



**HAL**  
open science

# Contribution à l'étude du test aléatoire de mémoires RAM

Antoine Fuentes

► **To cite this version:**

Antoine Fuentes. Contribution à l'étude du test aléatoire de mémoires RAM. Modélisation et simulation. Institut National Polytechnique de Grenoble - INPG, 1986. Français. NNT : . tel-00322092

**HAL Id: tel-00322092**

**<https://theses.hal.science/tel-00322092>**

Submitted on 16 Sep 2008

**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.

# THESE

présentée à

L'INSTITUT NATIONAL POLYTECHNIQUE

de GRENOBLE

pour obtenir le titre de

DOCTEUR-INGENIEUR

par

Antoine FUENTES

---

CONTRIBUTION A L'ETUDE

DU TEST ALEATOIRE DE MEMOIRES RAM

---

soutenu le 16 Décembre 1986 devant la commission d'Examen

## JURY

Monsieur	C. FOULARD	<i>Président</i>
Messieurs	A. COSTES	
	B. COURTOIS	
	A. DAVID	<i>Examineurs</i>
	R. JAZE	
Madame	P. THEVEROD	



INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Président : Daniel BLOCH

Année universitaire 1985-1986

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

Professeurs des Universités

BARIBAUD	Michel	ENSERG	JOUBERT	Jean-Claude	ENSIEG
BARRAUD	Alain	ENSIEG	JOURDAIN	Geneviève	ENSIEG
BAUDELET	Bernard	ENSIEG	LACOUME	Jean-Louis	ENSIEG
BEAUFILS	Jean-Claude	ENSEEG	LESIEUR	Marcel	ENSHG
BESSON	Jean	ENSEEG	LESPINARD	Georges	ENSHG
BLIMAN	Samuel	ENSERG	LONGUEUE	Jean-Pierre	ENSIEG
BLOCH	Daniel	ENSIEG	LOUCHET	François	ENSEEG
BOIS	Philippe	ENSHG	MASSELOT	Christian	ENSIEG
BONNETAIN	Lucien	ENSEEG	MAZARE	Guy	ENSIMAG
BONNIER	Etienne	ENSEEG	MOREAU	René	ENSHG
BOUVARD	Maurice	ENSHG	MORET	Roger	ENSIEG
BRISSONNEAU	Pierre	ENSIEG	MOSSIERE	Jacques	ENSIMAG
BRUNET	Yves	ENSIEG	OBLÉD	Charles	ENSHG
BUYLE-BODIN	Maurice	ENSERG	PARIAUD	Jean-Charles	ENSEEG
CAILLERIE	Denis	ENSHG	PAUTHENET	René	ENSIEG
CAVAIGNAC	Jean-François	ENSIEG	PERRET	René	ENSIEG
CHARTIER	Germain	ENSIEG	PERRET	Robert	ENSIEG
CHENEVIER	Pierre	ENSERG	PIAU	Jean-Michel	ENSHG
CHERADAME	Hervé	UERM CPP	POLOUJADOFF	Michel	ENSIEG
CHERUY	Arlette	ENSIEG	POUPOT	Christian	ENSERG
CHIAVERINA	Jean	UERM CPP	RAMEAU	Jean-Jacques	ENSEEG
COHEN	Joseph	ENSERG	RENAUD	Maurice	UERM CPP
COUMES	André	ENSERG	ROBERT	André	UERM CPP
DURAND	Francis	ENSEEG	ROBERT	François	ENSIMAG
DURAND	Jean-Louis	ENSIEG	SABONNADIERE	Jean-Claude	ENSIEG
FONLUPT	Jean	ENSIMAG	SAUCIER	Gabrielle	ENSIMAG
FOULARD	Claude	ENSIEG	SCHLENKER	Claire	ENSIEG
GANDINI	Alessandro	UERM CPP	SCHLENKER	Michel	ENSIEG
GAUBERT	Claude	ENSIEG	SERMET	Pierre	ENSERG
GENTIL	Pierre	ENSERG	SILVY	Jacques	UERM CPP
GUERIN	Bernard	ENSERG	SOHM	Jean-Claude	ENSEEG
GUYOT	Pierre	ENSEEG	SOUQUET	Jean-Louis	ENSEEG
IVANES	Marcel	ENSIEG	TROMPETTE	Philippe	ENSHG
JAUSSAUD	Pierre	ENSIEG	VEILLON	Gerard	ENSIMAG

Professeurs Université des Sciences Sociales (Grenoble II)

BOLLIET	Louis	CHATELIN	Françoise
---------	-------	----------	-----------

Chercheurs du C.N.R.S

CARRE	René	Directeur de recherche	DAVID	René	Maître de recherche
CAILLET	Marcel	"	DEPORTES	Jacques	"
FRUCHART	Robert	"	DRIOLE	Jean	"
JORRAND	Philippe	"	EUSTATHOPOULOS	Nicolas	"
LANDAU	Ioan	"	GIVORD	Dominique	"
ALLIBERT	Colette	Maître de recherche	JOUD	Jean-Charles	"
ALLIBERT	Michel	"	KAMARINOS	Georges	"
ANSARA	Ibrahim	"	KLEITZ	Michel	"
ARMAND	Michel	"	LEJEUNE	Gerard	"
BINDER	Gilbert	"	MERMET	Jean	"
BONNET	Roland	"	MUNIER	Jacques	"
BORNARD	Guy	"	SENATEUR	Jean-Pierre	"
CALMET	Jacques	"	SUERY	Michel	"
			WACK	Bernard	"



Personnalités agréées à titre permanent à diriger  
des travaux de recherche (Décision du conseil scientifique)

E.N.S.E.E.G

BERNARD CAILLET CHATILLON CHATILLON COULON DIARD	Claude Marcel Catherine Christian Michel Jean-Paul	FOSTER GALERIE HAMMOU MALMEJAC MARTIN GARIN NGUYEN TRUONG	Panayotis Alain Abdelkader Yves Régina Bernadette	RAVAINE SAINFORT SARRAZIN SIMON TOUZAIN URBAIN	Denis Paul Pierre Jean-Paul Philippe Georges
---	---	--	--	---	---

E.N.S.E.R.G

BOREL CHOVET	Joseph Alain			DOLMAZON HERAULT	Jean-Marc Jeanny
-----------------	-----------------	--	--	---------------------	---------------------

E.N.S.I.E.G

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

E.N.S.H.G

ALEMANY BOIS	Antoine Daniel	DARVE MICHEL	Félix Jean-Marie	ROWE VAUCLIN	Alain Michel
-----------------	-------------------	-----------------	---------------------	-----------------	-----------------

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

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

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

CHARUEL	Robert
---------	--------

C.E.N.G

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

Laboratoires extérieurs :

C.N.E.T

DEMOULIN DEVINE	Eric	GERBER	Roland	MERCKEL PAULEAU	Gérard Yves
--------------------	------	--------	--------	--------------------	----------------

\*\*\*\*\*

# Avant-propos

Les travaux présentés dans ce mémoire ont été effectués au Laboratoire d'Automatique de Grenoble dans le cadre d'un contrat avec la Société IBM (Compec) à Bordeaux.

Je tiens à remercier Monsieur C. FOULARD, Directeur du LAG de m'avoir accueilli dans son laboratoire et de présider le jury de cette thèse.

Je veux rendre un hommage tout particulier à mon directeur de recherche, Monsieur R. DAVID, Directeur de Recherche au C.N.R.S.. Je le remercie très sincèrement pour la qualité de son encadrement et son soutien constant et amical.

Je tiens à remercier Monsieur R. JAZE, Ingénieur IBM, pour la suite qu'il compte donner à ce travail. Ses suggestions pertinentes et l'intérêt porté à cette étude m'ont permis d'avoir un contact sympathique et très enrichissant avec le milieu industriel.

J'exprime mes sincères remerciements à Madame P. THEVENOD, Chargé de Recherche au C.N.R.S. Ses remarques et encouragements me furent d'une grande utilité.

Je tiens à exprimer ma profonde reconnaissance à Monsieur A. COSTES, Professeur à l'Institut National Polytechnique de Toulouse, ainsi qu'à Monsieur B. COURTOIS, Chargé de Recherche au C.N.R.S. pour leurs conseils. Qu'ils sachent que leur contribution à ce jury a été très appréciée.

Je tiens à souligner la compétence et la gentillesse de tous ceux qui ont contribué à la réalisation matérielle de cet ouvrage. Qu'ils en soient remerciés.

Enfin, un grand merci à tout le personnel du L.A.G.

# Table des matières



# TABLE DES MATIERES

1. INTRODUCTION	1
PARTIE A :	
2. METHODES DE TEST FONCTIONNEMENT DES RAM	5
2.I- Introduction	7
2.II- Bref descriptif des différentes RAM	7
2.III- Hypothèses et modèles de fautes	15
2.IV- Algorithmes de test	20
2.V- Le test aléatoire de mémoires	22
2.VI- Conclusion	26
3. LONGUEUR DE TEST ALEATOIRE POUR LES MODELES CLASSIQUES	27
3.I- Introduction	29
3.II- Modélisation des fautes	29
3.III- Résultats	42
3.IV- Conclusion	48
4. LONGUEURS DE TEST ALEATOIRE POUR DES FAUTES MULTIPLES	51
4.I- Introduction	53
4.II- Fautes doubles	54
4.III- Commentaires su les résultats	56
4.IV- Autres fautes multiples	59
4.V- Comparaison test aléatoire - test déterministe	59
5. NOUVELLES HYPOTHESES DE FAUTES	63
5.I- Introduction	65
5.II- Nouvelles hypothèses de fautes	65
5.III- Calcul des longueurs de test	77
5.IV- Conclusion	88

**PARTIE B**

<b>6. METHODOLOGIE DE DIAGNOSTIC BASEE SUR DES EXPERIENCES DE TEST ALEATOIRE</b>	<b>89</b>
6.I- Introduction	91
6.II- Bases générales	91
6.III- Un Algorithme de caractérisation	100
6.IV- Un Algorithme de localisation	109
<b>7. CAHIER DES CHARGES POUR UNE MACHINE DE TEST</b>	<b>113</b>
7.I- Introduction	115
7.II- Les fonctions de la machine de test	115
7.III- Schéma de principe de la machine	118
7.IV- Conclusion	120
<b>8. CONCLUSION</b>	<b>121</b>
<b>9. ANNEXE - TARAM UN LOGICIEL DE SYNTHESE</b>	<b>125</b>
9.I- Introduction	127
9.II- Le Mode détection	127
9.III- Le Mode diagnostic	134
9.IV- Conclusion	135
<b>BIBLIOGRAPHIE</b>	<b>137</b>

# Chapitre 1

## INTRODUCTION





Pendant la dernière décennie, la taille des mémoires RAM a quadruplé à peu près tous les 2 à 4 ans. Aujourd'hui, la densité d'intégration permet de produire des RAM de 1 Mbit, alors que les 4 Mbit sont annoncées sur le marché. Mais si les performances techniques ont permis d'accroître la capacité de stockage d'information d'un circuit, elles ont aussi augmenté la vulnérabilité des produits. Cette formidable évolution ne va pas sans poser un grand nombre de problèmes de testabilité.

Tester un produit est devenu une nécessité primordiale. L'évolution de la part des tests dans le coût total des circuits en est d'ailleurs significative : de 10 % en 1970, leur contribution financière est passée à 45 % du coût global en 1985. De ce fait, il s'avère nécessaire d'optimiser les techniques et donc les temps de test.

Pour une mémoire, la tâche pourrait sembler aisée : vérifier que chacun des points mémoire peut être mis à 0 et à 1. Ce n'est malheureusement pas si simple. En effet, à cause de la très haute densité d'intégration, on peut avoir des problèmes d'interaction entre cellules et le fonctionnement du point mémoire n'est perturbé que par certaines configurations de la mémoire. Une solution serait alors de tester de façon exhaustive la mémoire dans toutes ces configurations, soit un temps de test de  $10^{110}$  ans pour une 256K avec un cycle de 100 ns ! D'où la nécessité de concevoir des stratégies de test.

Les procédures de test de circuits peuvent être regroupées en trois catégories : les tests paramétriques qui vérifient les caractéristiques statiques du circuit, les tests dynamiques qui permettent de vérifier l'évolution des grandeurs électriques et les tests fonctionnels qui vérifient le bon fonctionnement logique.

C'est à cette dernière catégorie que nous nous intéressons. On y distingue les tests déterministes, pour lesquels on fait des hypothèses de fautes puis on écrit une séquence d'entrée permettant leur détection, et le test aléatoire dont le principe est d'envoyer une séquence d'entrée aléatoire (syntaxiquement correcte) et de calculer la longueur nécessaire à la détection des fautes pour une probabilité donnée.

Dans notre étude nous avons développé une stratégie de test aléatoire des mémoires. La première partie est consacrée à la détection des fautes par du

test aléatoire et à l'évaluation des longueurs de test. Après avoir présenté rapidement un panorama du test fonctionnel de mémoires, nous décrivons nos résultats sur le test aléatoire, puis nous donnons une comparaison avec les méthodes déterministes. Dans la deuxième partie, nous proposons une méthodologie d'aide au diagnostic de pannes basée sur des expériences de test aléatoire, ainsi qu'un cahier des charges d'une machine à tester.

PARTIE A

DETECTION DE FAUTES



## Chapitre 2

# METHODES DE TEST FONCTIONNEL DES RAM



## 2.I- INTRODUCTION

Nous donnons dans le présent chapitre une brève synthèse des résultats sur le test des RAM au début de notre étude.

Dans un premier temps, nous avons cru bon de décrire les différents types de RAM existant ainsi que les technologies utilisées. Nous détaillons aussi les divers mode d'accès se développant actuellement.

Nous présentons ensuite les hypothèses de fautes utilisées dans la littérature. Ces hypothèses sont regroupées en différentes catégories pour former des modèles de fautes.

A partir de ces modèles, des algorithmes de test permettant la détection des fautes ont été établis par différents auteurs, nous fournissons un tableau récapitulatif de ces algorithmes.

Enfin, nous avons clos ce chapitre en présentant le test aléatoire des RAM et les résultats obtenus sous des hypothèses de collage.

## 2.II- BREF DESCRIPTIF DES DIFFERENTES RAM

### 2.II- 1. Introduction

Nous nous limitons dans ce paragraphe à la description des mémoires vives ou RAM (Random Access Memory). [LIL 77, MET 74, PIC 84].

Ces mémoires, dites à lecture et écriture, servent à stocker temporairement des données. En effet, une coupure de l'alimentation provoque l'effacement de l'information.

On peut les regrouper en deux grandes catégories :

- les RAM statiques : l'information y est conservée sans précautions particulières, même pour un fonctionnement à la fréquence nulle. Cela parce que la cellule de base de la mémoire est un flip-flop.



- les RAM dynamiques : l'information se dégrade rapidement et on doit la "rafraichir" périodiquement. La cellule de base est en effet une capacité de structure de transistor (famille MOS et en bipolaire  $I_2L$ ). Sans rafraichissement, le fonctionnement à la fréquence nulle est interdit, par conséquent ces mémoires exigent des circuits d'intendance destinés à pourvoir ce rafraichissement.

Nous allons par la suite détailler chacune de ces deux catégories.

## 2.II- 2. Les RAM statiques

Pour ce type de mémoire, la cellule de base est le flip-flop. Quatre transistors sont nécessaires à sa réalisation. Si on tient compte des nécessités d'adressages, amplification de lecture et écriture, la structure du point mémoire nécessite quelques six transistors en moyenne d'où un obstacle à la réalisation de mémoires de très fortes capacités (une mémoire de 256 k nécessiterait 1 600 000 transistors) qui demanderait des niveaux d'intégration que l'on n'est en mesure de produire avec un rendement acceptable actuellement. Ce sont donc des mémoires à capacité modeste.

Dans la suite du paragraphe, nous distinguons les différentes technologies utilisées.

### **TTL**

Ces RAM de capacité modeste (maximum 1 à 4 K) sont historiquement les plus anciennes. Ce sont des mémoires rapides mais handicapées par une consommation excessive très difficile à réduire. Elles sont surtout utilisées comme mémoires tampon.

### **BIPOLAIRE**

De structure très voisine des RAM TTL, cependant la présence supplémentaire d'une ligne E ("chip enable") permet de réduire la consommation de la mémoire. En effet si E est à 0, la mémoire est sélectionnée et tous les circuits sont alimentés, alors que si E est à 1, pratiquement seule la matrice mémoire est alimentée d'où conservation de l'information tout en réduisant la consommation d'énergie (de l'ordre de cinq fois moins par rapport à une RAM TTL

de même capacité).

#### **MOS**

Ces mémoires nécessitent un découplage extrêmement soigné et une implantation plus délicate (générateurs de parasite). Elles sont aussi moins rapides. Leur assez forte capacité (jusqu'à 16 K) les rend bien adaptées aux mémoires de microprocesseurs : elles permettent de s'affranchir du rafraîchissement des mémoires dynamiques.

#### **CMOS**

Ces mémoires sont caractérisées par une consommation remarquablement faible notamment en "stand by" (c'est-à-dire quand seule la matrice de cellules est alimentée) d'où leur grande utilisation pour du matériel portatif alimenté par batteries.

### **2.II- 3. Les RAM dynamiques**

On l'a vu, les RAM statiques du fait de leur structure ne permettent pas la réalisation de mémoires à très forte capacité. On a donc cherché à réduire le nombre de transistors par point mémoire.

On en est arrivé à utiliser, pour stocker l'information, la capacité de structure d'un transistor, d'où les cellules à 1 MOS. Mais du fait des courants de fuite, le maintien de l'information ne se fera qu'à condition de rafraîchir périodiquement les cellules.

Evidemment, l'utilisation de ce point mémoire complique sérieusement la circuiterie annexe à la matrice de cellules.

En augmentant le nombre de bits par puce, on augmente le nombre de lignes d'adresse (64 K nécessitent 16 lignes d'adresse). Si on veut éviter l'utilisation de boîtiers à grand nombre de broches, on est amené à multiplexer ces lignes : les adresses transmises en deux fois (ligne puis colonne) sont stockées au niveau des décodeurs ligne et colonne dans des bloqueurs (latches) activés par RAS et CAS (respectivement Row Adress Strobe et Column Adress Strobe) (figure 1).

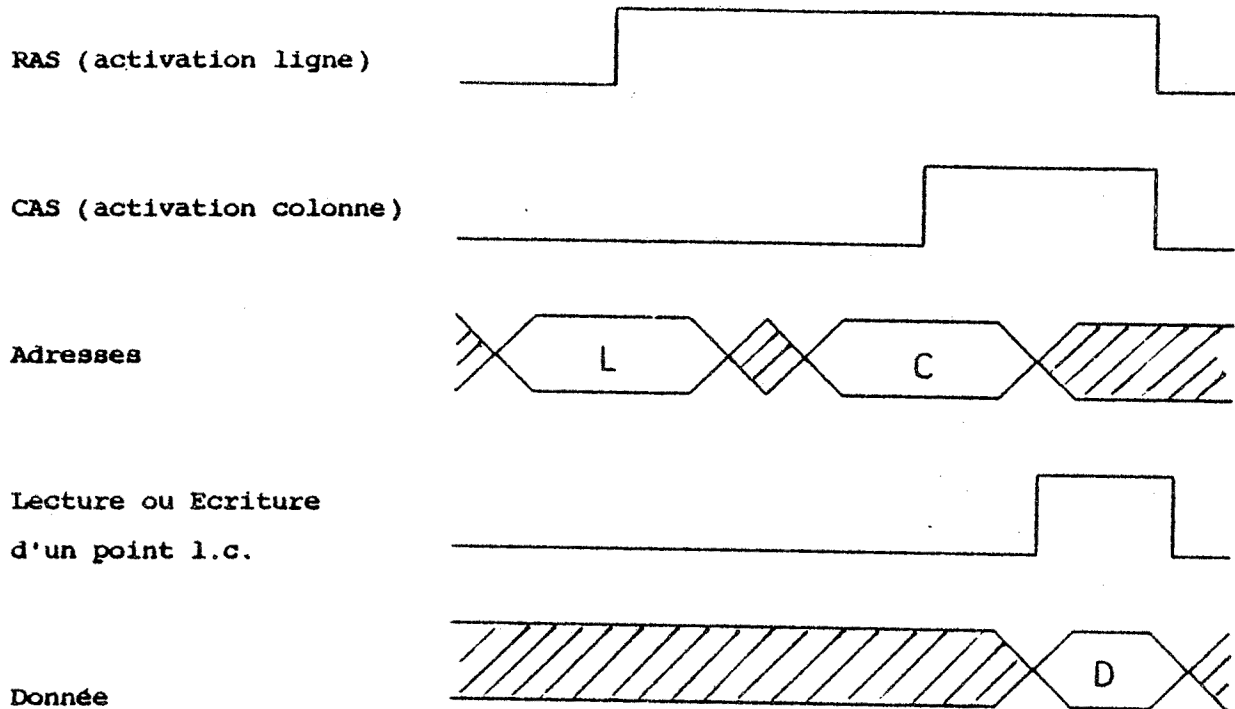


Figure 1 : Adresse d'une point l.c.

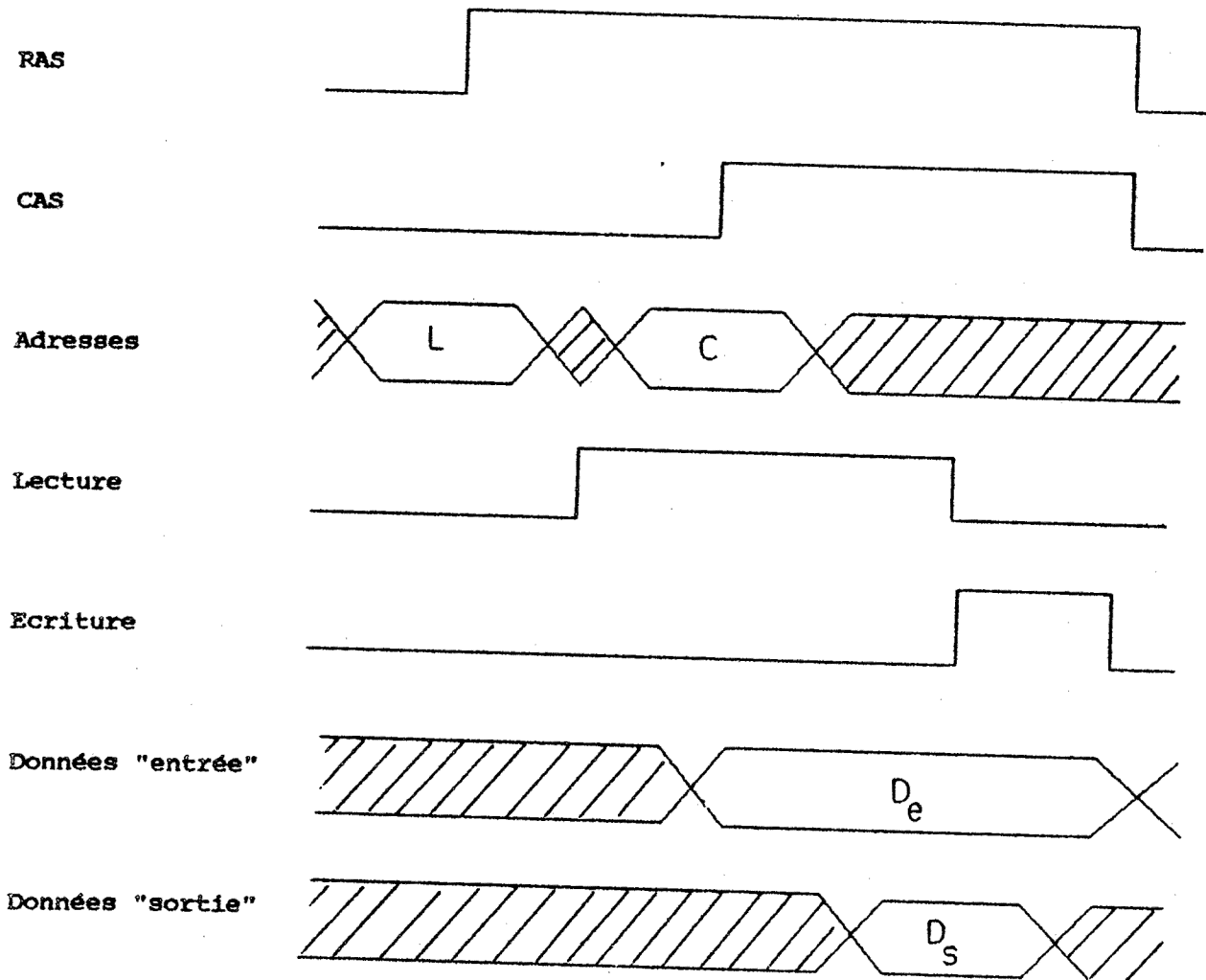
Ce procédé va encore augmenter la complexité des circuits annexes.

D'autre part, de nouvelles fonctions mémoire apparaissent : elles contribuent à augmenter les vitesses d'accès et extraction de données :

le double accès : il permet à la fois la lecture de données mémorisées et l'enregistrement de nouvelles données (figure 2).

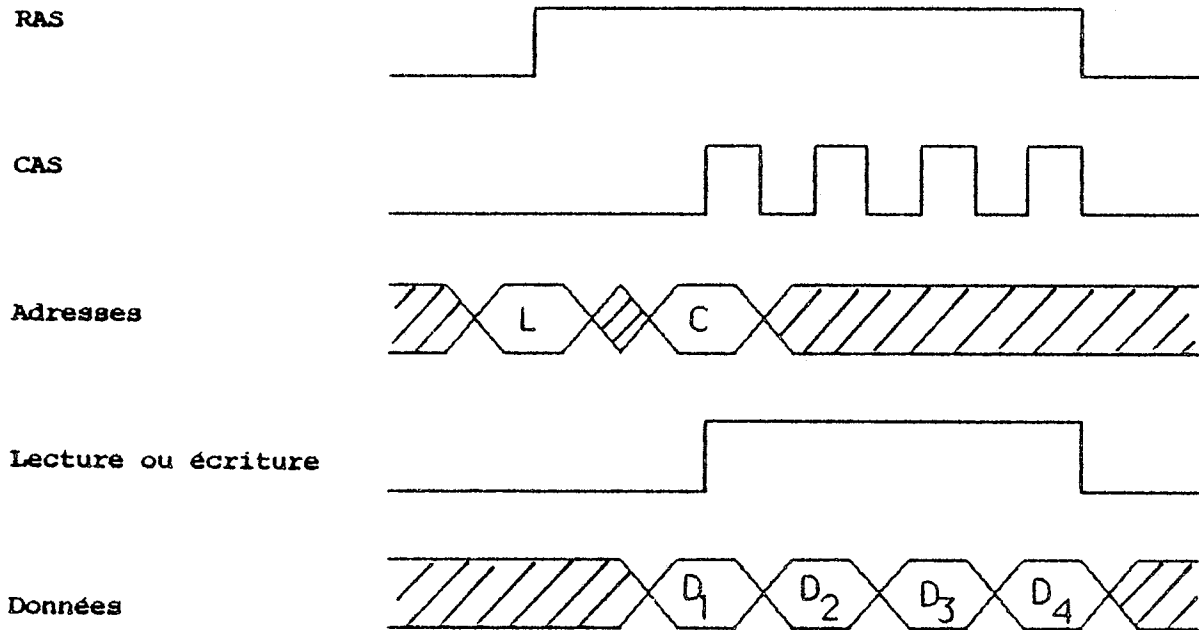
Ces données sont en général stockées de façon classique, mais elles sont transférées dans un registre (output buffer) au moment de la lecture. Après lecture, le cycle mémoire permet l'extraction en série des données du registre.

Cette fonction s'applique en particulier aux écrans de visualisation où les hautes définitions exigent des mémoires rapides et des extractions de données par train en série.



**Figure 2 : Le double accès**

**Le mode "grignotage"** : (nibble en anglais, on trouve l'expression mode "quartet" dans la littérature en Français). Néanmoins, on peut penser que ce mode d'adressage pourrait se faire par octet par exemple). Dans ce mode, l'organisation physique de la mémoire se fait par groupes de 4 bits. Cela donne quatre zones de mémoires identiques sur la puce, mais leurs données sont recueillies sur une seule broche sous forme d'une salve de 4 bits en série (shift register) (figure 3).



**Figure 3 : Nibble mode**

**Le mode page** : ici on accède à une ligne de mémoire que l'on "maintient" pendant le temps de sélection des colonnes. Dans une mémoire de 256 K, normalement organisée en 512 x 512 bits, une ligne d'adresse commande 512 points mémoire (ou une page).

Le temps de décodage et de verrouillage pour la ligne sont éliminés à partir du 2ème accès sur la même ligne. Seuls les temps de décodage et de verrouillage de la colonne sont nécessaires à chaque accès (figure 4).

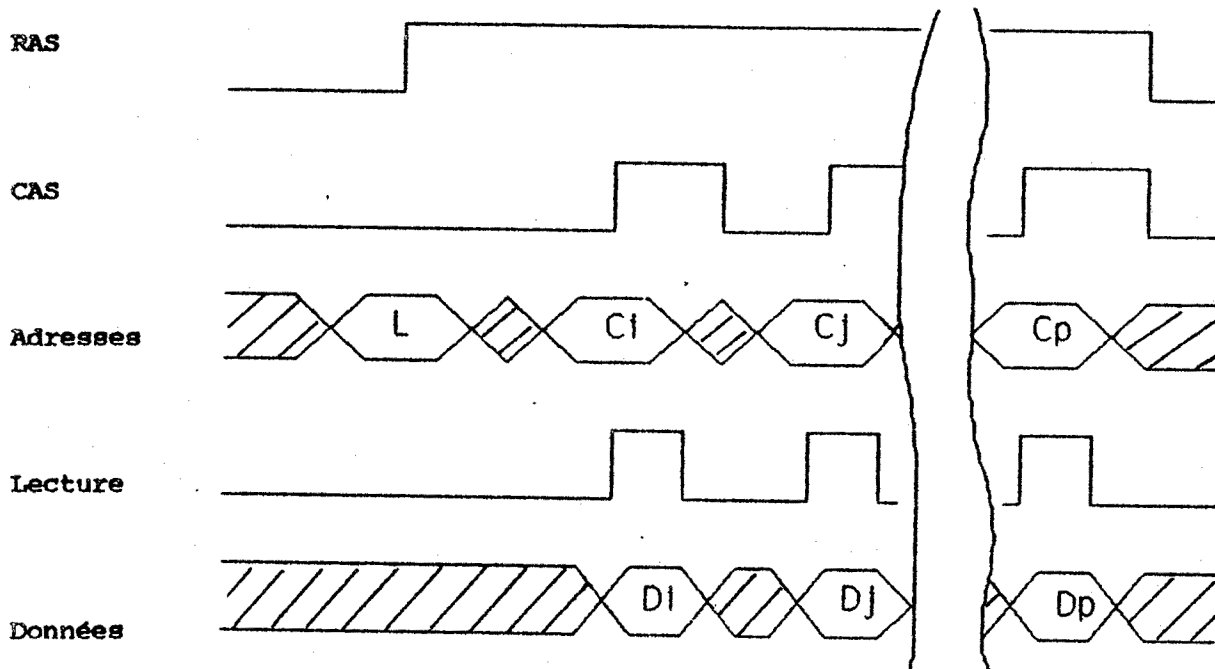


Figure 4 : Adressage par mode page

Le mode en cascade : (en anglais ripple mode). Le mode page est facilité par traitement "pipe-line" des adresses lignes.

2.II- 4. Remarques [Pic 84]

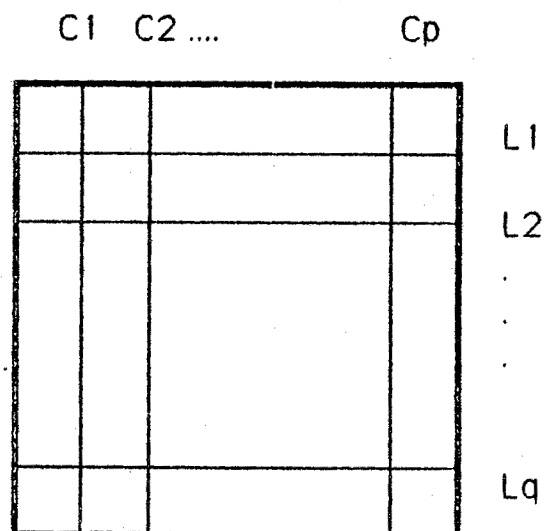


Figure 5 : Répartition des lignes et colonnes  
dans une matrice idéale

Dans la pratique on n'a jamais, sur des mémoires de grosse capacité, une répartition des adresses lignes et colonnes telle que celle représentée à la figure 5 mais plutôt celle de la figure 6.

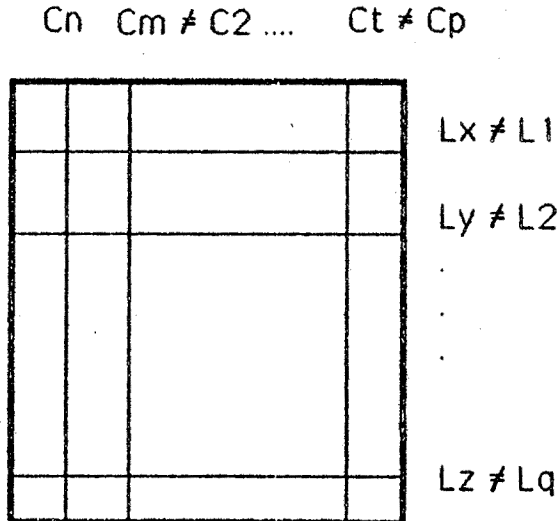


Figure 6 : Répartition des lignes et colonnes  
dans une matrice de mémoire

En d'autres termes, les colonnes où les rangées correspondant à des adresses consécutives ne sont pas physiquement adjacentes sur la puce.

Les raisons de ce "brouillage" sont dues essentiellement à des raisons technologiques et de facilité de réalisation.

D'autre part, avec la densité croissante d'intégration, il devient de plus en plus difficile d'être certain qu'une puce soit bonne. Pour améliorer le rendement de fabrication on commence à voir apparaître des RAM comportant un nombre de rangées ou de colonnes supérieur à celui nécessaire. Lors du test préalable au sciage pour l'obtention de la puce on peut détecter la colonne ou rangée défailante, la ligne x ou y correspondante est alors coupée au laser et le décodeur correspondant de l'élément réserve est personnalisé. Ceci signifie que d'une pièce à l'autre, il sera impossible de connaître l'implantation physique exacte des lignes et/ou colonnes.

## 2.III- HYPOTHESES ET MODELES DE FAUTES

## 2.III- 1. Introduction

Si on voulait tester une mémoire dans toutes ses configurations possibles, on obtiendrait des temps de test aberrants : une mémoire de 64 K avec un cycle de 100 ns nécessiterait  $10^{110}$  ans de test.

En conséquence, pour tester les mémoires on a été amené, dans un premier temps, à faire des hypothèses de fautes. La stratégie du test consistera en la détection, pour un modèle donné, de ces fautes.

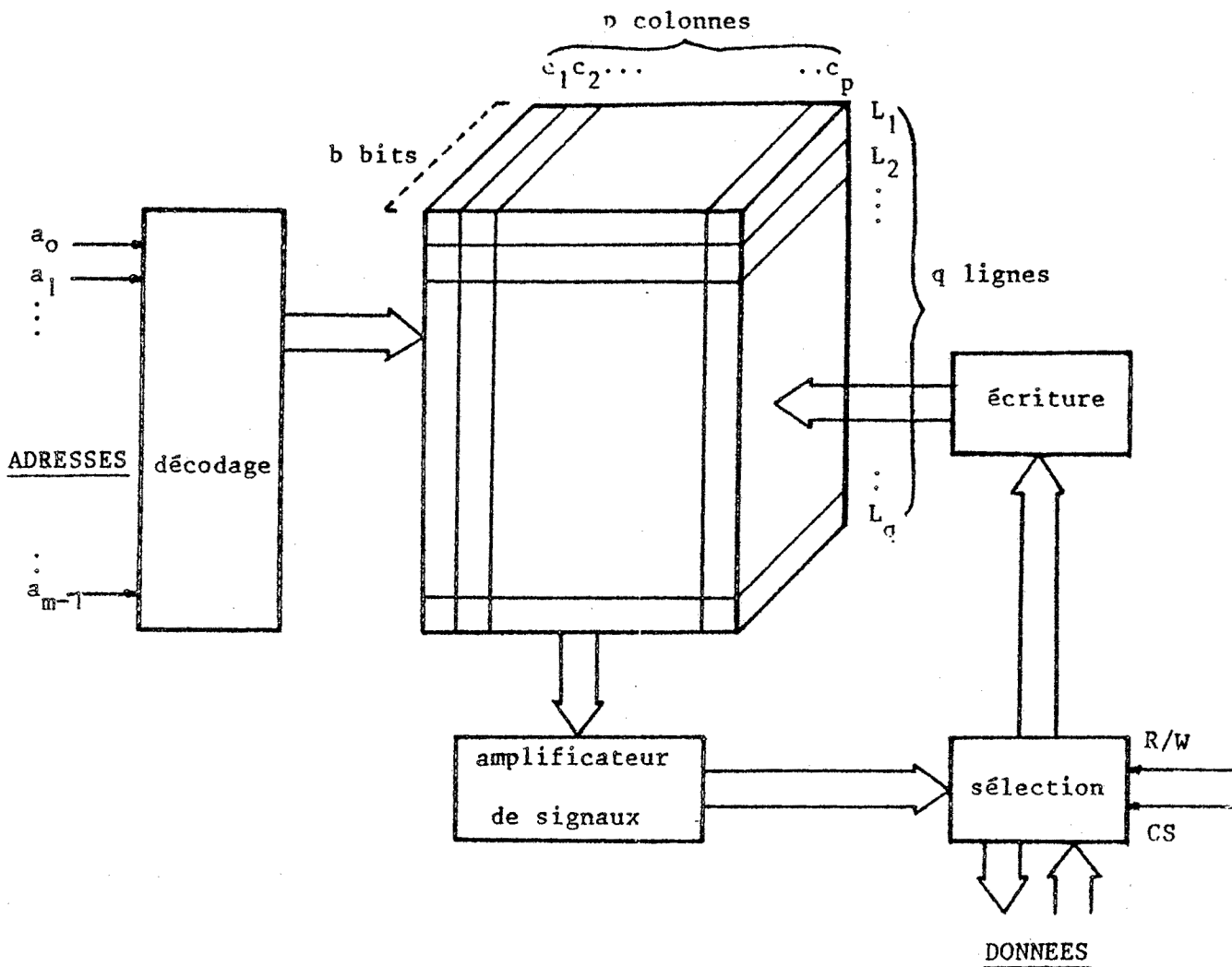


Figure 7 : Modèle de mémoire



## 2.III- 2. Modèle de mémoire [Tha 77]

On supposera que la mémoire est composée de trois blocs fonctionnels :

- matrice de cellules
- logique de décodage
- logique de lecture écriture

Dans la suite, on ne considèrera que les fautes dans la matrice mémoire. En effet :

- fautes dans la logique de décodage
  - le décodeur n'accède pas à la cellule adressée, il peut en revanche adresser d'autres cellules.
  - le décodeur peut adresser plusieurs cellules (y compris la cellule adressée).

Sous l'hypothèse que le décodeur ne transforme pas la logique combinatoire en logique séquentielle, toute faute du type précédent se ramène à un collage ou couplage de points sur la matrice de cellule [Tha 77].

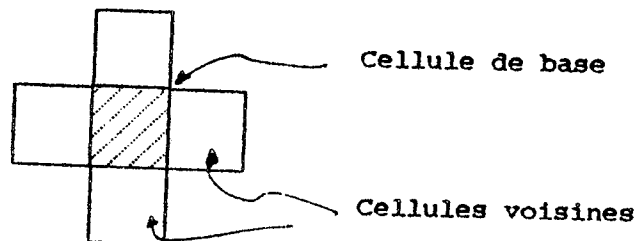
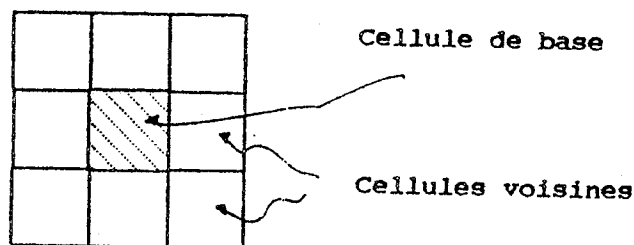
- fautes sur la logique de lecture écriture
  - lignes de sortie de l'ampli collées à 0 ou 1 : équivalent à un collage de point sur la matrice
  - lignes d'entrées court-circuitées : équivalent à un couplage de points mémoires.

On peut donc toujours avec ce modèle se ramener à une faute sur la matrice. Cette matrice par la suite sera constituée de  $n$  mots de  $b$  bits.

## 2.III- 3. Définitions

- a) Etat : une cellule  $i$  est à l'état  $x$  ( $x = 0$  ou  $1$ ) si la lecture à l'adresse  $i$  donne la valeur  $x$ .

- b) Transition : (↑ ou ↓ c'est-à-dire passage de 0 à 1 ou de 1 à 0 respectivement) : il y a transition sur la cellule  $i$  si la cellule  $i$  étant à l'état  $x$  on écrit la valeur  $\bar{x}$  à l'adresse  $i$ .
- c) Collage : la cellule  $i$  est collée à l'état  $x$  si quelque soit la valeur ( $x$  ou  $\bar{x}$ ) que l'on écrit à l'adresse  $i$ , la cellule  $i$  reste toujours à l'état  $x$  ceci ayant lieu quelque que soit l'état des  $n-1$  autres cellules.
- d) Influence idempotente : la transition ↑ (ou ↓) de la cellule  $j$  a une influence idempotente sur la cellule  $i$  si une transition de  $j$  provoque une transition de  $i$  seulement quand cette dernière est dans un certain état  $x$  (dépendant de  $i, j$ ). Si la cellule  $i$  est à l'état  $\bar{x}$  la transition de la cellule  $j$  n'a pas d'effet sur  $i$ .
- e) Influence d'inversion : la transition ↑ (resp ↓) de la cellule  $j$  a une influence d'inversion sur la cellule  $i$  si la transition ↑ (resp ↓) de la cellule  $j$  change l'état de la cellule  $i$ , quelque soit l'état de cette dernière avant la transition.
- On remarque qu'une influence est soit une influence d'inversion soit idempotente.
- f) Cellules voisines : on sera amené dans certains cas à considérer un voisinage de cellules, on parlera de  $v$  cellules.

ex  $v = 5$  $v = 9$ 

## 2.III- 4. Modèle utilisé par Abraham, Nair, Thatte et Marinescu

Les hypothèses de fautes rencontrées dans ce modèle sont regroupées en deux types : les collages et les couplages.

Ce type de faute correspond à des défauts physiques de la mémoire tels que courts-circuits, qui empêchent le bon fonctionnement logique : on ne peut écrire ou lire la valeur désirée en un point.

### 2.III- 4.1. Les collages

Une ou plusieurs cellules peuvent être collées à 0 ou 1. On remarquera qu'ici le collage d'un point mémoire est indépendant de l'état des  $n-1$  autres cellules.

### 2.III- 4.2. Les couplages

On dit qu'il y a couplage entre cellules lorsqu'il existe une influence entre ces cellules.

\* 2 - couplage : on se limite au cas où on suppose qu'il n'y a influence (idempotente ou d'inversion) qu'entre paires de cellules.  
[MAR 80-1], [MAR 80-2] exemple :  $\uparrow j \Rightarrow \uparrow i$  : le front 0  $\nearrow$  1 de la cellule  $j$  provoque le changement de  $i$  si  $i$  valait 0.

\*  $v$  - couplage : pour un ensemble de  $v$  cellules, une transition sur une des  $v$  cellules provoque une transition sur une autre cellule, ceci pour certaines valeurs des  $v - 2$  autres cellules.

exemple à 3  $(i, j, k)$  :  $\uparrow j \Rightarrow \uparrow i$  si  $k = 1$ .

Papachistou et Sahgal [Pap 85] ont établi une liste exhaustive de tous les 2 couplages possibles. La distinction entre fautes interactives et non interactives y est faite. On peut dire, de façon informelle, que 2 fautes sont interactives si leurs effets peuvent s'annuler.

## 2.III- 5. Modèle utilisé par Reddy et Suk [Suk 79]

Les temps d'accès, de communication, de stockage de données..., dépendent en fait de la configuration de la mémoire. Des effets parasites, couplages capacitifs... peuvent apparaître alors dans certaines zones de la mémoire. Ces défauts donnent lieu à des fautes de type PSF (Pattern Sensitive Faults, fautes sensibles à la configuration de la mémoire) qui ont été modélisées comme suit.

Pour ce modèle, on considère une cellule de base  $i$  et ses  $v-1$  voisines. On suppose que la faute sur  $i$  va dépendre de l'état des  $v-1$  voisines.

On distingue :

- les PSF actives : une transition sur une cellule influente ( $j$ ) a une action sur la cellule de base ceci pour une configuration donnée des  $v-2$  cellules déterminantes. Exemple avec  $v=4$  ( $i, j, k_1, k_2$ ) :  $\uparrow j \Rightarrow \uparrow i$  si  $k_1 k_2 = 01$ .
- les PSF passives : pour une configuration donnée des  $v-1$  cellules voisines une action ( $\uparrow$  ou  $\downarrow$ ) sur la cellule de base ne provoque aucun changement de cette dernière.  
Exemple avec  $v = 3$  ( $i, j, k$ ) :  $i$  garde sa valeur si  $jk = 00$ , quelles que soient les écritures que l'on veut effectuer sur  $i$ .

## 2.III- 6. Modèle unifié proposé par Courtois [Cour 81]

Courtois présente un modèle unifié en se rattachant au modèle présenté par Reddy et Suk : il est basé notamment sur le fait que :

- les collages sont des cas particuliers de PSF passives.  
En effet, pour un des modèles, une cellule peut être collée quel que soit l'état des autres, pour l'autre modèle, le collage dépend d'une configuration des autres cellules.
- les couplages sont des PSF actives. Par exemple, pour le couplage à 2 on ne considère que la cellule  $i$  et la cellule  $j$ , l'état des  $n-2$  autres cellules peut être quelconque.

Néanmoins, afin de généraliser la notion de PSF pour le couplage de cellules non voisines, il a été amené à définir d'autres notions.

Les fautes sont définies d'après les notions suivantes :

- les fautes : elles sont toutes de type PSF, une PSF pouvant être active ou passive.
- le domaine d'influence pour une cellule de base : il comprend l'ensemble des cellules qui peuvent avoir une influence sur la cellule de base c'est-à-dire qu'une transition d'une de ces cellules pourra avoir une influence sur la cellule de base.
- le domaine dont l'état est déterminant pour la manifestation de l'influence. Autrement dit, l'influence de j sur i sera effective pour un certain état du domaine déterminant. Ce domaine peut être différent du domaine d'influence, il peut aussi ne pas exister pour certaines fautes par exemple les collages et les couplages à 2.

## 2.IV- ALGORITHMES DE TEST [Cour 81], [Aba 83]

### 2.IV- 1. Introduction

Les algorithmes étudiés permettent de réaliser des séquences de test de type fonctionnel : on vérifie le bon comportement logique du circuit. Son principe est le suivant : on fait une série d'hypothèses de fautes sur la mémoire (modèle) et on écrit une séquence de vecteurs de test qui doit détecter ces fautes. Cette séquence de vecteurs (opérations de lecture et écriture dans la mémoire) est appliquée au circuit à tester. On compare alors le comportement du circuit sous test à celui d'un circuit de référence.

Trois critères sont à considérer pour évaluer un algorithme de test :

- la couverture de fautes : ensemble des fautes testées par la séquence obtenue.
- le coût du test : c'est-à-dire le nombre d'accès faits à la mémoire sous test.
- la simplicité qui est à considérer pour l'implantation de l'algorithme.

REFERENCES	COLLAGES		COUPLAGES		HYPOTHESES DE PANNEES			DUREE
	1 collage	2 collages	active ou passive	Domaine d'influence	Domaine dont état déterminant	Conditions restrictives ou particulières sur les hypothèses	obtenue	
(Mars 78)	oui	oui		non		Aucune	4n-2n	?
(Mars 78)	oui	non		non		pas collages dans la logique de décodage	4n	4n
(Mars 78)	oui	non		non		Hypothèse sur décodeur et technologie	4n	4n
(Mars 78)	oui	non		non		Hypothèse sur décodeur	13n/3	4n
(Mars 78)	oui	non		non		Hypothèse sur technologie	4n	4n
(Mars 78)	oui	non		non		néant	5n-2	4n
(Mars 78)	oui	infl. inv. cell. vois.		non		Adresse physique connue Localisation panne	10n	8n
(Mars 78)	oui	oui		non		néant	18n Log n	?
(Mars 78)	oui	oui		non		Cellule ne peut influencer et être influencée	11n	9n-2
(Mars 78)	oui	influence idempotente		non		Cellule ne peut influencer et être influencée	15n	?
(Mars 78)	oui	influence idempotente		non		Aucune	17n	?
(Mars 78)	oui	oui		non		Aucune	50n	?
(Suk 80)	oui	non	passive	néant	1 cellule voisine	Adresse physique connue Localisation panne	10n	3n
(Suk 80)	oui si pas collages	oui si pas collages	active et passive	1 voisin	1 cellule voisine	Adresse physique connue Localisation panne	14n	12n
(Suk 80)	oui	non	passive	néant	2 voisins	Adresse physique connue Localisation panne	18n	16n
(Suk 80)	oui	avec 2 voisins ?	active	2 voisins	1 voisin	Adresse physique connue Localisation panne	26n	28n
(Suk 80)	oui	avec 2 voisins ?	active et passive	2 voisins	active : 1 voisin passive : 2 voisins	Adresse physique connue Localisation panne	34n	32n
(Suk 80)	oui	non	passive	néant	3 voisins	Adresse physique connue Localisation panne	34n	32n
(Suk 78)	oui	Infl. idemp. cell. vois.	active	4 voisins	cellule considérée	PGF active idempotente Adresse physique connue	38n	34n
(Suk 80)	oui	avec 3 voisins ?	active	3 voisins	2 voisins	Adresse physique connue Localisation panne	64n	64n
(Suk 80)	oui	non	passive	néant	4 voisins	Adresse physique connue Localisation panne	64n	64n
(Suk 80)	oui	non	passive	néant	4 voisins	Adresse physique connue Localisation panne	67,5n	64n
(Suk 80)	oui	avec 3 voisins ?	active et passive	3 voisins	active : 2 voisins passive : 3 voisins	Adresse physique connue Localisation panne	92n	94n
(Suk 80)	oui	avec 4 voisins ?	active	4 voisins	3 voisins	Adresse physique connue	99,5n	97n
(Suk 80)	oui	non	passive	néant	3 voisins	Adresse physique connue Localisation panne	130n	128n
(Suk 80)	oui	avec 4 voisins ?	active	4 voisins	2 voisins	Adresse physique connue Localisation panne	162n	160n
(Suk 80)	oui	avec 4 voisins ?	active	4 voisins	3 voisins	Adresse physique connue Localisation panne	163,5n	160n
(Suk 80)	oui	avec 4 voisins ?	active et passive	4 voisins	active : 3 voisins passive : 4 voisins	Adresse physique connue	165n	161n
(Suk 80)	oui	avec 4 voisins ?	active et passive	4 voisins	active : 3 voisins passive : 4 voisins	Adresse physique connue Localisation panne	194n	192n
(Suk 80)	oui	avec 4 voisins ?	active et passive	4 voisins	active : 3 voisins passive : 4 voisins	Adresse physique connue Localisation panne	195,5n	192n
(Mars 78)	oui	oui	active	1 cellule quelconque	1 cellule quelconque	Aucune	n-32nlog n	?
(Mars 80)	oui	avec k voisins ?	active et passive	(k-1) voisins	(k-1) voisins	Adresse physique connue	(k-2)2 n	?
(Mars 78)	oui	oui	active et passive	cellule quelconque	active : n-2 cellules passive : n-1 cellules	Aucune	0(2n)	?

Table 1

AT&T TELECOMMUNICATIONS  
 FRANCE  
 38400 ST MARTIN D'HERES CEDEX  
 FRANCE  
 TEL (76) 51.46.36

## 2.IV- 2. Tableau récapitulatif

La Table 1 s'est principalement inspirée de [Cour 81] et [Aba et Reg 84]. Elle présente également des résultats qui ont été donnés dans d'autres articles qui sont en référence.

Cette table propose un récapitulatif des différents algorithmes existant. Pour chaque algorithme, nous avons détaillé le modèle de faute utilisé ainsi que la couverture correspondante.

Malgré l'intérêt théorique que présente l'unification des modèles de faute proposées par Courtois, il nous a paru préférable de présenter l'ensemble des fautes en trois catégories, ce qui permet une compréhension plus aisée. Cela correspond aussi à des classes d'algorithmes différents. Ces trois catégories sont 1) les collages, 2) les couplages à 2 (idempotence et inversion) et 3) les PSP, qui incluent les autres types de fautes y compris les couplages de type ET ou OU entre deux ou plusieurs cellules.

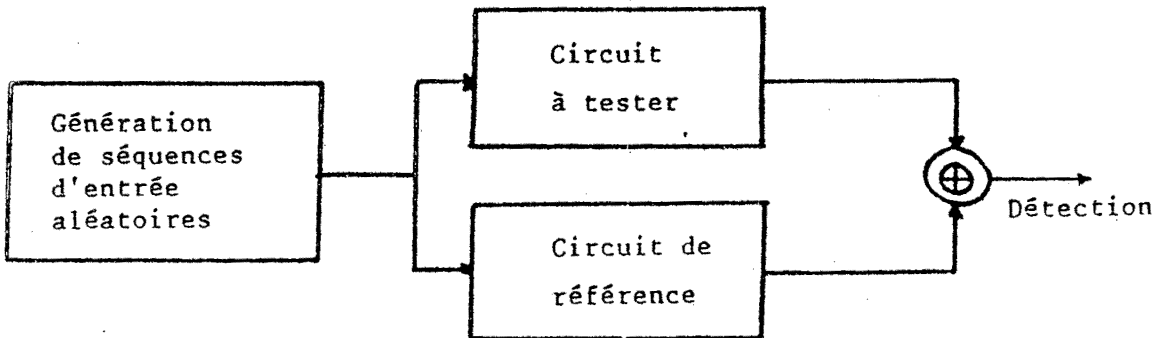
## 2.IV- LE TEST ALEATOIRE DE MEMOIRES

Plusieurs travaux sur le test aléatoire de mémoires ont donné lieu à certains articles.

Nous allons, dans ce paragraphe, présenter une brève synthèse des différents résultats obtenus.

### 2.V- 1. Notions sur le test aléatoire

Le test aléatoire de circuits est un test de type fonctionnel : il vérifie le bon comportement logique du circuit. Son principe est illustré à la figure 8.



**Figure 8 : Principe du test aléatoire**

Une séquence d'entrée aléatoire est appliquée simultanément au circuit à tester et à un circuit de référence. Les sorties des deux circuits sont comparées.

Le résultat du test peut prendre deux valeurs : bon (aucune erreur n'a été détectée) ou défectueux (au moins une erreur a été détectée).

Le résultat du test est juste si sa valeur est défectueux quand le circuit est défectueux.

La probabilité pour que le résultat d'un test soit juste dans le cas le plus défavorable n'est pas la probabilité d'avoir testé chaque faute mais la probabilité d'avoir testé la faute la plus difficile à détecter [Dav 76]. La qualité de détection notée  $1 - Q_D$  est la probabilité pour que le résultat du test soit juste.

On définit la longueur de la séquence de la faute la plus difficile à détecter pour une qualité de détection donnée.



## 2.V- 2. Premiers résultats obtenus par Thevenod-Fosse et David

Les résultats d'une première approche du test aléatoire de mémoires sous des hypothèses de collage [Dug 77] ont montré que les fautes les plus difficiles à détecter par un test aléatoire sont celles qui sont équivalentes au collage à 0 (ou 1) d'un seul point mémoire.

Sous ces hypothèses de collage, une étude théorique donnant une relation entre la longueur nécessaire à la détection d'un collage de point mémoire et la qualité de détection a donné lieu à un article [The 78].

Dans cet article, on considère que la probabilité de chacune des entrées est de 0,5 sauf pour l'entrée lecture / écriture : la probabilité d'écrire est  $w$ .

On montre dans cet article que l'initialisation de la mémoire joue un rôle important : en effet, il sera plus difficile de détecter un collage à 0 dans une mémoire initialisée à 0 que dans une mémoire initialisée à 1. Pour l'étude, on se place donc dans le cas le plus défavorable.

Une relation entre  $L$  et  $Q_D$  est déterminée :

Pour une qualité de détection  $1 - Q_D = 10^{-3}$  et  $w = 0,5$  (équiprobabilité de lecture et écriture) on obtient une longueur  $L = 30.n$  où  $n$  est le nombre de mots mémoires.

En effet, il est démontré que la longueur  $L$  dépend du nombre de mots  $n$  de la mémoire mais est indépendant du nombre  $b$  de bits par mots.

## 2.V- 3. Résultats de simulation obtenus par Bardell [Bar 83]

Des résultats de simulation ont été effectués concernant des mémoires de petite capacité (de 1 à 128 mots de 1 bit) pour diverses valeurs de  $w$  et de  $L$ . Les résultats obtenus ne correspondent pas aux résultats théoriques présentés par Thévenod-Fosse et David.

#### 2.V- 4. Résultats améliorés [Dav 85]

Au vu des résultats précédents, une deuxième approche du problème a permis de déterminer que l'erreur était due à l'approximation suivante :

On faisait l'hypothèse implicitement que la probabilité de détection à  $(t+1)$  est indépendante de la probabilité de détection à  $t$ . Or cette hypothèse permet en fait de calculer la longueur moyenne entre deux détections successives qui est différente de la longueur pour la première détection.

Une modification du calcul de  $L$  a permis d'obtenir des résultats qui coïncident bien avec les résultats de simulation obtenus par Bardell.

Une autre approche du problème à l'aide chaîne de Markov donne aussi des résultats théoriques en accord avec la simulation.

Les principaux résultats sont les suivants :

- $L$  ne dépend pas du tout du nombre de bits dans un mot
- $L = 46 n$  où  $n$  est le nombre de mots mémoires pour une qualité de détection  $1 - Q_D = 10^{-3}$ , et si l'état initial du bit collé est inconnu.

#### 2.V- 5. Résultats Mc Anney, Bardell et Gupta [MCA 84]

Dans cet article Bardell propose certains résultats concernant le test de collages pour une mémoire "plongée" dans un environnement de logique. Du point de vue théorique, cela revient à considérer que l'on peut attribuer n'importe quelle probabilité à chacune des entrées (les résultats présentés correspondant à une probabilité identique pour toutes les entrées). Cet article s'intéresse à la valeur de la qualité de détection en fonction du nombre de test  $L$ , et à la probabilité qu'un vecteur détecte un collage en fonction de  $t$  ceci en faisant varier les différentes probabilités d'entrée.

Ces résultats permettent de vérifier que la longueur de test minimum est obtenue par des probabilités 0,5 sur toutes les entrées.

## 2.VI- CONCLUSIONS

Au vu de ces résultats il nous a semblé intéressant de poursuivre l'étude commencée sur le test aléatoire de mémoire, mais sous d'autres hypothèses de fautes : couplages et PSF, de calculer les longueurs de test nécessaires à la détection de ces fautes et de les comparer aux longueurs proposées par les algorithmes de test déterministe.

Il faut remarquer que les algorithmes déterministes qui sont présentés dans la table 1 présentent des faiblesses assez importantes concernant les hypothèses. D'abord la plupart des méthodes sophistiquées (traitant les PSF) ne concernent que des mémoires dont les mots sont de 1 bit chacun. Ensuite, il est nécessaire de connaître l'adresse physique des différents mots mémoire (afin de définir quels sont ses voisins). Notons que la notion de voisinage avec des mots de b bits est plus délicate à caractériser. Enfin les modes d'adressages modernes (mode page, nibble et ripple mode...) ne sont nullement pris en compte. Il s'ensuit que les fautes couvertes par les algorithmes peuvent n'avoir qu'un assez lointain rapport avec les fautes qui pourront affecter réellement les mémoires.

## Chapitre 3

LONGUEURS DE TEST ALEATOIRE  
POUR LES MODELES CLASSIQUES



### 3.I- INTRODUCTION

Nous présentons dans ce chapitre les résultats obtenus sur le test aléatoire de mémoire sous des hypothèses classiques (collages, 2-couplages et PSF). Nous supposons d'autre part que ces fautes sont simples (ou du moins non interactives).

Dans un premier temps, nous détaillons la modélisation des différentes fautes par des chaînes de Markov, ainsi que les calculs permettant d'obtenir les longueurs de test nécessaires à la détection de chaque type de faute.

Ensuite, nous présentons les résultats concernant les calculs de longueur de test ainsi que l'influence de divers paramètres sur ces dernières.

### 3.II- MODELISATION DES PANNES

#### 3.II- 1. Introduction

Notre étude concerne le test aléatoire de RAM sous des hypothèses classiques (collages, 2-couplages, PSF). Son but est de calculer les longueurs de séquences à appliquer aux mémoires afin de permettre la détection des fautes avec une probabilité donnée. Il nous a donc fallu dans un premier temps modéliser ces fautes, à cette fin nous avons utilisé un outil mathématique : les chaînes de Markov. Ce chapitre présente la modélisation de ces fautes par des processus markoviens.

#### 3.II- 2. Chaînes de Markov

##### 3.II- 2.1. Quelques rappels

Une chaîne de Markov permet de représenter l'évolution d'un système pouvant prendre  $q$  états possibles  $S_1, S_2, \dots, S_q$  au cours du temps, sous certaines conditions. La probabilité que le système soit dans l'état  $S_i$  à la date  $t$  sera appelé  $p_i(t)$ . Les  $p_i(t)$  ( $i = 1, 2, \dots, q$ ) forment l'ensemble des vecteurs d'état décrivant le système pour toutes les dates futures considérées.

On suppose que le passage d'un état  $S_i$  à un état  $S_j$  ne dépend pas de ces deux états. A tout couple  $(S_i, S_j)$  on associe une probabilité  $p_{ij}$  que le système se trouve dans l'état  $S_j$  à la date  $t + 1$  sachant qu'il était dans l'état  $S_i$  à la date  $t$ .

Le système est alors régi par les équations :

$$p_j(t+1) = \sum_{i=0}^q p_i(t) \cdot p_{ij} \quad j = 1, 2, \dots, q$$

ou en utilisant l'écriture matricielle :

$$[P(t+1)] = [P(t)] \cdot [M]$$

où  $M = [p_{ij}]$  et  $[P(t)] = [p_j(t)]$

Ce processus permet donc, à partir d'un état initial  $P(0)$  de connaître les probabilités d'être dans chaque état  $S_i$  à tout instant  $t$ .

### 3.II- 2.2. Application à la modélisation des fautes dans une mémoire

La modélisation des différents défauts correspondant aux hypothèses classiques a été faite par des processus Markoviens.

On appelle erreur une valeur fautive. Par exemple si un bit est collé à 1, il y a une erreur si et seulement si la valeur qui devrait y être écrite est 0. Une erreur est détectée lorsqu'elle apparaît sur une sortie.

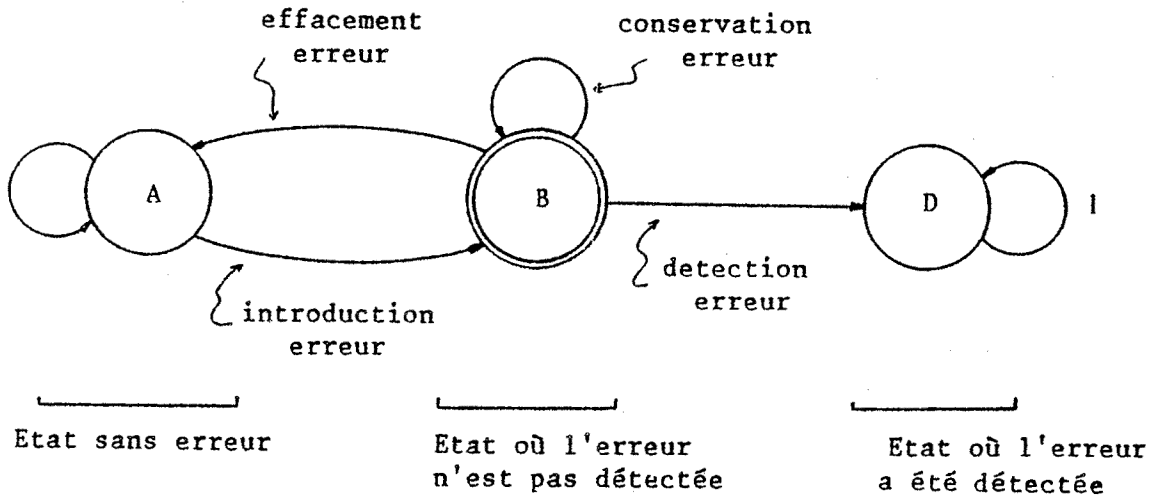
On définit trois types d'états dans une mémoire :

- 1) les états où la mémoire ne contient pas d'erreur et tels qu'aucune erreur n'a encore été détectée.
- 2) les états où la mémoire contient une erreur mais cette erreur n'est pas encore détectée (exemple : le bit  $i$  est collé à 1, on a écrit 0 à l'adresse  $i$  mais on n'a pas encore lu le bit  $i$ ).
- 3) un état tel qu'une erreur a déjà été détectée, il s'agit d'un état dont on

ne sort plus (dit état absorbant).

A partir de ces états, on définit les opérations qui permettent de changer d'état. Ce sont les opérations de lecture ou écriture dans la mémoire.

Pour chaque opération, on calcule la probabilité de passer de l'état  $S_i$  à l'état  $S_j$ . exemple : Probabilité d'écrire 1 à l'adresse a. On peut alors construire le graphe de Markov associé à la faute considérée (cf figure 9).



**Figure 9 : Modélisation d'une exemple de faute**

A partir de ce graphe et de la matrice  $M$  associée, on calcule la probabilité d'être dans l'état D au bout d'un temps  $t$  qui est la probabilité de détection  $1 - Q_D$  de la faute. Réciproquement, on peut calculer le nombre d'étapes nécessaires (ou longueur de séquence à appliquer à la mémoire) pour avoir une qualité de détection  $1 - Q_D$ .

### 3.II- 3. Définition

On présente dans ce paragraphe quelques notations utiles à la modélisation des fautes par graphes Markoviens.

#### 3.II- 3.1. Etats

$i^x$  : la cellule  $i$  devrait être dans l'état  $x$  ( $x = 0$  ou  $1$ ).

$i_y$  : la cellule  $i$  est dans l'état  $y$  (la lecture à l'adresse  $i$  donne la



valeur y)

- $i^x_y$  : la cellule i devrait être dans l'état x mais est dans l'état y  
 $i$  : notation équivalente à  $i^0_0$  ou  $i^1_1$  (cellule dans un état juste)  
 $i_*$  : notation équivalente à  $i^0_1$  ou  $i^1_0$  (cellule dans un état faux)

### 3.II- 3.2. Opérations de changement d'état

- $L_i$  : lecture de la cellule i  
 $E^x_i$  : écriture de la valeur x dans la cellule i  
 $E_i$  : écriture dans la cellule i  
 $A_i$  : adressage de la cellule i ( $A_i = L_i$  ou  $E_i$ ).

$l_i, e^x_i, e_i, a_i$  désignent les probabilités de chacune de ces opérations.

### 3.II- 3.3. Probabilités de changement d'états

On considère des mémoires de  $n = 2^m$  mots de 1 bit. On suppose qu'il y a équiprobabilité d'adressage de chaque cellule de la mémoire.

On peut donc calculer  $a_i =$  probabilité d'adresser la cellule i.

$$a_i = \Pr[A_i] = \frac{1}{n} = \frac{1}{2^m}$$

On suppose d'autre part qu'on a équiprobabilité de lecture ou écriture dans la mémoire. On en déduit :

$e_i =$  probabilité d'écrire à l'adresse i  
 $=$  probabilité d'adresser la cellule i  $\times$  probabilité d'écrire

$$e_i = \Pr[E_i] = \frac{1}{n} \times \frac{1}{2} \quad \text{donc} \quad e_i = \frac{1}{2n} = \frac{1}{2^{m+1}}$$

De la même façon on obtient :

$l_i =$  probabilité de lire la cellule i

soit 
$$l_i = \Pr[L_i] = \frac{1}{2n} = \frac{1}{2^{m+1}}$$

Si on choisit aussi une équiprobabilité d'écriture des valeurs 0 ou 1 on a :

$e_i^x$  = probabilité d'écrire la valeur  $x$  à l'adresse  $i$   
 = probabilité d'écrire à l'adresse  $i$  x probabilité d'écrire la valeur  $x$ .

$$e_i^x = \Pr [E_i^x] = \frac{1}{2n} \times \frac{1}{2}$$

$$e_i^x = \frac{1}{4n} = \frac{1}{2^{m+2}}$$

Remarque : Tous les calculs présentés ici supposent les équiprobabilités qui viennent d'être mentionnées. Néanmoins, la méthode d'évaluation de la longueur de test qui sera présentée, pourrait être utilisée avec n'importe quelle autre distribution des probabilités.

### 3.II- 4. Modélisation du collage d'un point mémoire

A l'aide des notations précédentes nous allons pouvoir définir pour chacune des hypothèses de fautes classiques (collage, 2-couplages, PSF) un modèle de Markov qui permettra de calculer la longueur de la séquence nécessaire à cette détection avec une probabilité  $1-Q_D$  donnée.

Dans ce paragraphe, nous allons détailler la modélisation du collage à 1 de la cellule  $i$  (cf figure 10).

Ce modèle a déjà été utilisé pour le collage d'un point mémoire dans [MCA 84] et [Dav 85].

A partir de l'état sans erreur ( $i_1^1$ ) on reste dans un état sans erreur si on effectue n'importe quelle opération sauf écriture de 0 à l'adresse  $i$ .

On introduit une erreur en écrivant 0 à l'adresse  $i$ , d'où passage dans un état où il y a présence d'erreur ( $i_1^0$ ), on peut alors :

- détecter l'erreur en lisant la cellule  $i$  (passage à l'état D)
- effacer l'erreur en écrivant 1 dans  $i$  (retour à  $i_1^1$ )
- conserver l'erreur en faisant n'importe quelle opération qui ne soit ni une lecture de  $i$ , ni une écriture de 1 dans  $i$ .

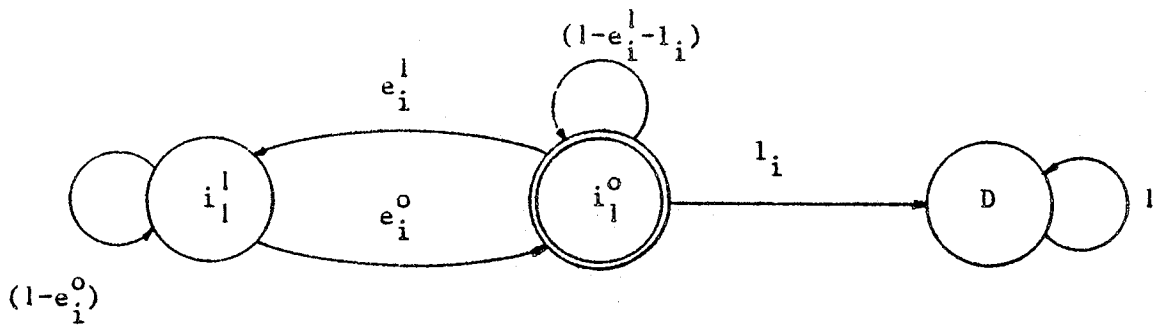


Figure 10 : Modélisation du collage à 1 de la cellule  $i$

La matrice associée s'écrit :

$$M = \begin{array}{c} \begin{array}{ccc} i_1^1 & i_1^0 & D \end{array} \\ \begin{array}{c} i_1^1 \\ i_1^0 \\ D \end{array} \left[ \begin{array}{ccc} 1-e_i^0 & e_i^0 & 0 \\ e_i^1 & 1-e_i^1-1_i & 1_i \\ 0 & 0 & 1 \end{array} \right] \end{array}$$

A partir d'un état initial  $P(0)$  on calcule le nombre d'étapes (ou longueur de séquence) permettant d'atteindre  $D$  avec une probabilité  $1-Q_D$ .

On remarque que ce nombre d'étapes dépend de  $P(0)$ . En effet, il sera plus difficile de détecter un collage à 1 dans une mémoire où toutes les cellules sont initialisées à 1 que dans une mémoire initialisée à 0 [The 78]. Ce point sera développé au paragraphe suivant.

## 3.II- 5. Modélisation des 2-couplages

Il existe deux types de 2-couplage entre cellules.

Le 2-couplage d'inversion : la transition  $\uparrow$  (ou  $\downarrow$ ) de la cellule  $j$  change l'état de la cellule  $i$  quel que soit l'état de cette dernière avant la transition (exemple :  $\uparrow j \Rightarrow \downarrow i$ ).

Le 2-couplage idempotent : la transition  $\uparrow$  (ou  $\downarrow$ ) de la cellule  $j$  change l'état de la cellule  $i$  seulement quand cette dernière est dans un certain état  $x$  (exemple :  $\uparrow j \Rightarrow \uparrow i$ ).

La figure 11 propose une modélisation du 2-couplage d'inversion.

De l'état  $j_0i$  sans erreur on ne sort qu'en écrivant 1 à l'adresse  $j$ . On introduit alors une erreur et on passe à l'état  $j_1i_*$ .

De cet état  $j_1i_*$  on peut

- 1) détecter l'erreur en lisant  $i$  (passage à  $D$ )
- 2) effacer l'erreur en écrivant dans  $i$  (passage à  $j_1i$ )
- 3) conserver l'erreur

soit en changeant d'état : l'écriture de 0 dans  $j$  conserve l'erreur mais amène à l'état  $j_0i_*$ .

soit en restant dans le même état : toute opération qui ne soit ni un adressage de  $i$  ni une écriture de 0 dans  $j$ .

De l'état  $j_0i_*$  on peut

- 1) effacer l'erreur en passant à  $j_0i$  par écriture dans  $i$  ou à  $j_1i$  en écrivant 1 dans  $j$ .
- 2) détecter l'erreur par lecture de  $i$  (passage à  $D$ ).
- 3) conserver l'erreur en restant dans ce même état : toute opération sauf adressage de  $i$  ou écriture de 1 dans  $j$ .

Enfin de  $j_1i$  on ne peut que

- 1) passer dans un état sans erreur ( $j_0i$ ) par écriture de 0 dans  $j$ .

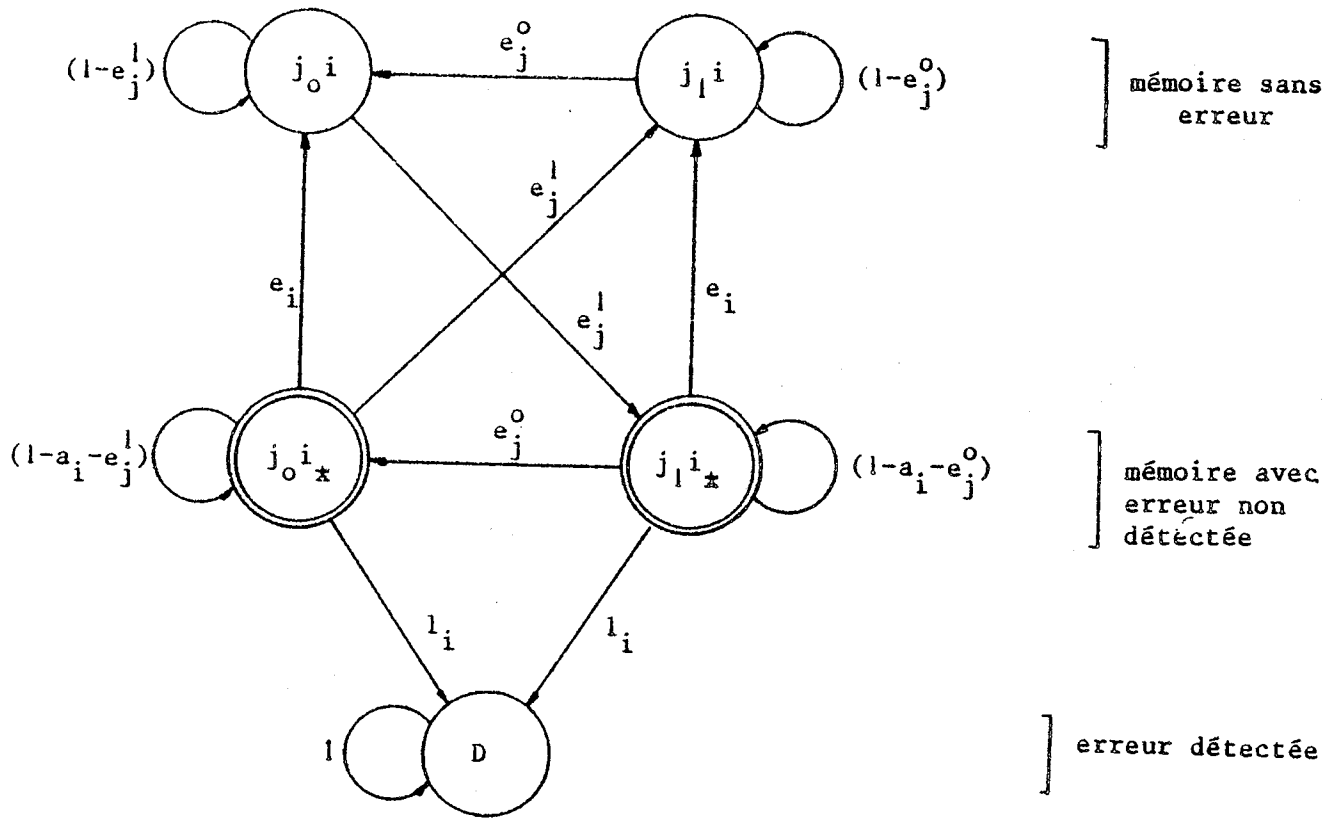


Figure 11 : Modélisation du 2-couplage d'inversion ( $\uparrow j \rightarrow \downarrow i$ )

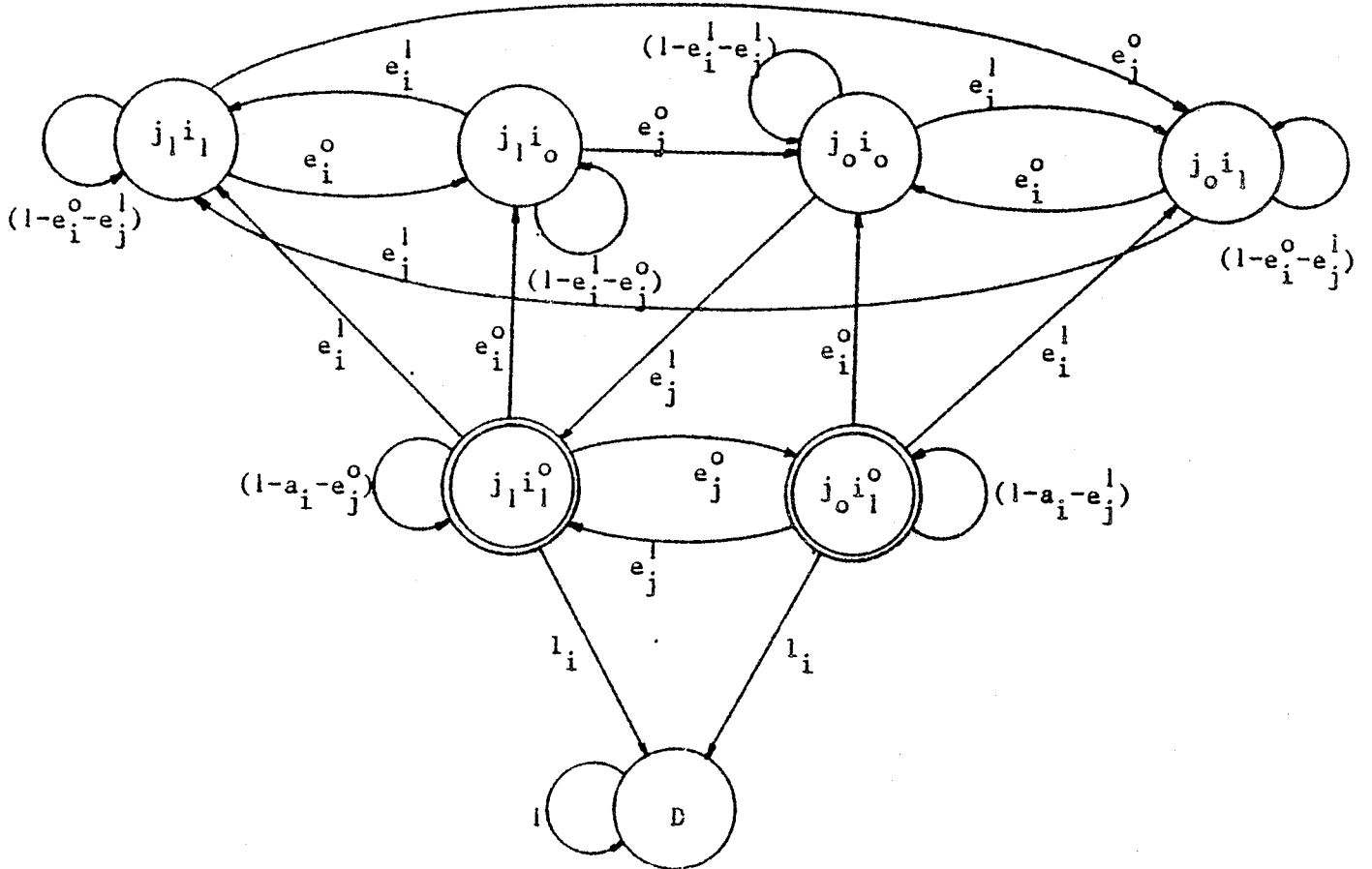


Figure 12 : Modélisation du 2-couplage idempotent ( $\uparrow j \rightarrow \uparrow i$ )

2) rester dans cet état pour toute autre opération.

On modélise de même le 2-couplage idempotent, ceci est représenté à la figure 12.

### 3.II- 6. Modélisation des PSP (Pattern Sensitive Faults)

Pour les PSP, il y a possibilité d'erreur uniquement pour une configuration donnée de la mémoire.

Exemple  $\uparrow j \Rightarrow \uparrow i$  si  $\cdot = k_1 k_2 k_3 = 010$

Il faut donc définir de nouveaux états :

- ceux dans lesquels on peut introduire une erreur. Pour l'exemple choisi c'est quand  $k_1 k_2 k_3 = 010$ .

- ceux dans lesquels on ne peut introduire une erreur.

ex :  $k_1 k_2 k_3 = 111, 000, 100, \dots$  (sauf 010)

et les possibilités de passage d'un état à un autre.

On traite les PSP à  $v$  cellules : on considère qu'il y a PSP si  $k_1 k_2 \dots k_{v-1}$  a une valeur donnée (ex : 010 ... 1).

Pour simplifier l'exposé on suppose qu'il y a possibilité d'erreur pour  $k_1 k_2 \dots k_{v-1} = 11 \dots 1$  et on pose dans ce cas  $K = 1$ , pour toute autre combinaison des  $k_i$  on pose  $K = 0$ .

On cherche alors les probabilités de passage de  $K = 0$  à  $K = 1$  et réciproquement.

#### 3.II- 6.1. Cas où l'on a deux cellules influentes ( $K = k_1 k_2$ )

On notera  $\Pr(E_K^0 / K = 1)$  la probabilité pour que  $K$  prenne la valeur 0 sachant que  $K = 1$ .

$$\Pr[E_K^0 / K = 1] = \Pr[E_{k_1}^0 / k_1 = 1] + \Pr[E_{k_2}^0 / k_2 = 1]$$

car l'écriture simultanée dans  $k_1 k_2$  est impossible ; on peut donc ajouter les probabilités.

=  $\Pr[E_{k_1}^0] + \Pr[E_{k_2}^0]$  étant donné que la probabilité d'écrire 0 dans  $k_1$  est indépendante de l'état de  $k_1$ .

$$\Pr[E_K^0 / K = 1] = 2 \times \Pr[E_{k_1}^0]$$

Pour ne pas alourdir les notations, on notera

$$e_K^0 = \Pr[E_K^0 / K = 1] \quad \text{ici } e_K^0 = 2 \times e_{k_1}^0$$

De même on note  $\Pr[E_K^1 / K = 0]$  la probabilité pour que  $K$  prenne la valeur 1 sachant que  $K = 0$ .

$$\begin{aligned} \Pr[E_K^1 / K = 0] &= \Pr[k_1 k_2 = 01 / K = 0] \times \Pr[E_{k_1}^1] \\ &\quad + \Pr[k_1 k_2 = 10 / K = 0] \times \Pr[E_{k_2}^1] \\ &= 1/3 \Pr[E_{k_1}^1] + 1/3 \Pr[E_{k_2}^1] \end{aligned}$$

$$\Pr[E_K^1 / K = 0] = 2/3 \Pr[E_{k_1}^1]$$

On note aussi  $e_K^1 = \Pr[E_K^1 / K = 1]$

### 3.II- 6.2. Cas où il y a $s$ cellules influentes

On généralise le résultat précédent et on obtient

$$e_K^0 = (v-1) \times e_k^0$$

$$e_K^1 = \frac{v-1}{2^{v-1}-1} e_k^1$$

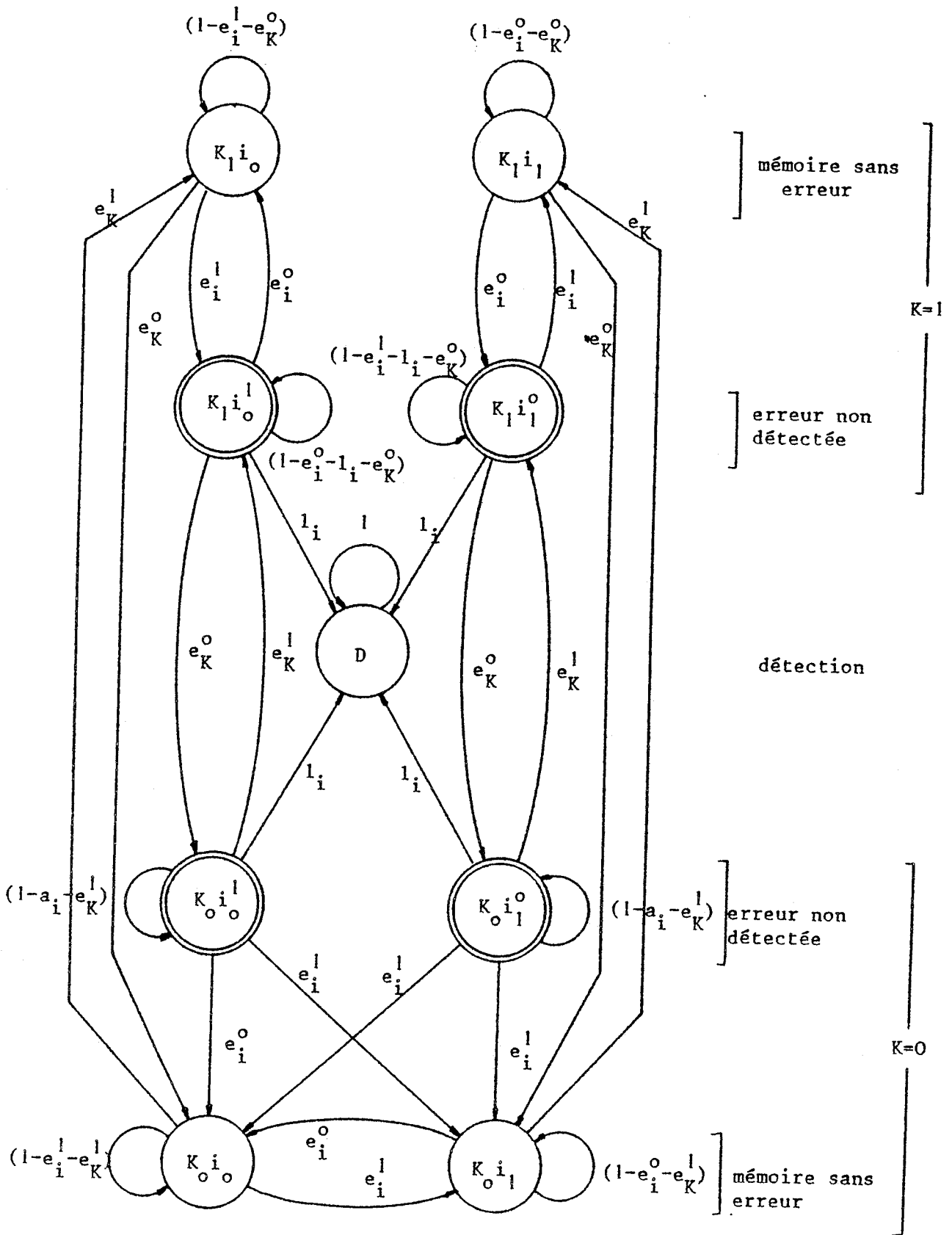
PSP passives

$$e_K^0 = (v-2) e_k^0$$

$$e_K^1 = \frac{v-2}{2^{v-2}-1} e_k^1$$

PSP actives

On peut ainsi construire les graphes de Markov des PSP qui sont donnés dans les figures 13, 14, et 15.



**Figure 13 : Modélisation d'une PSF passive**

(on ne peut écrire dans  $i$  si  $K=1$ )



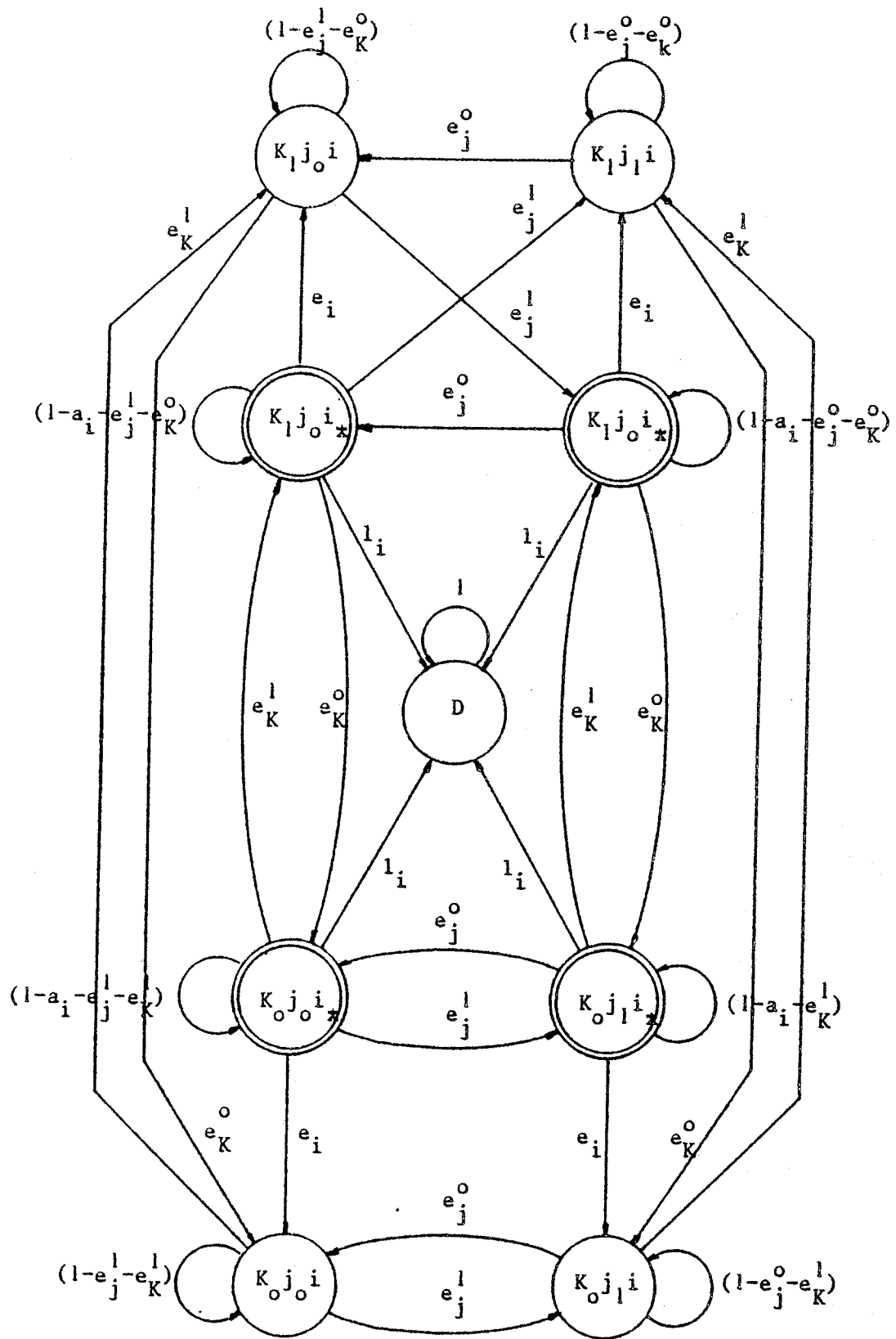


Figure 14 : Modélisation d'une PSF active d'inversion

( $\uparrow j \Rightarrow \uparrow i$  si  $K=1$ )

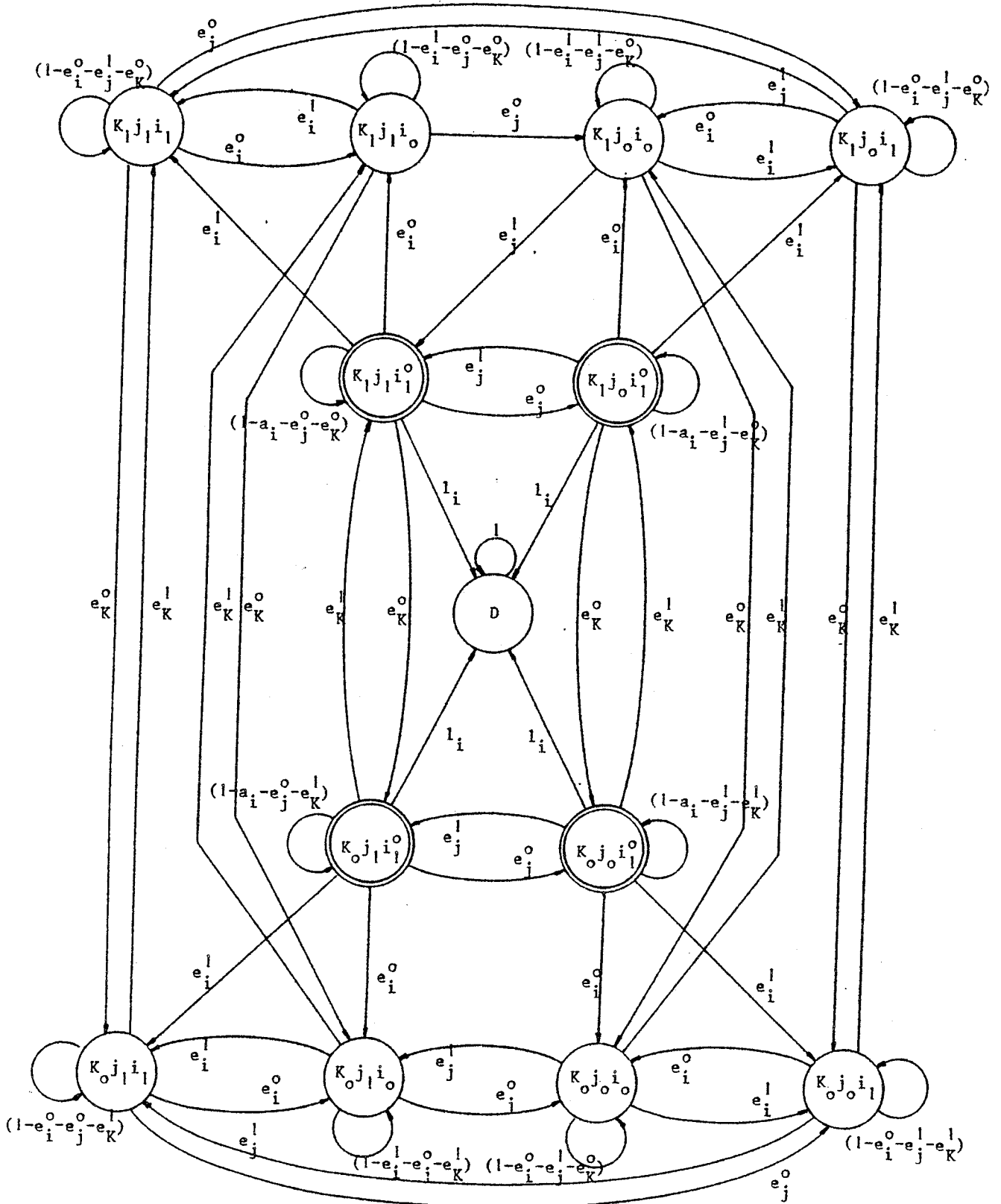


Figure 15 : Modélisation d'une PPSF active idempotente  
 ( $\uparrow j \rightarrow \uparrow i$  si  $K=1$ )

### 3.II- 7. Conclusion

A l'aide de ces modélisations, nous allons pouvoir faire les calculs des longueurs de test à appliquer aux mémoires et étudier l'influence des différents paramètres. Les résultats obtenus font l'objet du paragraphe suivant.

### 3.III- RESULTATS

#### 3.III- 1. Introduction

Afin de pouvoir comparer les résultats du test aléatoire à ceux du test déterministe nous nous sommes intéressés au rapport

$H = L/n$  où  $L$  est la longueur de la séquence à appliquer à la mémoire pour détecter une faute avec une probabilité  $1-Q_D$ , et  $n$  est la capacité de la mémoire ( $n = 2^m$  mots de 1 bits).

On pourra ainsi dire qu'il faut  $Kn$  accès mémoire pour détecter tel type de faute.

Ce chapitre présente les résultats du calcul de  $H$  pour les diverses fautes ainsi que l'influence des divers paramètres.

#### 3.III- 2. Influence de la capacité mémoire

Nous avons cherché à étudier le comportement de  $H$  en fonction de la capacité de la mémoire ( $n = 2^m$ ) ceci avec une probabilité de détection  $1-Q_D$  fixée et pour chaque hypothèse de faute.

La table 2 présente les résultats des différentes expériences effectuées, les calculs sont faits pour  $1-Q_D = 0,999$ .

On constate que pour chaque hypothèse de faute,  $H$  est pratiquement constante. En effet, pour  $m > 4$  (c'est à dire des mémoires de capacité supérieure à 16 bits) entre  $m = 4$  et  $m = 20$  (1M bit) la variation de  $H$  est de l'ordre de 1 pour mille.

## I-a Collages

Initialisations	mémoire de 1 bit			
	m = 0	16 bits m = 4	1 Kbit m = 10	1 Mbit m = 20
favorable	H=39	H=42,25	H=42,40	H=42,43
équiprobable	H=42	H=45,94	H=46,08	H=46,09
défavorable	H=44	H=48,25	H=48,43	H=48,45

I-b 2-Couplage d'inversion ( $\uparrow j \Rightarrow \downarrow i$ )

Initialisations	mémoire de 1 bit			
	m = 1	m = 4	m = 10	m = 20
favorable	H=94,5	H=96	H=96,2	H=96,2
équiprobable	H=98,5	H=100,1	H=100,3	H=100,3
défavorable	H=101,5	H=103,3	H=103,6	H=103,6

I-c 2-couplage idempotent ( $\uparrow j \Rightarrow \uparrow i$ )

Initialisations	mémoire de 1 bit			
	m = 1	m = 4	m = 10	m = 20
favorable	H=202	H=203,2	H=203,4	H=203,4
équiprobable	H=217	H=218,9	H=219,2	H=219,2
défavorable	H=223	H=224,6	H=224,9	H=224,9

## I-d PSF passive (une cellule influente)

Initialisations	mémoire de 1 bit			
	m = 1	m = 4	m = 10	m = 20
favorable	H=100,5	H=102,1	H=102,4	H=102,4
équiprobable	H=108	H=109,6	H=109,8	H=109,8
défavorable	H=113	H=114,7	H=114,9	H=114,9

I-e PSF active d'inversion (2 cellules influentes :  $\uparrow j \Rightarrow \downarrow i$  si  $k = 1$ )

Initialisations	mémoire de 1 bit			
	m = 2	m = 4	m = 10	m = 20
favorable	H=205,3	H=205,9	H=206,2	H=206,2
équiprobable	H=216,5	H=217,2	H=217,4	H=217,4
défavorable	H=224,8	H=225,5	H=225,7	H=225,7

I-f PSF active idempotente (2 cellules influentes :  $\uparrow j \Rightarrow \uparrow i$  si  $k = 1$ )

Initialisations	mémoire de 1 bit			
	m = 2	m = 4	m = 10	m = 20
favorable	H=412,8	H=413,6	H=413,8	H=413,8
équiprobable	H=445,8	H=446,5	H=446,8	H=446,8
défavorable	H=457,8	H=458,4	H=458,7	H=458,7

Table 2 : Influence de m sur k ( $1 - Q_D = 0,999$ )

Ces résultats sont très importants :

- a) la longueur de la séquence à appliquer à une mémoire pour détecter une faute est proportionnelle à la taille de la mémoire (fonction linéaire de  $n$ ).
- b) Le rapport  $H$  dépend du type de la faute.

### 3.III- 3. Influence de l'initialisation

Prenons le cas simple du collage représenté sur la figure 10. L'état initial est  $P(0) = (Pr[i_1^1], Pr[i_1^0], Pr[D])$ .

On peut considérer 3 états initiaux possibles:

- 1) un état favorable  $P(0) = (0, 1, 0)$ , celui qui conduira probablement le plus rapidement à la détection. (cet état est noté  $i_1^1$  dans la table 3).
- 2) un état défavorable  $P(0) = (1, 0, 0)$
- 3) un état équiprobable  $P(0) = (0.5, 0.5, 0)$ .

Si on considère le 2-couplage d'inversion (cf fig. 11) l'état initial est  $P(0) = (Pr[j_0i_1], Pr[j_1i_1], Pr[j_0i_{1*}], Pr[j_1i_{1*}], Pr[D])$ .

On distingue trois classes d'initialisations :

- 1) les favorables avec  $P(0) = (0, 0, 1, 0, 0)$  notée  $j_0i_{1*}$  dans la table 3 et  $P(0) = (0, 0, 0, 1, 0)$  notée  $j_1i_{1*}$   
on peut penser aussi à une initialisation  $P(0) = (0, 0, 0.5, 0.5, 0)$  notée moyenne dans la table 3.2.
- 2) les défavorables avec  $P(0) = (1, 0, 0, 0, 0)$  notée  $j_0i_1$  et  $P(0) = (0, 1, 0, 0, 0)$  notée  $j_1i_1$  et une moyenne  $P(0) = (0.5, 0.5, 0, 0, 0)$
- 3) l'équiprobable : on suppose qu'on a équiprobabilité d'être dans chaque état (sauf D) à l'initialisation  $P(0) = (0.25, 0.25, 0.25, 0.25, 0)$ .

Nous avons appliqué ce type d'initialisation aux graphes de chaque hypothèse de faute.

3-a Collage

Initialisations	Favorable		Equiprobable	Défavorable	
	$i_1^o$			$i_1^l$	
H(collage)	42,43		46,09		48,45

3-b 2-couplage d'inversion

Init.	Favorables			Equi.	Défavorables		
	$j_o i_x$	$j_l i_x$	moy.		$j_o i$	$j_l i$	moy.
H(2-coup.)	96,0	96,4	96,2	100,3	101,1	105,7	103,6

3-c 2-couplage idempotent

Init.	Favorables			Equi.	Défavorables				
	$j_o i_o^o$	$j_l i_l^o$	moy.		$j_o i_o$	$j_o i_l$	$j_l i_o$	$j_l i_l$	moy.
H(2-coup.)	202,0	204,8	203,4	219,2	219,1	225,9	225,9	227,9	224,9

3-d PSF passive (1 cellule influente)

Initialisations	Favorables					Equi.
	$K_l i_o^l$	$K_l i_l^o$	$K_o i_o^l$	$K_o i_l^o$	moy.	
H(PSF passive)	98,8	98,8	105,3	105,3	102,4	109,8

Initialisations	Défavorables				
	$K_l i_o$	$K_l i_l$	$K_o i_o$	$K_o i_l$	moy.
H(PSF passive)	112,5	112,5	117,0	117,0	114,9

**Table 3 : Influence de l'initialisation sur H**

3-e PSF active d'inversion (2 cellules influentes)

Initialisations	<u>Favorables</u>					moy.	<u>Equi.</u>
	$K_{1j_0i^*}$	$K_{1j_1i^*}$	$K_{0j_1i^*}$	$K_{0j_0i^*}$			
H(PSF act. inv.)	205,7	205,7	206,4	206,8	206,2	217,4	

Initialisations	<u>Défavorables</u>				
	$K_{1j_0i}$	$K_{1j_1i}$	$K_{0j_1i}$	$K_{0j_0i}$	moy.
H(PSF act. inv.)	220,0	226,8	226,8	228,8	225,8

3-f PSF active idempotente (2 cellules influentes)

Initialisations	<u>Favorables</u>					<u>Equi.</u>	<u>Défavorables</u>	
	$K_{1j_0i^0}$	$K_{1j_1i^0}$	$K_{0j_1i^0}$	$K_{0j_0i^0}$	moy.		$K_{1j_0i^0}$	$K_{1j_0i^1}$
H(PSF act. ide.)	410,4	414,3	414,3	416,0	413,8	446,8	448,0	458,3

Initialisations	<u>Défavorables</u>						
	$K_{1j_1i^1}$	$K_{1j_1i^0}$	$K_{0j_1i^1}$	$K_{0j_1i^0}$	$K_{0j_0i^0}$	$K_{0j_0i^1}$	moy.
H(PSF act. ide.)	458,3	458,3	461,1	461,1	461,1	462,4	458,7

Table 3 : Influence sur l'initialisation sur H (suite)

Les résultats sont regroupés dans la table 3. Les calculs sont faits pour une mémoire de 1 Mbit, avec  $Q_D$  fixé ( $1-Q_D = 0.999$ ). Sur l'ensemble des résultats, on vérifie que suivant l'initialisation,  $H$  varie sur une plage qui va de  $H_e - 10\%$  dans le cas le plus favorable à  $H_e + 5\%$  dans le cas le plus défavorable, ou  $K_e$  correspond à une initialisation équiprobable pour chaque état sauf  $D$ .

Dans la suite des expériences on pourra donc se limiter à prendre  $H_e$  (initialisation équiprobable) qui est une approximation convenable du fait qu'on ne sait pas si l'initialisation est favorable ou non. Si on désire une approximation très robuste on prendra  $H_e + 5\%$ .

### 3.III- 4. Influence de $Q_D$

Nous nous sommes aussi posé la question de savoir comment variait  $H$  en fonction de  $Q_D$ .

Les résultats sont regroupés dans la table 4. Les calculs sont effectués pour une mémoire de 1 Mbit ( $m = 20$ ) et initialisation équiprobable des états de la chaîne. On remarque que  $H \approx a \log Q_D$  avec  $a$  constante dépendant du type de la faute.

$1-Q_D$	0,9	0,99	0,999	0,9999	0,99999
H(collages)	15	32,69	46,09	61,53	77,17
H(2-coup. inv.)	32,14	66,24	100,3	134,4	168,6
H(2-coup. idemp.)	69,86	144,5	219,2	293,8	368,4
H(PSF passive)	33,70	71,76	109,8	147,8	185,9
H(PSF act. inv.)	67,55	142,5	217,4	292,3	367,3
H(PSF act. idemp.)	141,6	294,2	446,8	599,4	752,0

Table 4 : Influence de  $Q_D$  sur  $H$



## 3.III- 5. Influence du nombre de cellules

Il est évident que pour les PSF, suivant le nombre de cellules influentes, la longueur de la séquence permettant la détection de la faute varie.

Nous donnons dans la table 5 les valeurs obtenues dans nos expériences. Les calculs sont faits pour  $m = 20$ ,  $1-Q_D = 0.999$ , initialisations équiprobables.

On peut noter qu'en fait, rajouter une cellule influente pour un type de faute revient à multiplier la longueur de détection (et donc H) par le facteur 2.

nb de cellules dont état est déterminant	1	2	3	4	5	6	7
H(PSF passive)	109,8	222,7	443,1	878,8	1744	3467	6901
H(PSF Fact. inv.)	217,4	439,5	876,9	1745	3472	6916	13786
H(PSF act. idemp.)	446,8	889,5	1767	3512	6992	13940	27810

Table 5 : Influence du nombre de cellules dans les PSF

## 3.IV- CONCLUSION

Nous pouvons retenir de cette étude que :

- 1) pour une classe d'hypothèses de fautes que nous pouvons noter  $C_1$  la longueur de test nécessaire est proportionnelle au nombre de points mémoire soit  $L = H(C_1) \times n$ , où  $H(C_1)$  est pratiquement une constante, indépendante de  $n$ .
- 2) les constantes relatives aux fautes classiques H(collage), H(2-couplages), H(PSF) sont représentées sur la figure 16 où il apparaît que les collages (respectivement 2-couplage) sont les cas limites des PSF passives (respectivement actives lorsqu'on a 0 cellule influente).

3)  $H$  est une fonction linéaire de  $L_n(Q_D)$ .

Dans toute la suite du mémoire, lorsque rien n'est précisé, on suppose qu'on a pris pour le calcul de  $H$ , une initialisation équiprobable et  $Q_D = 10^{-3}$ .

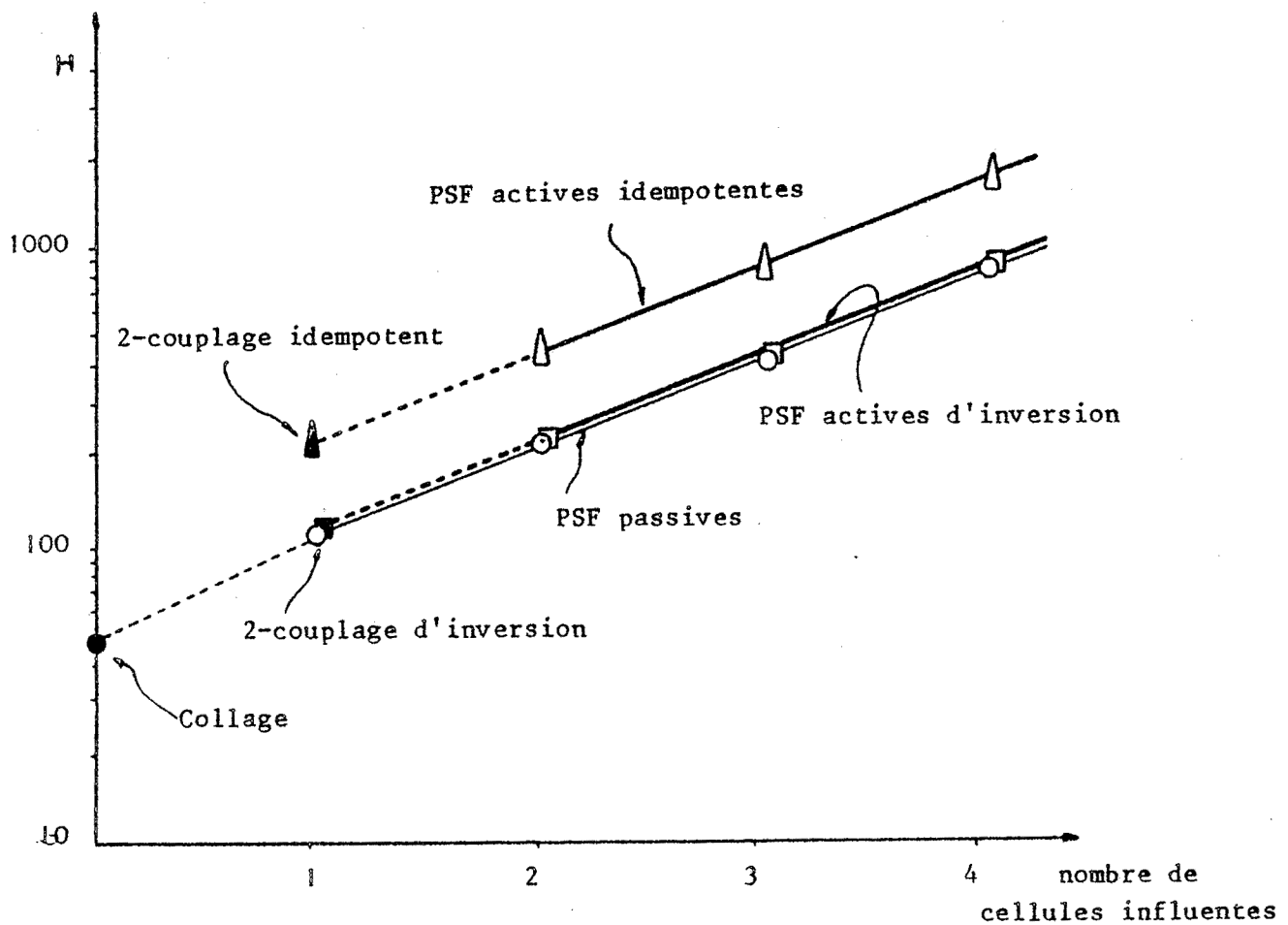


Figure 16 : Valeur de  $H$  en fonction du type de faute



## Chapitre 4

LONGUEURS DE TEST ALEATOIRE  
POUR DES FAUTES MULTIPLES



## 4.1- INTRODUCTION

Nous nous intéressons dans ce chapitre au problème des fautes multiples. Jusqu'à présent nous avons supposé que la mémoire n'était affectée que d'une faute, nous allons étudier l'influence des fautes multiples sur la longueur de détection. Deux fautes sont compatibles si elles peuvent être présentes simultanément dans le même circuit. Par exemple le collage de 1 à 1 et le collage à 0 de 1 ne sont pas compatibles.

Pour une meilleure compréhension, nous introduisons les notations suivantes :

$(f_1 \text{ OU } f_2)$  présence de la faute  $f_1$  ou de la faute  $f_2$  mais pas des deux

$(f_1 \text{ ET } f_2)$  présence simultanée des fautes  $f_1$  et  $f_2$

$$\begin{aligned} \uparrow j \rightarrow \uparrow \downarrow i &= (\uparrow j \rightarrow \uparrow i \text{ OU } \uparrow j \rightarrow \downarrow i) \\ \uparrow \downarrow j \rightarrow \uparrow i &= (\uparrow j \rightarrow \uparrow i \text{ OU } \downarrow j \rightarrow \uparrow i) \\ \uparrow j \rightarrow \downarrow i &= (\uparrow j \rightarrow \uparrow i \text{ ET } \uparrow j \rightarrow \downarrow i) \\ \uparrow \downarrow j \rightarrow \uparrow i &= (\uparrow j \rightarrow \uparrow i \text{ ET } \downarrow j \rightarrow \uparrow i) \end{aligned}$$

Nous pouvons alors définir des notations telles que  $\uparrow \downarrow j \rightarrow \uparrow \downarrow i$  (un des 4 cas possibles) ou  $\uparrow \downarrow j \rightarrow \uparrow i$  (un des 2 cas possibles).

Enfin soient  $F_S$  l'ensemble des collages de point mémoires

$F_V$  l'ensemble des couplages d'inversion

$F_D$  l'ensemble des couplages idempotents

$F_{P_x}$  l'ensemble des PSF passives à  $x$  cellules

$F_{V_x}$  l'ensemble des PSF actives d'inversion à  $x$  cellules

$F_{D_x}$  l'ensemble des PSF actives d'idempotents à  $x$  cellules

Conjecture : Soit  $f_{1,2,\dots,r}$  la faute multiple correspondant à la présence simultanée des fautes compatibles  $f_1, f_2, \dots, f_r$ .

Si  $f_1, f_2, \dots, f_r \in F_S \cup F_V \cup F_D \cup F_{P_2} \cup F_{P_3} \cup \dots \cup F_{V_3} \cup F_{V_4} \cup \dots \cup F_{D_3} \cup F_{D_4} \cup \dots$  alors

$$H(f_{1,2,\dots,r}) \leq \text{Max}(H(f_1), H(f_2), \dots, H(f_r))$$

□

Soit  $C$  une mémoire RAM et  $Z$  l'ensemble des séquences de longueur  $L$  qu'on peut lui appliquer. Si  $C$  possède  $m$  lignes d'adresse, 1 ligne de donnée et une ligne de lecture écriture, alors  $Z$  contient  $2^{(m+2)L}$  séquences.

$Z_u$  (avec  $Z_u \subset Z$ ) est l'ensemble des séquences de longueur  $L$  qui testent la faute  $f_u$ , pour un état initial  $q$ . Soit  $s$  une séquence aléatoire de  $Z$ , alors  $P_r(s \text{ teste } f_u / \text{état initial } q) = P_r(s \in Z_u / s \in Z)$ . Etant donné une incertitude de détection  $Q_D$  et une initialisation équiprobable (Eq. init)  $L_1$  et  $L_2$  sont les longueurs de test nécessaires à la détection de  $f_1$  et  $f_2$  respectivement.

Il est clair que  $L_1 \leq L_2$  si et seulement si  $P_r(s \text{ teste } f_1 / \text{Eq. init}) \geq P_r(s \text{ teste } f_2 / \text{Eq. unit})$  quelque soit la longueur de la séquence aléatoire  $s$ .  
Donc

$H(f_1) \leq H(f_2)$  si et seulement si  $P_r(s \text{ teste } f_1 / \text{Eq. init}) \geq P_r(s \text{ teste } f_2 / \text{Eq. init})$  (1)

Propriété 1 : Si une des fautes, par exemple  $f_1$ , est un collage alors la conjecture est vrai.

□

En effet, toute séquence  $s$  qui teste  $f_1$  teste  $f_{1,2,\dots,r}$ . De l'équation 1 on tire  $H(f_{1,2,\dots,r}) \leq H(f_1)$ .

Dans le paragraphe suivant nous considérons des cas moins triviaux de fautes doubles.

#### 4.II- FAUTES DOUBLES

Soient  $f_1$  et  $f_2$  deux fautes présentes dans une RAM  $f_1$  est une faute telle que les opérations sur  $j$  ont une influence sur l'état de la cellule  $j$  (on notera  $j \xrightarrow{f_1} i$ ) et  $f_2$  une autre faute d'influence. Les différents cas sont présentés à la figure 17 (le cas  $j \xrightarrow{f_1} i$  et  $j \xrightarrow{f_2} i$  n'est pas présenté car on peut le considérer comme un nouveau type de faute  $j \xrightarrow{f_{1,2}} i$ ). On suppose qu'il n'y a pas itération d'influence (i.e si l'écriture dans  $j$  produit un changement sur  $i$  ce changement ne peut en produire un autre).

Propriété 2 : Pour toutes les fautes  $f_1$  et  $f_2$  le théorème est vrai dans les cas c, d et e de la figure 17.

□

En effet, toute séquence  $s$  qui teste  $f_1$  dans la figure 17 (resp d et d) teste  $f_{1,2}$ . On a bien

$$H(f_{1,2}) \leq H(f_1)$$

Cette propriété 2 est vraie pour n'importe quel type d'influence. On remarquera que si on remplace les cellules  $i, j, k, l$  tel par les ensembles de cellules  $I, J, K, L$  cette propriété 2 est encore vraie. Pour  $K \cap (I \cup J) = \emptyset$ . Nous allons dorénavant analyser les types a et b, seuls cas dans lesquelles les fautes sont réellement interactives.

#### 4.II- 1. $f_1$ et $f_2$ sont des couplages d'inversion

On a  $H(f_1) : H(f_2) = 100$

Type a  $f_{1,2} = (\uparrow j \Rightarrow \downarrow i \text{ et } \uparrow i \Rightarrow \downarrow j)$ . Ce cas est le seul (toute autre faute double est obtenue par symétrie). Le calcul nous donne  $H(f_{1,2}) = 48 < H(f_1) = H(f_2) = 100$ .

Type b :  $f_{1,2} = (\uparrow i \Rightarrow \downarrow j \text{ et } \uparrow k \Rightarrow \downarrow j)$ , seul cas (et les symétriques). On obtient  $H(f_{1,2}) = 66 < H(f_1) = H(f_2) = 100$ .

#### 4.II- 2. $f_1$ et $f_2$ sont des couplages idempotents

On a alors  $H(f_1) = H(f_2) = 219$

Type a 4 cas sont possibles (et leurs symétriques)

$$f_{1,2} = (\uparrow j \Rightarrow \uparrow i \text{ et } \uparrow i \Rightarrow \uparrow j) \quad H(f_{1,2}) = 110$$

$$f_{1,2} = (\uparrow j \Rightarrow \uparrow i \text{ et } \uparrow i \Rightarrow \downarrow j) \quad H(f_{1,2}) = 124$$

$$f_{1,2} = (\uparrow j \Rightarrow \uparrow i \text{ et } \downarrow i \Rightarrow \uparrow j) \quad H(f_{1,2}) = 124$$

$$f_{1,2} = (\uparrow j \Rightarrow \uparrow i \text{ et } \downarrow i \Rightarrow \downarrow j) \quad H(f_{1,2}) = 76$$

Type b il existe 2 cas (et leurs symétriques)

$f_{1,2} = (\uparrow j \Rightarrow \uparrow i \text{ et } \uparrow k \Rightarrow \uparrow i)$ . Cette faute est trivialement plus facile à détecter que  $f_1$  seule, étant donné que  $f_2$  ne peut en aucun cas effacer une erreur produite par  $f_1$ .



$f_{1,2} = (\uparrow j \Rightarrow \uparrow i \text{ et } \uparrow k \Rightarrow \downarrow i)$ . On obtient  $H(f_{1,2}) = 114$ .

#### 4.II- 3. Fautes mixtes

Nous présentons deux exemples de présence simultanée d'un 2 couplage d'inversion et d'un 2 couplage idempotent. (De telles fautes multiples sont très difficilement prises en compte par des algorithmes déterministes).

Type a par exemple  $f_{1,2} = (\uparrow j \Rightarrow \downarrow i \text{ et } \uparrow i \Rightarrow \uparrow j)$   
 $H(f_{1,2}) = 76 \leq \text{Max}(100, 219)$ .

Type b Par exemple  $f_{1,2} = (\uparrow j \Rightarrow \uparrow i \text{ et } \uparrow k \Rightarrow \uparrow i)$   
 $H(f_{1,2}) = 81 < \text{Max}(100, 219)$

#### 4.III COMMENTAIRES SUR LES RESULTATS

##### 4.III- 1. Probabilités d'avoir une valeur fautive dans une cellule

Considérons la faute  $f_1 = (\uparrow j \Rightarrow \downarrow i)$ . L'état de la cellule  $i$  dépend des deux événements suivants :

- l'écriture dans la cellule  $i$  ( $w_i$ ) dont la probabilité est  $w_i$
- l'écriture de la valeur 1 dans la cellule  $j$  sachant que  $j$  vaut 0.

Nous notons cet événement  $w_j^{1/0}$  et sa probabilité  $w_j^{1/0}$ . La figure 18 représente une chaîne de Markov dans laquelle  $q_x$  est tel que  $x$  événement  $w_j^{1/0}$

sont survenus depuis le dernier événement  $w_i$ . La cellule  $i$  contient une erreur si et seulement si  $x$  est un nombre impair.

On note  $y = w_i / (w_i + w_j^{1/0})$ . Etant donné que  $w_i = 1/2n$  et  $w_j^{1/0} = 1/8n$  on obtient

$$y = 0.2. \text{ D'où}$$

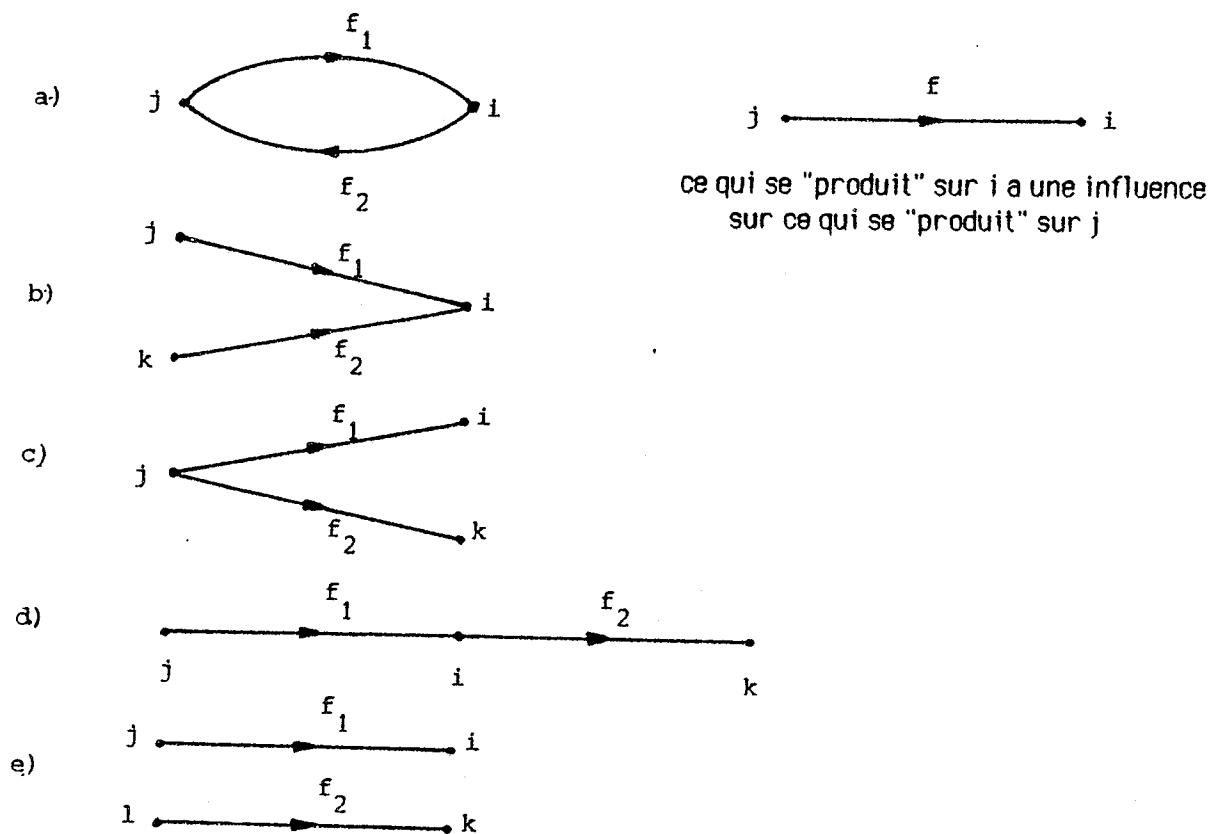
$$P_r [q_0] = 1-y = 0.80$$

$$P_r [q_1] = (1-y)y = 0.16$$

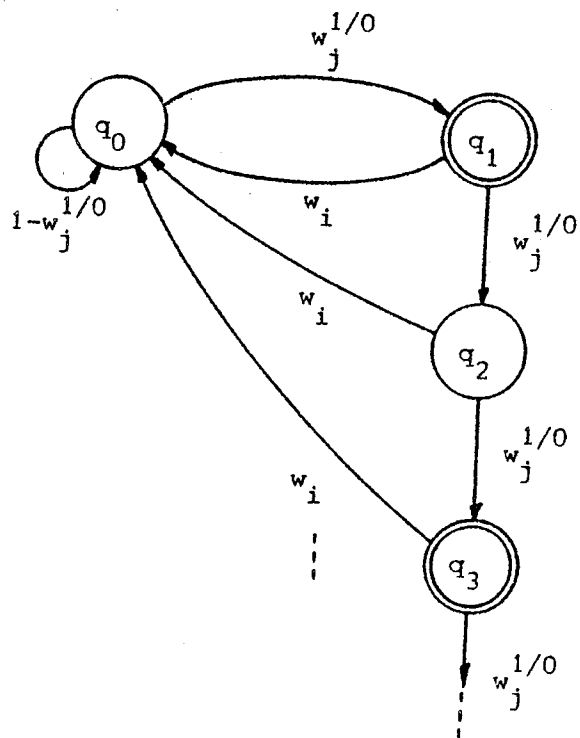
$$P_r [q_2] = (1-y)y^2 = 0,032 \text{ etc...}$$

$$P_r [i \text{ avec erreur} / f_1] = P_r [q_1] + P_r [q_3] + \dots = (1-y)(y+y^3+y^5).$$

$$\text{On obtient } P_r \left[ i \text{ avec erreur} / f_1 \right] = \frac{y}{1+y} = 0,167.$$



**Figure 17 : Différents types influence double**



**Figure 18 : Etat de la cellule  $i$  pour  $j \uparrow \Rightarrow \downarrow i$**

Considérons la faute double  $f_{1,2} = (j \Rightarrow i \text{ et } k \Rightarrow i)$ . On peut utiliser la même chaîne de Markov ( $w_j^{1/0} + w_k^{1/0}$ ) jouant le rôle de  $w_j^{1/0}$ . On obtient  $\gamma = 0,33$  et  $P_r [i \text{ avec erreur} / f_{1,2}] = 0,25$ .

Il faut faire attention  $P_r [i \text{ avec erreur } f_1]$  ne permet pas de calculer  $H(f)$ . Néanmoins quand l'évènement lecture de  $i$  ( $R_1$  survient, nous avons une meilleure chance de détection pour une des plus grande valeur de  $P_r [i \text{ avec erreur}]$ .

On peut s'attendre alors à ce que  $H(f_{1,2}) / H(f_1)$  ait une valeur qui soit de l'ordre de  $P_r [i \text{ avec erreur} / f_1] / P_r [i \text{ avec erreur} / f_{1,2}] = 2/3$ . Ceci est vérifié puisque nous avons obtenu  $H(f_{1,2}) = 66$  et  $H(f_1) = 100$ .

Considérons les types e et c de la figure 18 où  $f_1$  et  $f_2$  sont toutes deux des influences idempotentes : pour le type e, les détections relatives à  $f_1$

et  $f_2$  sont indépendantes, on peut s'attendre à ce que  $H(f_{1,2}) \approx H(f_1) / 2$ . On obtient  $H(f_{1,2}) = 49$ . Pour le type c, les deux influences s'ajoutent bien que non indépendantes, on peut s'attendre à ce que  $H(f_1) / 2 < H(f_{1,2}) < H(f_1)$  on obtient  $H(f_{1,2}) = 68$ .

(On peut montrer que  $P_r [i \text{ ou } k \text{ avec erreur} / f_{1,2}] = 0,25$ ).

#### 4.III- 2. PROBABILITE POUR UN BIT D'AVOIR UNE VALEUR DONNEE

Pour une mémoire sans fautes la  $P_r [i = 1]$  est 0.5 étant donné les propriétés de la séquence de test. La présence d'une faute peut modifier cette probabilité. Grâce à une chaîne de Markov telle que chaque état représente la valeur réelle des cellules  $j$  et  $i$ , on peut montrer que pour  $f_1 = (\uparrow j \Rightarrow \uparrow i)$  on a  $P_r [i = 1 / f_1] = 0,5 = P_r [i = 1 / \text{pas de faute}]$  et pour  $f_3 = (\uparrow j \Rightarrow \uparrow i)$   $P_r [i = 1 / f_3] = 0,61 > P_r [i = 1 / \text{pas de faute}]$ .

Cette probabilité peut modifier ( $f_3$ ) ou non ( $f_1$ ) les effets d'une autre faute  $\uparrow i \Rightarrow X$  ou  $\downarrow i \Rightarrow Y$ .

Considérons  $f_{1,2} = (\uparrow j \Rightarrow \uparrow i)$  et  $\downarrow i \Rightarrow \downarrow j$ ). La faute  $f_1$  augmente  $P_r [i = 1]$ , donc augmente  $P_r [w_1^{0/1} / w_1^0]$ . Donc  $f_1$  augmente la probabilité d'avoir un effet de  $f_2$  et réciproquement. Ceci explique la petite valeur de  $K$  obtenue au paragraphe 4.II-2.

Considérons le type d de la figure 17 où  $f_1$  et  $f_2$  sont toutes deux des couplages d'inversion. Bien que  $f_1$  et  $f_2$  ne soient pas indépendantes  $P_r [i = 1 / f_1] = 0,5$  donc  $P_r [k \text{ avec erreur} / f_{1,2}] = P_r [k \text{ avec erreur} / f_2]$ . Ceci est bien confirmé par le fait que  $K(f_{1,2}) = 49$  (comme pour le type e).

#### 4.IV AUTRES FAUTES MULTIPLES

Nous n'avons pas encore considéré la faute  $f_4 = (\downarrow j \Rightarrow \uparrow i)$ . Cette faute  $f_4$  a évidemment une plus grande chance de produire une valeur fautive dans  $i$  que la faute  $f_3 = (\uparrow j \Rightarrow \uparrow i)$ . On peut donc supposer que  $H(f_4) < H(f_3)$ . On obtient  $H(f_4) = 113$ . On peut comparer  $H(f_4) / H(f_3) = 113/219 = 0,52$  à  $P_r [i \text{ avec erreur} / f_3] / P_r [i \text{ avec erreur} / f_4] = (1/9) / (1/5) = 0,56$ .

Si on prend  $f_5 = (\downarrow k \Rightarrow \uparrow i)$ , on peut supposer que  $H(f_{4,5}) < H(f_4)$ , on obtient bien  $H(f_{4,5}) = 77$ .

On a donc examiné toutes les fautes doubles possibles du type 2-couplage et elles sont toutes plus faciles à détecter que le 2-couplage simple le plus difficile à détecter.

Si on considère les fautes doubles telles que une au moins d'entre elles est une PSF, on peut s'attendre aux mêmes résultats. Pour les fautes multiples contenant plus de trois fautes simples, nous pensons que la conjecture reste vraie. Quand une séquence  $s_1$ , teste  $f_1$  on n'est pas sûr (d'un point de vue déterministe) qu'elle teste  $f_{1,2}, \dots, r$ . Mais d'autres séquences  $s_2, s_3, \dots$ , peuvent tester  $f_{1,2}, \dots, r$  alors qu'elles ne testent pas  $f_1$ .

La probabilité de tester  $f_{1,2}, \dots, r$  par une séquence aléatoire  $s$  peut être plus importante que la probabilité de tester  $f_1$ . On se rappellera du cas de la faute double  $f_{1,2} = (\uparrow j \Rightarrow \downarrow i \text{ et } \uparrow k \Rightarrow \downarrow i)$   $f_2$  peut effacer une erreur due à  $f_1$ , mais la probabilité de cet effet est moins importante que la probabilité que  $f_2$  produise une erreur sur  $i$ .

#### 4.V COMPARAISON TEST ALÉATOIRE - TEST DÉTERMINISTE

Nous comparons ici les temps de test d'algorithmes déterministes optimaux aux temps de test pour une séquence aléatoire. La table 6 donne des valeurs pour différents modèles de fautes.

Pour les collages, les influences idempotentes et les influences non interactives, les algorithmes déterministes sont plus rapides que les séquences aléatoires.

Mais la comparaison devient plus impressionnante si nous considérons des fautes de type 2-couplages interactives. La longueur de test pour une séquence aléatoire est de  $219 n$  alors que les algorithmes déterministes sont en  $O(n \log n) : (36 n + 24 n \log n)$ .

De même si l'on considère les fautes du type 3-couplage ou les PSF. En effet, pour les 3-couplages, les algorithmes déterministes (connus jusqu'à présent) demande des temps en  $O(n \log n)$ . En revanche, un test aléatoire demande un temps qui est toujours proportionnel à  $n : 447 n (Q_D = 10^{-3})$ . Ceci est dû au fait que les calculs sont fait indépendamment de la situation relative des cellules du domaine. Il est évident que cette comparaison serait d'autant plus impressionnante si on considérait les  $V$ -couplages (avec  $V > 3$ ) (mais personne n'a essayé de calculer les longueurs d'un tel test déterministe).

En fait, pour être capable d'utiliser des tests déterministes pour ce type de fautes, il faut considérer un voisinage, i.e connaître la structure de la mémoire (i.e la correspondance, adresse physique, adresse logique). On voit là une des puissances du test aléatoire. Si l'on tient compte des résultats numériques obtenus pour les fautes doubles et si on suppose que la conjecture du IV-1 est vrai pour toute faute multiple, il est intéressant de noter que la longueur de test décroît quand la multiplicité des fautes augmente alors que la difficulté d'obtenir un algorithme déterministe croît dramatiquement.

Du point de vue de l'utilisateur, ces résultats peuvent être utilisés directement. Supposons que quelqu'un utilise un des  $11 n, 14 n, 15 n, 17 n, 36 n$  algorithmes de test optimaux, il pourra aussi bien utiliser ces algorithmes ou choisir d'utiliser le test aléatoire qui avec une longueur non prohibitive ( $200 n$ ) peut être très facilement mis en oeuvre. Il est de noter qu'il vaut beaucoup mieux utiliser le test aléatoire avec une longueur de test raisonnable (quelques centaines de  $n$ ) qu'un très long et inefficace algorithme déterministe tel que GALPAT, en  $4 n^2$  et dont la couverture n'est pas connue. Utiliser une bonne longueur de test aléatoire permet de couvrir un nombre important de fautes bien définies .... et un ensemble de fautes inconnues. Par exemple, il est presque certain que le test aléatoire couvre les influences par lecture [Pap 84] (les

influences ici sont dues à l'écriture) alors qu'en test déterministe de nouveaux algorithmes seraient à définir.

Fautes		LONGUEURS DE TEST			
		Algorithmes déterministes	test aléatoire		
			$Q_D=10^{-2}$	$Q_D=10^{-3}$	$Q=10^{-4}$
n mots de 1 bit	• collages	4n	33n	46n	62n
	• infl. idempotent	15n	145n	219n	294n
	• infl. non interactive	36n	145n	219n	294n
	• infl. interactive	$36n + 24n \log n$	145n	219n	294n
	• PSF active, V = 3 à voisinage	28n	294n	477n	599n
	• PSF active, V = 3 n'importe où	$n + 32n \log n$	294n	477n	599n
	• PSF active, V = 5 à voisinage	195n	1200n	1805n	2410n
	• PSF active, V = 5 n'importe où	?	1200n	1805n	2410n
	• infl. inv. plus infl. idempotent	?	145n	219n	294n
	• PSF active, V = 3 plus infl. inv.	?	294n	447n	599n
n mots de 1 bit	• 2-couplages	?	145n	219n	294n
	• PSF active, V = 5 n'importe où	?	1230n	1845n	2465n

**Table 6 : Longueur de test déterministe et  
aléatoire pour quelques fautes**

Remarque : La deuxième partie de la table 6 donne des résultats expliqués au chapitre 5.



## Chapitre 5

# NOUVELLES HYPOTHESES DE FAUTES





## 5.I- INTRODUCTION

Nous nous intéressons ici à des nouvelles hypothèses de fautes. Toutes les hypothèses trouvées dans la littérature sont établies pour des RAM statiques à mots de 1 bit. Nous avons d'une part étendu ce hypothèses aux RAM à mots de  $b$  bits ( $b > 1$ ) et nous avons été amené à en définir de nouvelles. Nous avons aussi déterminé l'influence de type d'adressages spécifiques tels que le mode page et le nibble mode sur les longueurs de détection. Enfin, certaines fautes spécifiques aux RAM dynamiques ont été introduites et modélisées.

Dans ce chapitre, nous donnons d'abord l'ensemble des nouvelles hypothèses introduites puis leur modélisation ainsi que le calcul des longueurs de test correspondantes.

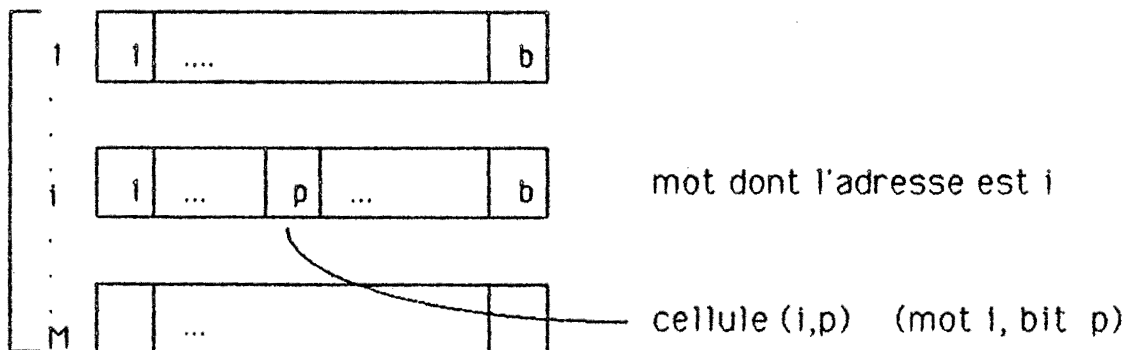
## 5.II- NOUVELLES HYPOTHESES DE PANNES

### 5.II- 1. Mémoires à mots de $b$ bits

Dans ce paragraphe, nous considérons des mémoires à  $M$  mots de  $b$  bits. Nous essayons de déterminer ce que deviennent les hypothèses classiques telles que collages, couplages et PSF et de conclure quant à leur signification pour ce modèle de mémoires.

La mémoire sera donc composé de :

$M$  mots



5.II- 1.1. Collages

## a) Collage simple

Un bit  $p$  du mot  $i$  est collé à la valeur  $\alpha$  ; on notera  $(i,p)/\alpha$

## b) Collage multiple

- plusieurs bits d'un même mot sont collés

ex :  $[(i,p)/\alpha, (i,p)/\beta, \dots]$ .

- Plusieurs bits de mots différents sont collés

ex :  $[(i,p)/\alpha, (j,q)/\beta, \dots]$ .

5.II- 1.2. Couplages

## a) Idempotence

On rappelle qu'il y a influence d'idempotence entre les cellules  $a$  et  $b$  si une transition ( $\uparrow$  ou  $\downarrow$ ) de  $b$  provoque un changement de  $a$  quand  $a$  a une valeur donnée  $x$  (ex  $\uparrow b \Rightarrow \uparrow a$ ).

- entre cellules du même mot

ex :  $\uparrow (i,p) \Rightarrow \uparrow (i,q)$  cette faute est sans réalité physique puisque la valeur est écrite simultanément dans  $(i,p)$  et  $(i,q)$  (cellules du même mot).

- entre cellules de mots différents

ex :  $\uparrow (j,q) \Rightarrow \uparrow (i,p)$ . C'est le même type de faute que l'on rencontrait dans les RAM de mots de 1 bit.

## b) Inversion

Il y a influence d'inversion entre les cellules  $a$  et  $b$  si une transition de  $b$  ( $\uparrow$  ou  $\downarrow$ ) provoque un changement de  $a$  quelle que soit la valeur dans  $a$ .

- entre cellules d'un même mot

Pour les mêmes raisons que celles de l'idempotence, cette faute n'existe pas.

- entre cellules de mots différents

ex :  $\uparrow(j,q) \Rightarrow \downarrow(i,p)$ . Ici aussi on rencontre le même principe de fautes que celui des RAM à 1 bit.

c) Couplage de types ET, OU

Les influences d'idempotence et d'inversion entre 2 bits d'un même mot n'ont pas de signification comme nous l'avons vu. On peut en revanche considérer un autre type de couplages entre cellules d'un même mot. Ce couplage serait de type logique : c'est à dire qu'il réaliserait la fonction logique ET ou OU de 2 bits d'un même mot. Ex  $(i,p) = (i,q) = (i,p) + (i,q)$  pour le OU logique des bits  $(i,p)$  et  $(i,q)$ .

### 5.II- 1.3. PSF actives

La cellule  $j$  a une influence (idempotence ou inversion) sur la cellule  $i$  pour une configuration donnée des  $k$  voisines.

On se limitera à parler d'influence dans ce paragraphe, la différence entre inversion et idempotence n'intervenant que lors de la modélisation par graphes markoviens.

$$\text{ex : } \uparrow(j,q) \Rightarrow \uparrow(i,p) \text{ si } K = k_1 k_2 = 11$$

On a vu au paragraphe précédent qu'il fallait que la cellule influente  $(j,q)$  et la cellule influencée  $(i,p)$  n'appartiennent pas au même mot.

D'autre part, les cellules  $k_1 k_2 \dots$  ne doivent pas appartenir au même mot que  $(j,q)$  si on veut que la faute ait un sens. On notera  $(j,-)$  un bit quelconque du mot  $j$ .

La faute considérée devient :

$$\uparrow(j,q) \Rightarrow \uparrow(i,p) \text{ si } k_1 k_2 = 11$$

#### 5.II- 1.4. PSF passives

On ne peut écrire dans une cellule que pour une configuration donnée des voisines.

ex : on ne peut écrire dans  $(i,p)$  si  $k_1 k_2 = 11$ . De même que précédemment on doit avoir  $k_1 k_2 \dots \in (i,-)$ .

#### 5.II- 1.5. Conclusion

On a vu que l'on pouvait dans la majorité des cas étendre les modèles de fautes des RAM à 1 bit aux RAM à b bits. Néanmoins, des restrictions sont nécessaires dans certains cas.

D'autre part, il faut voir que ces modélisations vont nécessiter de nouvelles représentations par graphes de Markov, si dans le concept elles sont similaires, les états ainsi que les probabilités de changement d'état sont très différents.

### 5.II- 2. Les RAM Dynamiques

Nous étudions dans ce paragraphe les problèmes relatifs aux RAM dynamiques.

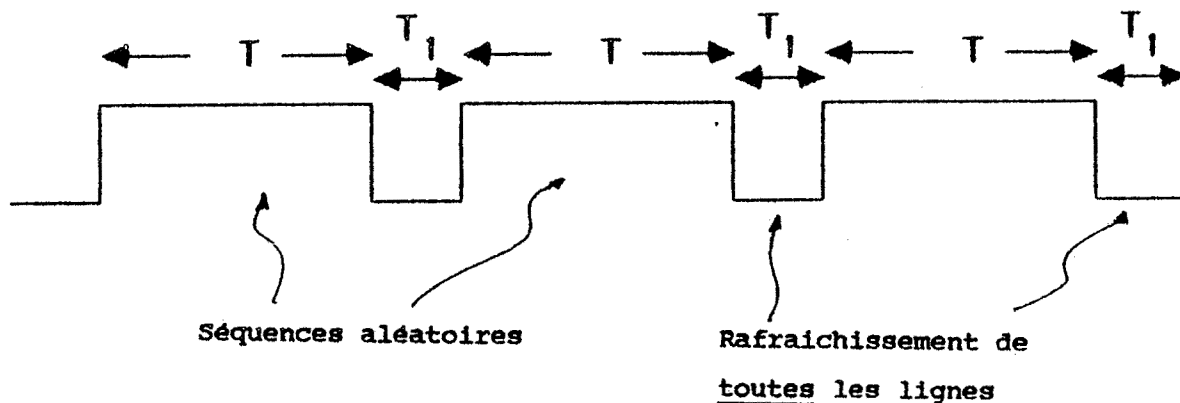
Dans un premier temps, nous transposons les hypothèses classiques des RAM statiques, puis nous étudions les problèmes liés au rafraichissement des cellules.

#### 5.II- 2.1. Hypothèses classiques

On peut considérer que toutes les hypothèses de fautes classiques affectant les RAM statiques sont transposables sans modification aux RAM dynamiques.

La longueur de test aléatoire permettant leur détection peut être la même que celle des RAM statiques multipliées par un facteur S.

En effet, la stratégie de test applicable aux RAM dynamiques est la suivante :



On rafraichit toutes les  $T$  secondes, toutes les cellules de la mémoire pour être sûr d'être dans les normes de fonctionnement de la mémoire. Cette opération dure  $T_1$  secondes.

Si  $L_S$  est la longueur nécessaire à la détection d'une faute  $p$  dans une RAM statique, la longueur  $L_D$  nécessaire à la détection de la même faute dans une mémoire dynamique de même capacité est :

$$L_D = \frac{T_1 + T}{T} L_S$$

### 5.II- 2.2. Nouvelle classe de fautes

Ce paragraphe est consacré aux problèmes spécifiques des RAM dynamiques, dus au rafraichissement de l'information dans les cellules.

Nous avons fait trois hypothèses de fautes :

- Disparition trop rapide de l'information.
- Le rafraichissement ne se fait pas
- Le rafraichissement introduit une erreur.

Nous détaillons ci-après chacune de ces fautes.

a) Disparition trop rapide de l'information

Chaque cellule de la mémoire doit être rafraîchie au moins toutes les T secondes.

Deux types de phénomènes sont à prendre en compte :

Volatilité : En fonctionnement "rapide" de la mémoire : temps de cycle de l'ordre de 250 ns.

On peut supposer qu'une ou plusieurs cellules ne conservent l'information que pendant une durée  $T' < T$ . Il semble raisonnable de penser que si cette faute affecte une cellule, elle affecte au moins toutes les cellules de la même ligne. Donc dans ce cas, on peut avoir perdu l'information avant le rafraîchissement suivant.

Non persistance : En fonctionnement "lent" : temps de cycle de l'ordre de  $10 \mu s$  : l'information en sortie de la mémoire (ligne de données) n'est présente que pendant un temps  $\tau$  inférieur à celui donné par le constructeur. Suivant la période d'observation de la donnée quand on fait une lecture on pourra avoir deux cas. Dans le premier cas où on observe la sortie pendant toute la durée où elle doit être présente, on s'aperçoit de sa disparition à la fin de cette période : au point de vue détection, c'est équivalent à un collage. Dans le deuxième cas, on a pu observer une sortie correcte. Si cette valeur a disparu après l'observation, on n'observera l'erreur que lors d'une prochaine lecture à cette adresse.

b) Le rafraîchissement ne se fait pas

Cette faute équivaut à un mauvais fonctionnement de la lecture ou de l'écriture. Ce type de faute est donc couvert par les hypothèses classiques.

5.II- 3 - Adressage en mode page

Nous ne cherchons à tester que le fonctionnement en mode page de la

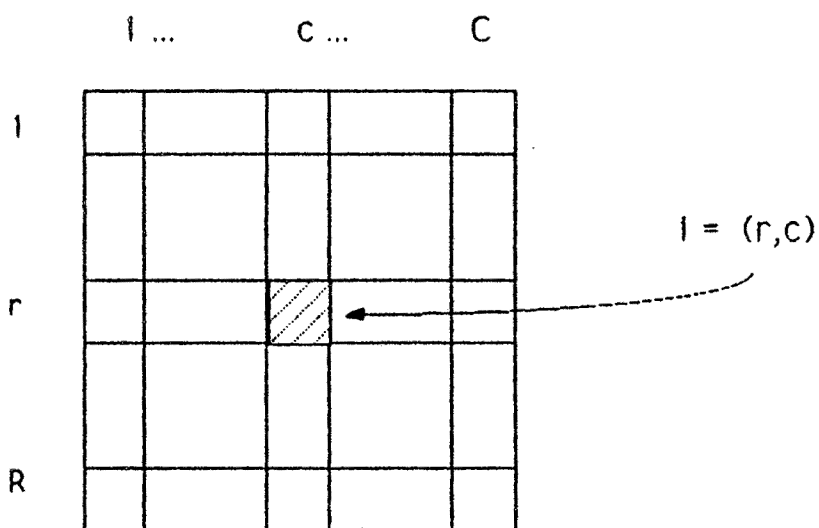
mémoire. Si la mémoire est utilisée en fonctionnement classique, cela revient à tester une RAM dynamique classique.

De ce fait, on peut considérer plusieurs stratégies de test :

- a) tirage aléatoire d'une ligne et de l'opération lecture ou écriture.  
lecture (ou écriture) de toute la ligne.
- b) tirage aléatoire d'une ligne.  
lecture puis écriture sur toute la ligne.
- c) tirage aléatoire d'une ligne  
tirage aléatoire de lecture ou écriture sur  $x$  cellules de la ligne.

Ce sera cette dernière stratégie qui sera retenue, la valeur  $x$  pouvant être fixée, ou aléatoire avec une valeur moyenne fixée (dans le cas où la valeur  $x$  est fixée à 1, on retrouve le cas de l'adressage classique).

La matrice mémoire est modélisée par  $R$  lignes et  $C$  colonnes. Une cellule  $i$  est repérée par  $i = [r, c]$  où  $r$  est l'indice de ligne  
 $c$  est l'indice de colonne





5.II- 3.1. Collages

On peut envisager soit des collages simples

ex :  $[r, c]/\alpha$

soit des collages multiples

Dans ce cas, on différencie le cas où :

\* plusieurs cellules d'une même ligne sont collées

ex :  $[r, c]/\alpha; [r, d]/\beta...$

\* plusieurs cellules de lignes différentes sont collées.

ex :  $[r, c]/\alpha, [l, d]/\beta, ...$

5.II- 3.2. 2-Couplages

On doit considérer les influences d'idempotence et d'inversion. On distinguera :

- le cas où les 2 cellules appartiennent à la même ligne

ex :  $\uparrow [r, c] \Leftrightarrow \uparrow [r, d]$

- le cas où les cellules appartiennent à des lignes différentes

ex :  $\uparrow [r, c] \Rightarrow \uparrow [l, d]$

5.II- 3.3. PSF actives

De même pour les PSF actives (inversion et idempotence) on distingue

- le cas où la cellule influente  $[r, c]$  et la cellule influencée  $[r, d]$  appartiennent à la même ligne.

ex :  $\uparrow [r, c] \Rightarrow \uparrow [r, d]$  si  $k_1 k_2 \dots = 11 \dots$

- le cas où elles appartiennent à des lignes différentes :

ex :  $\uparrow [r, c] \Rightarrow \uparrow [l, d]$  si  $k_1 k_2 \dots = 11 \dots$

On remarque que la position des cellules  $k_1, k_2, \dots$  en mémoire (i.e. appartenance ou non à la ligne considérée) n'a aucune influence.

#### 5.II- 3.4. PSF passive

Ici le fait qu'on soit en mode page ne modifie en rien l'hypothèse de faute. ex : on ne peut écrire dans  $[r, c]$  si  $k_1 k_2 = 11$ .

#### 5.IV- 4. Adressage en nibble mode

On ne teste ici que le fonctionnement en nibble mode. La stratégie est la suivante : on tire aléatoirement une adresse et l'opération lecture ou écriture. Puis on lit ou écrit sur les 4 cellules "voisines".

On note les quatre zones mémoires a, b, c, d.

a	b
c	d

On note une cellule  $(i, a)$  où :  
i correspond au poids fort de l'adresse et  
a correspond au poids faible

On appelle quartet les 4 cellules  $(i, a), (i, b), (i, c), (i, d)$ .

#### 5.II- 4.1. Collages

On peut envisager des collages

- simples : ex  $[i, a] / \alpha$
- multiples :
  - dans le même quartet : ex  $[i, a]/\alpha, [i, b]/\beta$
  - dans des quartets différents : ex  $[i, a]/\alpha, [j, b]/\beta$

#### 5.II- 4.2. 2-Couplages

On doit distinguer le cas où l'influence se fait sur des cellules du

même quartet ou de quartets différents.

$$\begin{array}{l} \text{ex } \uparrow [i, b] \Rightarrow \uparrow [i, a] \\ \uparrow [j, b] \Rightarrow \uparrow [i, a] \end{array}$$

On considère les deux couplages d'idempotence et d'inversion.

#### 5.II- 4.3. PSF actives

a) influence entre cellules du même quartet ; deux cas seront à prendre en compte :

$$\text{- ex : } \uparrow [i, a] \Rightarrow \uparrow [i, b] \text{ si } K = 1 \quad K = [i, c], [j, b]$$

il existe au moins une cellule dont l'état est déterminant qui appartient au quartet.

$$\text{- ex : } \uparrow [i, a] \Rightarrow \uparrow [i, b] \text{ si } K = 1 \quad K = [j, b], [k, c], \dots$$

aucune cellule dont l'état est déterminant n'appartient au quartet.

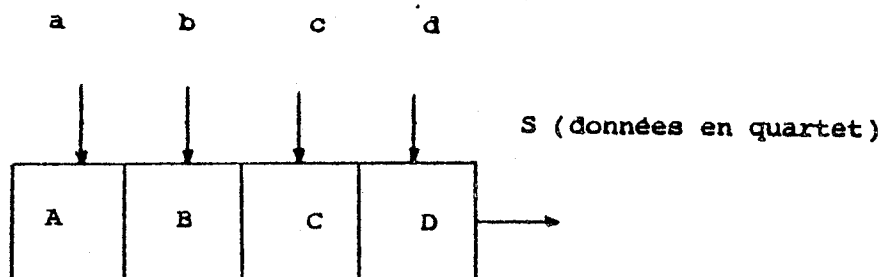
#### 5.II- 4.4. PSF passives

On ne peut écrire dans  $[i, a]$  si  $K = 1$   
avec  $K = [i, b], [j, c], \dots$

#### 5.II- 4.5. Pannes dans les registres

On considère les différentes fautes qui peuvent affecter les registres tant d'entrée que de sortie des données.

exemple : registre de sortie



Nous envisageons les collages

- des lignes a, b, c, d, S
- des cellules A, B, C, D
- des transitions A - B, B - C, C - D, D - S

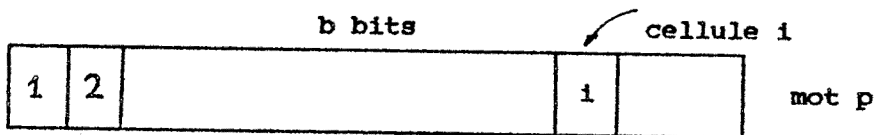
Les fautes du registre d'entrée sont similaires.

### 5.III- CALCUL DES LONGUEURS DE TEST

#### 5.III- 1. Mémoires à mots de b bits

Nous donnons dans ce paragraphe les différents résultats obtenus concernant les hypothèses de fautes adoptées pour les mémoires à M mots de b bits, soit  $n = M \times b$  bits.

Chaque mot m de la mémoire contient b bits (ou cellules) notés i, j, k, ...



Tous les calculs sont effectués pour une incertitude de détection  $1 - Q_D = 0.999$ . On suppose également une initialisation avec équiprobabilité des états de la mémoire.

#### 5.III- 1.1. Collages

On suppose que le bit i du mot p est collé à la valeur  $\alpha$ . La chaîne de Markov associée est similaire à celle définie pour les mots de 1 bit.

La longueur L nécessaire à la détection de cette faute est  $L = 46 \times M$ . On remarque donc que la longueur de détection d'un collage simple dans une mémoire à mots de b bits est indépendante de b, cette longueur ne dépend que du nombre de mots M. Pour  $b = 1$  (hypothèse classique) on obtient bien le même résultat.

### 5.III- 1.2. 2-Couplages

#### a) Idempotence et inversion

On suppose que ces 2-couplages n'interviennent qu'entre cellules de mots différents.

Les graphes de Markov associés à ces fautes sont les mêmes que ceux existant pour les mémoires à mots de 1 bit. Seules diffèrent les probabilités de changement d'état, où  $n$  est remplacé par  $M$ . (ex :  $a_{(1,p)} = \text{Prob [adresser le bit } i \text{ du mot } p] = 1/M$ ).

On obtient donc les valeurs suivantes

$$L(\text{inversion}) = 100 \times M$$

$$L(\text{idempotence}) = 219 \times M$$

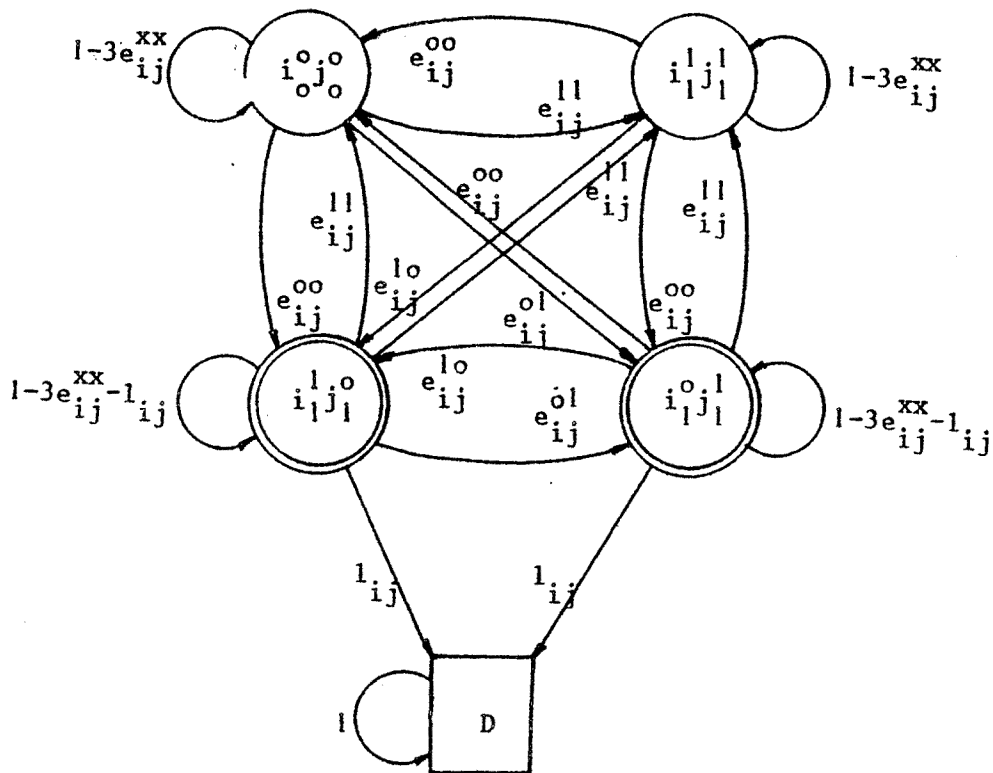
Ici aussi mêmes résultats que pour les mémoires à mots de 1 bit ( $b=1$ ,  $M = n$ ).

#### b) Couplage de types ET, OU

Nous avons été amenés à définir ce nouveau type de couplage. Il représente le couplage logique de 2 bits d'un même mot. (ex :  $(i,p) = (i,q) = (i,p) + (i,q)$ ).

La chaîne de Markov associée est représentée à la figure 19.

On obtient  $L = 46 \times M$ . Cette faute est donc plus facile à détecter que 2-couplage classique.



**Figure 19 : Modélisation du couplage "i-j=i+j"**

### 5.III- 1.3. PSF

La faute sur la cellule  $i$  dépend de l'état des  $k_j$  voisines.

Pour ce type de faute également, les graphes de Markov sont les mêmes que pour des PSF à mots de 1 bit. En revanche, les probabilités de passage entre les états où existe une faute et les états sans faute sont à évaluer. En effet, suivant que les cellules influentes appartiennent à des mots différents ou pas ces probabilités changent.

#### a) Cas de 2 cellules influentes $k_1, k_2$

On cherche les probabilités de passage de  $K = 0$  à  $K = 1$  et réciproquement.  $K = k_1 k_2$ .

On notera :

$\Pr[E_K^0 / K=1]$  la probabilité pour que  $K$  prenne la valeur 0 sachant que  $K = 1$

$E_{\begin{smallmatrix} \alpha_1 & \alpha_2 & \dots \\ k_1 & k_2 & \dots \end{smallmatrix}}$  = écriture de la valeur  $\alpha_1$  dans  $k_1$ ,  $\alpha_2$  dans  $k_2$ ...

$(k_1, k_2, \dots)$  appartiennent au même mot.

On définit :

la probabilité d'écrire un mot  $p$  tel que le bit  $(k_1, p)$  ait une valeur donnée  $x_1$  qui vaut  $1/4M$ .

et plus généralement la probabilité d'écrire dans un mot  $p$  tel que les bits  $(k_1, p), \dots, (k_q, p)$  prennent respectivement les valeurs  $x_1, \dots, x_q$  qui vaut  $1/2M \times 1/2^q$ .

a. 1. Les 2 cellules appartiennent au même mot



$\Pr[E_K^1 / K=0] = \Pr[E_{\begin{smallmatrix} 11 \\ k_1 k_2 \end{smallmatrix}}^{11}]$  ] étant donné que la probabilité d'écrire 11 dans

$k_1 k_2$  est indépendante de l'état  $k_1 k_2$ .

donc  $\Pr[E_K^1 / K = 0] = \frac{1}{4} \times \frac{1}{2M}$

de même on a :

$$\begin{aligned} \Pr[E_K^0 / K = 1] &= \Pr[E_{\begin{smallmatrix} 01 \\ k_1 k_2 \end{smallmatrix}}^{01}] + \Pr[E_{\begin{smallmatrix} 00 \\ k_1 k_2 \end{smallmatrix}}^{00}] + \Pr[E_{\begin{smallmatrix} 10 \\ k_1 k_2 \end{smallmatrix}}^{10}] \\ &= \frac{3}{4} \times \frac{1}{2M} \end{aligned}$$

a.2. Les 2 cellules n'appartiennent pas au même mot



On a alors

$$\begin{aligned} \Pr[E_K^1 / K = 0] &= \Pr[E_{k_1}^1] \times \Pr[k_1 k_2 = 01 / K = 0] \\ &+ \Pr[E_{k_2}^1] \times \Pr[k_1 k_2 = 10 / K = 0] \\ &= \frac{1}{3} \Pr[E_{k_1}^1] + \frac{1}{3} \Pr[E_{k_2}^1] \end{aligned}$$

soit  $\Pr[E_K^1 / K = 0] = \frac{1}{3} \times \frac{1}{2.M}$

et  $\Pr[E_K^0 / K = 1] = \Pr[E_{k_1}^0 / k_1 = 1] + \Pr[E_{k_2}^0 / k_2 = 1]$

$$= \frac{1}{2.M}$$

a. 3. Calculs

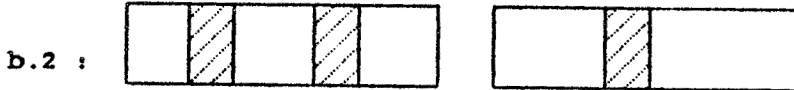
Nous pouvons à partir de l'établissement de ces probabilités, évaluer les rapports  $H = L/M$  correspondant aux diverses fautes

	cellules $k_1 k_2$ dans le même mot	cellules $k_1 k_2$ dans 2 mots différents	RAM mots de 1 bit
H(PSF passive)	231.5	222.7	222.7
H(PSF act. inv.)	448.2	439.5	439.5



b) Cas de 3 cellules influentes

Cette fois trois possibilités sont à envisager :



Les calculs sont similaires à ceux présentés dans a. 2.

	Cas b1	Cas b2	Cas b3	RAM mot de 1 bit
H(PSF passive)	474.2	383.2	443.1	443.1
H(PSF act. inv.)	909,1	754,0	876,9	876,9

5.III- 1.4. Commentaires

Suivant la position des cellules influentes les valeurs de H varient de moins de 10 % par rapport à H obtenus pour des RAM de M x 1 bit. On remarque que le cas le plus difficile à détecter est quand toutes les cellules influentes appartiennent au même mot.

On obtient les mêmes valeurs de H pour la détection des PSF si, dans une RAM à M x b bits les cellules appartiennent toutes à des mots différents, que pour des RAM à M x 1 bit.

On peut dire en bonne approximation que les résultats sur H, obtenus pour des mémoires à mots de 1 bit sont généralisables aux mots de b bits. La longueur de détection ne dépend pas du nombre de bits dans un mot mais du nombre total de mots.

## 5.III- 2. Problème spécifique aux RAM dynamiques

Nous étudions dans ce paragraphe un problème spécifique des RAM dynamiques, à savoir, la disparition trop rapide de l'information : une ou plusieurs cellules ne conservent l'information que pendant une durée  $T' < T$  ( $T$  temps donné par le constructeur).

On note  $\rho$  le rapport  $\rho = T'/T$ .

Prenons l'exemple où la valeur stockée en mémoire est 1, au bout d'un temps  $T'$ , si la cellule n'a pas été adressée, elle prendra la valeur fautive 0.

Calculons la probabilité de passage  $P$ , d'un état juste ( $i^1_1$ ) à un état faux ( $i^1_0$ ).

$$P = P_1 \times P_2 \text{ avec}$$

$P_1 = \text{Pr}[(\text{pas d'évènement écriture dans } i \text{ ou lecture de } i) \text{ pendant } T']$ . On suppose que la cellule n'est rafraichie que dans on vient la lire ou l'écrire.

En prenant  $T \approx 4\text{ms}$  et un temps d'accès de 200 ns on a environ :

$$\frac{4}{2 \cdot 10^{-4}} = 2 \cdot 10^4 \text{ évènement pendant } T.$$

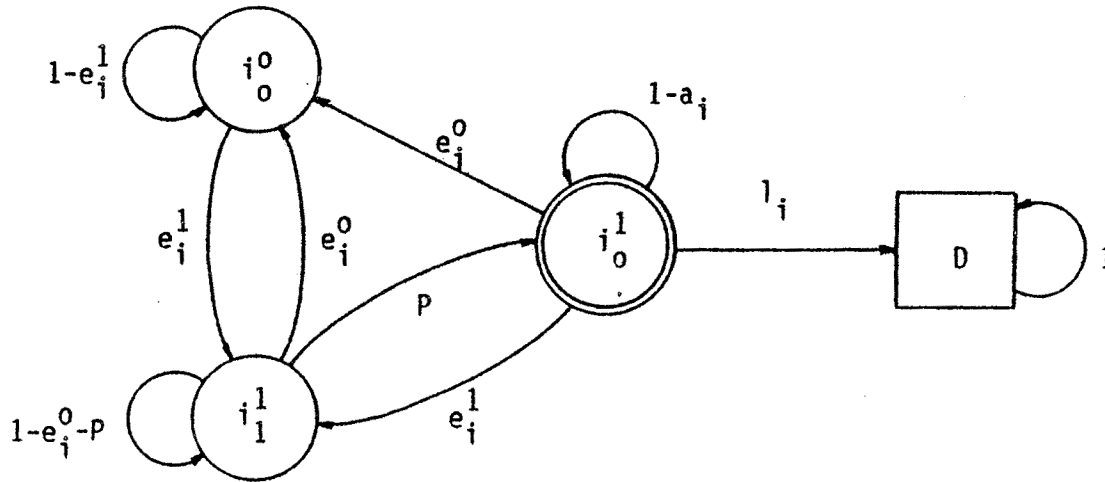
$$\text{d'où } P_1 = [1 - (e_i + l_i)]^\rho \times 2 \cdot 10^4$$

$$P_2 = \text{Pr}[\text{pas d'évènement "rafraichissement" pendant } T']$$

$$\text{d'où } P_2 = \frac{T-T'}{T} = (1 - \rho)$$

$$\text{On obtient } P = (1 - \rho) (1 - (e_i + l_i))^\rho \times 2 \cdot 10^4$$

Le graphe de Markov représentant ce type de faute est donné à la figure 20.



**Figure 20 : Modélisation d'une perte trop rapide  
d'information dans la cellule 1**

Dans l'interprétation des résultats, il faut tenir compte du fait que la faute étudiée ne concerne qu'une cellule. Or, il semble vraisemblable de penser que si cette faute affecte une cellule, elle affecte toutes les cellules de la mémoire. Il est donc une borne supérieure. On a d'autre part supposé que le rafraichissement ne s'effectuait que pour l'adressage de la cellule concernée, peut être devrait-on prendre en compte toutes les cellules de la ligne.

A partir du graphe de la figure 20 nous avons fait les calculs suivants :

$\rho = T'/T$	H(16 K)	H(256 K)
0.001	46,09	46,09
0.1	46,09	46,09
0.9	46,12	46,09
0.99	46,53	46,10
0.999	50,62	46,18
0.9999	97,03	46,97
0.99999	607,8	55,3
0.999999	5752	151,9

On remarque que pour une mémoire de 256 K si la durée relative de l'information  $T'/T < 0.9999$  la faute a une aussi bonne probabilité de détection qu'un collage. La modélisation de la faute de rafraichissement de toute la mémoire n'a pas été faite. Mais elle ne semble pas nécessaire étant donné la grande probabilité de détection de ce type de faute.

### 5.III- 3. Adressage en nibble mode

La stratégie de test aléatoire est celle du fonctionnement en nibble mode : tirage aléatoire d'une adresse et de l'opération lecture ou écriture puis lecture (ou écriture) sur les 4 cellules du quartet concerné.

On note  $(i, q_x)$  une cellule de la mémoire de  $(4 \times n)$  bits

$q_1$	$q_2$
$q_3$	$q_4$

où  $i$  correspond au poids fort de l'adresse  
et  $q_x$  correspond au poids faible de l'adresse

Calculons la probabilité  $a_{(i, q_x)}$  d'adresser la cellule  $(i, q_x)$ . Etant donnée la stratégie de test, adresser  $(i, q_x)$  équivaut à adresser le quartet  $[(i, q_1) (i, q_2) (i, q_3) (i, q_4)]$ .

$$\text{On a donc } a_{(i, q_x)} = 4 \times \frac{1}{4n}$$

$$\text{soit } a_{(i, q_x)} = \frac{1}{n}$$

On en déduit aisément

$$e^{\alpha}_{(i, q_x)} = \frac{1}{4n}$$

$$l_{(i, q_x)} = \frac{1}{2n}$$

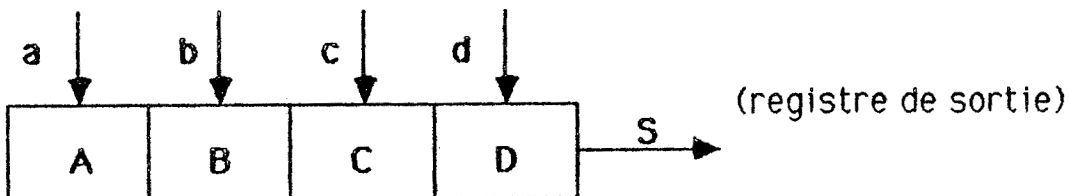
Pour une mémoire (4 x n bits) en nibble mode les probabilités d'opération sur les cellules sont les mêmes que pour une RAM classique à n mots de 1 bit et indépendantes des positions respectives des cellules dans les quartets. D'autre part, les hypothèses de fautes, nous l'avons vu, restent identiques.

On pourra donc appliquer tous les résultats concernant les longueurs L et valeurs de H obtenus pour des mémoires à mots de 1 bit. Soit  $L(4 \times n \text{ en nibble mode}) = L(n \text{ mots de 1 bit})$  pour une même hypothèse de faute.

### Pannes sur les registres à décalages

Il nous faut pour ce type de mémoire considérer des fautes spécifiques concernant les registres d'entrée et de sortie des données.

Ces registres peuvent être symbolisés de la façon suivante :



Sous des hypothèses de collages les fautes simples sont :

- collages des lignes a, b, c, d,
- collages des cellules A, B, C, D,
- collages des transitions A-B, B-C, C-D, D-S

elles sont toutes plus faciles à détecter qu'un collage de point mémoire.

En effet, pour les collages de chacune des lignes a, b, c, d c'est équivalent d'un point de vue détection au collage d'un quart de la matrice mémoire.

Le collage de S lui équivaut à un collage de tous les points mémoire.

Le collage d'une des cellules : le plus difficile à détecter serait

celui de la cellule A, or il est aussi facile à détecter qu'un collage d'un quart de la mémoire.

Les transitions : la plus difficile à détecter est la transition A-B qui est aussi équivalente (en probabilité) au collage d'un quart de la mémoire.

### Conclusion

On ne rencontre aucun problème spécifique au test du fonctionnement en nibble mode. On peut appliquer directement les résultats de H trouvés pour des RAM classiques.

#### 5.III- 4. Adressage en page mode

Nous exposons dans ce chapitre les résultats obtenus sur les longueurs de test pour des mémoires en fonctionnement mode page. La stratégie de test utilisée est la suivante :

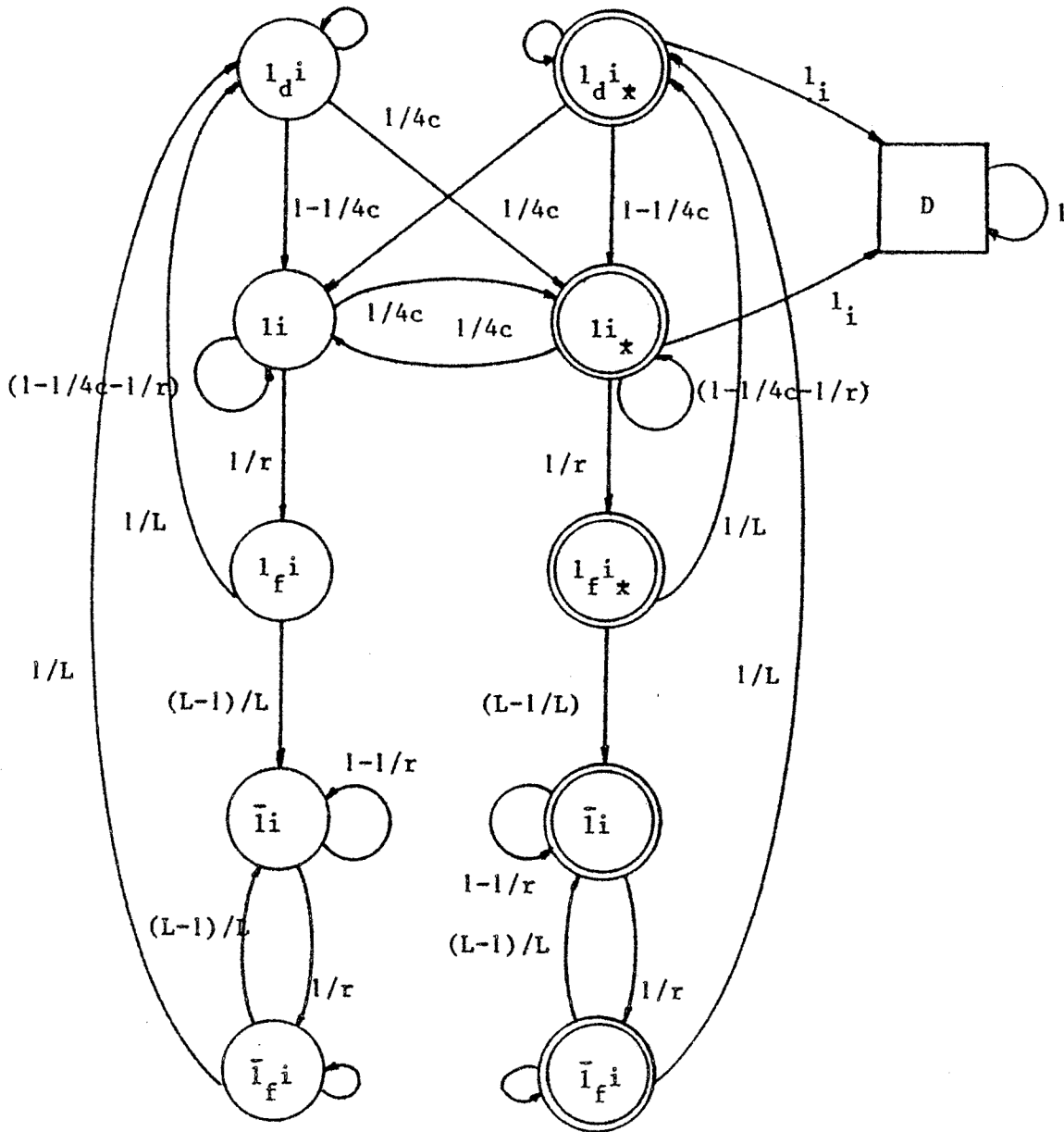
- tirage aléatoire d'une ligne
- tirage aléatoire de lecture ou écriture sur  $r_1$  cellules de la ligne ( $r_1$  est une variable aléatoire de moyenne  $r$ ).

Nous n'avons modélisé que le collage d'un point mémoire. La modélisation par chaîne de Markov du processus est représenté à la figure 21.

Les résultats numériques obtenus sont les suivants :

On s'est limité par valeur supérieure à  $r = 100$  puisque de toutes manières nous avons un temps de maintien du RAS qu'on ne peut dépasser.

r	H (collage)
1	46.20
10	46.35
50	46.33
100	48.56



**Figure 21 : Modélisation du collage i en mode page**

Les valeurs de  $H$  obtenues pour la détection d'un collage simple sont pratiquement indépendantes de  $r$  dans la fourchette considérée. On remarque que pour  $r = 1$  on retrouve bien la valeur obtenue pour un adressage classique.

Néanmoins, la modélisation par chaîne, multiplie par 5 le nombre d'états nécessaires par rapport aux modélisations dans une RAM classique.

Ceci amènerait à la construction de chaînes comprenant de 40 à 60 états pour les PSF, ce qui pratiquement semble peu raisonnable. On peut cependant penser, au vu des collages, que les valeurs de  $H$  trouvées sont très proches de celles pour les fautes classiques.

#### 5. IV- CONCLUSION

On peut dire en bonne approximation que les résultats obtenus concernant les longueurs de détection pour des fautes classiques étendues aux mémoires à mots de  $b$  bits sont les mêmes que pour les mémoires à mots de 1 bit. Cette longueur est donc fonction du nombre de mots dans la mémoire et non pas du nombre de bits dans le mot.

De même, l'extension au mode page et nibble mode, si elle change la stratégie de test, ne change pas les longueurs.

Enfin, la "perte d'information" d'une cellule dans une RAM dynamique ne semble pas poser de problème de détection (longueur équivalente à celle d'un collage).





PARTIE B

DIAGNOSTIC DE FAUTES



## Chapitre 6

METHODOLOGIE DE DIAGNOSTIC  
BASEE SUR DES EXPERIENCES  
DE TEST ALEATOIRE



## 6.I- INTRODUCTION

Nous avons vu comment évaluer la longueur de test nécessaire pour détecter les fautes dans une RAM. Grosso modo, on peut dire qu'une longueur de quelques centaines de fois  $N$  ( $N$  étant le nombre de mots) donne une bonne qualité de détection à la fois pour des hypothèses classiques (influences, PSF) et pour des hypothèses nouvelles (mots de  $b$  bits, adressage en mode page et nibble, RAM dynamiques).

Nous savons comment détecter par le test aléatoire, nous allons montrer comment diagnostiquer par des expériences de test aléatoire.

Le but est de définir une méthode (sous forme d'algorithme) pour cerner progressivement la faute dans un circuit défectueux.

Après avoir donné les bases générales nécessaire à la compréhension de notre algorithme, nous présentons l'algorithme de caractérisation de fautes.

## 6.II- BASES GENERALES

Dans ce chapitre, nous introduisons d'abord la liste des fautes que nous avons établie et sur laquelle nous avons travaillé. Nous donnons ensuite quelques définitions et notions de base nécessaires à la compréhension de l'algorithme.

### 6.II- 1. Hypothèses de fautes

Si l'on veut déterminer à partir d'une mémoire en faute quelle faute affecte cette mémoire, il nous faut établir une liste des fautes possibles. Nous donnons ici la liste adoptée.

#### 6.II- 1.1. Collage de point mémoire (type a).

On suppose qu'il existe un point mémoire collé (à 0 ou 1). Dans la suite, on notera cette faute : faute de type a.

### 6.II- 1.2. Adressage éventuellement faux (type b)

Ce type de faute est caractérisé par le collage d'un (ou plusieurs) bit d'adresse.

ex : Soit  $a_n a_{n-1} \dots a_1 a_0$  l'adresse d'un point mémoire. On suppose le bit  $a_0$  collé à 1. Les points mémoires  $i$  d'adresse  $a_n \dots a_1 0$  et  $i^+$  d'adresse  $a_n \dots a_1 1$  auront alors la "même" adresse pour la mémoire qui sera l'adresse  $a_n \dots a_1 1$ . Les opérations lecture (resp. écriture) de  $i$  et lecture (resp. écriture) de  $i^+$  seront les mêmes. Cela revient en fait à n'adresser qu'une moitié de la mémoire.

Cette faute sera dite de type b.

### 6.II- 1.3 Multi adressages

#### \* Ligne toujours décodée (type c)

Pour cette faute, lorsque l'on sélectionne un point d'une autre ligne il y a bi-adressage. On écrit et on lit en deux points mémoire à la fois. Si le point mémoire que l'on n'a pas sélectionné est à 0 il impose sa valeur en sortie, s'il est à 1, il n'influe pas sur la sortie. Ceci correspond donc au ET logique des deux points. (Suivant la technologie utilisée, la fonction peut être un ET ou un OU logique).

Exemple : Soit la ligne  $L$  toujours décodée. Si on note  $(L_x, C_y)$  l'adresse d'un point mémoire où  $L_x$  est l'adresse de la ligne et  $C_y$  l'adresse de la colonne : on aura toujours bi-adressage entre  $(L_x, C_y)$  et  $(L, C_y)$  quand  $(L_x, C_y)$  sera adressé, sauf si  $L = L_x$ .

Cette faute sera dite de type c.

#### \* Colonne toujours décodée (type d)

Cette faute est identique à celle où la ligne est toujours décodée en remplaçant le rôle de la ligne par celui de la colonne. Colonne  $C$  toujours décodée = adressage simultané de  $(L_x, C_y)$  et  $(L_x, C)$ . Cette faute est notée faute de type d.

#### \* Point mémoire toujours valide (type e)

Il y a bi-adressage lorsque l'on sélectionne un autre point mémoire.

Si le point  $(L,C)$  est toujours validé, on a adressage simultané de  $(L,C)$  et  $(L_x, C_y)$  pour tout point  $(L_x, C_y)$ .

Cette faute est dite faute de type e.

#### 6.II- 1.4. Panne de logique lecture/écriture (type f)

On suppose que la ligne de lecture/écriture peut être collée soit en position lecture soit en position écriture (faute de type f).

#### 6.II- 1.5. 2-couplages

Il existe deux types de 2-couplages entre cellules :

- le 2-couplage d'inversion : la transition  $\uparrow$  (ou  $\downarrow$ ) de la cellule  $j$  change l'état de la cellule  $i$  quel que soit l'état de cette dernière avant la transition (ex  $\uparrow_j \rightarrow \uparrow_i$ ).

Cette faute est dite de type g.

- le 2-couplage idempotent : la transition  $\uparrow$  (ou  $\downarrow$ ) de la cellule  $j$  change l'état de la cellule  $i$  seulement quand cette dernière est dans un certain état  $x$  (ex  $\uparrow_j \rightarrow \uparrow_i$ ).

Cette faute est dite de type h.

#### 6.II- 1.6. PSF

Ce sont les fautes sensibles à la configuration de la mémoire : il y a possibilité d'erreur uniquement pour une configuration donnée des cellules influentes.

On notera :

- . faute de type l les PSF actives idempotentes
- . faute de type m les PSF actives d'inversion
- . faute de type n les PSF passives.



### 6.II- 1.7. Panne spécifique aux RAM dynamiques

Elle est liée à une disparition trop rapide de l'information contenue dans les cellules :

. faute de type p, volatilité

### 6.II- 1.8. Fonctionnement en nibble mode (type q)

On tient compte des collages dans les registres d'entrée ou sortie des données (collage de lignes ou de cellules). Ces fautes sont schématisées sur la figure 22.

Ces fautes sont dites de type q.

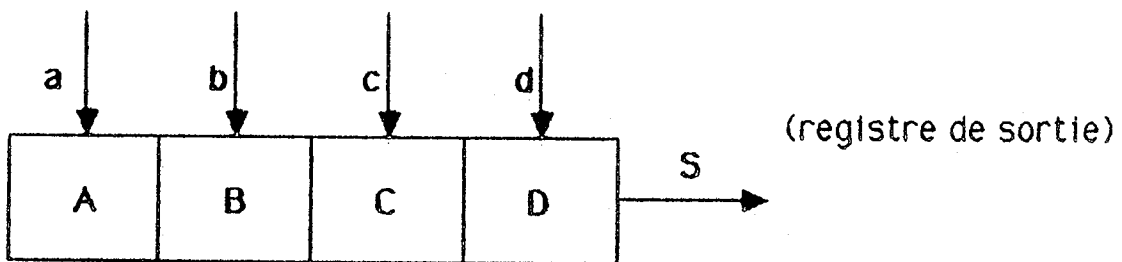
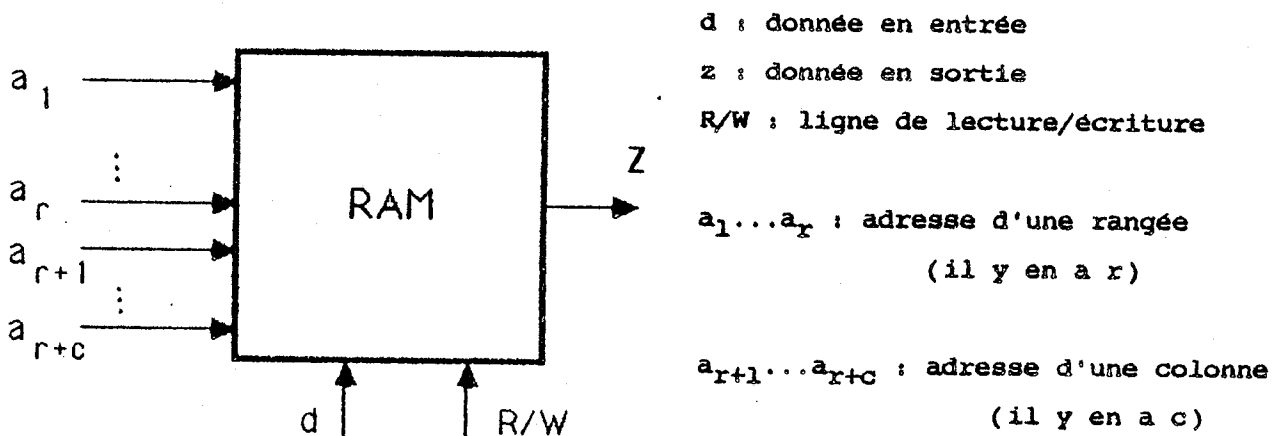


Figure 22 : Hypothèses de collages dans les registres E/S en nibble mode

### 6.II- 2. Modèle de mémoire

Nous définissons dans ce paragraphe le modèle de mémoire utilisé ainsi que quelques notations



## 6.II- 3. Définition d'une expérience

La stratégie utilisée par l'algorithme est la suivante. On veut déterminer à partir d'une mémoire en faute quelle est la faute qui l'affecte si elle est unique, ou au moins une des fautes si elle est multiple. Pour cela nous allons faire des expériences de test sur certaines zones de la mémoire et à partir des résultats obtenus nous tirerons des conclusions.

Pour chaque expérience, on définit des conditions initiales, des opérations de test et une observation.

Notion de zone

La première condition à définir est la zone  $Z$  de la mémoire sur laquelle nous allons faire l'expérience.

On notera :

$M$  : la mémoire entière. On suppose que l'on peut adresser tout point mémoire.

$M/a_1$  : l'ensemble des points mémoires pour lesquels  $a_1$  vaut 0.

$M/a_1$  : l'ensemble des points mémoires pour lesquels  $a_1$  vaut 1.

$M/a_1 \dots a_i$  : l'ensemble des points mémoires pour lesquels  $a_1 = 1, \dots, a_i = 1$ .

D'un point de vue pratique, restreindre l'ensemble  $M$  à une zone quelconque sera facile à condition que l'on puisse commander chaque ligne d'adresse  $a_i$  de la façon suivante :

- $a_i$  fixée à 1
- $a_i$  fixée à 0
- $a_i$  libre (varie comme la séquence qui lui est envoyée).

Notion d'initialisation

Pour toute zone Z avant de faire un test, il va falloir initialiser l'ensemble des points appartenant à Z. On peut envisager trois types d'initialisations :

- initialisation à 0 de tous les points appartenant à Z, on notera cela  $I_0(Z)$ .
- initialisation à 1 de tous les points appartenant à Z, notée  $I_1(Z)$ .
- initialisation mixte. Certains points à 0, d'autres à 1, notée  $I_\alpha(Z)$ .

Notion d'opération

Nous devons ensuite définir les opérations effectuées au cours de l'expérience. Elles sont au nombre de trois :

- lecture d'une zone Z. On ne fait que des lectures sur les points mémoires appartenant à Z. Notée  $R(Z)$ .
- écriture sur Z. On écrit sur les points appartenant à Z. Notée  $W(Z)$ .
- Test aléatoire de la zone. Succession de lectures et écritures aléatoires sur les points de Z. Notée  $T.A(Z)$ .

En pratique, il nous suffira de pouvoir commander la ligne R/W :

- soit fixée à 1 : lecture
- soit fixée à 0 : écriture
- soit libre : varie comme la séquence aléatoire qui lui est envoyée.

Observation

Après le test d'une zone Z, deux résultats sont possibles :

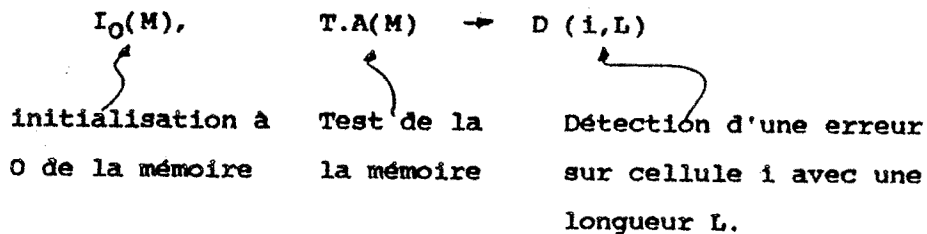
- Détection d'une erreur (Notée D).
- Pas de détection (Notée D').

Dans le cas où il y a eu détection, on peut mémoriser la cellule qui contient l'erreur et la longueur de la séquence qui a permis la détection.

En bref, on peut dire qu'une expérience est une suite d'opérations comprenant :

- initialisations de zones.
- opération sur une zone.
- observation du résultat.

exemple :



## 6.II- 4. Rôle de l'initialisation

Soit une zone Z à tester. Nous allons voir que suivant le type d'initialisation, ( $I_0, I_1, I_\alpha$ ) et le domaine à initialiser, les observations peuvent être différentes.

a) Influence de la zone Z

Supposons que l'on veuille faire une expérience sur la zone Z. Notons Z l'ensemble des points mémoires de M qui n'appartiennent pas à Z.

Il est clair que suivant que l'on initialise Z ou M avant de faire l'expérience sur Z, l'ensemble des erreurs présentes dans Z peut être différent. Exemple : Si on a une faute de type  $\uparrow j \Rightarrow \uparrow i$  avec  $j \notin Z$  et  $i \in Z$ . Si on initialise Z, il n'y aura pas d'erreur à l'initialisation (on n'écrit pas dans j) alors qu'il peut très bien y en avoir une si on initialise M (on fait une

écriture dans j).

Il conviendra donc de bien préciser la zone d'initialisation pour chaque expérience.

#### b) Influence du type d'initialisation

Nous allons analyser successivement l'influence de chacune des initialisations  $I_0$ ,  $I_1$  et  $I_\alpha$  sur une zone Z.

##### Initialisation de Z à 0, $I_0(Z)$ .

Lors de l'initialisation, on peut introduire une erreur. Nous allons déterminer quelles sont parmi les fautes considérées celles qui peuvent amener une erreur dans la zone considérée.

type a : s'il existe un collage à 1.

type b : l'erreur peut être présente dans la partie non adressable, mais sera alors indétectable.

Exemple :  $Z = M$ . Le bit  $a_1$  est collé à 0. On fait  $I_0(M)$ . Dans ce cas, tous les points pour lesquels  $a_1 = 1$  auront une valeur quelconque 0 ou 1 puisqu'on ne peut les adresser, alors qu'ils devraient valoir 0. Mais cette erreur est indétectable puisqu'on ne peut y accéder.

type c : ligne toujours décodée, on n'introduit pas d'erreur à l'initialisation.

type d : colonne toujours décodée, on n'introduit pas d'erreur à l'initialisation.

type e : point toujours valide, on n'introduit pas d'erreur à l'initialisation.

type f : collage de R/W, on peut introduire une erreur si la ligne est collée en position lecture.

type g : influence idempotente, on peut introduire une erreur s'il existe des fautes du type  $\downarrow j \Rightarrow \uparrow i$  et que j vaille 1 avant l'initialisation et qu'on écrive dans i avant d'écrire dans j.

type h : influence d'inversion, on introduit une erreur s'il existe une faute  $\downarrow j \Rightarrow \downarrow i$  et que j = 1 avant l'initialisation et on écrit dans i avant d'écrire dans j.

type l : PSF active idempotente. On peut introduire une erreur s'il existe des

fautes du type  $(\downarrow j \Rightarrow \uparrow i \text{ si } k = \alpha)$  et que  $j$  vaille 1 avant l'initialisation, il faut qu'on écrive dans  $i$  avant d'écrire dans  $j$ .

type m : PSF active d'inversion. On peut introduire une erreur s'il existe des fautes du type  $(\downarrow j \Rightarrow \downarrow i \text{ si } k = \alpha)$  et que  $j$  vaille 1 avant l'initialisation. Il faut écrire dans  $i$  avant d'écrire dans  $j$ .

type n : PSF passive. On introduit une erreur s'il existe des fautes du type (on ne peut écrire dans  $i$  si  $k = \alpha$ ) et que  $i = 1$  avant l'initialisation,  $k = \alpha$  au moment où on écrit dans  $i$ .

type p : Volatilité. Panne spécifique aux RAM dynamiques. On n'introduit pas d'erreur à l'initialisation.

type q : Collages dans les registres (nibble mode) s'il existe un collage à 1.

### Initialisation de Z à 1

On a exactement les mêmes résultats que pour l'initialisation à 0 en inversant les rôles des 0 et 1.

### Initialisation quelconque

Ici l'erreur introduite n'est jamais bien déterminée.

type a : on peut introduire une erreur si collages à 0 ou 1.

type b : on peut avoir une erreur si on a 2 points différant par un bit d'adresse, possédant des valeurs différentes et que l'on fait une écriture sur la cellule non adressable en dernier.

type c : il y a erreur s'il existe une colonne telle que la dernière ligne écrite ne soit pas la ligne toujours décodée et que la valeur écrite soit différente de la valeur correspondante sur la ligne toujours décodée.

type d : idem en inversant les rôles de lignes et colonnes.

type e : il existe un point où l'on écrit avec valeur différente.

type f : oui si la ligne est collée en lecture.

type g : oui mais il serait trop compliqué de détailler les diverses configurations.

type l, idem type g ou h.

m, n

type p : on n'introduit pas d'erreur

type q : s'il existe un collage à 0 ou 1.

On a donc porté notre choix sur une initialisation à zéro. En effet, l'initialisation quelconque présente beaucoup plus de cas où l'état initial est faux, sans que cela apporte quelque renseignement que ce soit sur la présence d'erreurs.

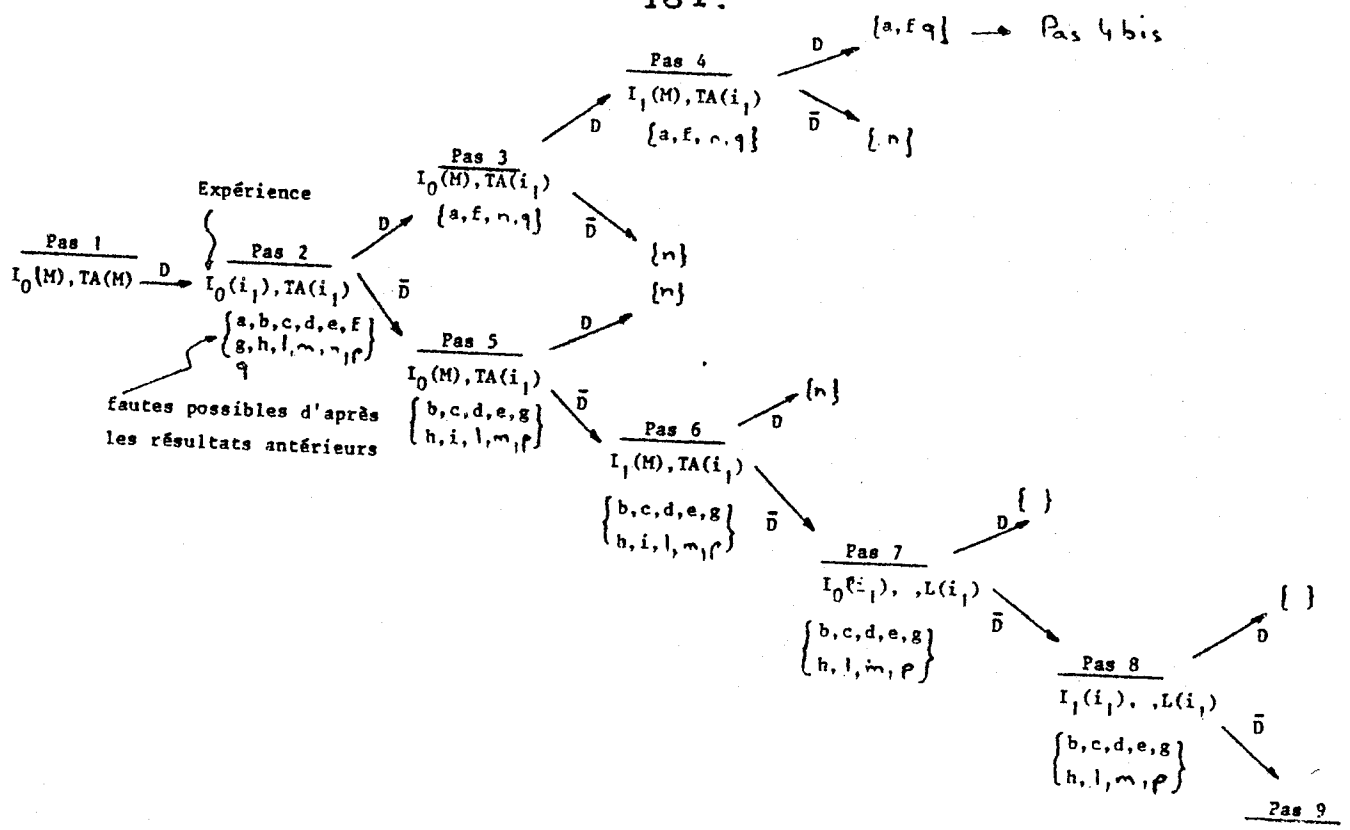
#### 6.II- 5. Conclusion

Ayant défini un ensemble de fautes, nous allons maintenant présenter un algorithme permettant la caractérisation d'au moins une des fautes affectant une mémoire en faute. Cet algorithme sera une succession d'expériences comme nous l'avons défini au paragraphe 6.II- 3.

### 6.III- ON ALGORITHME DE CARACTERISATION

#### 6.III- 1. Description

Nous donnons ici sous forme de schéma le déroulement de l'algorithme (figure 23). La stratégie utilisée est la suivante : on localise une cellule contenant une erreur puis on augmente le nombre de cellules adressées en variant la zone. Nous avons essayé [Fue 86] une autre "philosophie" qui était de cerner la zone mémoire adressée : fixant des bits d'adresse. Cette dernière méthode ne permettait pas de différencier les fautes si elles étaient multiples.



fautes possibles d'après les résultats antérieurs

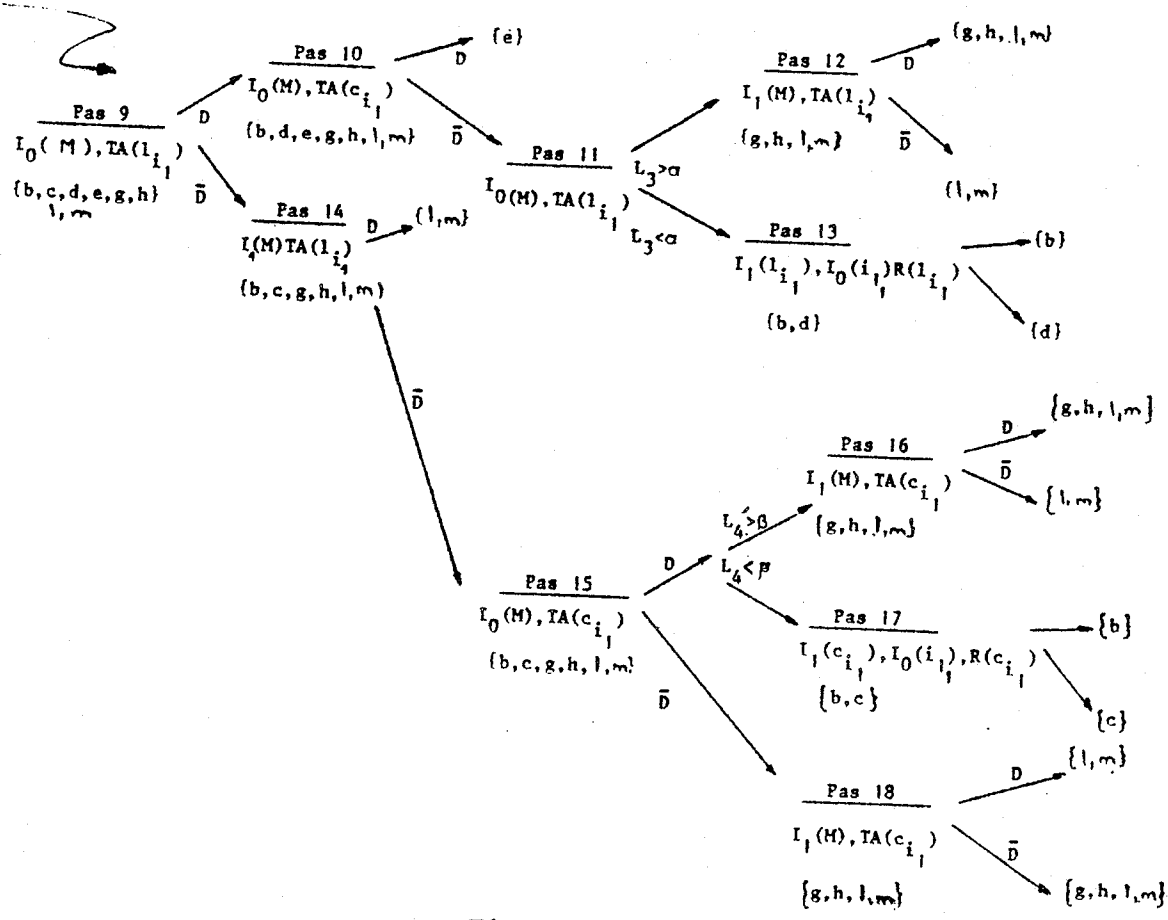


Figure 23 : "Algorithme"



### 6.III- 2. Etude de l'algorithme

Nous allons dans ce paragraphe décrire chacun des pas de l'algorithme et expliciter les conclusions que l'on en tire. Le pas 1 donne l'adresse de la cellule  $i$  contenant une erreur et la longueur  $L_1$  de la séquence ayant permis la détection.

#### 6.III- 2.1. Etude du Pas 2

Initialisation à 0 de  $i$  et test aléatoire de  $i$

\* si on a détection d'une faute

alors on obtient une longueur  $L_2$

la faute peut être :

de type a : collage de  $i$

de type f : collage de lecture écriture

de type n : PSF passive

de type q : collage en nibble mode

\* si on n'a pas détection

on est alors en présence d'une des fautes "b, c, d, e, g, h, m, n, p".

#### 6.III- 2.2. Etude du Pas 3

Initialisation à 0 de la mémoire et test aléatoire de la cellule  $i$ . Si on a détection on mémorise  $L_3$ . En fait ce pas permet de déterminer si la faute est une PSF passive du type : on ne peut écrire dans  $i$  si  $k = 1$

#### 6.III- 2.3. Etude du Pas 4

Initialisation à 1 de la mémoire et test aléatoire de la cellule  $i$ . Si détection, on mémorise  $L_4$ . Ce pas permet de caractériser une PSF passive pour le cas  $k = 0$ .

Si on a détection, il nous reste les fautes a, f et q à distinguer. Deux cas de figures peuvent se présenter :

a) la mémoire possède un fonctionnement en nibble mode

Dans ce cas, il nous faudra effectuer le pas 4bis.

b) la mémoire ne possède pas de fonctionnement en nibble mode.

Dans ce cas, la longueur  $L_1$  peut nous permettre de conclure :

- Si  $L_1 > 20$  alors on peut considérer que la faute est un collage de  $i$ . En effet, si c'était une faute de lecture écriture cette faute affecterait toutes les cellules ; elle est plus facile à détecter qu'un collage. La longueur de détection pour  $1-Q_D = .999$  est donc inférieure à 46, l'espérance de  $L_1$  est de  $46/\text{Log } 10^3 \approx 46/7 \approx 7$ .

Nous avons pris une borne supérieure 20. On peut donc dire que si  $L > 20$ , ce n'est pas une faute de logique R/W, c'est un collage.

- Si  $L_1 < 20$ , on ne peut conclure directement. On a une certaine probabilité d'être tombé rapidement sur la cellule en faute. On peut alors refaire une initialisation et un test de la mémoire. On obtient une cellule  $i'$  et une longueur  $L'_1$ .

si  $i' = i$  alors on a un collage de  $i$

si  $i' \neq i$

si  $L'_1 > 20$  on peut supposer que c'est un collage double de  $i$  et  $i$ .

si  $L'_1 < 20$  : faute de lecture écriture.

Ces dernières conclusions ne sont pas absolues mais ont une très grande probabilité d'être exactes. L'expérimentateur peut alors faire des essais supplémentaires s'il veut avoir une certitude.

#### 6.III- 2.4. Etude du Pas 4bis

Ce pas détermine les problèmes de collage dans les registres de fonctionnement en nibble mode.

On note A, B, C, D et les collages des lignes d'entrée du registre et E, F, G les collages des cellules (cf. figure 24).

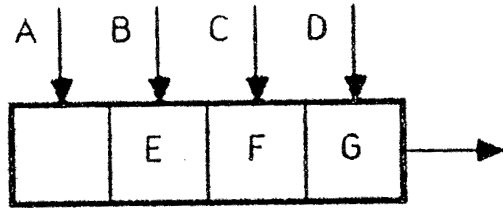


Figure 24 : Collages dans les registres

On fait une initialisation de la mémoire à 0 puis une lecture en nibble mode de M en fixant les deux adresses qui servent au fonctionnement nibble mode à 00 (on note L (M00)).

L'observation de la séquence de sortie nous permet de déterminer le type du collage. Le tableau suivant permet de donner la faute en fonction de la séquence de sortie S

Collage de A	S = 0 0 0 1 0 0 0 1 ...
Collage de B	S = 0 0 1 0 0 0 1 0 ...
Collage de C	S = 0 1 0 0 0 1 0 0 ...
Collage de D	S = 1 0 0 0 1 0 0 0 ...
Collage de E	S = 0 0 1 1 0 0 1 1 ...
Collage de F	S = 0 1 1 1 0 1 1 1 ...
Collage de G	S = 1 1 1 1 1 1 1 1 ...
Collage de R/W	S = aléatoire
Collage de i	S = 0 0 0 0 0 0 0 0 ...

On remarque que ce pas permet aussi de caractériser les fautes a et f.

#### 6.III- 2.5. Etude des Pas 5 et 6

Ces 2 pas permettent de mettre en évidence la présence d'une PSF passive. La détection au pas 5 montre une PSF passive avec  $k = 0$ , la détection au pas 6 montre une PSF passive avec  $k = 1$ .

#### 6.III- 2.6. Etude des Pas 7 et 8

Nous testons ici, pour les RAM dynamiques, si la cellule maintient la

valeur entre deux rafraichissements.

### 6.III- 2.7. Etude du Pas 9

Initialisation à 0 de  $L_1$  et test aléatoire de  $L_1$ .

\* Si on a détection alors la faute peut être de :

type "b" : possibilité de collage de bit d'adresse mais c'est alors un bit de colonne

type "d" : colonne toujours décodée

type "e" : point toujours décodé

type "g" : 2-couplage idempotent : possible si les deux points appartiennent à la ligne  $L_{1_1}$ .

type "h" : 2-couplage d'inversion : possible si les deux points appartiennent à la ligne  $L_1$

type "l" : PSF active idempotente : i et j appartiennent à la ligne  $L_1$ .

type "m" : PSF active d'inversion : i et j appartiennent à la ligne  $L_1$ .

\* Si on n'a pas eu détection, alors elle est de :

type "b" : collage de bit d'adresse mais bit de ligne

type "c" : ligne toujours décodée

type "g" : j est n'importe où ( $i \in L_1$  évidemment)

type "h" : j est n'importe où ( $i \in L_1$ )

type "l" : j et k n'importe où ( $i \in L_1$ )

type "m" : j et k n'importe où ( $i \in L_1$ )

### 6.III- 2.8. Etude du Pas 10

On initialise la colonne contenant i et on fait un test aléatoire de la colonne. On sait qu'il y a eu détection au Pas 9.

\* Si on détecte alors la faute est

de type "e" : on a un point toujours validé.

\* Si on ne détecte pas alors c'est une faute

type "b" : bit de colonne collé  
 type "d" : colonne toujours décodée  
 type "g" : les deux points appartiennent à la ligne i  
 type "h" : les deux points appartiennent à la ligne i  
 type "l" : j et k n'importe où ( $i \in L_1$ )  
 type "m" : j et k n'importe où ( $i \in L_1$ )

On peut donc déjà diagnostiquer le cas d'un point mémoire toujours décodé : détection au Pas 10. Nous allons maintenant essayer de différencier les fautes b, d, g, h, l, m.

### 6.III- 2.9. Etude du Pas 11

On initialise la ligne contenant i et on teste cette ligne. On va obligatoirement avoir détection puisque toutes les fautes à différencier sont détectables par cette expérience. Soit  $L_{11}$  la longueur de détection et  $i_{11}$  la cellule contenant une erreur. Pour différencier ces fautes, nous allons raisonner sur l'espérance des longueurs permettant leur détection.

L'espérance de L pour une faute de type g est de l'ordre de  $110/7 \times r$  soit  $E(L_g) = 15r$  si r est le nombre de lignes (et donc le nombre de cellules sur la colonne). De même, pour une faute de type f elle est de  $220/7 \times r$  soit  $E(L_f) = 30r$ . Pour les fautes de type l et m, cette espérance est de l'ordre de 60 r et 30 r respectivement.

Pour une faute de type b, cette même espérance est de :

$$\begin{aligned}
 \text{pour une faute de type b } 1/E(L_b) &\approx (\text{proba de lire}) \times (\text{proba } i > i^+) \\
 &\approx 1/2 \times 1/4
 \end{aligned}$$

$$\text{donc } E(L_b) \approx 8$$

de même pour une faute de type d  $E(L_d) \approx 8$ .

Comme les mémoires à tester sont de capacité supérieure à 1K on peut supposer que  $r > 30$

on a donc  $E(L_D) \approx 8$

$E(L_M) = E(L_G) > 300$

$E(L_H) > 600$

$E(L_1) > 1200$

On a donc des ordres de grandeurs très différents (8 et 300).

On peut considérer en bonne approximation que :

si  $L_{11} < 20$ , on a soit un collage de bit d'adresse, soit une ligne toujours décodée (b, d). Pour les différencier on aura le pas 12.

si  $L_{11} > 20$ , on a une faute d'influence (g, h) ou une PSF active (l, m). Afin de connaître l'adresse des cellules on utilisera l'algorithme 2.

L'expérimentateur pourra renouveler l'expérience du pas 11 s'il veut s'assurer de la conclusion.

#### 6.III- 2.10. Etude du Pas 12

Initialisation à 1 de la mémoire et test aléatoire de la ligne  $L_1$ . Ce pas permet de donner des précisions supplémentaires sur les positions respectives de k et j et i.

Si on a détection on a une faute de :

type g : i et j appartiennent à  $L_1$

type h : i et j appartiennent à  $L_1$

type l : i et j appartiennent à  $L_1$

type m : i et j appartiennent à  $L_1$

Si on n'a pas détection c'est une faute de :

type l : i et j appartiennent à  $L_1$  et k n'appartient pas à  $L_1$  k = 0

type m : i et j appartiennent à  $L_1$  et k n'appartient pas à  $L_1$  k = 0

6.III- 2.11. Etude du Pas 13

On veut différencier les fautes :

- b bit d'adresse de colonne collé
- d colonne toujours décodée

Pour cela on utilise la stratégie suivante :

- Initialisation à 1 de la ligne  $L_1$  puis initialisation à 0 de  $i$ .
- lectures de  $L_1$ 
  - Si D alors on est en présence de la faute d et la colonne toujours décodée est  $C_1$ .
  - Si D' on mémorise  $i^*$  et on fait  $I_1(L_1)$  et  $I_0(j)$  avec  $j \neq i$  et  $j \in L_1$ . La détection par lectures de  $L_1$  donne la détection de  $j^*$ .  
si  $i^* = j^*$  alors la colonne contenant  $i^*$  est toujours décodée  
si  $i^* \neq j^*$  on a un collage de bit de colonne

On a donc différencié les fautes b et d.

6.III- 2.12. Etude du Pas 14

Ce pas permet de caractériser les fautes de type l et m dans le cas où :

- i et j appartiennent à  $L_1$
- ( $k = 1$  et  $k \notin L_1$ ) ou ( $k \in L_1$ )

6.III- 2.13. Etude des Pas 15, 16 et 17

Les pas 15, 16 et 17 sont l'équivalent des pas 11, 12 et 13 respectivement si l'on remplace le rôle de  $L_1$  par celui de  $C_1$ .

6.III- 2.14. Etude des Pas 18

Si on a détection on est en présence d'une PSF (type l ou m) où  $k = 0$  et ( $j \notin L_1$  et  $j \notin C_1$ ).

Si on n'a pas détection on est en présence d'une faute de type g, h l ou m mais  $j \notin L_1$ ,  $j \notin C_1$ .

## 6.III- 3. Conclusion

Nous avons donc un algorithme qui nous permet de caractériser le type de faute affectant la mémoire. Il nous a semblé intéressant dans le cas de fautes du type 2-couplage en PSF de localiser les cellules en faute. Ceci fait l'objet du chapitre suivant.

## 6.IV- UN ALGORITHME DE LOCALISATION

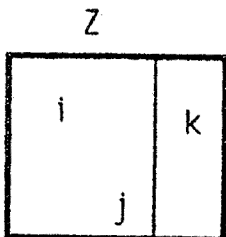
Dans ce chapitre, nous tentons de localiser la cellule  $j$  dans le cas des fautes de 2-couplage ( $\uparrow j \Rightarrow \uparrow i$  ou  $\uparrow j \Rightarrow \downarrow i$ ) ou PSF active ( $\uparrow j \Rightarrow \uparrow i$  si  $k=0$  ou  $\uparrow j \Rightarrow \downarrow i$  si  $k=1$ ). On connaît la cellule  $i$  (grâce à l'algorithme de caractérisation) et une zone de mémoire  $Z$  dans laquelle se trouve  $j$  (cette zone est soit  $L_1$ , soit  $C_1$ , soit la mémoire  $M$ ).

On a donc  $a_1^i \dots a_{r+c}^i$  adresse de la cellule  $i$  influencée connue. On cherche  $a_1^j \dots a_{r+c}^j$  adresse de la cellule influente. On note  $a_1 \dots a_{r+c}$  l'adresse d'un point quelconque et on suppose qu'on peut soit fixer  $a_1$  à une valeur donnée (0 ou 1) soit laisser varier  $a_1$  en fonction de la séquence aléatoire qui lui est envoyée.

Les fautes de type 2-couplage sont des cas particuliers de PSF actives ( $k$  n'a pas d'influence) étant donné que dans ce chapitre nous nous intéressons à la localisation de  $i$  et  $j$ , une autre expérience d'initialisation des autres cellules à 0 ou 1 permet, une fois déterminés  $i$  et  $j$ , de savoir si on a un 2-couplage ou une PSF.

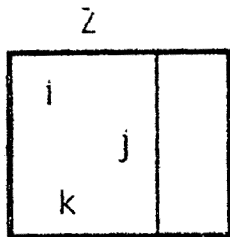
Deux cas de figures peuvent se produire :

a)  $i, j \in Z$   $k \notin Z$





b)  $i, j, k \in Z^*$



Nous allons étudier ces deux cas successivement.

On peut définir cette zone Z par un ensemble de lignes d'adresse  $a^*_1 \dots a^*_p$  (dans le cas où  $Z = L_1$ ,  $a^*_1 \dots a^*_p = a_1 \dots a_r$ , dans le cas où  $Z = C_1$ ,  $a^*_1 \dots a^*_p = a_r \dots a_{r+c}$  et dans le cas où  $Z = M$ ,  $a^*_1 \dots a^*_p = a_1 \dots a_{r+c}$ ).

1)  $i$  et  $j \in Z$ ,  $k \in Z$

Soit Z la zone contenant i et j et  $Z^* = M - Z$ . On fait  $I_0(M) TA(Z)$  puis  $I_1(M) TA(Z)$ . On doit alors avoir soit D puis D' (si la faute se produit pour  $k=0$ ) soit D' puis D (si la faute se produit pour  $k=1$ ). On déroule ensuite l'algorithme 1 qui donne l'adresse de j.

Faire BOUCLE pour  $l = 1$  jusqu'à  $l = p$

DEBUT BOUCLE

Fixer  $a_1 = a^i_1$

Initialiser la zone adressée

Test aléatoire de la zone

- si D mémoriser  $a^j_1 = a^i_1$

laisser  $a_1$  fixé à  $a^i_1$

- si D' mémoriser  $a^j_1 = a^i_1$

laisser  $a_1$  aléatoire

FIN BOUCLE

FIN ALGO

Nous avons donc déterminé l'adresse de j. Une simple expérience permettra de déterminer si cette influence est de type inversion ou idempotente. On notera que pour déterminer l'adresse de k il faudrait procéder par "intuition".

2)  $i, j, k \in Z^*$

On se propose ici aussi de déterminer l'adresse de  $j$  soit  $a_1^j \dots a_{r+c}^j$ .

L'algorithme est le suivant :

Faire boucle pour  $l = 1$  jusqu'à  $l = p$

DEBUT BOUCLE

Initialiser la zone adressée

Fixer  $a_1 = a_1^i$

Test aléatoire de la zone adressée

\* si D mémoriser  $a_1^j = a_1^i$

laisser  $a_1^i$  fixée

\* si D' laisser  $a_1$  libre

Initialiser  $Z^*$  a 1

Fixer  $a_1 = a_1^i$

Test aléatoire de la zone adressée

- si D alors mémoriser  $a_1^j$

laisser  $a_1$  fixé

- si D' alors mémoriser  $a_1^j = a_1$

laisser  $a_1$  aléatoire

FIN BOUCLE

FIN ALGO



## Chapitre 7

# CAHIER DES CHARGES POUR UNE MACHINE DE TEST



## 7.I- INTRODUCTION

Ce chapitre est consacré à la définition des principes de bases utiles à la mise en application du test aléatoire de mémoire sur une machine de test. Nous n'entrerons pas dans les détails, nous donnons simplement une idée de ce que sont les fonctions à réaliser par la machine. Cette machine pourra soit être un testeur existant que l'on adapte, cette adaptation étant en grande partie logicielle, soit un testeur à construire et nous donnons alors quelques idées sur la réalisation du générateur de séquence aléatoire.

Après une définition des diverses fonctions de cette machine, nous proposons une solution matérielle ou logicielle.

## 7.II- LES FONCTIONS DE LA MACHINE DE TEST

### 7.II- 1. Introduction

Dans le test d'un circuit, nous devons considérer deux parties différentes :

- a) la partie amont ou génération de la séquence de test, c'est cette séquence qui peut être aléatoire ou déterministe.
- b) la partie aval ou observation des sorties qui donne le résultat du test.

Nous nous intéressons ici surtout à la partie amont. En effet, la détection d'erreur en sortie est un problème classique qui n'entre pas dans le cadre de notre étude.

Pour mémoire nous pouvons distinguer :

- la comparaison avec un (ou plusieurs) circuit de référence. On remarquera que dans le cas d'une génération par logiciel on peut être bien utiliser un plan mémoire "fictif".
- les méthodes d'analyse de signature (comptages, registres à décalages bouclés, ...).

## 7.II- 2. Les fonctions à réaliser

Si nous faisons la synthèse des différentes fonctions utilisées, aussi bien pour la détection de pannes que dans l'algorithme d'aide ou diagnostic, nous obtenons la liste suivante :

### a) l'initialisation

La machine doit pouvoir initialiser, soit à la valeur 0, soit à la valeur 1, une zone de la mémoire (cette zone pouvant être la mémoire entière).

### b) le Test aléatoire

Ce test doit pouvoir adresser une zone donnée.

### c) Lecture aléatoire

Pour une zone Z donnée on veut pouvoir faire des lectures en adressant aléatoirement des cellules appartenant à Z.

Enfin lors d'une détection d'erreur il faut :

- mémoriser l'adresse de la cellule contenant l'erreur
- mémoriser le nombre de vecteurs jusqu'à la détection de l'erreur.

On remarque que ces deux derniers points nous interdisent la détection d'erreur par analyse de signature. En effet, il faut pouvoir déterminer l'instant où la faute est détectée ce qui est incompatible avec une méthode d'analyse de signature.

En revanche la méthode de comparaison avec un plan fictif comme référence est parfaitement adaptée.

## 7.II- 3. Composantes des fonctions

## a) Zone Z

Toutes les zones mémoires que nous avons utilisé dans nos expériences sont déterminées par des bits d'adresse fixé à 0 ou 1.

Exemple : on fixe  $a_0$  à 1 ; on adresse alors simplement une moitié de la mémoire.

## b) Le type d'adressage

Etant donné une zone Z cette zone pourra être adressé de façon séquentielle S (par exemple lors d'une initialisation) soit de façon aléatoire A (lors d'un test aléatoire).

## c) La ligne R/W

On veut pouvoir soit écrire W (lors d'une initialisation) soit lire R, soit tirer aléatoirement une lecture ou une écriture A (lors d'une séquence de test).

## d) Les données

Elles peuvent être fixées à 0 ou 1 (lors d'une initialisation) ou aléatoires A (lors d'un test).

Nous donnons dans le tableau ci-dessous l'ensemble des fonctions à réaliser ainsi que la valeur de chacun des paramètres.

Fonction	Zone	Adressage	R/W	Données
1) Initialisation	Z	S	W	0 ou 1
2) Test aléatoire	Z	A	A	A
3) Lecture aléatoire	Z	A	R	0

4) Mémorisation de l'adresse de la cellule contenant une erreur.

5) Mémorisation du nombre de vecteur jusqu'à cette détection.



## 7.II- 4. Les paramètres de la mémoire

Plusieurs paramètres concernant la mémoire sont nécessaires :

- le nombre de bits d'adresse
- si multiplixage des adresses : nombre de lignes et nombre de colonnes
- le nombre de bits de données
- le mode adressage (direct, cycle R/W, nibble mode et mode page)

## 7.II- 5. Conclusion

Ces notions de base vont nous permettre d'envisager un schéma de principe pour la machine de test. Ce schéma de principe pourra aussi bien s'appliquer à une solution logicielle qu'à une solution matérielle.

## 7.III- SCHEMA DE PRINCIPE DE LA MACHINE

La figure 24 donne le schéma de principe du dispositif complet de test.

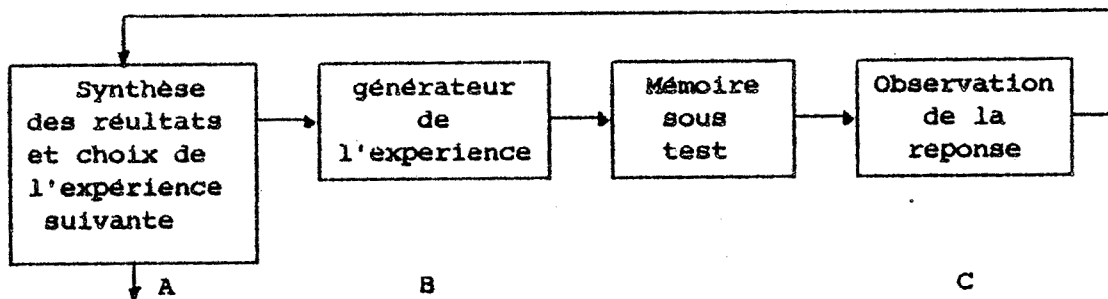


Figure 24 : Dispositif complet de test

A. Synthèses des résultats et choix de l'expérience suivante. Cette partie a été programmé. Chaque expérience se compose de plusieurs phases, par exemple l'initialisation d'une zone puis le test aléatoire de cette zone. Chaque phase est caractérisée par les grandeurs qui sont les entrées du générateur d'expérience et qui sont les composantes de fonction précédemment défini, plus la longueur dans le cas d'adressage aléatoire.

B. Le générateur d'expérience est décrit sur la figure 25.

C. L'observation de la sortie est un problème qu'on sait résoudre, et qui sort du cadre de cette étude.

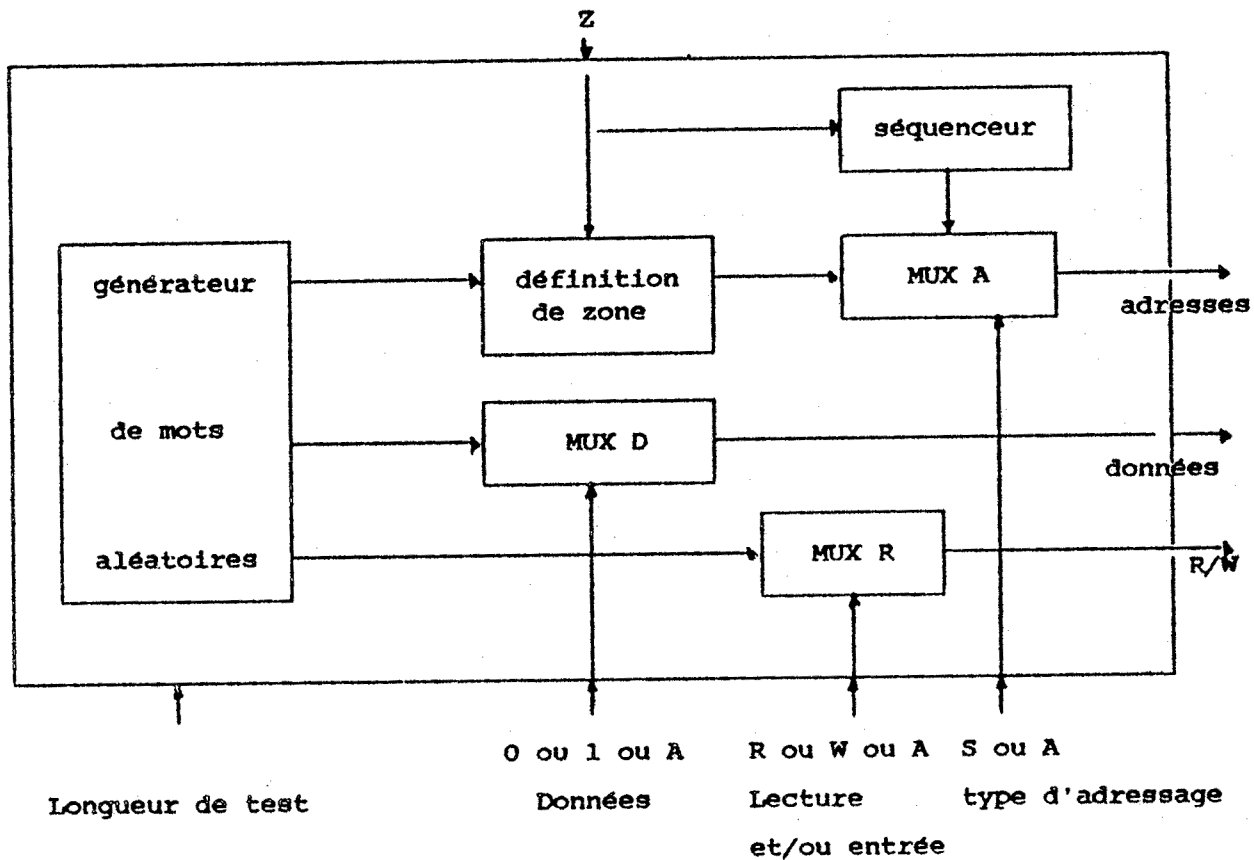


Figure 25 : Générateur d'expérience

### Le générateur d'expérience

le générateur de mots aléatoire peut être soit un registre à décalage , soit une génération logicielle.

la définition de la zone revient à fixer certains bits d'adresse à 0 ou 1.

le séquenceur est en fait un compteur qui décrit la zone adressée.

le multiplicateur d'adresse MUX A permet soit d'avoir un adressage aléatoire soit un adressage séquentiel.

le multiplicateur de donnée MUX D permet soit de fixer les données à 0 ou 1, soit d'avoir une séquence aléatoire.

le multiplicateur MUX R permet de choisir entre lecture, écriture ou séquence

aléatoire.

#### 7.IV- CONCLUSION

Bien que les deux solutions testeur câblé, adaptation sur un testeur existant par logiciel soient réalisables, il semblerait que la seconde soit préférable.

En effet, la réalisation d'un testeur demande de gros moyens de mise en oeuvre alors qu'il semble assez simple sur certains testeurs d'implanter du test aléatoire. On pourrait alors commander ces testeurs de façon à ce qu'ils déroulent la séquence de l'algorithme soit manuellement soit à partir d'un ordinateur personnel. Les seuls paramètres à transmettre étant ceux définis en B.II- 1. La réalisation du séquençement de l'algorithme pourrait être donnée par un logiciel simulateur à celui présenté dans l'annexe.

Chapitre 8

CONCLUSION



Nous voudrions souligner plusieurs points dans cette conclusion. Le premier concerne l'application du test aléatoire aux mémoires. D'un point de vue théorique, les résultats sont encourageants : pour tout type de panne, la longueur de la séquence nécessaire à sa détection est proportionnelle à la capacité de la mémoire  $N$  (nombre de points mémoire). En fait, une longueur de quelques centaines de  $N$  est nécessaire pour détecter tous les types de pannes étudiés. D'un point de vue pratique, ces résultats sont facilement applicables. Nul besoin de générer des séquences d'entrée compliquées, l'ensemble des vecteurs à envoyer est aisément réalisable : des adresses et des données aléatoires, on lit ou on écrit.

Contrairement aux tests déterministes, la détection d'un nouveau type de panne ne nécessite pas la modification de la séquence, tout au plus sa longueur. Il ne paraît pas nécessaire de vouloir développer de nouveaux algorithmes de test déterministes qui, de plus en plus compliqués, sont difficiles à mettre en œuvre sur des testeurs alors que des séquences aléatoires de longueurs raisonnables sont tout aussi efficaces.

Nous pensons qu'il est intéressant et même nécessaire, lorsqu'un circuit est défaillant, de déterminer la panne. Jusqu'à présent ceci était difficilement réalisable par du test aléatoire. L'existence d'une méthodologie de diagnostic devrait permettre d'augmenter la crédibilité de cette stratégie auprès des utilisateurs, même si les notions de test aléatoire et de diagnostic peuvent être au premier abord contradictoires. D'autre part, l'introduction du système expert dans le diagnostic devrait pouvoir augmenter sa puissance.

Autant de points qui sont en faveur de cette stratégie. Nous regrettons de n'avoir pu effectuer des expériences, ce serait là un début d'application de ces résultats. Le travail théorique étant fait nous espérons que son application dans le monde industriel soit proche.



ANNEXE

TARAM

UN LOGICIEL DE SYNTHÈSE



## 9.I- INTRODUCTION

Nous présentons dans cette annexe le logiciel de mise en oeuvre de nos résultats de l'étude sur le test aléatoire de mémoires RAM.

Ce logiciel permet d'une part de calculer les longueurs de test pour la détection de fautes et d'autre part donne un algorithme d'aide au diagnostic : c'est-à-dire un séquençement d'expériences de test conduisant à la détermination du type de faute présente.

Nous allons successivement développer chacun des modes de fonctionnement.

## 9.II. LE MODE DETECTION

### 9.II- 1. Introduction

Etant donné une mémoire et un modèle de faute, la longueur de test aléatoire  $L$  nécessaire à sa détection avec une probabilité  $(1-Q_D)$  est fonction de plusieurs paramètres (on suppose qu'il y a équiprobabilité de chacun des vecteurs de la séquence de test).

- $f$  la faute
- $n$  la capacité de la mémoire
- l'incertitude de détection ( $Q_D$ )
- l'état initial de la mémoire

En bonne approximation, nous pouvons supposer qu'il y a équiprobabilité d'être dans chacun des états de la chaîne à l'état initial.

On peut donc écrire  $L = L(f, n, Q_D)$ .

Le logiciel permet de fixer chacun de ces paramètres notamment de choisir la faute parmi un ensemble prédéfini et de calculer la longueur de détection  $L$  en fixant  $Q_D$ . Dans le cas où plusieurs fautes seraient possibles, le programme détermine la faute la plus difficile à détecter et calcule la longueur de détection, il calcule aussi l'incertitude de détection pour chacune des autres fautes : Ce mode de calcul est le "MODE 1".

Il nous a semblé intéressant de pouvoir fixer la longueur  $L$  et de calculer l'incertitude de détection pour chacune des fautes du modèle : C'est le "MODE 2".

## 9.II- 2. Organigramme de principe

Le déroulement du programme en mode de détection est le suivant

1) Définition du composant à tester

Avec acquisition des différents paramètres nécessaires aux calculs (capacité, temps de cycle...).

2) Choix du mode de calcul : MODE 1 ou MODE 23) En MODE 1 :- entrées

- l'opérateur fixe l'incertitude de détection et choisit les fautes possibles parmi un ensemble prédéfini.

- sorties

- le logiciel détermine la faute la plus difficile à détecter,  
- calcule la longueur permettant sa détection,  
- calcule l'incertitude de détection pour chacune des autres fautes du modèle,  
- affiche les résultats.

4) En MODE 2- entrées

- l'opérateur fixe la longueur de test et choisit un modèle de fautes.

-sorties

- le logiciel calcule l'incertitude de détection pour chacune des fautes du modèle et affiche les résultats.

5) FIN

### 9.II- 3. Définition du composant

Les différents paramètres concernant la mémoire dont nous avons besoin pour les calculs sont les suivants :

- le type de la mémoire : statique ou dynamique (si la mémoire est statique, la durée séparant deux rafraichissements),
- la capacité de la mémoire en nombre de mots,
- la nombre de bits par mots,
- la durée du cycle mémoire,
- l'existence de mode page ou nibble.

L'acquisition de ces différents paramètres est effectuée par le sous programme DEFMEM.

### 9.II- 4. Le choix du modèle de fautes

Une ou plusieurs fautes peuvent être choisies parmi la liste ci-dessous

#### a) Pour toutes les mémoires

- collage de point mémoire
- 2 couplage d'inversion ( $\uparrow i \Rightarrow \downarrow j$ )
- 2 couplage idempotent ( $\uparrow i \Rightarrow \downarrow j$ )
- 2 couplage logique ( $i = j = i \text{ ET } j$ )
- PSF passive ( $V_1$  cellules)
- PSF active inversion ( $V_2$  cellules)
- PSF active idempotente ( $V_3$  cellules)

#### b) Pour les RAM dynamiques seulement

Pour cette faute, une cellule, si elle n'est pas adressée, ne maintient sa valeur que pendant un temps  $t'$  inférieur au temps  $t$  donnée par le constructeur (durée entre deux rafraichissements). L'opérateur fixe le rapport  $t'/t$ .

c) Pour les RAM à mots de b bits

2-couplage logique dans un même mot.

Ce choix s'effectue à l'aide du sous programme CHOPAN. A chacune de ces fautes correspond une modélisation par graphe de Markov. Les matrices correspondantes sont stockées dans le sous programme DEFPAN.

## 9.II- 5. Détermination de la faute la plus difficile à détecter

A partir d'un ensemble de fautes choisi, le sous programme PANMAX détermine de façon approximative la longueur de détection pour chacune des fautes et détermine la faute la plus difficile à détecter.

La formule approchée pour la détermination de  $L(f)$  est la suivante :

$$L(f) = \frac{L_0(f) \times \log\left(\frac{1}{Q_D}\right)}{\log 10} \quad \text{avec } L_0(f), \text{ longueur de détection de } f \text{ pour } Q_D = 0.1$$

En effet, nous avons vu qu'on pouvait dire en bonne approximation que  $L(f)$  était proportionnelle au "nombre de 9" de  $1 - Q_D$ . Cette approche donne de bons résultats.

## 9.II- 6. Calcul de la longueur de test pour une faute donnée

Chaque faute est modélisée par une chaîne de Markov. On note  $M$  la matrice associée,  $P_0$  le vecteur d'état initial et  $P$  le vecteur d'état courant. On cherche à déterminer la longueur  $L$  de détection avec une probabilité  $c=1 - Q_D$ .

Cette longueur est donnée par la résolution du système

$$P_L = P_0 \times M^L$$

$$P_L(1) = c_L$$

où  $P_L(1)$  est la 1<sup>ère</sup> coordonnée du vecteur  $P_L$  et  $c_L = 1 - Q_D(L)$ .

Pour des mémoires de 1 M bit, on peut obtenir des longueurs de test de l'ordre de  $10^9$  ce qui rend prohibitif le calcul de L par une méthode qui calculerait successivement  $M^1, M^2, M^3, M^4, \dots$  puis vérifierait que  $P_i(1) = c_L$  (avec  $P_i = P_0 M^i$ ).

Nous avons donc utilisé l'artifice suivant :

1) On calcule successivement  $M^2, M^4, M^8, \dots$ , c'est-à-dire les puissances  $2^i$  de M jusqu'à obtenir  $M^{2^r}$  telle que

$$P_{2^{r-1}} = P_0 M^{2^{r-1}} \quad \text{et} \quad P_{2^r} = P_0 M^{2^r}$$

$$P_{2^{r-1}}(1) < c_L \quad \text{et} \quad P_{2^r}(1) > c_L$$

on obtient donc un intervalle  $[2^{r-1}, 2^r]$  contenant L, on garde en mémoire d'autre part les 20 dernières matrices  $[M^{2^{r-19}}, M^{2^{r-18}}, \dots, M^{2^r}]$ .

2) Ensuite, on affine le résultat, à savoir, on réduit l'intervalle contenant L par dichotomie.

Un exemple de calcul est détaillé ci-après, pour la clarté de l'exemple on a utilisé le stockage de 7 matrices ce qui diminue la précision de calcul de L. Dans la pratique, nous avons choisi de stocker 20 matrices pour avoir une précision relative sur L de  $1/(2^{20}-1)$  soit de  $5 \cdot 10^{-5}$  ce qui est largement suffisant pour le calcul de L.

### Exemple

1<sup>er</sup> pas : Stockage des matrices nécessaires et obtention de l'intervalle contenant L.

$M_1 = M^1$	$P_1 = P_0 \times M^1$	tel que $P_1(1) < Q_D$
$M_2 = M^2$	$P_2 = P_0 \times M^2$	tel que $P_2(1) < Q_D$
$M_3 = M^4$	$P_4 = P_0 \times M^4$	tel que $P_4(1) < Q_D$
$M_4 = M^8$	$P_8 = P_0 \times M^8$	tel que $P_8(1) < Q_D$
$M_5 = M^{16}$	$P_{16} = P_0 \times M^{16}$	tel que $P_{16}(1) < Q_D$
$M_6 = M^{32}$	$P_{32} = P_0 \times M^{32}$	tel que $P_{32}(1) < Q_D$
$M_7 = M^{64}$	$P_{64} = P_0 \times M^{64}$	tel que $P_{64}(1) < Q_D$

Deuxième remplissage (afin de ne pas dépasser le nombre de matrices mémorisées que nous nous sommes fixé, les premières calculées seront remplacées par les nouvelles obtenues)

$$M_1 = M^{128} \quad P = P_0 \times M^{128} \quad \text{tel que } P_{128}(1) < Q_D$$

$$M_2 = M^{256} \quad P = P_0 \times M^{256} \quad \text{tel que } P_{256}(1) > Q_D$$

donc  $128 < L < 256$

2<sup>ème</sup> pas : Affinement du résultat

$$M_1 \times M_7 = M^{192} \quad P_{192} = P_0 \times M^{192} \quad \text{tel que } P_{192}(1) > Q_D$$

donc  $128 < L < 192$

$$M_1 \times M_6 = M^{160} \quad P_{160} = P_0 \times M^{160} \quad \text{tel que } P_{160}(1) > Q_D$$

donc  $128 < L < 160$

$$M_1 \times M_5 = M^{144} \quad P_{144} = P_0 \times M^{144} \quad \text{tel que } P_{144}(1) > Q_D$$

donc  $128 < L < 144$

$$M_1 \times M_4 = M^{136} \quad P_{136} = P_0 \times M^{136} \quad \text{tel que } P_{136}(1) > Q_D$$

donc  $128 < L < 136$

$$M_1 \times M_3 = M^{132} \quad P_{132} = P_0 \times M^{132} \quad \text{tel que } P_{132}(1) < Q_D$$

donc  $132 < L < 136$  Résultat final.

$$\text{avec } P_{132}(1) = 0.9988 \quad \text{et} \quad P_{136}(1) = 0.9991$$

#### 9.II- 7. Détermination de $Q_D$ pour une faute donnée connaissant L

Pour ce sous programme, on a fixé L. On approxime L par une somme de puissance de 2 (à  $5 \cdot 10^{-5}$  près).

$$L = a_p 2^p + a_{p-1} 2^{p-1} + \dots + a_{p-19} 2^{p-19}$$

avec  $a_i = 0$  ou  $1$  pour  $i = p, \dots, p-19$ .

Ces coefficients  $a_i$  sont donnés par un sous programme DECLON.

On n'utilise que 20 puissances de 2 pour avoir la même précision de calcul que

lors du calcul de L (une précision relative de  $5 \cdot 10^{-5}$ ).

$$\text{On calcule ensuite } M = \sum_{i=p-20}^p a_i M_0^{2i} \quad \text{puis} \quad P_L = M \times P_0$$

et on obtient  $c_L = P_L(1)$ . Ceci est obtenu par le sous programme CALQD.

Etant donné qu'on calcule  $c_L$  et que  $Q_D$  peut varier de  $10^{-1}$  à  $10^{-100}$  (ou plus), le calculateur ne travaillant que sur 15 chiffres significatifs pour des valeurs de  $Q_D < 10^{-10}$ , on a des résultats de calculs sur  $c_L$  qui ne signifient rien. Il a donc fallu trouver un moyen de calculer  $Q_D$  pour des valeurs inférieures à  $10^{-10}$ . Nous avons utilisé une fonction qui est une bonne approximation [1], à savoir, L est proportionnelle au nombre de 9 de  $Q_D$ .

$$c_L = c_{L_0}$$

Pour des valeurs de  $c_L$  inférieures à  $10^{-10}$ , nous obtenons des valeurs "estimées" par ce moyen.

#### 9.II- 8. Affichage des résultats

a) En MODE 1 le programme affiche :

- la faute la plus difficile à détecter
- la longueur de la séquence de test (en nombre de cycles mémoire)
- le temps de test : il faut tenir compte pour les RAM dynamiques du temps de rafraichissement.
- le nombre d'accès mémoire  $K = \frac{L}{n}$
- l'incertitude de détection pour chacune des fautes de l'ensemble des fautes incluses dans les hypothèses.

b) En MODE 2 le programme affiche :

L'incertitude de détection pour chacune des fautes de l'ensemble des fautes incluses dans les hypothèses.

### 9.III- LE MODE DIAGNOSTIC

A partir d'un ensemble de fautes prédéfini, nous donnons ici une suite d'expériences qui permettent de déterminer le type de fautes qui affecte la mémoire

#### A) L'ensemble des fautes possibles est le suivant :

##### 1) Pour toutes les mémoires

- collages de point mémoire (a)
- collage de bit d'adresse (b)
- ligne toujours décodée (c)
- colonne toujours décodée (d)
- point mémoire toujours validé (e)
- faute de logique lecture/écriture (f)
- 2 couplages d'inversion (g)
- 2 couplages idempotent (h)
- PSF passive ( $V = 2$ ) (h) (l)
- PSF active d'inversion ( $V = 3$ ) (m)
- PSF active idempotente ( $V = 3$ ) (n)

##### 2) Pour les RAM dynamiques

- Non maintien du temps séparant deux rafraichissements. (p)

##### 3) En nibble mode

- fautes dans les registres (q)

#### B) Les expériences

Elles sont toutes caractérisées par une initialisation (à 0 :  $I_0$  ou à 1 :  $I_1$ ) d'une zone de la mémoire (cette zone,  $Z$ , est obtenue en fixant certains bits d'adresse à des valeurs 0 ou 1) et une séquence de test qui peut-être,

soit un test aléatoire d'une zone, ( $TA(Z)$ )

soit une lecture des cellules d'une zone  $L(Z)$ .



Suivant qu'on a ou pas détection d'erreur, on en tire des conclusions quant à la présence des diverses fautes.

Le séquençement des expériences est donné à la figure 23

A chaque expérience l'opérateur indique

- si la séquence a détecté une faute, (D) ou non (D')
- la longueur de détection L
- l'adresse de la cellule contenant une erreur.

L'algorithme indique alors l'ensemble des fautes possibles et donne la nouvelle expérience à effectuer.

Ce sous programme est nommé DIAGNO.

#### 9.IV- CONCLUSION

Ce logiciel pourrait servir de guide à quelqu'un voulant appliquer le test aléatoire sur une machine. Il permet de connaître les longueurs des séquences à appliquer pour détecter une faute, il donne pour une longueur la probabilité de détection. La partie diagnostic donne le séquençement des opérations à effectuer pour trouver la faute.

Il a été écrit en FORTRAN, pour plus de détails se reporter au listing (ou fichier source).

REFERENCES  
BIBLIOGRAPHIQUES

- [Aba 83] Abadir M.S. et Reghbatl H.K. "Functionnal testing of Semiconductor RAM", Computing Surveys, vol. 15, n° 3, Septembre 1983.
- [Bar 83] Bardell P.H. Notes sur le programme et les résultats de simulation transmises à R. David le 20 Avril 1983.
- [Cou 81] Courtois B. "Functionnal RAM testing. A unified model", rapport IMAG RR n° 235, Février 1985.
- [Dav 76] David R. et Blanchet B. "About random testing of combinational circuits", I.E.E.E. Trans. on Comp., Juin 1976.
- [Dav 85] David R. et Fedi X. "Résultats améliorés sur le test aléatoire de mémoires", RAIRO APII, vol. 19, n°6, 1985.
- [Dug 76] Dugard L. et Vray B. "Test aléatoire de mémoires MOS", Laboratoire d'Automatique, Projet d'élèves, I.N.P.G., Juin 1976.
- [Fos 77] Fosse P. and David R. " Random testing of memories", 7th Conf. Gesellschaft fu Informatik (GP'77), Nuremberg (R.F.A.), September 1977.
- [Fue 86] Fuentes A., David R. and Courtois B. "Random testing versus Deterministic Testing of RAMs", 16th Fault Tolerant Computing Symposium, Vienne (Autriche), Juin 1986, pp 266-271.
- [Hay 75] Hayes J.P. "Detection of pattern sensitive faults in random access memories", I.E.E.E. trans. on Comp., vol. C-24, n° 2, Février 1975.
- [Hay 80] Hayes J.P. "Testing memories for single cell patern sensitive faults", I.E.E.E. Trans. on Comp., vol. C-29, n° 3, Mars 1980.

- [Kem 60]           Kemeny J. and Snell J. "Finite Markov Chain", Van Nostrand, Princeton, 1960.
- [Kna 77-1]         Knaizuk J. et Hartman J. et C. "An algorithm for testing RAM", I.E.E.E. Trans. on Comp., Avril 1977.
- [Kna 77-2]         Knaizuk J. et Hartman J. et C. "An optimal algorithm for testing stuck-at faults in RAM", I.E.E.E. Trans. on Comp., vol. C-26, n° 11, November 1977.
- [Lil 77]           Lilen H. "Mémoires intégrées" Ed. Radio.
- [Mar 80-1]         Marinescu M. "Test fonctionnel de mémoires vives à large couverture de fautes", rapport IMAG RR n° 191, Mars 1980.
- [Mar 80-2]         Marinescu M. "Test fonctionnel de mémoires vives à large couverture de fautes", Proc. of Int. Conf. on Reliability and Maintainability, Perros-Guirec, Tregastel, 8-12 Septembre 1980.
- [Mar 82]           Marinescu M. "Simple and efficient algorithms for fonctionnal RAM testing", I.E.E.E. Test Conf., 1982.
- [MCA 84]           Mc Anney W.H., Bardell P.H. et Gupta V.P. "Random testing for stuck-at storage cells in an embedded memory", I.E.E.E. Int. Test Conf., 1984.
- [Met 74]           Metger G. et Vabre J.P. "Les mémoires électroniques", Ed. Masson & Cie.
- [Nai 78]           Nair R., Thatte S.M., and Abraham S. "Efficient algorithms for testing semiconductor RAM", I.E.E.E. Trans. on Comp., vol. C-27, n° 6, Juin 1978.
- [Nai 79]           Nair R. "Comments on an optimal algorithm for testing stuck-at faults in RAM", I.E.E.E. Trans. on Comp., vol. C-28, n° 3, Mars 1979.

- [Pap 85] Papachistou C. et Sagall N. "An improved method for detecting fonctionnal faults in semiconductor random access memories" I.E.E.E. Trans. on Comp., Février 1985.
- [Pic 84] Notes manuscrites d'un cours sur les mémoires transmises à R. David par S. Picard (Compec IBM) Février 1984.
- [Sah 84] Sahami H. et Courtois B. "Functionnal testing vs structural testing of RAM", Congrès de Bonn, Septembre 1984.
- [She 76] Shedletsky J.J. and Mc Cluskey E.J. "The error latency of a fault in a sequential digital circuit", I.E.E.E. Trans. on Comp., Juin 1976, pp 655-659.
- [Sri 77] Srini V.P. "API tests for RAM chips", Computer, Juillet 1977.
- [Suk 78] Suk D.S. "Functionnal and pattern sensitive fault testing algorithms for semiconductor random access memories", Ph. D. thesis, University of Iowa, Juillet 1978.
- [Suk 79] Suk D.S. et Reddy S.M. "An algorithm to detect a class of PSF in RAM", 9th Fault Tol. Comp., Symp., Juin 1979.
- [Suk 80] Suk D.S. et Reddy S.M. "Test Procedures for a class of PSF in RAM", I.E.E.E Trans. on Comp., vol. C-29, n° 6, Juin 1980.
- [Suk 81] Suk D.S. et Reddy S.M. "A march test for fonctionnal faults in RAM", I.E.E.E Trans. on Comp., vol C-30, n° 12, Décembre 1981.
- [Tha 77] Thatte S.M. "Fault diagnostic of semiconductor random access memories", Coordinated Science Laboratory, University of Illinois, Mai 1977.
- [Tha 77-2] Thatte S.M. et Abraham J. "Testing of semiconductor RAM", Proc. 7th FTCS Juin 1977.
- [The 78] Thevenod-Fosse P. et David R. "Test aléatoire de mémoires", RAIRO, vol. 12, n° 1, 1978.

## AUTORISATION de SOUTENANCE

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

VU les rapports de présentation de Messieurs

- . R. DAVID, Directeur de recherche
- . R. JAZE, Ingénieur

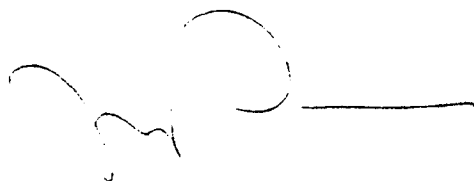
**Monsieur Antoine FUENTES**

est autorisé à présenter une thèse en soutenance en vue de l'obtention du diplôme de DOCTEUR-INGENIEUR, spécialité "Automatique et traitement du signal".

Fait à Grenoble, le 2 décembre 1986

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

*P.O. le Vice-Président,*



*Il cherche la fameuse machine à peser les balances !  
Jacques PREVERT (Paroles)*

*à Gisèle*