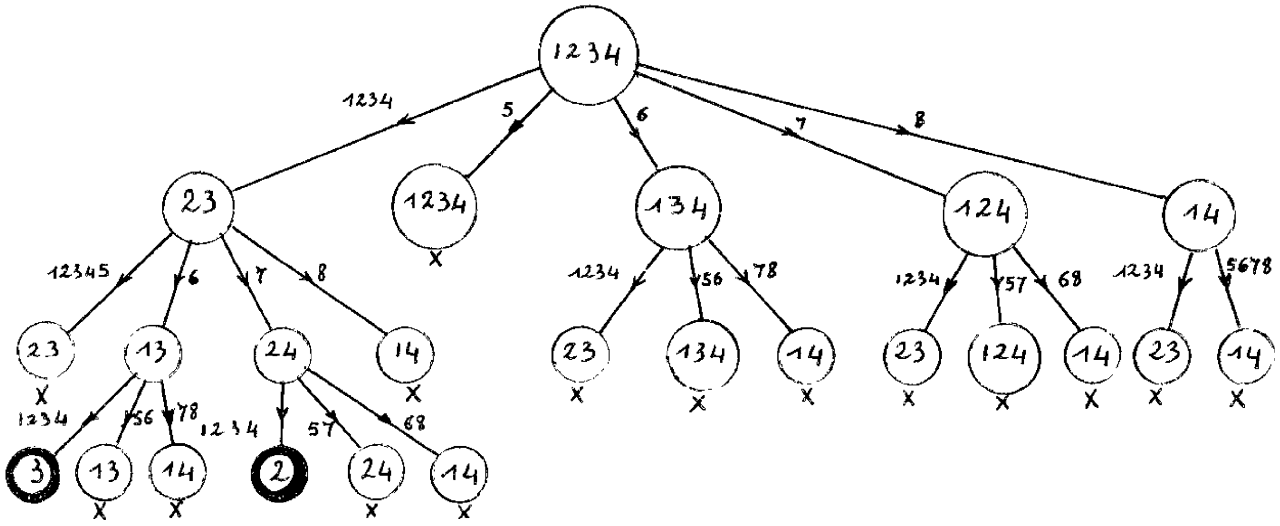
L'état de départ étant inconnu, on regarde en appliquant chacun des vecteurs d'entrée,sur quels états stables ils orientent le système. On effectue ainsi un partitionnement des états. Afin d'obtenir les séquences de synchronisation les plus courtes, un test d'arrêt suit l'élaboration de chaque couche :

- une branche prend fin lorsque :
  - la dernière partition calculée existe déjà sur une couche précédente.

	1	2	3	4	5	6	7	8 ← vecteurs d'entrée
1	3	3	3	3	1	1	1	1
2	2	2	2	2	2	1	2	1
3	3	3	3	3	3	3	4	4
4	2	2	2	2	4	4	4	4

↑ Etats

Matrice d'excitation codée



Arbre des séquences de synchronisation

Deux séquences de synchronisation

$$\begin{cases} 1234 \quad 6 \quad 1234 \rightarrow (3) \\ 1234 \quad 7 \quad 1234 \rightarrow (2) \end{cases}$$

Bascule 7473

Figure V 2

- la dernière partition calculée ne comporte qu'un seul élément. Cette branche constitue alors une séquence de synchronisation amenant le système de n'importe quel état, dans l'état représenté par la partition à 1 élément.

Ex: figure V.2

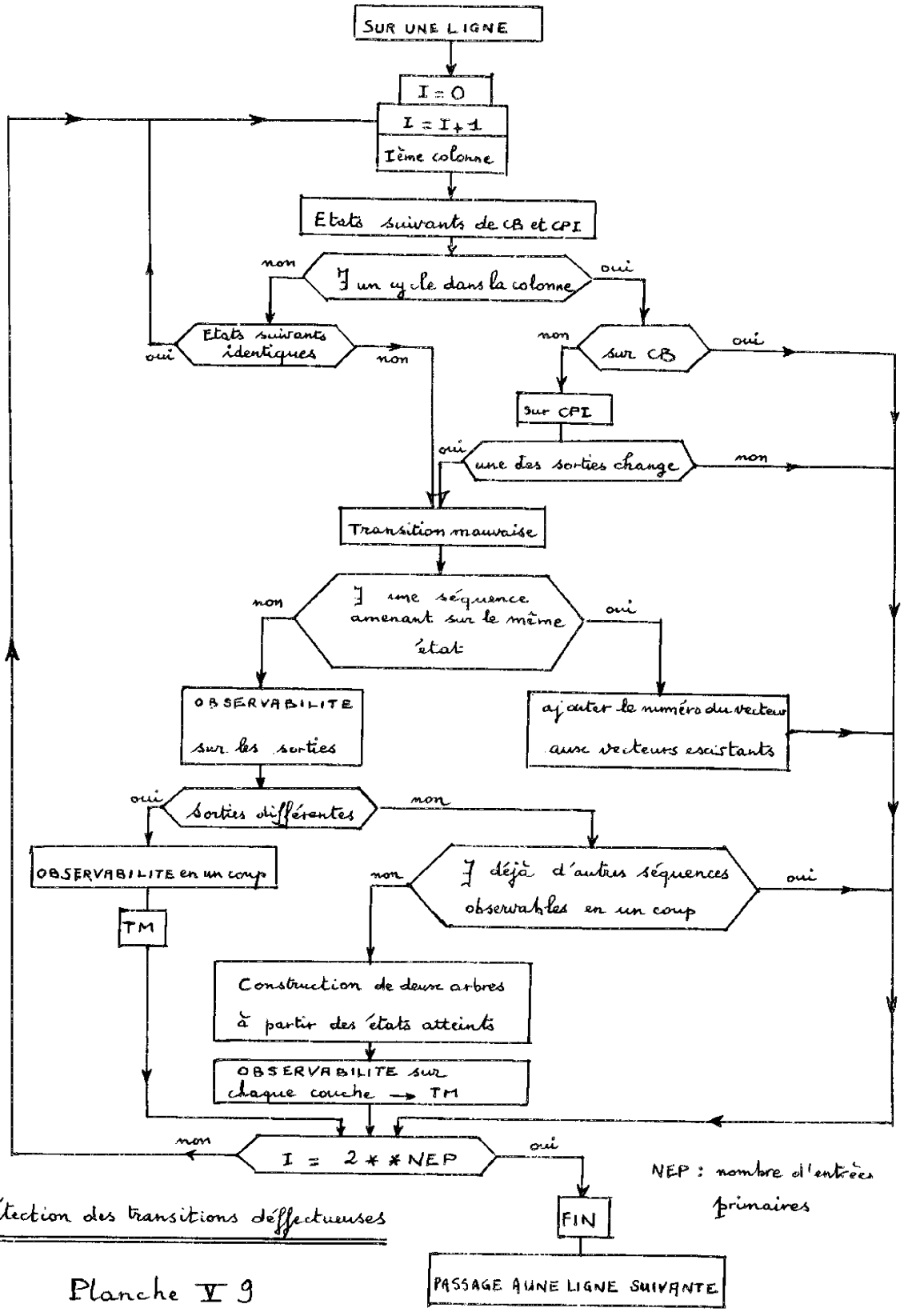
On détermine deux S.S. 1 2 3 4 – 6 – 1 2 3 4 ⟶ ③
1 2 3 4 – 7 – 1 2 3 4 ⟶ ②

V.3.3.3. Détection des transitions défectueuses (planche V.9)

Après que deux lignes stables de même numéro aient été déterminées, il faut examiner où amènent dans CPI et dans CB les transitions à partir de cet état commun. La recherche s'effectue simultanément sur les deux matrices, colonne par colonne c'est-à-dire pour toutes les combinaisons des entrées primaires. Le déplacement dans une colonne indique les états suivants atteints par les deux circuits (on trouvera un exemple à la fin de ce paragraphe). Plusieurs cas peuvent se produire.

1 - cycle dans la colonne CB : nous excluons comme lors des programmes précédents les circuits comportant des cycles. Par conséquent, le vecteur d'entrée qui provoque ce cycle ne sera pas examiné.

2 - cycle dans la colonne de CPI : - si à l'intérieur de ce cycle la valeur des sorties de CPI ne change pas, cette transition mauvaise (TM) sera observable



Détection des transitions défectueuses

Planche V 9

seulement si dans l'état atteint par CB, les sorties ont une valeur différente.

- si la valeur des sorties de CPI change au cours du cycle, cette TM sera observable.

3 - si les états suivants sont identiques, la transition due au vecteur d'entrée correspondant est bonne dans CPI; la comparaison repart sur une autre colonne.

4 - dans le cas où les états suivants diffèrent, une TM apparait. On regarde en premier lieu si un autre vecteur d'entrée conduisant CB et CPI dans les mêmes états que ceux que l'on vient d'atteindre, n'a pas déjà été traité. Dans l'affirmative ces deux vecteurs appartiennent à la même TM. Il suffit de les réunir.

S'il n'en est pas ainsi, le vecteur d'entrée correspondant à la colonne examinée indique une nouvelle TM. Il se pose alors le problème de l'observabilité de cette TM sur les sorties du système.

OBSERVABILITE

Nous sommes partis d'un état commun à CB et CPI, et nous venons de déterminer une transition conduisant CB et CPI dans deux états différents. Si les sorties correspondant à ces deux états différents également, l'observabilité sera réalisée en un seul coup, conduisant à une séquence de test de TM de longueur 1.

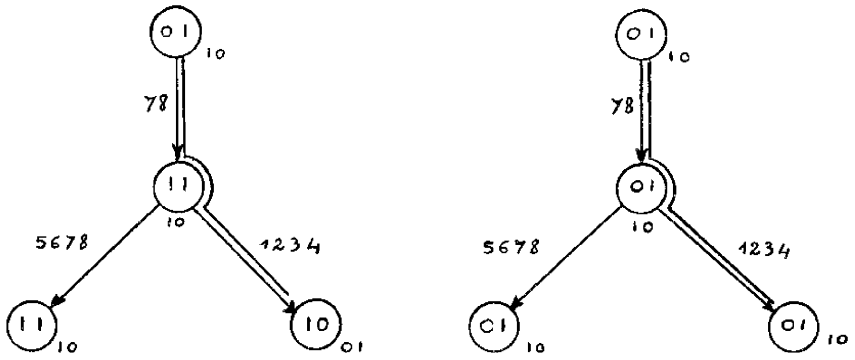
Ex : planche V.2.j : A partir de l'état (00) les vecteurs 1 2 3 4 devraient amener dans l'état (01) . Le fait que ces vecteurs restent dans l'état (0 0) est observable en un seul coup puisque les sorties diffèrent pour ces deux états.

Par contre lorsque les deux états atteints possèdent des sorties identiques, on ne peut prétendre avoir montré l'existence d'une TM.

Il faut, à partir de ces deux états, chercher à nouveau d'autres transitions rendant observable la TM. Cela nécessite la construction simultanée de deux arbres. En tête de chacun d'eux figurent respectivement les deux états atteints dans CB et CPI par la TM. Puis,en explorant chaque colonne des lignes correspondant à ces deux états, on détermine les états atteints à partir d'eux. Ceci forme une couche sur laquelle s'effectue la comparaison des vecteurs de sortie. Dans le cas où

les sorties sont encore identiques, une nouvelle couche se déduit à partir des états de la précédente. Nécessairement l'observabilité doit être réalisée à la (n - 1) ième couche (si le système a n états) si non il ne peut y avoir diagnostic de pannes et le module collé est logiquement inutile.

Ex : planche V.2.b :



La séquence 78 - 1234 assure l'observabilité de la TM 78

V.3.3.4 Formation des séquences complètes de test

Une fois l'algorithme précédent appliqué sur tous les états stables communs à CB et CPI, on connait les TM caractéristiques de la ième panne. Il faut alors chercher s'il existe des séquences capables d'amener à la fois CB et CPI dans les états de départ des TM et cela quel que soit l'état initial du système,qu'il fonctionne correctement (CB), ou qu'il comporte la ième panne (CPI). Si cela est possible, les deux dernières transitions de la séquence complète indiqueront si le système fonctionne. Pour mener à bien la démarche précédent, nous allons faire appel aux S.S.du circuit correct. Celles-ci conduisent le système dans des états particuliers connus et selon que les états de départ des TM sont égaux ou non à ces derniers, il suffira d'appliquer la S.S. correspondante, ou bien il faudra ajouter une séquence de rajustement (SR).

Mais ici se pose un problème de compatibilité à deux niveaux :

- Le circuit affecté de la ième panne admet-il au moins une séquence de synchronisation?
- Dans l'affirmative les S.S. de CB sont-elles compatibles avec celles de CPI?

Nous répondrons à ces deux questions et donnerons les solutions apportées en expliquant plus en détail les algorithmes.

V.3.3.4.1. Recherche des S.S. du circuit en panne

Il suffit de soumettre la matrice d'excitation de CPI à l'algorithme du § V.3.3.2.

Deux cas peuvent se produire : - le circuit en panne possède des S.S.
- il n'en possède pas.

THEOREME : Si un système séquentiel fortement connecté a des S.S. et que l'effet d'une panne soit tel qu'il n'en possède plus, c'est que son graphe de fluence a été divisé en plusieurs sous graphes indépendants ou est devenu complètement symétrique sur les états et sur les transitions.

En effet :

- dans le cas où le système reste fortement connecté malgré la panne on retrouve le théorème du § V.3.1.
- s'il n'est plus fortement connecté cela se traduit par l'apparition d'états sources ou d'états puits (ou les deux), ou de sous graphes fermés indépendants.

. Si seuls existent des états sources on retrouve le cas précédent car un tel circuit peut fort bien avoir des S.S.

. Si par contre apparaissent des états puits, deux cas restent à envisager.

- 1 seul état puits : on trouve nécessairement une S.S. conduisant dans cet état puisque le système se bloque sur lui.

- Plusieurs états puits : ce morcellement du graphe initial en plusieurs sous graphes non fermés implique que si l'état initial n'est pas connu il est impossible de savoir dans lequel de ces graphes le système va évoluer jusqu'à un état puits, donc aucune S.S. n'existe.

. Si le graphe initial se divise en sous graphes fermés, les conclusions ci-dessus s'appliquent.

V.3.3.4.2. Formation des séquences complètes dans le cas où CPI possède des S.S.

L'algorithme suivant s'applique à chaque TM de CPI. On distingue deux cas :

a - l'état à partir duquel la transition considérée est défectueuse, coïncide avec un état atteint par une S.S. de CB.

La séquence complète de test de cette transition sera constituée par la réunion de la S.S. et de la séquence de test de TM à condition que la S.S. se montre compatible sur CPI : pour celà on applique la S.S. de CB à CPI et elle doit l'amener sur le même état que celui atteint par CB. Dans le cas contraire, la réunion des deux séquences ci-dessus ne peut assurer le test de TM. S'il existe d'autres S.S. de CB aboutissant au même état, elles sont essayées à leur tour.

b - l'état à partir duquel la transition considérée est défectueuse ne correspond à aucun des états atteints par les S.S. de CB.

On va chercher quel est l'état le plus proche dans lequel conduit une S.S. Pour celà, on construit un arbre depuis l'état de départ de TM, en remontant les transitions. On obtient ainsi tous les états à partir desquels en une seule transition, le système se positionne dans l'état de départ de TM, directement ou par l'intermédiaire d'états instables. Si aucun de ces états ne constitue le point d'aboutissement d'une S.S., la construction de l'arbre reprend et la comparai-

son s'effectue sur les nouvelles couches formées. Cette recherche s'effectuant sur le circuit correct, on trouvera toujours dans l'arbre un état "fin de S.S.".

La séquence complète de test sera alors constituée par la réunion de :

- la S.S. amenant dans l'état précédemment déterminé.
- la S.R. effectuant la liaison entre cet état et l'état origine de la TM
- la séquence de test de la TM.

Il y aura autant de séquences complètes de test que de S.S. qui aboutissent à l'état (ou aux états) déterminé par l'arbre. Cependant une séquence de test ne sera formée qu'après avoir satisfait les conditions suivantes :

1) Il faut que la S.S. de CB soit compatible à CPI. Pour cela elle doit l'amener :

- dans le même état que celui où elle conduit CB
- ou bien dans l'état de départ de test de TM
- ou dans l'un de ces deux états indifféremment.

2) S'il en est ainsi on doit vérifier que la S.R. assure la transition entre les deux états ci-dessus, ou bien, appliquée depuis l'état de départ de TM, qu'elle laisse CPI dans cet état.

Si une de ces deux conditions ne peut se vérifier, il n'y aura pas de séquence de test possible pour cette transition défectueuse.

Lorsque toutes les TM de CPI ont été soumises à cet algorithme, l'alternative suivante se présente :

- aucune séquence de test n'a pu être déterminée : la ième panne ne sera pas diagnosticable par ce programme.

- il existe une ou plusieurs séquences et dans ce cas, toutes les solutions déterminées seront conservées. Le nombre maximum de solutions correspond à la somme des S.S. capables de tester chaque TM.

V.3.3.4.3. Formation de séquences de test dans le cas où CPI ne possède pas de S.S.

Si aucune S.S. n'apparait dans CPI, nous sommes dans l'impossibilité de positionner le système dans un état particulier. Nous allons essayer d'établir des séquences de test mais en général elles ne caractériseront pas la panne correspondante de façon unique, c'est-à-dire qu'en les appliquant aux entrées d'un système, celui-ci pourra donner une réponse en accord avec la panne pour laquelle elles ont été calculées, alors qu'une panne différente affecte le circuit.

On regroupe tous les états à partir desquels une TM apparait. On recherche ensuite sur l'arbre de formation des S.S. du circuit en panne (arbre qui n'a pas indiqué de S.S.) s'il existe au moins une séquence amenant dans un groupe d'états identique au précédent. S'il n'en existe pas, le test de cette panne est abandonné, si non il y aura autant de séquences de test que de transitions défectueuses, chacune résultant de l'assemblage de :

- la séquence amenant dans le groupe d'états.
- et de la séquence de test de TM.

V.3.3.5 Regroupement des pannes indiscernables - Séquences de Diagnostic

Les algorithmes précédents fournissent, pour chaque panne, une ou plusieurs séquences qui les caractérisent.

Une séquence complète de test est en réalité composée de deux séquences : - entrées
- sorties.

Toutes les séquences de sortie donneront la valeur de la combinaison binaire des sorties pour les deux dernières transitions de la séquence d'entrée. La raison en est la suivante : lorsqu'on applique une séquence aux entrées

du système, comme l'état initial dans lequel il se trouve demeure inconnu, la valeur des sorties diffère selon cet état. L'observabilité de la panne s'effectue sur la dernière transition. Il suffit donc de connaître la valeur que doivent prendre les sorties juste avant cette transition, et juste après.

- si on observe une valeur correcte des sorties avant et après, le circuit n'est pas affecté par la panne i (il peut l'être par contre par une panne J)

- si la valeur des sorties est bonne avant, mais pas après la dernière transition, la panne i est présente.

- si la valeur des sorties est anormale avant la transition, une autre panne existe qui sera déterminée par une autre séquence.

Lorsque plusieurs pannes possèdent les mêmes séquences d'entrée-sortie, on les regroupe en ensembles de pannes indiscernables. Ainsi, l'application d'un groupe de séquences au circuit montrant qu'il fonctionne mal, indique que la panne cause de ce mauvais fonctionnement est l'une de celles constituant l'ensemble de pannes indiscernables correspondant aux séquences esseyées :

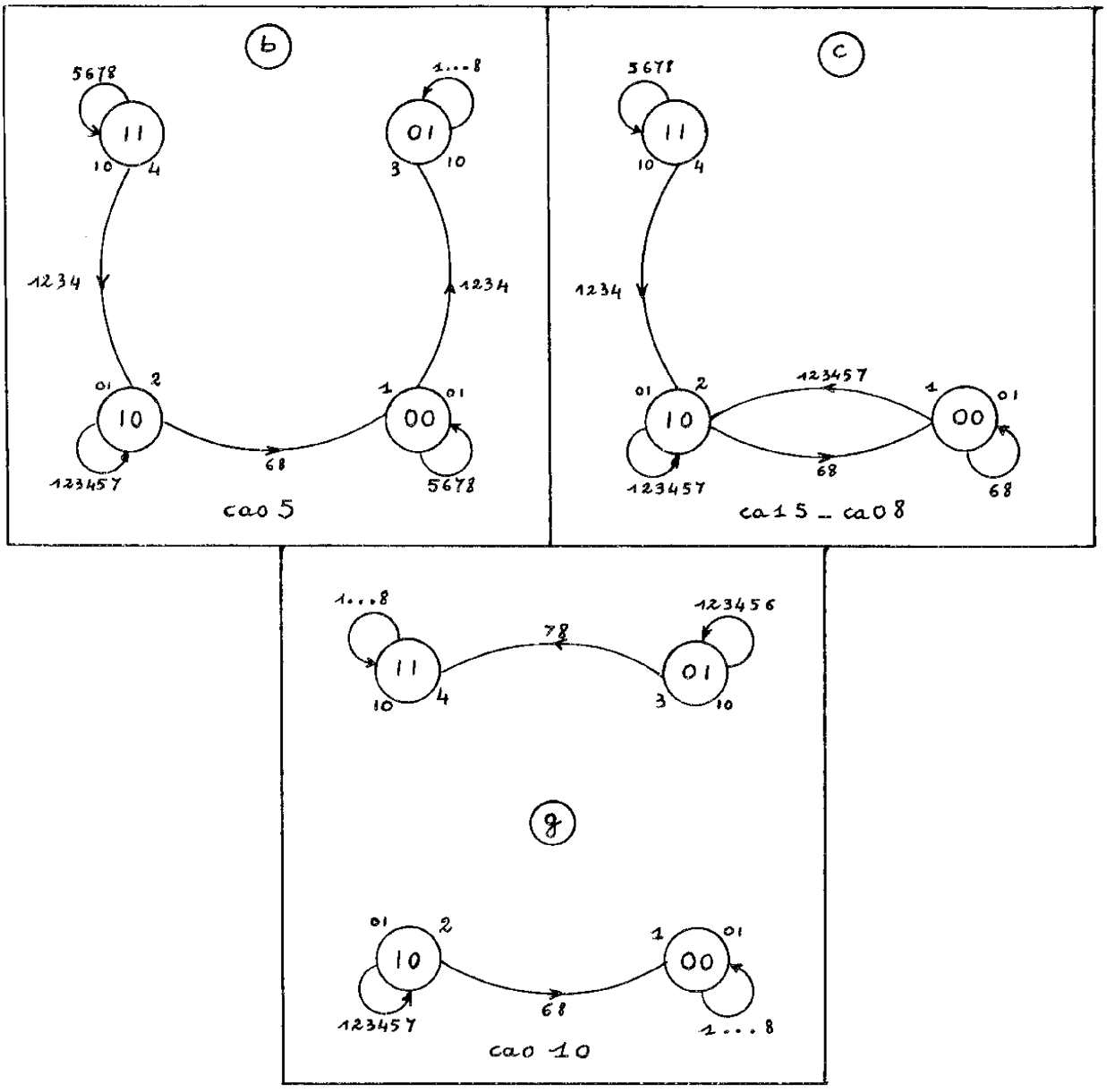
REMARQUE :

Au cours de l'établissement des séquences de test, nous n'avons pas traité le problème de l'adjacence entre les transitions, d'abord pour ne pas accroître la complexité des algorithmes et ensuite parce que les transitions regroupent la plupart plusieurs vecteurs le choix de ces vecteurs en vue de satisfaire l'adjacence pouvant se faire facilement au dépouillement des résultats.

Il est bien sûr possible de l'ajouter en adaptant l'algorithme du § IV.5.2.3.4.

V.3.4. Exemple

Afin de clarifier et d'illustrer la méthode exposée au § V.3.3., nous allons reprendre l'exemple de la bascule JK 7473 (pl. V.2 et V.2.bis) et examiner comment s'établissent les séquences de test sur quelques uns des graphes transformés.



Bascule JK 7473 (extrait de la Planche V 2)

Les S.S. du circuit correct déterminées au § V.3.3.2 (fig. V.2) sont :

1 2 3 4	6	1 2 3 4	amenant dans l'état	(3)
1 2 3 4	7	1 2 3 4	amenant dans l'état	(2)

(1) cac 5 (pl. V.2.5.)

- TM : . depuis l'état 1, pas de TM

. depuis l'état 2, pas de TM

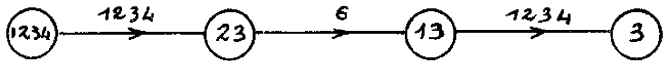
. depuis l'état 3, la TM est 78 qui devrait amener en 4 alors qu'elle bloque le système en 3. Mais les sorties ont une valeur identique dans ces deux états. Cette TM n'est pas observable en un coup. La formation de l'arbre d'observabilité donne la transition 1 2 3 4 à ajouter. Ainsi le test complet de la TM s'effectue par 78-1 2 3 4 (fig. V.3.a)

. depuis l'état 4, pas de TM.

- formation des séquences de test : le circuit en panne admet une S.S. aboutissant dans l'état 3 : 1 2 3 4-68 — 1 2 3 4. Comme l'état 3 est un état de fin de S.S. de CB et que la séquence correspondante 1 2 3 4-6-1 2 3 4 amène également le circuit affecté de la panne collage à 0 du module 5 dans l'état 3 (compatibilité, fig. V.3.b), la séquence complète de test s'écrit :

entrées	1 2 3 4	6	1 2 3 4	78	1 2 3 4	
sorties :				1	2	

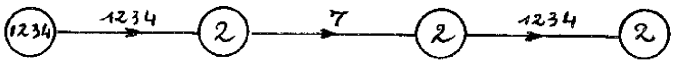
a- Test de la transition défectueuse (observabilité)



b- SS du circuit correct appliquée au circuit avec ca0 5

figure V 3

a- Transition défectueuse



b- SS du circuit correct appliquée au ca1 5, ca0 8

figure V 4

②- ca 1.5 - ca 0.6 (pl. V.2.c)

- TM : - état 1 : 1 2 3 4 est une TM observable en un coup (fig. V.4.a)
5 7 est une TM observable en ajoutant 1 2 3 4 parce que 57 faisant appel à 1234 il n'apporte aucun supplément d'information au sujet de la panne. Il ne constituera pas une transition à tester.

- état 2 : pas de TM
- état 3 : n'existe plus en tant qu'état stable
- état 4 : pas de TM.

- Formation des séquences de test.

1 n'est pas un état de fin de S.S.. En remontant depuis 1, on trouve par la transition 68, l'état 2, sur lequel amène la séquence 1234-7-1234 dans CB.

- Compatibilité : CB possède 3 S.S. amenant dans l'état 2.

1 2 3 4 , 5 7 - 1 2 3 4 , 6 8 - 1 2 3 4

La séquence 1234-7-1234 amène aussi dans l'état 2 (comme assemblage de 1234 et 57 - 1234), elle est donc compatible au circuit en panne (fig. V.4.b)

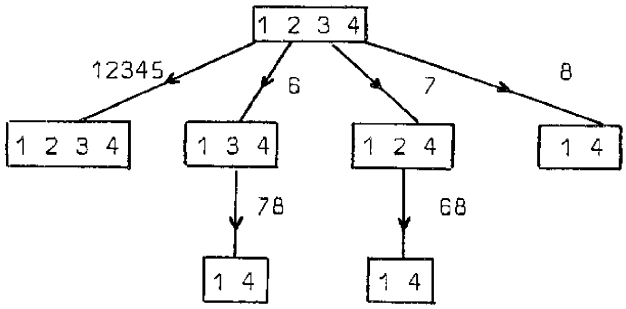
La séquence complète sera :

entrées :	1 2 3 4 7 1 2 3 4	6 8	1 2 3 4
	S.S.	S.R.	TM
sorties :		2	1

③- ca 0.10 (pl. V.2.g)

On a deux TM : à partir de l'état 1 : 1 2 3 4
à partir de l'état 4 : 1 2 3 4

L'ordre de formation des S.S. ci-dessous indique qu'aucune S.S. n'existe :



Le regroupement 14 des états de départ des TM se retrouve sur cet arbre. On formera donc les deux séquences de test suivantes :

Entrées : 8 1 2 3 4
Sorties : 1 2

Entrées : 8 1 2 3 4
Sorties : 2 1

qui ne sont pas caractéristiques de cette panne. En effet, si le système se trouve au départ dans l'état 3 affecté de la panne ca 0.5, l'application de la première donnera en sortie 1 1 résultat que l'on retrouve si le système affecté de la panne ca 0.10 est dans l'état initial 3 ou 4. On ne peut donc conclure et cette panne appartiendra à deux ensembles différents. Notons au passage que l'emploi de l'entrée de remise à zéro aurait permis de classer cette panne dans un seul ensemble. On voit en effet sur la pl. V.3. (h) que la remise à zéro a refermé le graphe.

V - 4 DEPOUILLEMENT DES RESULTATS

Nous allons indiquer ici la façon dont se présentent les résultats et le moyen de les interprêter. L'exemple choisi est la bascule D SN 7474 (TEXAS INSTRUMENTS) (fig. V.5.)

L'étude des séquences de diagnostic se fera en deux temps :

- sans tenir compte de la remise à zéro (R = 1)
- en considérant cette entrée au même titre que les autres.

V.4.1. 1er cas (R = 1)

La simulation systématique des pannes donne les graphes représentés sur la planche V.11. Le circuit correct possède les S.S. suivantes :

$$\begin{cases} 1\ 2\ -\ 3 \longrightarrow (2) \\ 1\ 2\ -\ 4 \longrightarrow (7) \end{cases}$$

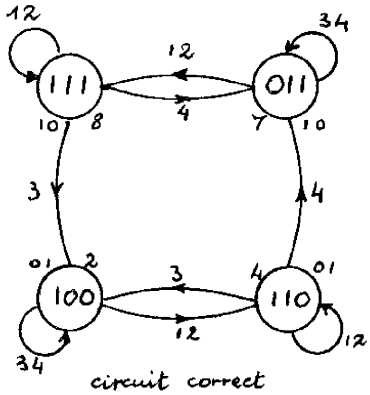
Les séquences de diagnostic et les groupements de pannes indiscernables sont résumés par le tableau V.2.. Les pannes strictement identiques n'y figurent pas car elles sont seulement indiquées au moment de la comparaison.

Il faut, sur ces résultats, vérifier que la séquence de diagnostic, correspondant à un groupement, donne lorsqu'elle est appliquée à tous les autres groupements des valeurs sur les sorties égales à celles du circuit correct, ou différentes de celles prévues lors de l'avant dernière transition (voir § V.3.3.5). Dans le cas contraire, les regroupements effectués ne seraient pas distincts les uns des autres.

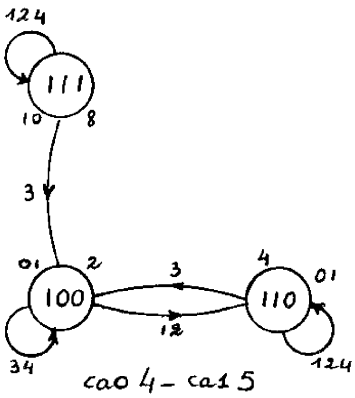
On obtient ici un partitionnement des pannes en trois ensembles distincts, d'où un diagnostic assez peu précis.

C.4.2. 2ème cas

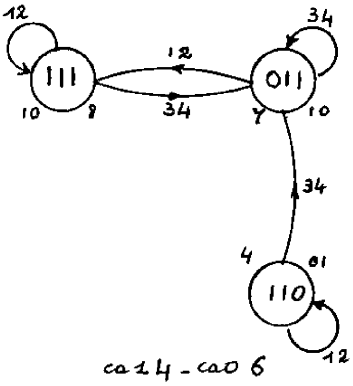
En incluant l'entrée de forçage les graphes deviennent ceux de la planche V.12



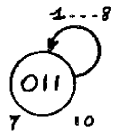
circuit correct



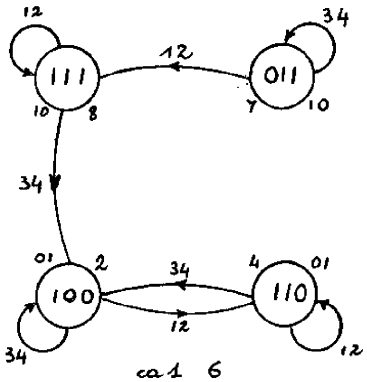
cao 4 - ca1 5



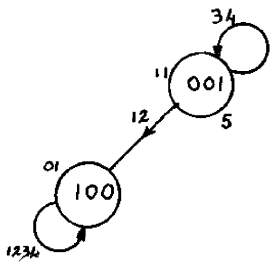
ca14 - cao 6



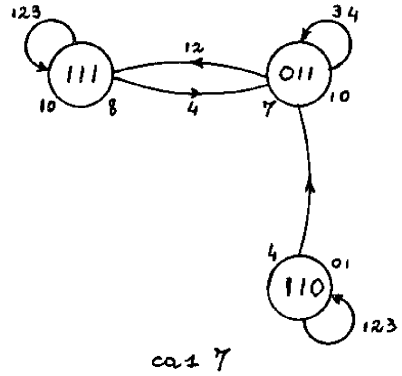
cao 5



ca1 6

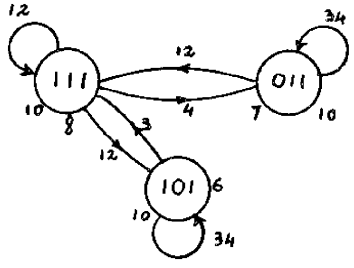


cao 7

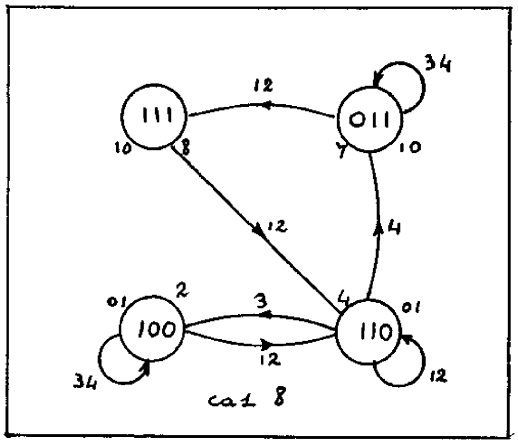


ca1 7

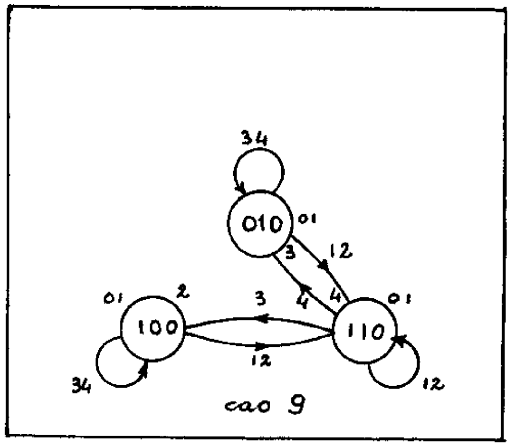
PL. V 11



cao 8 - ca1 9



cas 8



cas 9

codage des entrées

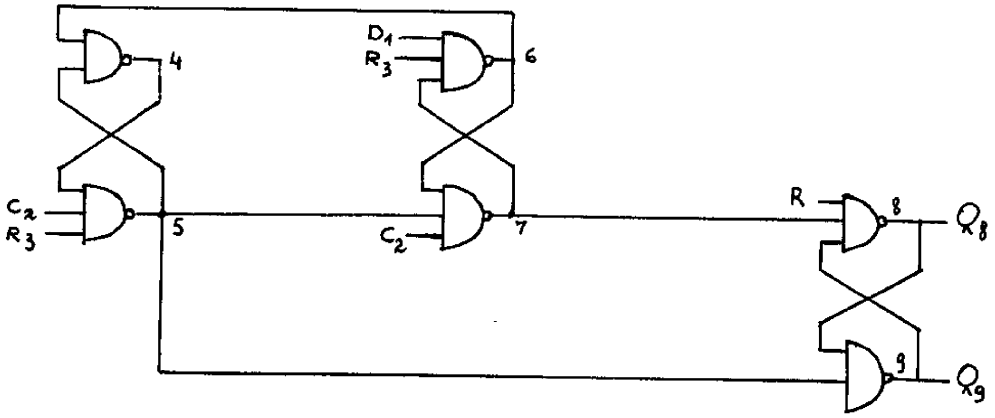
	D	C
1	0	0
2	1	0
3	0	1
4	1	1

codage des sorties

Q_8	Q_9
0	0
1	1
2	0
3	1

D 7474 sans forçage

Planche V 11 (suite)



Bascule D 7474

figure V 5

1 ----- LES SEQUENCES

ENTREE	12	3	12	4
SORTIE	2	1		

TESTENT LES PANNES SUIVANTES

COLLAGE A 0 DU MODULE 4
COLLAGE A 1 DU MODULE 6
COLLAGE A 0 DU MODULE 7
COLLAGE A 0 DU MODULE 9

QUI SONT INDISCERNABLES

2 ----- LES SEQUENCES

ENTREE	12	4	12	3
SORTIE	1	2		

TESTENT LES PANNES SUIVANTES

COLLAGE A 1 DU MODULE 4
COLLAGE A 0 DU MODULE 5
COLLAGE A 0 DU MODULE 6
COLLAGE A 1 DU MODULE 7
COLLAGE A 0 DU MODULE 8

QUI SONT INDISCERNABLES

3 ----- LES SEQUENCES

ENTREE	12	4	12
SORTIE	1	1	

TESTENT LES PANNES SUIVANTES

COLLAGE A 1 DU MODULE 8

- Tableau V.2 -

. S.S. du circuit correct

12
34 - 12 ⟶ (4)
78 - 12

34 - 567
56 ⟶ (2)
78 - 56

34 - 8 ⟶ (7)

L'entrée de forçage à elle seule donne des S.S. de longueur 1 :

1 2 qui amène en 4
5 6 qui amène en 2

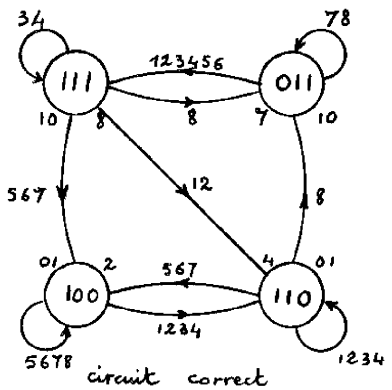
Les différents groupements sont représenté sur le tableau V.3.

Le dépouillement demande ici un peu plus d'attention car le programme fournit des ensembles de pannes indiscernables,en groupant les colleges qui possèdent les mêmes séquences de diagnostic, mais il n'applique pas les séquences caractéristiques d'un ensemble, à un autre (ceci pourrait être ajouté) de telle sorte qu'il peut exister des ensembles qui se distinguent par des séquences légèrement différentes,donnant en réalité des résultats identiques. C'est le cas par exemple des ensembles 2 et 3, 1 et 5, 3 et 7. Par contre l'ensemble 6 se distingue des ensembles 2 et 4 grâce à sa première séquence, et en choisissant le vecteur 7 : 12– 7– 8.

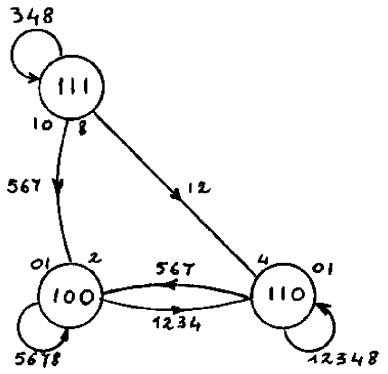
On obtient donc cinq groupements distincts

1 et 5
2 et 4
3 et 7
6
8

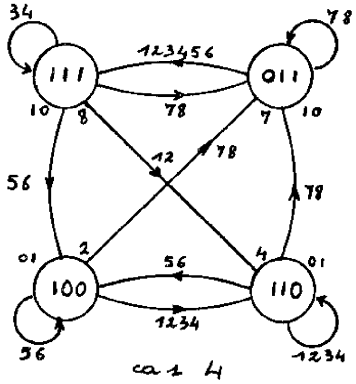
Ainsi, grâce à l'entrée de forçage, on a pu extraire les pannes { ca 0.5 d'une part, ca 0.8 et ca 1.7 d'autre part du deuxième ensemble de pannes indiscernables donné par



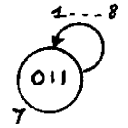
circuit correct



cao 4 - ca1 5



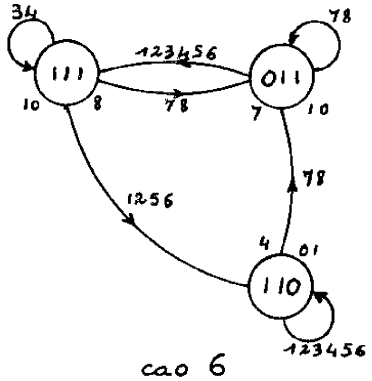
ca1 4



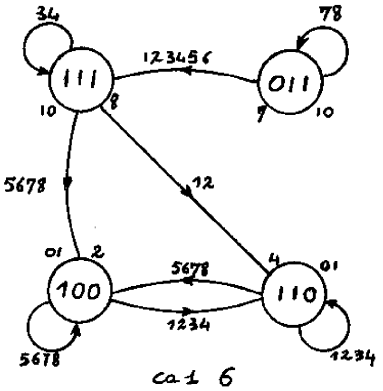
S : 11 pour 1256

S : 10 pour 3478

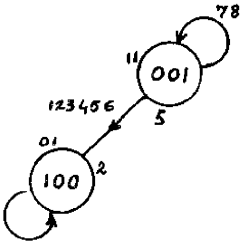
cao 5



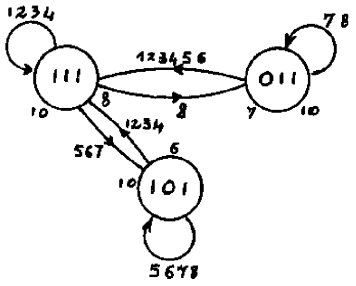
cao 6



ca1 6

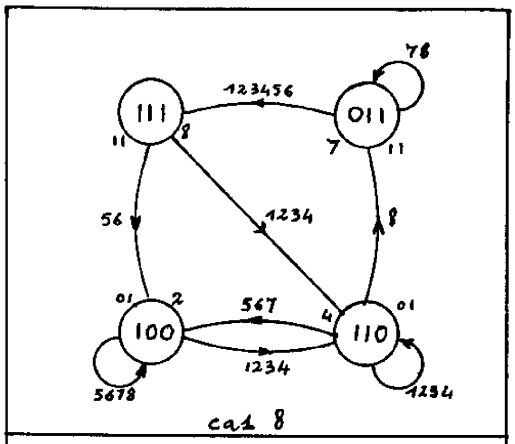


cao 7

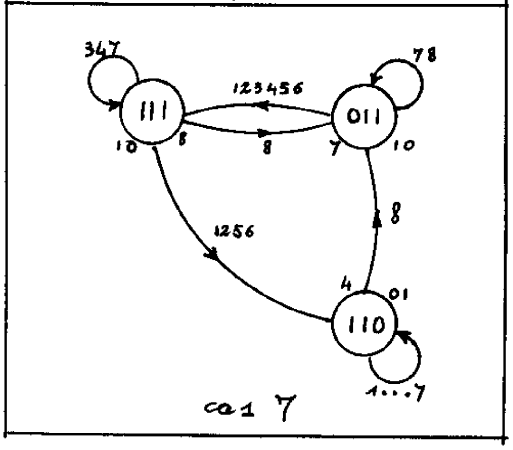


cao 8 - ca1 9

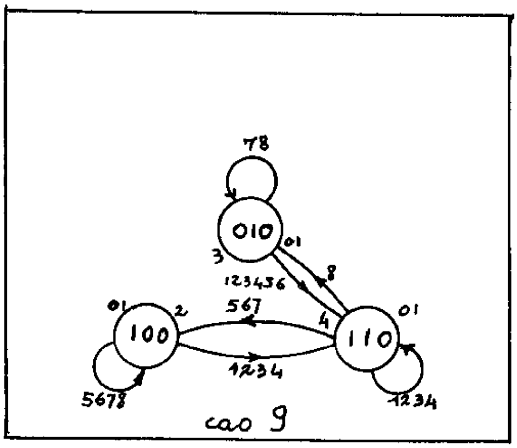
PL. V 12



cas 8



cas 7



cas 9

codage des entrées

	D	R	C
1	0	0	0
2	1	0	0
3	0	1	0
4	1	1	0
5	0	0	1
6	1	0	1
7	0	1	1
8	1	1	1

codage des sorties

	Q_8	Q_9
0	0	0
1	1	0
2	0	1
3	1	1

Planche V 12 (suite)

1 ----- LES SEQUENCES

ENTREE	12	8		
SORTIE	2	1		
ENTREE	34	12	8	
SORTIE	2	1		
ENTREE	78	12	8	
SORTIE	2	1		

TESTENT LES PANNES SUIVANTES

COLLAGE A 0 DU MODULE 4

COLLAGE A 1 DU MODULE 6

COLLAGE A 0 DU MODULE 9

QUI SONT INDISCERNABLES

2 ----- LES SEQUENCES

ENTREE	56	78			
SORTIE	2	2			
ENTREE	78	56	78		
SORTIE	2	2			
ENTREE	12	7			
SORTIE	2	2			
ENTREE	34	12	7		
SORTIE	2	2			
ENTREE	78	12	7		
SORTIE	2	2			
ENTREE	34	8	34	7	
SORTIE	1	2			

TESTENT LES PANNES SUIVANTES

COLLAGE A 1 DU MODULE 4

3 ----- LES SEQUENCES

ENTREE	34	8	1256
SORTIE	1	2	

TESTENT LES PANNES SUIVANTES

COLLAGE A 0 DU MODULE 5

4 ----- LES SEQUENCES

ENTREE	12	56	78	
SORTIE	2	2		
ENTREE	34	12	56	78
SORTIE	2	2		
ENTREE	78	12	56	78
SORTIE	2	2		
ENTREE	12	7		
SORTIE	2	2		
ENTREE	34	12	7	
SORTIE	2	2		
ENTREE	78	12	7	
SORTIE	2	2		
ENTREE	34	8	34	7
SORTIE	1	2		

TESTENT LES PANNES SUIVANTES

COLLAGE A 0 DU MODULE 6

5 ----- LES SEQUENCES

ENTREE	34	567	1234	8
SORTIE	2	1		
ENTREE	56	1234	8	
SORTIE	2	1		
ENTREE	78	56	1234	8
SORTIE	2	1		

TESTENT LES PANNES SUIVANTES

COLLAGE A 0 DU MODULE 7

6 ----- LES SEQUENCES

ENTREE	12	567	8		
SORTIE	2	2			
ENTREE	34	12	567	8	
SORTIE	2	2			
ENTREE	78	12	567	8	
SORTIE	2	2			
ENTREE	34	8	56	8	
SORTIE	2	2			
ENTREE	34	8	34	56	8
SORTIE	2	2			
ENTREE	34	8	34	7	
SORTIE	1	2			

TESTENT LES PANNES SUIVANTES

COLLAGE A 1 DU MODULE 7

7 ----- LES SEQUENCES

ENTREE	34	8	123456
SORTIE	1	2	

TESTENT LES PANNES SUIVANTES

COLLAGE A 0 DU MODULE 8

8 ----- LES SEQUENCES

ENTREE	34	8	34
SORTIE	1	1	

TESTENT LES PANNES SUIVANTES

COLLAGE A 1 DU MODULE 8

- Tableau V.3 -

le tableau V.2. Le diagnostic du système est donc plus fin.

Remarquons pour finir que malgré l'entrée de remise à zéro, nous n'avons pas supposé connu un état initial.

Donc, si un circuit possède des entrées de forçage, nous les feront toujours intervenir en les considérant comme des entrées normales. Elles permettront d'obtenir des séquences de synchronisation plus courtes et en règle générale, un diagnostic plus évolué du système.

V-5 UTILISATION DES PROGRAMMES - POSSIBILITES

V.5.1 Programme de simulation de pannes

Il s'utilise de la même façon que le programme d'analyse (chap. III) puisqu'il emploie les mêmes algorithmes moyennant quelques adaptations. Le nombre d'instructions FORTRAN est porté à 2500.

Il existe en deux versions :

- La première dont les résultats ne sont pas destinés à être exploités par le programme de diagnostic, n'effectue pas la mise en mémoire des matrices. La limite sur le nombre d'entrées reste égale à 9 ou 10, celle sur le nombre de variables secondaires et de sorties dépend toujours de la capacité du calculateur employé.

- La seconde met en mémoire les matrices des Excitations et des sorties de tous les collages effectués et les transcrit sur cartes de résultat. La limite actuelle sur le nombre d'entrées, de variables secondaires, et de sorties est de trois. Nous en avons indiqué la raison au § V.2.1.. Le temps de calcul dépend de la complexité du circuit et surtout du nombre de modules. Pour les circuits que nous étudions, il est de l'ordre de 5' (IBM 7044)

V.5.2 Programme d'élaboration des séquences de diagnostic

Il comporte environ 1300 instructions FORTRAN.

Les principales données qu'il utilise sont fournies directement par le programme précédent.

La limitation sur le nombre d'entrées, de variables secondaires et de sorties est fixée également à trois, et le temps de calcul en ce qui concerne les mêmes exemples se situe aux environs de la minute.

Améliorations :

Pour un codage différent des matrices et des vecteurs calculés la limite donnée ci-dessus pourrait s'accroître dans des proportions appréciables.

En ne comparant pas chacune des matrices aux autres, ce qui nécessite leur présence simultanée en mémoire, il serait possible de stocker des matrices beaucoup plus importantes, au détriment bien sûr du temps calcul puisque les pannes strictement identiques ne seraient plus détectées.

- C O N C L U S I O N -

En l'absence de toute méthode théorique générale d'Analyse au sens large de systèmes séquentiels, et compte tenu de la grande complexité du problème à tous les niveaux d'étude, il nous a paru indispensable de faire une prospection aussi complète que possible sur l'ensemble des problèmes qui peuvent se poser dans ce domaine. Par ailleurs, le but final d'une étude de détection et de diagnostic étant toujours l'Analyse de circuits séquentiels réels, nous avons appliqué systématiquement les méthodes que nous proposions à des circuits commerciaux de taille moyenne.

De ce fait, sans avoir à nous préoccuper outre mesure du problème purement informatique (optimisation de programmes : temps de calcul, place mémoire ...), nous avons pu élaborer un ensemble de programmes qui constituent un moyen à peu près complet d'étude des systèmes séquentiels et donc un support indispensable à toute investigation théorique ultérieure.

La première partie de cette étude a permis de mettre au point un programme d'Analyse logique des systèmes séquentiels qui, s'il demande certaines améliorations, constitue un outil désormais nécessaire et riche en enseignements. De par sa conception, il est aisément transformable et adaptable à tout problème d'analyse logique (par ex : détermination d'une séquence de sortie pour une séquence d'entrée donnée).

La deuxième partie ayant trait au test fonctionnel, a fait l'objet d'une méthode dont l'esprit est sensiblement différent de celles jusqu'à présent proposées et appliquées. Elle présente l'intérêt de ne nécessiter que peu d'hypothèses, la nature des pannes n'étant pas du tout limitée. La restriction qui se traduit par la suppression du terme $C\,\overline{y}_{(T)}$ dans l'équation $y_{(T+1)} = A\,y_{(T)} + B + C\,\overline{y}_{(T)}$ peut si on le désire être enlevée en donnant des critères d'assignements pour ce terme. Cependant cette méthode nécessiterait une justification théorique plus rigoureuse. Nous nous sommes limités à son établissement, sa systématisation et son application par l'écriture d'un programme. La vérifica-

tion des résultats et leur concordance avec ceux donnés par les spécifications (3) constitue seulement une justification à postériori. Cette lacune n'a pas été comblée par le fait que ceci aurait dépassé le cadre de notre étude.

L'établissement des séquences de détection, grâce aux notions de vecteur transfert et d'observabilité,a donné lieu à une méthode systématique, d'autant plus intéressante qu'elle génère des séquences de longueur fort acceptable tenant compte de l'adjacence.

Ces trois programmes constituent un ensemble cohérent et complet, apportant une solution au problème du test fonctionnel,par la seule description du circuit logique (matrice topologique).

Le programme d'Analyse et de simulation de pannes,nous a fait prendre conscience de l'énormité du problème qu'est le diagnostic des systèmes séquentiels. Les résultats obtenus et les nombreuses remarques qu'on a pu déduire, montrent le grand intérêt de la simulation par ordinateur.

La méthode de diagnostic que nous avons développée,a nécessité l'emploi des matrices des Excitations et des sorties,limitant par l'occupation en place mémoire,la taille des circuits testables. L'emploi des séquences de synchronisation lève toutes les hypothèses que l'on rencontrait jusqu'à présent au sujet des entrées de forçage et permet d'envisager le problème de façon plus réaliste. Cette méthode permet d'envisager toute transformation d'un graphe (état instable devenant stable, états supprimés, blocages, cycles, états stables devenant instables) et localise toute panne de la partie séquentielle d'un système fortement lié, à condition que cela soit possible.

Les algorithmes et les méthodes décrits utilisent la décomposition d'un circuit au niveau du module élémentaire. Il est bien évident que si l'on veut aborder le test de gros ensembles logiques, cette décomposition devient impossible. On sera alors obligé de décrire un système par blocs et le problème qui se pose est le suivant : après avoir mis au point une méthode effectuant la détection de pannes, localiser ensuite le bloc défaillant. Si celui-ci est accessible et de taille assez réduite,on pourra le soumettre aux méthodes ci-dessus (dans la mesure où celà présente un intérêt), si non le problème reste entier et on risque fort de ne

pas pouvoir aller plus loin. De plus, il faudra élaborer des méthodes absolument générales,auxquelles on devra accorder une confiance illimitée,car la vérification des résultats sur de gros systèmes,peut rapidement devenir inextricable.

Le but précis du L.A.M.* et du L.A.A.S.,est justement d'élaborer des méthodes théoriques générales,telles que leurs résultats soient à même de rejaillir sur la synthèse des systèmes. Il semble en effet normal, étant donnée la complexité toujours croissante des circuits séquentiels, que leur synthèse soit établie de façon à faciliter ou du moins à rendre possible leur test non exhaustif.

* L.A.M. : LABORATOIRE D'AUTOMATIQUE de la Faculté des Sciences de Montpellier, dirigé par Monsieur DURANTE, Maître de Conférences.

- A N N E X E I -

TRANSFORMATION DU PROGRAMME D'ANALYSE

A-1.I. Recherche des variables secondaires

La méthode (3) exposée au paragraphe III.2.2.4. a été programmée, et donne de très bons résultats du point de vue de la rapidité, critère essentiel à respecter. Comme le minimum de variables secondaires n'est pas atteint avec une totale certitude, l'algorithme de choix s'est trouvé simplifié : si le choix donné par l'utilisateur recouvre le tableau des boucles fondamentales, il est pris en considération, même si le nombre de variables secondaires est loin d'être minimal. Dans le cas contraire, ce choix est totalement abandonné et le premier ensemble déterminé le remplace.

Par le gain important en place mémoire, on augmente la taille des circuits analysables. Mais il apparait alors une autre limitation : la recherche des boucles et boucles fondamentales, nécessite la mise en mémoire de toutes les boucles. Leur nombre peut devenir très important et cette partie du programme introduit à son tour une restriction sur la taille des circuits, en satisfaisant mal les critères rapidité, économie.

On est donc conduit à chercher les variables secondaires sans utiliser les boucles fondamentales. On peut penser ainsi gagner sur le temps et la place mémoire.

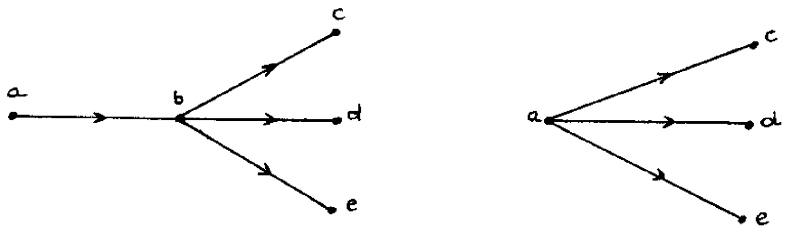
L'algorithme que nous allons brièvement décrire vise à réaliser ces deux objectifs.

Connaissant la matrice topologique du circuit, on commence à supprimer la colonne de codage (deuxième colonne), et tout ce qui ne contribue pas au caractère séquentiel du circuit (entrées primaires, modules appartenant à un

réseau combinatoire en entrée ou en sortie). On obtient ainsi une matrice représentant le graphe de la partie bouclée du système. Celui-ci est composé d'arêtes qui correspondent aux connexions entre modules, ceux-ci étant figurés par des noeuds.

On définit les degrés incidents et sortants d'un noeud comme le nombre d'arêtes incidentes et sortantes en ce noeud.

En un premier temps, le graphe est simplifié : tous les noeuds de degré incident 1 peuvent être supprimés car l'ouverture de l'arête incidente bloque la propagation de l'information sur les arêtes sortantes, donc sur les noeuds auxquels elles aboutissent (fig. A.1.I)



- Fig. A.1.I -

On calcule ensuite les degrés incidents et sortants de tous les noeuds. On recherche dans la liste obtenue parmi les noeuds à sortance la plus élevée,un de ceux dont l'entrance est maximale. La sortie du module correspondant constitue une des variables secondaires. Le graphe se trouve modifié par l'ouverture des branches issues du noeud choisi, et simplifié à nouveau en reconsidérant les noeuds de degré incident 0 et 1.

Le processus reprend avec le calcul des degrés de noeuds et le choix d'une nouvelle variable secondaire.

Lorsque le graphe se réduit à un seul point (comportant toujours au moins une boucle propre), celui-ci donne la dernière variable secondaire de l'ensemble.

La figure A-1.2 représente les différentes transformations d'un graphe qu'implique cet algorithme.

Cette méthode qui par rapport à celles exposées au paragraphe III.2.2.4., donne des temps de calcul et un encombrement mémoire bien moins importants, constitue une première amélioration intéressante du programme.

A-1.II. Codage et calcul des expressions logiques

Les améliorations proposées au paragraphe III.3.4., c'est-à-dire la non limitation du nombre d'entrées et le rec a lage des variables secondaires, ont été apportées. Elles permettent un gain intéressant en place mémoire et en temps calcul.

A-1.III. Réponse d'un système à une séquence d'entrée

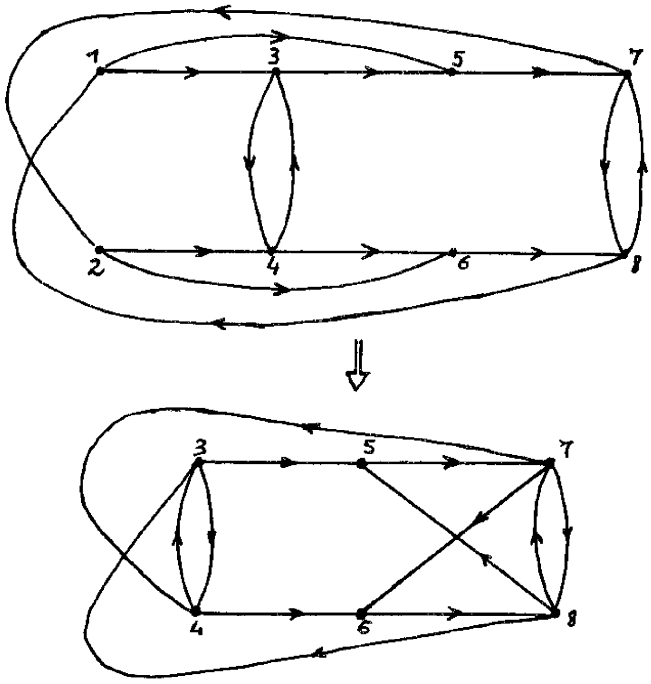
Ceci a été ajouté pour donner au programme une plus grande souplesse d'emploi.

Par la première combinaison des entrées, le calcul de l'état initial doit être possible. Cela impose que le système possède une entrée de forçage, ou, dans la colonne de la matrice des excitations correspondant à cette première combinaison, un seul état stable.

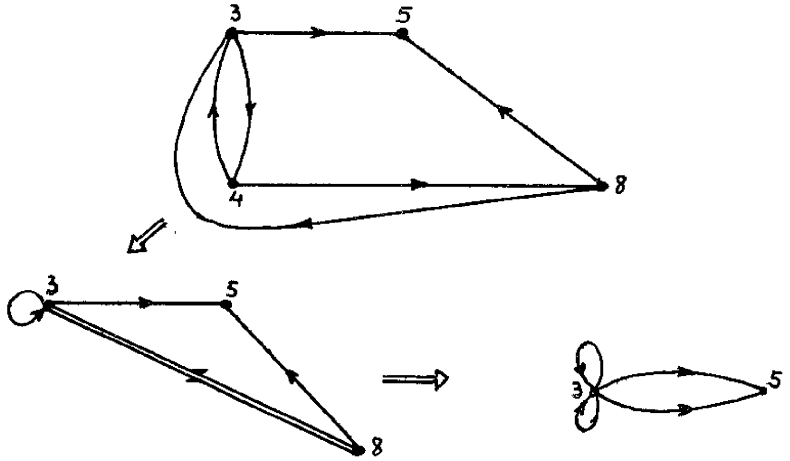
La détermination de l'état initial s'effectue de la façon suivante :

1. On regarde l'effet des variables d'entrée sur les équations des variables secondaires. Celles-ci se trouvent transformées (orientées à 0, 1, simplifiées) ou inchangées si elles ne sont pas fonction des entrées.

2. On soumet les variables secondaires non orientées à 0 ou 1, aux valeurs de celles qui l'ont été en 1).



- Ouverture de 7:



- Ouverture de 3

•3 Ensemble : 3, 7

figure A1 . 2

Le procédé se reboucle sur lui-même jusqu'à ce que les valeurs de toutes les variables secondaires soient déterminées, ce qui indique l'état initial.

Dès que ce dernier est connu, les états suivants se calculent en soumettant les équations des variables secondaires, aux valeurs présentes des entrées et des variables secondaires. On en déduit de la même façon les valeurs des sorties.

Il devient ainsi possible, après le calcul des équations logiques du système, de choisir entre l'observation de sa réponse à une séquence d'entrée, ou le tracé des tables.

La première option doit être adoptée lorsque les systèmes analysés deviennent très complexes. Elle constitue une aide intéressante à la conception, en ce sens qu'elle permet de savoir si un circuit fonctionne conformément à ce qu'on espérait, pour une séquence donnée.

La deuxième option, que nous avons déjà détaillée (paragraphe III.2.2.9), donne beaucoup plus de renseignements, que l'on paye bien sûr en temps calcul. Elle est intéressante pour les petits circuits (jusqu'à huit variables secondaires), et lorsqu'on désire connaître très précisément le fonctionnement d'un système, dans tous les cas possibles de configurations d'entrées.

A l'aide de ces modifications, nous avons pu analyser sans difficulté des circuits comprenant une centaine de modules et une vingtaine de variables secondaires (par exemple : Registre à décalage 8 bits) sur un ordinateur de taille moyenne : IBM 7044.

- A N N E X E II -

ETUDE DU PIRE CAS SUR LA BASCULE 74H72 (fig. III.5)

Les équations des variables secondaires 9 et 13 sont :

$$\begin{cases} Y_{9(T+1)} = \underbrace{(K' + C' + Y'_{13(T)})}_{A_1} \cdot Y_{9(T)} + \underbrace{J\,K'\,C + J\,C\,Y'_{13(T)}}_{B_1} \\ Y_{13(T+1)} = \underbrace{(Y_{9(T)} + K\,C)}_{A_2} \cdot Y_{13(T)} + \underbrace{J' \cdot Y_{9(T)} + C' \cdot Y_{9(T)}}_{B_2} \end{cases}$$

ISOLEMENT (transitions bouclées)

Etat ① ($Y_{9(T)} = 0$, $Y_{13(T)} = 0$)

Les conditions de pire cas sont : $A_1 = 1+$, $B_1 = 0-$, $A_2 = 1+$, $B_2 = 0-$

$A_1 = 1+$ d'où $K = 0$ ou $C = 0$ soit 3 0 0

$B_1 = 0-$ donne $J = 0$ ou $K = 1$ ou $C = 0$ 0 1 0

et $J = 0$ ou $C = 0$ 0 3 0

Ces conditions étant obligatoires on a : $\begin{bmatrix} 2 & 0\,1\,0 \\ 2 & 0\,3\,0 \end{bmatrix}$

$A_2 = 1+$ $K = 1$ et $C = 1$ 3 1 3

3 3 1

B_2 = 0- comme $Y_{9(T)}$ = 0, le pire cas est obtenu par J = 0 ou C = 0

soit 0 3 0, sans priorité,puisque $Y_{9(T)}$ = 0 réalise déjà la condition B = 0.

On obtient donc la liste suivante où les termes sont classés par ordre décroissant de poids :

2	0 1 0	la première colonne représente J
2	0 3 0	la deuxième colonne représente K
	3 1 3	la troisième colonne représente C
	3 3 1	
	3 0 0	
	0 3 0	

Dans la première colonne,la variable du premier terme est comparée aux suivantes. Elle rencontre une autre variable dans un terme de priorité 2. (C_{22}). Comme elles sont compatibles, la comparaison se poursuit dans la colonne. Le dernier terme sans priorité est également compatible (C_{20}).

A ce moment, la valeur J = 0 est admise et dans le premier terme 2 0 1 0, on va effectuer la transformation suivante :

Il correspond à J K' C dans B_1. Comme J = 0 assure la valeur 0 de ce monôme, on va inverser les valeurs données primitivement à K et C afin de rendre le 0 plus fragile, soit K = 0 et C = 1. Mais il faut alors affecter à J = 0 une priorité maximale; le terme devient donc :

2 0 1 0 ⟶	3	0 3 3
		3 0 3
		3 3 1

K = 0 et C = 1 sont souhaitables,en vue d'améliorer le pire cas mais ne peuvent être affectés d'un degré de priorité .

La liste devient :

3 0 3 3
2 0 3 0
3 0 3
3 3 1
3 1 3
3 3 1
3 0 0
0 3 0

La comparaison reprend en tête de la colonne puisque les degrés de priorité ont été modifiés. Le terme 2 0 3 0 étant compatible (C_{32}), on va également inverser la valeur de la variable C, comme ci-dessus, et on obtient : 2 0 3 0 ⟶ 3 3 1 . Il n'y a pas génération d'un terme 3 0 3 3 puisque celui-ci existe déjà.

d'où la liste :

3 0 3 3
3 0 3
3 3 1
3 3 1
3 1 3
3 3 1
3 0 0
0 3 0

On passe maintenant à la deuxième colonne. Le premier terme où la variable K apparait avec une valeur non indifférente (≠ 3) est 3 0 3. Ce terme comporte une seule variable. On recense tous les autres termes où K apparait seul (C_{00}) ainsi que la valeur de la variable. Il y a deux termes qui ne contiennent que K. Dans l'un (3 0 3), K = 0, dans l'autre (3 1 3), k = 1. On va donc supprimer ces deux termes puisqu'il n'est pas possible de régler le litige qui les oppose.

La liste devient :

3 0 3 3
3 3 1
3 3 1
3 3 1
3 0 0
0 3 0

et dans cette même colonne, le terme 3 0 0 impose la valeur K = 0.

Dans la troisième colonne (C), c'est également un problème entre termes de degré 0 qui intervient (C_{00}).

On trouve trois termes à une seule variable où C = 1, et aucun à une seule variable où C = 0. On en déduit C =1 et l'assignement C = 0 des deux termes 3 0 0 et 0 3 0 disparait.

d'où la liste

3 0 3 3
3 0 3
3 3 1
3 3 1
3 3 1
3 0 3
0 3 3

Les trois colonnes ont été rendues compatibles; le vecteur à tester s'obtient en effectuant l'intersection entre eux de ces termes (contraction) : 0 0 1

Donc, le test de la bascule dans l'état 0 0 est réalisé par J = 0, K = 0, C = 1, c'est-à-dire un seul vecteur d'entrée, au lieu de six possibles.

Etat ③ ($Y_9 = 0$, $Y_{13} = 1$)

$A_1 = 1$+ On retrouve K = 0 ou C = 0 soit 3 0 0

$B_1 = 0$ Il faut J = 0 ou K = 1 ou C = 0 soit 2 0 1 0

puisque $Y_{13} = 1$, on va améliorer le pire cas par

J = 1 et C = 1 soit

1 3 3
3 3 1

A_2 = 1-, comme $Y_9 = 0$ il faut K = 1 et C= 1, soit

1 3 1 3
1 3 3 1

B_2 = 0+

$Y_9 = 0$ en imposant J = 1 et C = 1

1 3 3
3 3 1

on obtiendra le meilleur 0.

On classe ensuite ces termes par priorité en inscrivant les termes de priorité 1 avant les termes de priorité 2 car un terme de priorité 1 peut contenir l'assignement d'une seule variable tandis qu'un terme de priorité 2 comporte plusieurs variables, et peut sans dommage voir une de ses variables modifiée

1 3 1 3
1 3 3 1
2 0 1 0
1 3 3
3 3 1
1 3 3
3 3 1
3 0 0

Dans la première colonne, la comparaison s'effectue par rapport au terme de priorité 2. Il supprime les 2 termes 1 3 3 incompatibles, puis, à la fin de la colonne, se transforme comme on l'a indiqué plus haut :

2 0 1 0 ⟶ 3 0 3 3
3 0 3
3 3 1

On obtient donc la liste :

3 0 3 3
1 3 1 3
1 3 3 1
3 0 3
3 3 1
3 3 1
3 3 1
3 0 0

La comparaison, reprise par rapport au terme de priorité 3, n'apporte pas de modification.

Dans la deuxième colonne, le terme 1 3 1 3 est pris pour référence. Il supprime 3 0 3 et transforme 3 0 0 en 3 3 0, ce qui donne :

3 0 3 3
1 3 1 3
1 3 3 1
3 3 1
3 3 1
3 3 1
3 3 0

Dans la troisième colonne le terme 1 3 3 1 est compatible avec les trois termes 3 3 1, mais supprime le terme 3 3 0.

On obtient :

3 0 3 3
1 3 1 3
1 3 3 1
3 3 1
3 3 1

et par contraction, la valeur 0 1 1 soit J = 0; K = 1, C = 1, vecteur à tester dans l'état 3 (0 1).

TRANSFERT :

Passage de l'état ① (0 0) à l'état ② (1 0)

B_1 = 1-

Comme Y_{13} = 0, on impose J = 1 et C = 1 avec une priorité maximale puisque la condition d'inscription B_1 = 1- est absolument nécessaire pour le passage de ① à ② .

soit : 3 1 3 3
3 3 3 1

Pour réaliser le pire cas, on mettra dans J K' C, J = 0 ou K =1, ou C = 0

soit : 0 1 0

A_2 = 1+

K = 1 et C = 1 donnent 3 1 3
3 3 1

B_2 = 0-

Comme Y_9 = 1, dans l'état d'arrivée, il faut, si l'on ne veut pas détruire l'isolement de Y_{13} à 0, placer de façon impérative, J = 1 et C = 1

soit : 3 1 3 3
3 3 3 1

d'où la liste classée : 3 1 3 3
3 3 3 1
3 1 3 3
3 3 3 1
3 1 3
3 3 1
0 1 0

Dans la première colonne 3 1 3 3 transforme 0 1 0 en 3 1 0

3 1 3 3
3 3 3 1
3 1 3 3
3 3 3 1
3 1 3
3 3 1
3 1 0

Dans la deuxième colonne, 3 1 3 et 3 1 0 sont compatibles donc K = 1.

Dans la troisième colonne 3 3 3 1 transforme 3 1 0 en 3 1 3.

La liste définitive devient

3 1 3 3
3 3 3 1
3 1 3 3
3 3 3 1
3 1 3
3 3 1
3 1 3

et le vecteur à tester, 1 1 1, c'est-à-dire J = 1, K = 1, C = 1.

Nous ne traiterons pas cet exemple complètement car cela serait trop long et ne montrerait pas, de toute façon, tous les cas possibles. Nous espérons à l'aide de ces quelques lignes, avoir éclairci l'algorithme de recherche de la compatibilité.

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

(1) J. LAGASSE
Logique combinatoire et séquentielle (DUNOD 1968)

(2) PERRIN - DACLIN - DENOUETTE
Systèmes Logiques (DUNOD 1967)

(3) PIEL
Spécification générale d'une famille de circuits intégrés logiques à usage civil (C.I.I.) 20.11.68

(4) HIDEHITO KUBO
A procedure for generating test séquences to detect séquential circuit failures (NEC Research Development n° 12 October 1969 pp. 69-78)

(5) R.S. LEWIS Jr.
An approach to test pattern generation for synchronous sequential circuits. Southern Methodist University, Ph. D. 1967 Engineering Electrical

(6) HENNIE
Fault detecting experiments for sequential circuits.
Proc. 5th Annual Symp. on Switching theory and logical design. Princeton University, Nov. 64, pp. 95-110

(7) C.R. KIME
An organization for checking experiments on sequential circuits
I.E.E.E. Transactions on electronic computers. Février 1966

(8) C.R. KIME
A failure detection method for sequential circuits
Dpt. of Electrical Engineering. University of Iowa. Technical report 66-13 January 1966.

(9) J. POAGE
Derivation of optimum tests to detect faults incombinational circuits.
Technical Report n° 18. Dpt. of Electrical Engineering, Digital Systems Laboratory Princeton University - Mars 1962

(10) J. POAGE
Derivation of optimum tests procedures for switching circuits.
PHD Thésis, Dpt. of Electrical Engineering, Princeton University Janvier 1963.

(11) J. POAGE - Mc. CLUSKEY
Derivation of optimum test sequences for sequential machines

(12) SESHU - FREEMAN
The diagnosis of asynchronous sequential swichting systems.
I.R.E. Transactions on electronic computers. AoÛt 1962

(13) SESHU
On an improved diagnosis program
I.E.E.E. Transactions on électronic computers. Vol. EC. 14, pp. 76-79, AoÛt 1962.

(14) T.L. BOOTH
Sequential machines and Automata theory
(John Wiley and Sons 1967) pp. 177-178.

(15) PIGNAL - ROUX - VINCENT-CARREFOUR
Une méthode de test automatique pour les ensembles logiques.
L'Onde Electrique vol. 48, n° 500, Nov. 1968

(16) Thomas A. KRIZ
Apath sensitizing algorithm for diagnosis of binary sequential logic.
I.B.M. Federal Systems division. Owego New-York - I.B.M. n° 70-825-2486 Mars 1970

(17) ROTH
Diagnosis of Automata failures : A calculus and a method
I.B.M. J. Research Develop. Vol. 10, pp. 278-291, Juillet 1966

(18) DURANTE - ERCEAU - RICHARD
Sur la détection des pannes dans les ensembles logiques.L.A.A.S.
Journées de l'Electronique de Toulouse

(19) E.R. JONES
"Automatic test génération méthods for LSI Logics"
I.E.E.E. Journal of Solid State circuits, vol. SC.2, n° 4, Déc. 67
(pp. 221 - 226)

(20) C. CHICOIX - J. ERCEAU - F. BLANCA - P. PRECHOUX
DIALOG : programme d'élaboration des séquences de test de détection et diagnostic des pannes logiques dans les multiples combinatoires.
Colloque International sur la Microélectronique avancée - Paris - Avril 1970.

(21) JL MICHELON - F. PRUNET
Conception et réalisation d'un testeur automatique de Circuits Logiques intégrés
L.A.A.S. n° 747 - 748.

- TABLE DES MATIERES -

CHAPITRE IV : TESTS DE DETECTION

CHAPITRE V : TESTS DE DIAGNOSTIC