# Conception de contrôleurs autotestables pour des hypothèses de pannes analytiques

Ingrid Eleonora Schreiber Jansch

# THESE

présentée à

**l'Institut National Polytechnique de Grenoble**

en vue d'obtenir le titre de
DOCTEUR-INGENIEUR
" Microélectronique"

par

Ingrid Eleonora SCHREIBER JANSCH

*********

# CONCEPTION DE CONTROLEURS AUTOTESTABLES POUR DES HYPOTHESES DE PANNES ANALYTIQUES

*********

Soutenue le 14 janvier 1985 devant la Commission d'Examen.

M. BOLLIET Louis PRESIDENT
MM. COURTOIS Bernard
DAVID René
DIAZ Michel
LARCHER Simon EXAMINATEURS

# INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

**Président:** Daniel BLOCH
**Vice-Présidents:** René CARRE
Hervé CHERADAME
Jean-Pierre LONGEQUEUE

Année universitaire 1983-1984

## Professeurs des Universités

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

| | | | |
|---|---|---|---|
| BONNETAIN | Lucien | PARIAUD | Jean-Charles |
| BONNIER | Etienne | RAMEAU | Jean-Jacques |
| DURAND | Françis | SOHM | Jean-Claude |
| GUYOT | Pierre | SOUQUET | Jean-Louis |
| LOUCHET | François | | |

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

| | | | |
|---|---|---|---|
| BARIBAUD | Michel | GENTY | Pierre |
| BLIMAN | Samuel | GUERIN | Bernard |
| BUYLE BODIN | Maurice | POUPOT | Christian |
| CHENEVIER | Pierre | SERMET | Pierre |
| COHEN | Joseph | ZADWORNY | François |
| COUMES | André | | |

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

| | | | |
|---|---|---|---|
| BARRAUD | Alain | JAUSSAUD | Pierre |
| BAUDELET | Bernard | JOUBERT | Jean-Claude |
| BESSON | Jean | JOURDAIN | Geneviève |
| BLOCH | Daniel | LACOUME | Jean-Louis |
| BRISSONNEAU | Pierre | LONGEQUEUE | Jean-Pierre |
| CAVAIGNAC | Jean-François | MASSELOT | Christian |
| CHARTIER | Germain | MORET | Roger |

| | | | |
|---|---|---|---|
| CHERUY | Arlette | PAUTHENET | René |
| DURAND | Jean-Louis | PERRET | René |
| FELICI | Noël | PERRET | Robert |
| FOULARD | Claude | POLOUJADOFF | Michel |
| GAUBERT | Claude | SABONNADIERE | Jean-Claude |
| IVANES | Marcel | SCHLENKER | Claire |
| JALINIER | Jean-Michel | SCHLENKER | Michel |

**E.N.S.H.G.**

| | | | |
|---|---|---|---|
| BOIS | Philippe | LESPINARD | Georges |
| BOUVARD | Maurice | MOREAU | René |
| LESIEUR | Marcel | PIAU | Jean-Michel |

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

| | | | |
|---|---|---|---|
| ANCEAU | François | MOSSIERE | Jacques |
| FONLUPT | Jean | ROBERT | François |
| LATOMBE | Jean-Claude | SAUCIER | Gabrielle |
| MAZARE | Guy | VEILLON | Gérard |

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

| | | | |
|---|---|---|---|
| CHERADAME | Hervé | RENAUD | Maurice |
| CHIAVERINA | Jean | ROBERT | André |
| GANDINI | Alessandro | SILVY | Jacques |

## Professeurs Associés

| | | |
|---|---|---|
| BLACKWELDER | Ronald | ENSHG |
| HAYASHI | Hirashi | ENSIEG |
| PURDY | Gary | ENSEEG |

## Professeurs Université des Sciences Sociales (Grenoble II)

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

## Chercheurs du C.N.R.S.

### Directeurs de recherche

| | |
|---|---|
| FRUCHARD | Robert |
| JORRAND | Philippe |
| VACHAUD | Georges |

### Maîtres de recherche

| | | | |
|---|---|---|---|
| ALLIBERT | Michel | JOUD | Jean-Charles |
| ANSARA | Ibrahim | KAMARINOS | Georges |
| ARMAND | Michel | KLEITZ | Michel |
| BINDER | Gilbert | LANDAU | Ioan-Dore |
| BORNARD | Guy | LASJAUNIAS | Jean-Claude |
| CARRE | René | MERMET | Jean |
| DAVID | René | MUNIER | Jacques |
| DESPORTES | Jacques | PIAU | Monique |
| DRIOLE | Jean | PORTESEIL | Jean-Louis |
| GIGNOUX | Damien | SUERY | Michel |
| GIVORD | Dominique | THOLENCE | Jean-Louis |
| GUELIN | Pierre | VERDILLON | André |
| HOPFINGER | Emile | | |

### Personnalités habilitées à diriger des travaux de recherche

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

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

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

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

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

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

**E.N.S.H.G.**

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

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

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

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

| | |
|---|---|
| CHARUEL | Robert |

**C.E.N.G.**

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

## Laboratoires extérieurs

### C.N.E.T.

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

### I.N.S.A. Lyon

| | |
|---|---|
| GAUBERT | C. |

# ECOLE NATIONALE SUPERIEURE DES MINES DE SAINT-ETIENNE

**Directeur** : M. M. MERMET
**Directeur des Etudes et de la formation** : M. J. LEVASSEUR
**Directeur des Recherches** : M. J. LEVY
**Secrétaire Général** : Mle M. CLERGUE

## Professeurs de 1èré catégorie

| | | | |
|---|---|---|---|
| COINDE | Alexandre | LOWYS | Jean-Pierre |
| FORMERY | Philippe | MATHON | Albert |
| GOUX | Claude | RIEU | Jean |
| LEVY | Jacques | SOUSTELLE | Michel |

## Professeurs de 2èré catégorie

| | | | |
|---|---|---|---|
| HABIB | Michel | TOUCHARD | Bernard |
| PERRIN | Michel | VERCHERY | Georges |

## Directeur de recherche

| | |
|---|---|
| LESBATS | Pierre |

## Maîtres de recherche

| | | | |
|---|---|---|---|
| BISCONDI | Michel | LANCELOT | Françis |
| DAVOINE | Philippe | LE COZE | Jean |
| FOURDEUX | Angeline | THEVENOT | François |
| KOBYLANSKI | André | TRAN MINH | Canh |
| LALAUZE | René | | |

## Personnalités habilitées à diriger des travaux de recherche

| | | | |
|---|---|---|---|
| DRIVER | Julian | THOMAS | Gérard |
| GUILHOT | Bernard | | |

## Professeur à l'UER de Sciences de Saint-Etienne

| | |
|---|---|
| VERGNAUD | Jean-Maurice |

# AVANT-PROPOS

Je tiens à remercier :

- Monsieur Bernard COURTOIS, Chargé de Recherche au CNRS et responsable de l'équipe d'Architecture des Ordinateurs / Sureté de Fonctionnement au Laboratoire TIM3 (IMAG), pour avoir pris la responsabilité de cette thèse. Que ce document soit pour lui le témoignage de ma reconnaissance pour l'orientation qu'il a su donner à mes travaux, pour son interêt, ses conseils, ses encouragements et surtout pour l'enthousiasme qu'il m'a toujours transmis.

- Monsieur Louis BOLLIET, Professeur à l'Université de Grenoble, pour l'honneur qu'il me fait en acceptant de présider le jury de cette thèse.

- Monsieur René DAVID, Directeur de Recherche au CNRS et responsable de l'équipe de Systèmes Logiques et Discrets au Laboratoire d'Automatique de Grenoble (LAG), qui a bien voulu participer au jury de cette thèse. Je le remercie aussi pour ses conseils au tout début de cette recherche, et pour les critiques fructueuses de ce texte.

- Monsieur Michel DIAZ, Maître de Recherche au CNRS et responsable de l'équipe Logiciel et Communication au LAAS, à Toulouse, qui a bien voulu juger mon travail et faire partie du jury. Je le remercie aussi pour ses conseils qui m'ont aidé à améliorer la présentation finale de cet exemplaire.

- Monsieur Simon LARCHER, Directeur Scientifique Adjoint à la Compagnie de Signaux et d'Entreprises Electriques (CSEE), à Paris, pour avoir accepté de faire partie du jury de ma thèse.

- Monsieur François ANCEAU, Professeur à l'Institut National Polytechnique de Grenoble, pour m'avoir accueilli dans son équipe de recherche à mon arrivée à Grenoble, et qui est, à présent, chez Bull, à Paris.

- Mon collègue Michaelis NICOLAIDIS qui s'est montré toujours

disponible, patient et décontracté pendant nos discussions. Celles-ci ont largement contribué à la progression de mes recherches.

- Mes collègues de l'équipe et du personnel administratif pour leur collaboration en des moments divers et à différents titres. Qu'ils sachent aussi que par leur sympathie et amitié, ils ont aidé à rendre agréable mon séjour en France.

- Mlle Isabelle CAILLAT, qui a assuré l'entrée de la totalité du texte dans l'éditeur d'une manière très rapide et intelligente.

- Au service de Reprographie de l'ENSIMAG qui a assuré le tirage de ce document.

- L'"Universidade Federal do Rio Grande do Sul", à Porto Alegre, et en particulier le "Departamento de Informática" du "Centro de Processamento de Dados" et la "Pós-Graduação em Ciência da Computação", pour leur autorisation qui m'a permis de réaliser ce programme de doctorat.

- La CAPES - "Coordenação para o Aperfeiçoamento de Pessoal de Nível Superior", entité appartennante au Ministère d'Education et Culture brésilien, pour le soutien financier.

## RESUME

Dans cette étude nous nous intéressons aux contrôleurs utilisés dans des systèmes autotestables, pour le test des sorties, combinatoires ou séquentielles, du bloc fonctionnel.

Deux classes de contrôleurs sont abordées : les "Strongly Code Disjoint" (SCD) qui vérifient une propriété combinatoire, et les "Strongly Language Disjoint" (SLD), où la propriété vérifiée est séquentielle.

Pour la première, nous examinons la conception des contrôleurs NMOS à partir de l'assemblage des cellules, des règles de conception pour celles-ci, et des hypothèses de pannes pouvant survenir dans les systèmes aussi bien que dans quelques structures spécifiques de contrôleurs.

Les contrôleurs "Strongly Language Disjoint" définis ici composent la plus large classe qui, associée à des circuits "sequentially self-checking", permet au système d'atteindre le "TSC goal" sous certaines hypothèses de pannes. Ils conservent la propriété "language-disjoint" même en présence de fautes. Des propositions pour la conception de ces contrôleurs sont également données -nous vérifions la possibilité de les construire à partir de blocs combinatoires.

Toutes les considérations pratiques sont basées sur des hypothèses de pannes analytiques.



## MOTS-CLES

circuits autotestables - circuits autocontrôlables - contrôleurs - codes - conception de contrôleurs - contrôleurs "Strongly Code Disjoint" - contrôleurs "Strongly Language Disjoint" - test (en ligne/hors ligne) - conception VLSI.

# SOMMAIRE

# CHAPITRE 1:

## INTRODUCTION

L'utilisation de circuits ayant des propriétés d'autotestabilité est intéressante dans les systèmes digitaux pour des applications où la sûreté de fonctionnement est nécessaire, car ils permettent la détection immédiate des erreurs. Les difficultés liées à la conception de ces circuits sont dues surtout au fait que l'on cherche à avoir des circuits conçus avec le minimum d'augmentation de surface (donc, minimum de redondance) possible, avec la capacité majeure de détection de pannes (voire la capacité totale de détection de toutes les pannes qui peuvent survenir selon les hypothèses considérées).

Par la suite, et dans tout le texte de la thèse, nous utilisons les mots "panne", "faute" et "défaut" comme des synonymes ayant le sens d'"une modification dans la structure physique interne du circuit (produite par une défaillance électrique) laquelle peut résulter ou non en un changement de la fonction éxécutée". En général, on préfère le terme "panne" dans le sens physique, tandis que les deux autres s'appliquent aussi pour le sens logique, comme par exemple aux valeurs des vecteurs d'entrée. L'"erreur" identifie toujours le résultat d'une faute : le signal incorrect généré à la suite d'une panne.

En outre, nous gardons la terminologie de définition des circuits selon la dénomination originale en anglais afin d'éviter toute confusion possible. Les expressions "autotestable" et "autocontrôlable" sont utilisées dans le sens générale d'un circuit avec des propriétés de surveillance de fonctionnement, en équivalence au terme anglais "self-checking". Toutes les deux sont trouvées dans la litérature.

Le système considéré a la structure classique représentée à la figure 1. Il se compose d'un circuit fonctionnel et d'un contrôleur. Le premier éxécute la fonction de base du circuit et a ses sorties codées de telle manière que, à partir de leur analyse, l'existence d'un défaut dans le circuit soit aperçue. Les valeurs en provenance du bloc fonctionnel sont reçues par le contrôleur lequel doit, à côté de sa capacité de détection des erreurs parmi les lignes d'entrée reçues, pouvoir aussi détecter les défauts qui surviennent à lui-même. C'est lui qui produit l'indication générale d'erreur. Les propriétés globales et les hypothèses d'occurrence de défauts dans le système dépendent de son organisation interne, des codes utilisés pour l'information et, avant tout, des définitions de chacun des blocs composants. Ces propriétés alliées à certaines hypothèses ont le but d'assurer que la première sortie erronée du circuit sera reconnue, donc détectée - ce but est référé comme le "totally self-checking goal" (objectif TSC).

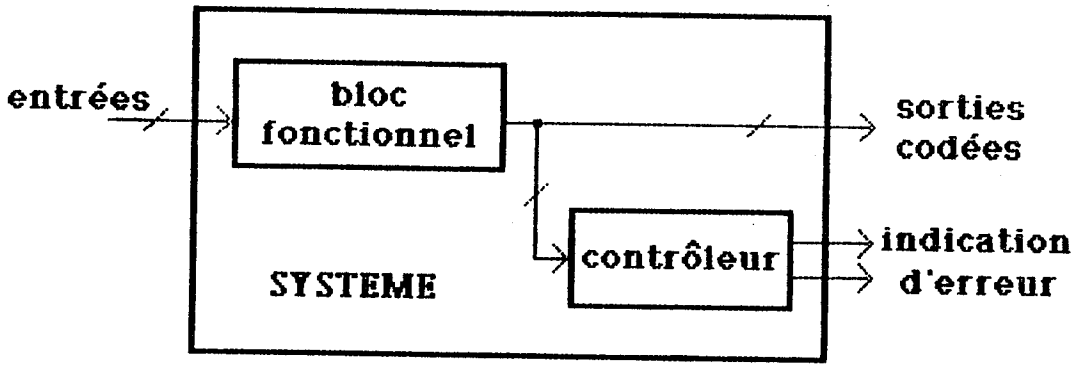



Figure 1 : Structure classique pour un système autotestable

Nicolaïdis et Courtois ont étudié la conception des blocs fonctionnels pour être utilisés dans ces systèmes, les circuits "strongly fault-secure", et ont défini des contrôleurs qui leur sont associés, les contrôleurs "strongly code disjoint". Poursuivons dans la conception de ces derniers et etendons ces définitions pour d'autres employées aux sorties des blocs fonctionnels ayant des propriétés séquentielles - donc, une classe plus ample.

Selon la théorie classique, on utilise un circuit "self-testing"(ST) (auto-testable) et "fault secure"(SF) (sûr à fautes) pour la partie fonctionnel et un circuit "self-testing", "fault secure" et "code disjoint"(CD) (code-disjoint) pour le contrôleur afin d'avoir un circuit "totally self-checking" (TSC) (totalement auto-contrôlable) avec lequel on peut assurer que toute panne y survenant est indiquée aux sorties du contrôleur (une erreur est produite).

Mais ces propriétés sont très restrictives et impliquent trop de contraintes pour les procédures de conception des circuits. Et la plupart des pannes qui ne causent pas de changement de la fonction éxécutée par le circuit, peuvent être acceptées. Les circuits "strongly code disjoint"(SCD) (fortement code-disjoint) considèrent cette possibilité et, utilisés sous certaines hypothèses de pannes assurent que la première sortie erronée n'appartient pas au code de sortie : par conséquent, le défaut est détecté et le "TSC goal" est achevé. L'ensemble circuit SFS / contrôleur SCD compose ainsi un système combinatoire autocontrôlable. Les définitions des circuits SFS et SCD étant moins restrictives que leurs "sources" classiques, sa conception peut être éxécutée plus facilement que celle des circuits SC/FS et SC/FS/CD.

Dans les systèmes où les sorties du bloc fonctionnel ont un comportement séquentiel, il faut disposer d'un contrôleur pour ces sorties

ayant des propriétés séquentielles : les contrôleurs "strongly language disjoint"(SLD) (fortement langage-disjoint) sont ainsi définis dans ce document. Le bloc fonctionnel "sequentially self-checking"(SeSC) (sequentiallement auto-contrôlable) produit des sorties décrites par un langage. Ce-ci est vérifié par le contrôleur qui donne l'indication d'erreur, étant le cas. Parmi les publications plus anciennes, nous ne trouvons pas de références à des contrôleurs séquentiels. Les systèmes possédant ces propriétés avaient, en général, les états internes codés et ceux-ci étaient vérifiés par un contrôleur combinatoire classique (TSC). Les sorties du système ne pouvaient être que combinatoires. Plus récemment, Viaud et David ont défini les contrôleurs "SeSC" et "language-disjoint" pour le test des sorties séquentielles - leur inconvénient est de ne pas assurer la propriété "language-disjoint" en présence de défauts. A l'exemple des circuits fonctionnels étudiés auparavant, pour les contrôleurs il faut aussi considérer que les fautes pour lesquelles le circuit est redondant peuvent survenir. La propriété "language-disjoint" devra être gardée dans ce cas là, de même en présence d'un défaut quelconque avant sa détection. Ceci est le concept de base des contrôleurs "strongly language disjoint". Ainsi, l'ensemble circuit SeSC/contrôleur SLD compose un système séquentiel autocontrôlable lequel, dans certaines hypothèses de pannes, accomplit le but du "TSC goal". Donc les contrôleurs SLD sont la plus large classe de contrôleurs et représentent pour les systèmes séquentiels ce que les contrôleurs SCD sont pour les systèmes combinatoires.

A partir des définitions des circuits SLD, leur conception peut sembler compliquée. Elle le serait sûrement, si l'on suivait les mêmes procédures que celles des circuits séquentiels classiques. Cependant, l'utilisation d'éléments de base suffisamment étudiés dans la théorie des circuits combinatoires autotestables apporte des solutions simples au problème. A côté de ces aspects, on envisage aussi la possibilité d'utiliser des structures régulières dans l'ensemble du dessin afin de pouvoir automatiser toute l'activité de conception.

Un autre aspect relatif à la théorie classique concerne les modèles de pannes y considérés. Avant, on utilisait des modèles logiques de pannes, les fautes étant représentées par des collages à des valeurs logiques aux entrées ou sorties des portes logiques. Galiay et alli [GAL 80] ont démontré l'inefficacité de ce modèle, ne pouvant pas représenter toutes les situations de fautes possibles. Leur proposition consiste à analyser les situations possibles de défauts à partir du schéma électrique.

En fait, ce modèle est suffisant pour la conception des circuits et pour la génération des vecteurs de test mais des possibilités y représentées ne

sont pas réelles en termes de l'implémentation résultante. La représentation optimale est celle qui correspond plus directement au circuit conçu: l'approche topologique. A partir des types de pannes pouvant survenir physiquement, on fait l'analyse sur le dessin topologique du circuit. Une part des fautes considérées possibles auparavant pour le circuit électrique peut être ainsi éliminée, car il n'y a pas de raison physique pour qu'elles surviennent.

Les modèles de pannes réelles pouvant se produire dans le circuit sont dits "hypothèses de pannes de bas-niveau" (analytiques, physiques ou réelles). Comme hypothèses, nous considérons l'ensemble réuni sous la Classe I selon la classification de Courtois en [COU 81]. Appartiennent à cette classe, les défauts suivants : un contact ou pré-contact défaillant, un MOS défaillant (s-on ou s-open), un conducteur coupé, et une grille flottante (à laquelle est associé un MOS s-on ou s-open comme mode de panne). Les courts-circuits sont admis entre une ligne d'aluminium et l'autre aluminium le plus proche géographiquement, et le cas similaire entre deux diffusions, bien que cela soit moins probable. On considère un seul défaut physique à la fois. Dans les exemples et pour les considérations concernant exclusivement l'implémentation, la technologie NMOS est utilisée.

Ensuite, nous expliquons le plan de présentation de la thèse.

Le chapitre II consiste en un résumé de ce que l'on trouve dans la théorie classique concernant les contrôleurs, reprise comme un appui initial pour la conception des contrôleurs "strongly language disjoint". Les définitions de base des circuits "totally self-checking" sont présentées, ainsi que les codes classiques utilisés pour la codification des sorties du bloc fonctionnel (donc, les entrées des contrôleurs). Des circuits logiques pouvant être associés à chacun des cas sont exemplifiés. Ces circuits sont passibles d'être employés comme une proposition initiale pour la conception des contrôleurs SCD mais nous vérifierons, plus tard, que ces propositions peuvent ne pas être optimales. En outre, des circuits rejettés pour ne pas satisfaire les paramètres classiques peuvent être étudiés en vue de l'implémentation d'un circuit SCD.

Dans le chapitre III, nous rassemblons les aspects concernants la conception de la plus large classe de contrôleurs utilisés dans les systèmes combinatoires, basée sur des hypothèses de pannes analytiques - les contrôleurs SCD. Tout d'abord sont données les définitions et les propriétés de ces contrôleurs. Une hypothèse nouvelle d'occurrence de défauts est développée: on considère la possibilité de fautes dans les différentes unités, le bloc fonctionnel et le contrôleur, avant la détection de la première (par

rapport au temps, à ne pas confondre avec l'ordre d'énumération). Ayant remarqué la difficulté de donner des règles précises pour la conception des contrôleurs en général, nous trouvons plus intéressant l'utilisation des cellules pré-dessinées - la conception devient une activité d'assemblage et d'interconnexion de ces cellules. Donc, la conception des cellules de contrôleurs est analysée vis-à-vis de la propriété SCD et en vue de leur assemblage (contrôleurs SCD cellulaires). Des exemples de contrôleurs double-rail et à parité sont présentés. Enfin, nous examinons la situation d'une structure multi-contrôleur SCD utilisée pour la détection de pannes pendant le fonctionnement en ligne aussi bien que hors ligne. Cette application en vue de la maintenance avait été suggérée initialement par Nicolaïdis et Courtois.

Les contrôleurs "strongly language disjoint" sont définis dans le chapitre IV. Ce sont les contrôleurs pour les systèmes séquentiels, et nous avons pris pour base les définitions des circuits "sequentially self-checking" données par Viaud et David. Le rapport entre les définitions utilisées pour les systèmes séquentiels et combinatoires est examiné en vue de la dérivation de ces dernières à partir des premières. Des hypothèses de pannes diverses qui peuvent être associées aux systèmes séquentiels sont aussi étudiées. L'utilisation de structures régulières étant aussi intéressante pour la conception des contrôleurs SLD motive la section IV.4., où nous regardons, soigneusement, l'emploi des blocs combinatoires internes dans la construction de ces contrôleurs.

Au chapitre V sont résumées les conclusions plus importantes des deux chapitres précédents et ajoutées des considérations en vue de l'application et de la poursuite de cette étude.

# CHAPITRE 2:

# CIRCUITS AUTOTESTABLES ET CONTROLEURS CLASSIQUES

# CIRCUITS AUTOTESTABLES ET CONTROLEURS CLASSIQUES

## II.1. DEFINITIONS DE BASE : CIRCUITS ET CONTROLEURS TSC

Nous commençons en introduisant des concepts et des définitions liées au domaine des circuits combinatoires autocontrôlables selon la théorie classique. Les conventions qui suivent sont reprises de [SMI 78]. Nous considérons un circuit G en logique combinatoire possédant des entrées multiples et des sorties multiples. Si on a un circuit G avec r entrées primaires, les $2^r$ vecteurs binaires de longueur r forment l'ensemble X des vecteurs d'entrée. L'ensemble Y de vecteurs de sortie est formé par les $2^q$ vecteurs binaires de longueur q, le nombre des sorties du circuit G étant égal à q.

Pendant le fonctionnement normal (avant l'occurrence des défauts), le circuit G reçoit à ses entrées un sous-ensemble A de vecteurs de X, appelé code d'entrée et il produit un sous-ensemble B de vecteurs de Y, appelé code de sortie. En présence de défauts, le circuit peut produire des sorties en dehors du code de sortie B.

En présence d'un défaut f, la sortie du circuit G qui reçoit un vecteur d'entrée x, peut être dénotée G(x,f). Avant l'occurrence des défauts, elle est dénotée G(x,ø).

Les définitions suivantes sont dues à ANDERSON [AND 71]. On considérera un bloc fonctionnel G, avec un code d'entrée A, un code de sortie B et un ensemble de défauts F.

**Définition D1**
Un circuit G est "fault secure" (FS) pour un ensemble de défauts F si pour toutes les fautes appartenant à F et pour tous les vecteurs d'entrée, soit la sortie est correcte soit elle est un mot hors code, i.e. :

$$\forall f \in F, \forall a \in A, \text{ soit } G(a,f) = G(a,\emptyset)$$
$$\text{soit } G(a,f) \notin B.$$

**Définition D2**
Un circuit G est "self-testing" pour un ensemble F de défauts si pour chaque défaut appartenant à F, il y a au moins un vecteur d'entrée que produit une sortie en dehors du code ; i.e.

$$\forall f \in F, \exists a \in A \text{ tel que } G(a,f) \notin B.$$

**Définition D3**
Un circuit G est "totally self-checking" (TSC) pour un ensemble de défauts F si le circuit est "fault secure" and "self-testing" pour l'ensemble F.

Donc, les définitions ci-dessus supposent qu'une sortie erronée provoquée par une faute dans un circuit "fault secure" est toujours un vecteur en dehors du code. Et un circuit "self-testing" peut être vérifié avec que des vecteurs d'entrée, dès que au moins un de ces vecteurs cause une sortie en dehors du code pour toute faute. Par conséquent, aux circuits qui sont à la fois "self-testing" et "fault secure" (les circuits TSC), toutes les fautes produiront une sortie erronée détectable.

**Définition D4**
Un circuit est "code disjoint" s'il produit toujours une sortie hors code pour un vecteur hors code appliqué aux entrées, pendant le fonctionnement normal; i.e.,

$$\forall a \in A, G(a,\varnothing) \in B$$
$$\forall a \notin A, G(a,\varnothing) \notin B.$$

**Définition D5**
Un circuit est un contrôleur TSC s'il est "totally self-checking" et "code disjoint".

Un bloc fonctionnel G accomplit le "Totally Self-Checking goal" (TSC goal) si la première sortie erronée due aux défauts de l'ensemble F, est en dehors du code de sortie. Donc, elle sera détectée par le contrôleur TSC. L'hypothèse d'occurrence de fautes normalement utilisée est référée comme hypothèse H1 et énoncée ci-dessous.

**Hypothèse H1**
Entre l'occurrence de deux défauts quelconques appartenant à l'ensemble F, il s'écoule un laps de temps suffisant pour que tous les vecteurs du code d'entrée A soient appliqués aux entrées du circuit G.

L'hypothèse H1 étant respectée, un circuit TSC accomplit le "totally self-checking goal".

## II.2. CODES ET CONTROLEURS CLASSIQUES

Dans ce chapitre, sont décrits des codes souvent utilisés à la conception des systèmes totalement autocontrôlables. Dans les prochaines sections, seront présentés les codes de parité, double-rail, m-parmi-n, Berger et Berger modifié et les principaux contrôleurs classiques associés à ces codes.

D'abord, afin d'éviter toute confusion, des expressions et des notations

utilisées dans les pages suivantes sont définies.

Couche logique : La couche logique correspond à une porte composée d'un ensemble des éléments comprenant un transistor de charge et les transistors de commande qui lui sont associés, ceux-ci pouvant être groupés en fonction OU et ET.

Circuit actif : Le circuit actif [AND 71] est appliqué aux circuits composés des lignes telles qu'elles prennent les valeurs 0 ou 1, pour les entrées qui appartiennent au code. Cette condition permet l'observabilité du circuit et est nécessaire (mais pas suffisante) pour la détection des défauts.

Les circuits "self-testing" ne peuvent pas avoir des lignes qui restent avec la valeur logique constante, pendant le fonctionnement normal. Un circuit passif est défini comme ayant des lignes qui prennent des valeurs constantes pour les entrées du code et ne les changent qu'en cas d'erreur.

Dans les figures nous utilisons les abréviations suivantes, les mêmes présentées par Crouzet [CRO 78] :

$n_T$ = nombre de transistors

$n_C$ = nombre de couches logiques

$n_P$ = nombre de portes MOS (comme défini dans "couche logique")

entrance max OU = nombre maximal d'entrées aux portes OU

entrance max ET = nombre maximal d'entrées aux portes ET.

## II.2.1. Codes de parité

### II.2.1.1. Généralités

Les codes de parité sont des codes séparables simples. Ils sont définis ci-dessous.

**Définition Dé**

Un ensemble de configurations binaires de n variables $e_i$ forme un code de parité si la relation suivante est verifiée:

$$e_1 \oplus e_2 \oplus e_3 \oplus \ldots\ldots\ldots\ldots \oplus e_n = a \qquad (2.1)$$

où a = 0 pour un code de parité paire,

a = 1 pour un code de parité impaire.

Le code est formé par adjonction d'une variable de parité. Il permet la détection d'une erreur simple. L'extension de ce code à la détection de fautes multiples est constituée par les codes b-adjacents. Ces codes sont obtenus en intercalant les variables de b codes de parité simple.

II.2.1.2. Circuits de contrôle

Du point de vue pratique, le contrôle d'un code de parité teste si l'équation 2.1. est respectée.

Une méthode pour l'éxécution de ce contrôle est effectuée [AND 71] :
(1) en divisant arbitrairement l'ensemble des variables de code en deux groupes : A et B ;
(2) en associant à chaque groupe un circuit de calcul d'addition module deux exécuté sur leurs variables. En général, la structure est un arbre de modules "ou-exclusif". Un module primaire est un circuit "ou-exclusif" à deux entrées (voir figure 2).

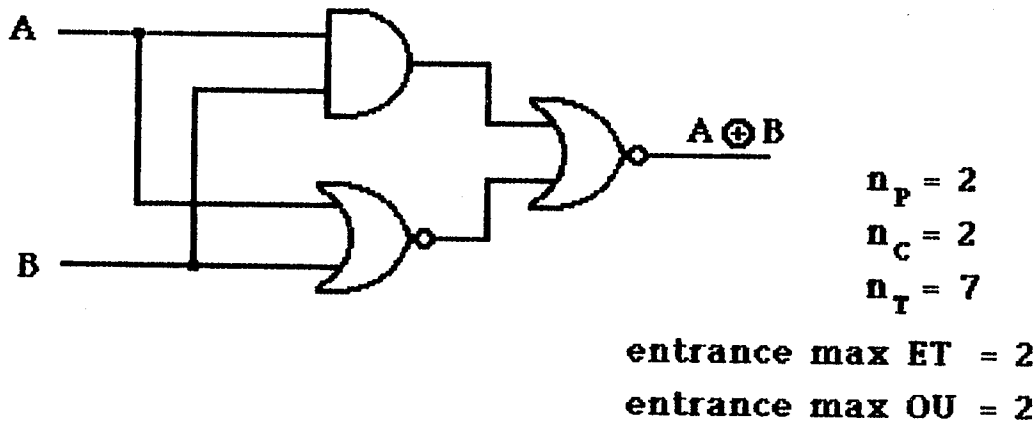



Figure 2 : Le module "ou-exclusif"

Pour obtenir les deux groupes A et B, il est possible d'appliquer une division des bits quelconque, cependant il faut que chaque groupe ait au moins un bit. Le nombre de couches logiques est le plus réduit possible si le nombre de variables contenues dans chaque groupe est égal (ou presque).

Si l'on considére des vecteurs composés de deux bits obtenus à partir de chacun des circuits de contrôle des groupes A et B respectivement, ces vecteurs pourront être 01 ou 10, pour un code de parité paire. D'autres valeurs aux sorties indiquent l'existence d'un défaut.

Pour un code de parité pair, une des sorties est inversée ( une des

branches de l'arbre, $f_0$ ou $g_0$) afin d'avoir aussi des sorties complémentaires. Cette solution est montrée à la figure 3.

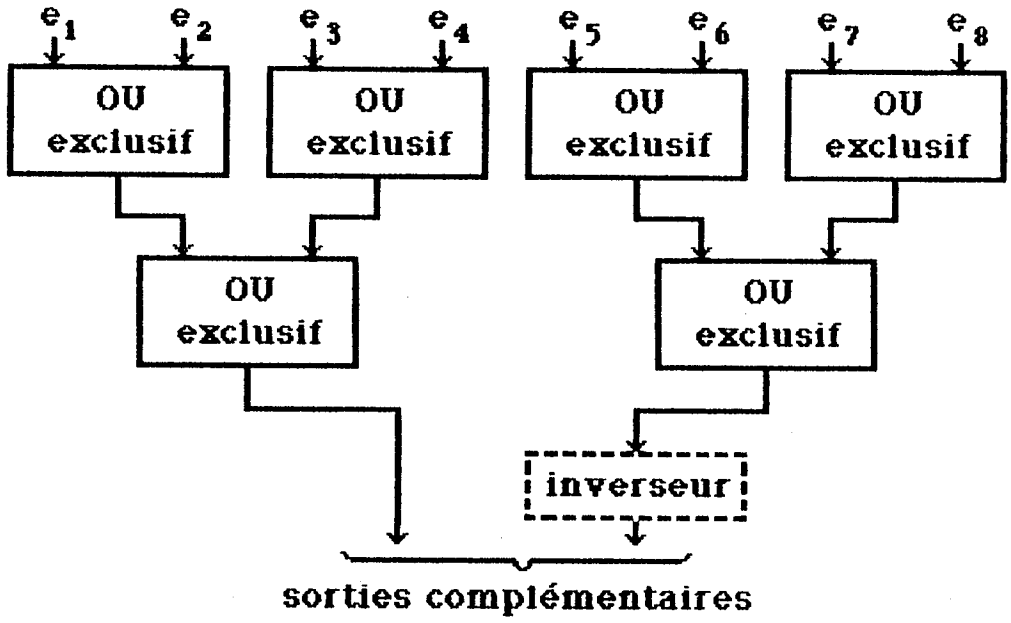



Figure 3 : Contrôleur de parité pair

Le circuit de contrôle d'un code b-adjacent est constitué par b circuits de contrôle de code de parité. A chacun de ces circuits est associé un contrôleur du code b/2b (les codes m-parmi-n seront étudiés plus tard ; le b/2b est un cas particulier d'un tel code).

## II.2.2. Codes à duplication et à complementation

### II.2.2.1. Généralités

La première idée d'un code à complémentation ou double-rail vient d'un code à duplication. Le code à duplication est composé par deux groupes des bits exactement égaux.

**Définition D7**
Un ensemble de 2n variables $e_i$ forme un code à duplication si

$$e_{i+n} = e_i \qquad \forall i \in (0,n).$$

Une méthode directe pour la construction d'un contrôleur d'un tel code est exécutée en faisant l'OU-EXCLUSIF de chaque paire de bits correspondants et après l'OU des résultats obtenus. Le zéro à la sortie de la

porte "OU" indique l'égalité de toutes les paires de bits correspondants ; les sorties des portes OU-EXCLUSIF sont toutes égales à zéro. Mais ce circuit n'est pas totalement autocontrôlable et un nombre très important des vecteurs d'entrée serait nécessaire. Alors, il faut trouver une structure totalement autocontrôlable qui exécute la même fonction ou il faut partir d'une définition différente : d'un code double-rail, par exemple.

**Définition D8**
Un ensemble de 2n variables $e_i$ forme un code à complémentation (ou double-rail) si :

$$e_{i+n} = \overline{e_i} \qquad \forall i \in (0,n).$$

Dans la codification double-rail, les vecteurs d'entrée sont des paires complémentaires (0,1) et (1,0) ; les paires (0,0) et (1,1) sont des entrées qui n'appartiennent pas au code. Ainsi, le même circuit que teste un code double-rail peut être utilisé pour réunir plusieurs paires de sorties en une paire.

## II.2.2.2. Circuits de contrôle

Un contrôleur double-rail totalement autocontrôlable transforme les n paires d'entrées étant $(a_1, b_1, a_2, b_2, \ldots, a_n, b_n)$ et divisés en deux groupes soit $(a_1, a_2, \ldots, a_n)$ et $(b_1, b_2, \ldots, b_n)$ en une paire de sorties, soit $(f_n, g_n)$. Durant l'opération normale f et g définissent un code 1-parmi-2 si et seulement si $a_i = \overline{b_i}$, pour toutes les valeurs de i tels que $1 \leq i \leq k$, i.e.,la paire de sorties doit être complémentaire si et seulement si les k paires d'entrée sont toutes complémentaires.

A partir de la définition, il est possible de vérifier que chaque vecteur du code a n bits du niveau logique "1" et n bits du niveau logique "0". Cet ensemble peut être traité comme un code k-parmi-2k, pour lequel nous présentons les circuits de contrôle dans la section suivante. Mais il faut remarquer qu'une solution d'un tel type ne peut pas détecter de défauts multiples.

Le circuits de contrôle utilisés en général pour des codes double-rail peuvent être conçus à partir de l'utilisation de cellules de base ou d'une solution récursive avec des portes ET et OU. Nous présentons des exemples pour ces deux cas, on considère un contrôleur pour k = 4, et sorties générales appelées $f_0$ et $g_0$. L'ensemble des bits d'information est I = { $a_1$,

$a_2, a_3, a_4, b_1, b_2, b_3, b_4$).

Solution recursive. Soient $f_1$, $g_1$ et $f_2$, $g_2$ les sorties intermédiaires du contrôleur. Pour k = 4, les fonctions seront

$$f_0 = f_1 g_2 + g_1 f_2$$
$$g_0 = f_1 f_2 + g_1 g_2 \qquad (2.2)$$

mais

$$f_1 = a_1 b_2 + b_1 a_2$$
$$g_1 = a_1 a_2 + b_1 b_2 \qquad (2.3)$$

et

$$f_2 = a_3 b_4 + b_3 a_4$$
$$g_2 = a_3 a_4 + b_3 b_4 \qquad (2.4)$$

Si l'on substitue (2.3) et (2.4) en (2.2), on aura :

$$f_0 = a_1 a_3 a_4 b_2 + a_1 b_2 b_3 b_4 + a_2 a_3 a_4 b_1 + a_2 b_1 b_3 b_4 + a_1 a_2 a_3 b_4 + a_1 a_2 a_4 b_3 + + a_3 b_1 b_2 b_4 + a_4 b_1 b_2 b_3$$

$$g_0 = a_1 a_3 b_2 b_4 + a_1 a_4 b_2 b_3 + a_2 a_3 b_1 b_4 + a_2 a_4 b_1 b_3 + a_1 a_2 a_3 a_4 + a_1 a_2 b_3 b_4 + + a_3 a_4 b_1 b_2 + b_1 b_2 b_3 b_4 .$$

Le circuit logique qu'implémente ces fonctions est présenté par la figure 4.

Selon [AND 71], une implémentation à deux niveaux peut être obtenu de la façon décrite ci-dessus, pour n'importe quel valeur de k, en utilisant $2+2^k$ portes logiques et le nombre des entrées aux portes OU au dernier niveau sera égal à $2^{k-1}$. Toutes les $2^k$ entrées appartenant au code d'entrée sont nécessaires pour son diagnostic complet.

Mais ces paramètres peuvent être diminués par l'interconnexion des blocs à deux niveaux en arbres multiniveaux de taille arbitraire, avec l'avantage additionnel que c'est l'utilisation de cellules conçues auparavant. Bien que le délai obtenu par l'utilisation de cette proposition à un niveau ET-OU soit petit, le circuit n'est pas pratique du point de vue du conceptor parce qu'il lui faut dessiner un nouveau circuit pour chaque nombre de bits d'information.

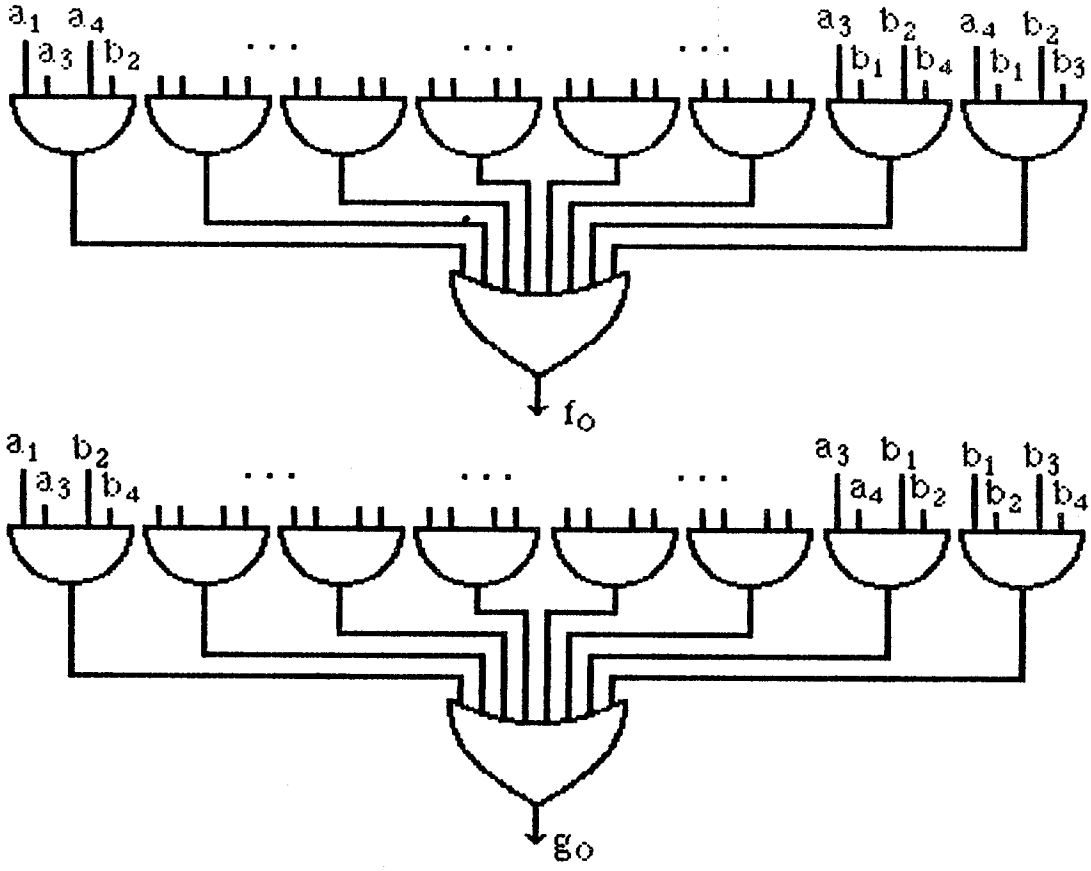



Figure 4 : Contrôleur double-rail pour k = 4

Une solution modulaire donnée par l'assemblage des cellules pré-dessinées est plus pratique.

Solution cellulaire. La structure en arbre d'un contrôleur double-rail pour k = 4, est montré dans la figure 5. Les cellules de base ont été conçues pour k = 2.

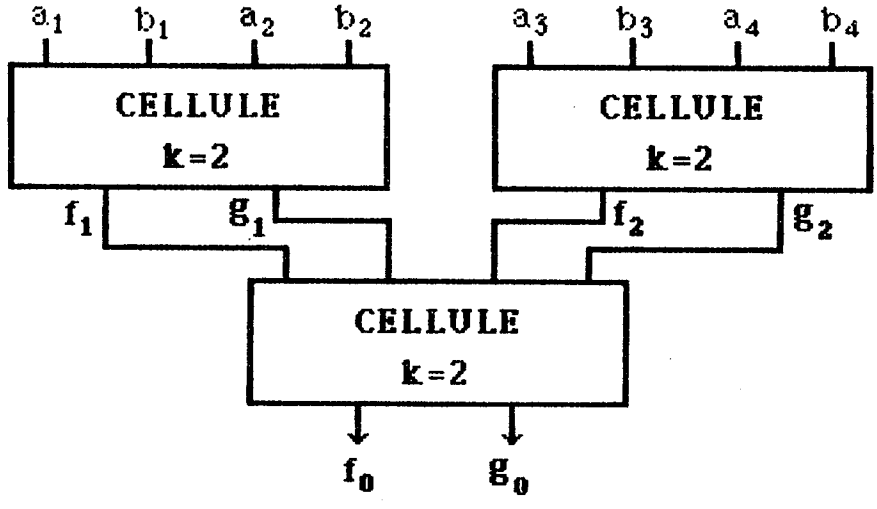



Figure 5 : Contrôleur double-rail avec la structure en arbre

Le circuit de contrôle de Carter [CAR 68] est utilisé comme cellule de base de cette solution modulaire. Chaque module exécute la comparaison entre deux paires de variables $(a_1, b_1)$ et $(a_2, b_2)$. Si $a_1 = \overline{b_1}$ et $a_2 = \overline{b_2}$, alors $(f_i, g_i)$ c (01,10) sinon $(f_i, g_i)$ e (00,11). Le circuit correspondant (figure 6) est basé sur les équations ci-dessous :

$$f_1 = a_1 b_2 + a_2 b_1 \quad \text{et} \quad g_1 = a_1 a_2 + b_1 b_2 .$$

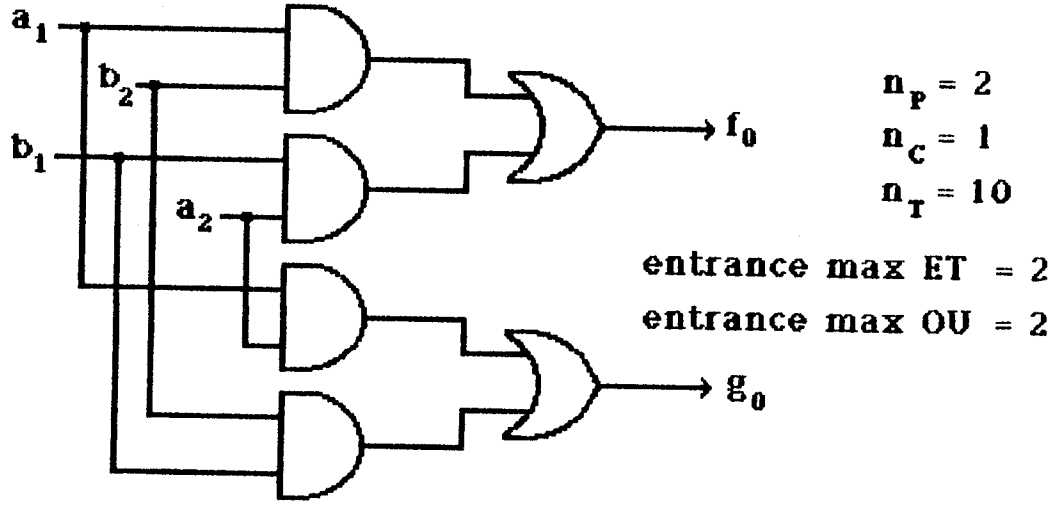



Figure 6 : Module primaire pour un contrôleur du code double-rail

En général, pour des arbres avec m paires d'entrée construites par l'interconnexion de cellules de k paires d'entrées, il faut $\lceil (m-1)/(k-1) \rceil$ cellules et $\lceil \log_k m \rceil$ niveaux de cellules. Le test de chaque cellule demande l'application des $2^k$ entrées et, en fait, selon [AND 71], un arbre entier peut être diagnostiqué avec ces $2^k$ entrées de test adéquatement générées.

### II.2.3. Codes m-parmi-n

#### II.2.3.1. Généralités

Ce sont des codes optimaux parmi les codes non-séparables, i.e., ils utilisent le plus petit nombre des bits supplémentaires.

**Définition D9**
Les codes m-parmi-n , représentés par m/n sont composés par des vecteurs où "m" bits ont la valeur logique "1" et "n-m" bits ont la valeur logique "0". Le nombre des vecteurs du code correspond au nombre de combinaisons

possibles, soit $C_n^m = (n!)/(m!(n-m)!)$; la valeur maximal est accomplie si $m=\lfloor n/2 \rfloor$ ou $m = \lfloor n/2 \rfloor$. La codification de $n_i$ bits d'information exige un code m/n tel que $C_n^m \geq 2^{n_i}$.

II.2.3.2. Circuits de contrôle

En vue de la conception des circuits de contrôle les codes m-parmi-n sont divisés en trois classes différentes, qui sont :

- m-parmi-2m
- 1-parmi-n
- m-parmi-n (où $m \neq 1$, $n \neq 2m$).

A chacune de ces classes, est associé un circuit de contrôle. Des propositions pour ces circuits seront présentées dans les prochaines sous-sections. D'autres papiers, comme [GAI 83] et [PIE 83], suggèrent des différentes méthodes pour la conception des contrôleurs des codes m-parmi-n, mais ne seront pas examinés ici. Les circuits résultants de ces procédures sont, peut-être, plus efficients dans certains configurations des codes m-parmi-n, mais ne résoudent et ne considérent pas tous les cas possibles.

*II.2.3.2.1. Contrôleurs m-parmi-2m*

Pour ces codes, Anderson [AND 73] a proposé une solution conçue à partir des fonctions majorité, qui est décrite à la suite.

Les bits d'entrée sont divisés en deux groupes, A et B. Soit n le nombre total de bits, $n_a$ le nombre de bits en A et $n_b$ le nombre de bits en B. Par conséquent, $n_a + n_b = n$.

Chaque groupe doit avoir au moins un bit, donc $n_a \neq 0$ et $n_b \neq 0$. Soit $m_a$ et $m_b$ le nombre de "1"s dans chaque groupe A et B, respectivement. Pour les vecteurs d'entrée, nous avons $m = m_a + m_b$. Au niveau logique, $T(m_a \geq i) = 1$ si la condition "$m_a \geq i$" est valable.

Le contrôleur est formé de deux sous-circuits indépendants séparés, à chacun desquels correspondent une sortie nommées respectivement f et g, et définies ci-dessous :

$$f = \Sigma\ T(m_a \geq i).\ T(m_b \geq m-i),\ i=(x_1,\ldots,x_2),\ \text{impair}$$
$$g = \Sigma\ T(m_a \geq i).\ T(m_b \geq m-i),\ i=(x_1,\ldots,x_2),\ \text{pair}.$$

Pour toutes les valeurs de i parmi les entiers, c'est facile à vérifier que les sorties du contrôleur sont égales à (01) ou (10), pour les vecteurs qui appartiennent au code d'entrée. Si le nombre de "1"s aux entrées est plus grand ou plus petit que m, les sorties sont égales à (11) ou (00), respectivement.

Le nombre de facteurs nécessaires sera réduit par l'élimination des termes-produit égaux à zéro et des variables qui sont toujours égales à "1", dont :

$$x_1 = \max(m - n_b\ ;\ -1)$$
$$x_2 = \min(n_a\ ,\ m + 1) \qquad \text{[AND 71]}.$$

Une technique pour l'implémentation d'une fonction de simple seuil $T(m_a \geq 1)$ est une réalisation double niveau ET-OU, où les différentes combinaisons des i bits sont utilisées (des $n_a$ disponibles) comme des entrées pour les $C_i^{n_a} = n_a!/(\ i!\ (n_a - i)!\ )$ portes ET. Les sorties de portes ET sont utilisées comme des entrées d'une porte OU simple. L'implémentation de $T(m_b \geq m - i)$ est faite d'une façon pareille.

La méthode est illustrée par l'implémentation d'un contrôleur 3-parmi-6, donc n = 6 et m = 3. En vue d'une exécution économique, nous faisons $n_a = n_b = 3$ ; alors $A = (a_1, a_2, a_3]$ et $B = (b_1, b_2, b_3]$, et

$$x_1 = \max(0, -1) = 0 \quad \text{et} \quad x_2 = \min(3, 4) = 3.$$

Les fonctions f et g seront :

$$f = [\ T(m_a \geq 1).\ T(m_b \geq 2)\ ] + [\ T(m_a \geq 3).\ T(m_b \geq 0)]$$
$$g = [\ T(m_a \geq 0).\ T(m_b \geq 3)\ ] + [\ T(m_a \geq 2).\ T(m_b \geq 1)].$$

Les substitutions pour les fonctions T et simplifications des expressions faites, nous aurons :

$$f = (a_1 + a_2 + a_3).\ (b_1b_2 + b_1b_3 + b_2b_3) + (a_1\ a_2\ a_3)$$
$$g = (b_1\ b_2\ b_3) + (a_1\ a_2 + a_1\ a_3 + a_2\ a_3)\ (b_1 + b_2 + b_3)$$

qui correspondent au circuit de la figure 7.

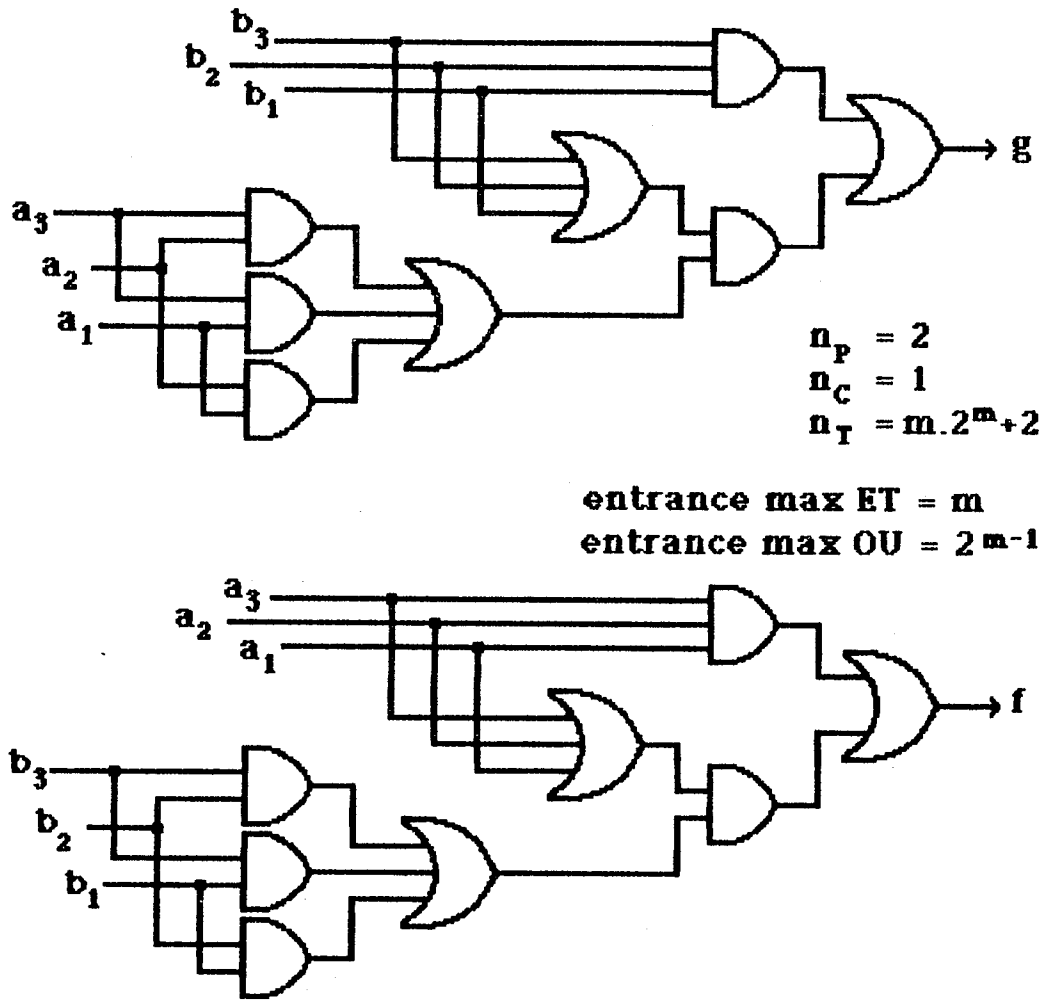



Figure 7 : Contrôleur pour le code 3-parmi-6, totalement autocontrolable

Ces circuits ne sont pas envisageables pour des valeurs de m élevée à cause du nombre important des entrées pour les fonctions ET et OU. Et aussi, le nombre de transistors augmente exponentiellement avec m. Reddy [RED 74a] apporte une solution à ces problèmes en introduisant des réseaux cellulaires facilement testables pour la génération des fonctions majorité.

La fonction de base, nommée $T_p^n$, est une fonction de seuil de n variables que reçoit la valeur "1" si et seulement si au moins p variables d'entrée sont égales à "1", étant $1 \leq p \leq n$. En utilisant les mêmes représentations déjà introduites, nous aurons :

$$T_p^n (m \geq p) = 1,$$

où m est le nombre de "1"s parmi les variables d'entrée ($m \leq n$).

Le réseau cellulaire $T^n$ synthétise toutes les n $T_p^n$ fonctions de n variables. Un réseau général de n variables est montré par le schéma de la figure 8. Les fonctions internes à chaque cellule de ce réseau sont montrées par un exemple d'un réseau $T^5$ (figure 9).

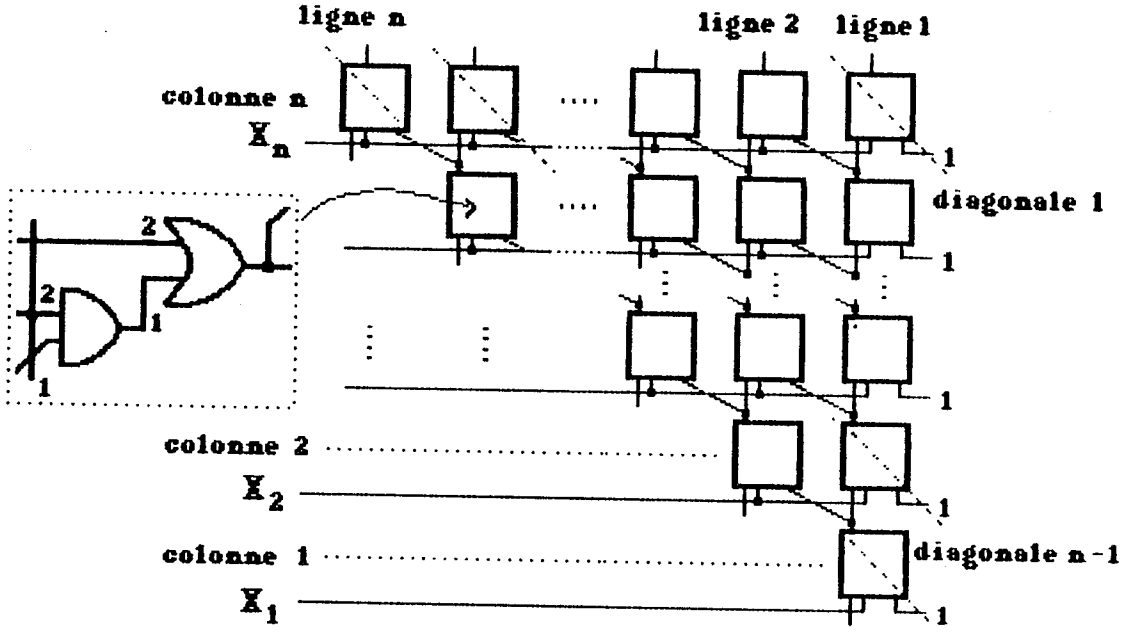



Figure 8 : Un réseau général de n variables

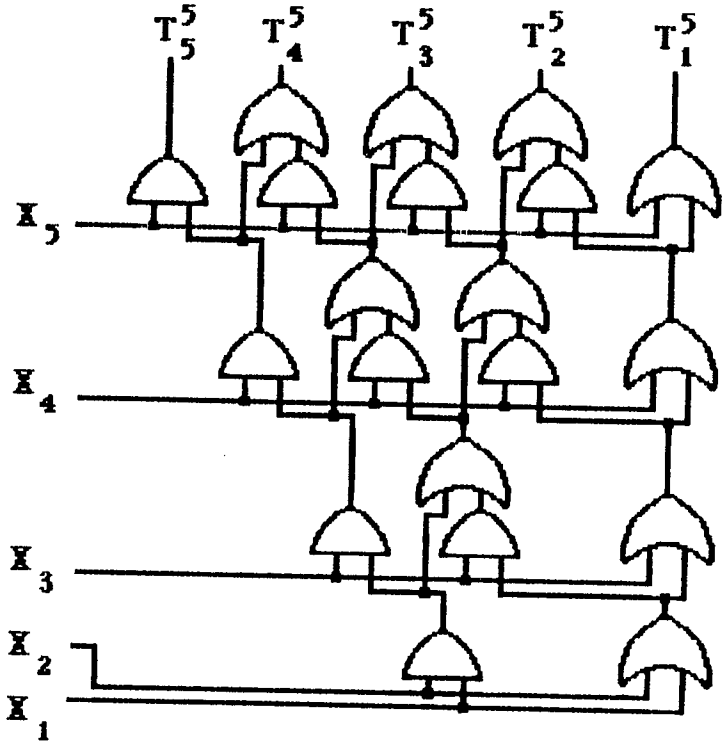



Figure 9 : Exemple d'un réseau $T^5$

Le choix d'un réseau de Reddy conduit aux paramètres suivants :

$n_p = m\,(m-1)$

$n_T = 4\,m^2 - 2m - 2$

$n_C = m$

entrance max OU - m

entrance max ET - 4

### *II.2.3.2.2. Contrôleurs 1-parmi-n*

Une solution possible pour un contrôleur de code 1/n est la translation entre ce code-ci et un code m/2m, tel que ($C_{2m}^{m} \geq n$). Ce dernier code peut être contrôlé pour des circuits présentés à la section II.2.3.2.1. Le schéma général (figure 10) est décomposé en deux blocs, le bloc de translation et le bloc contrôleur m/2m, cela pouvant être dessiné avec un PLA [AND 71].

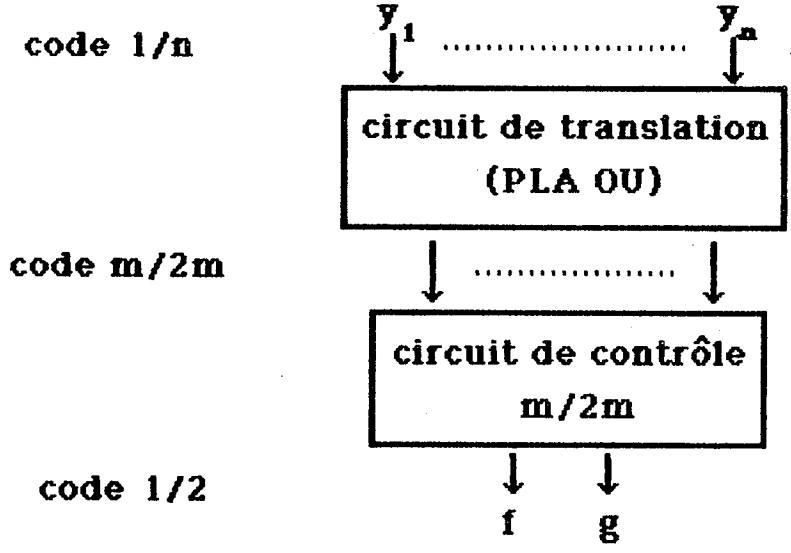



Figure 10 : Contrôleur de code 1/n avec une translation intermédiaire à un code m/2m

Une autre solution peut être obtenue pour des valeurs de n telles que $n = 2^N$. Il s'agit d'une réalisation cellulaire en arbre, chaque cellule de base étant un contrôleur du code 1/4 (figure 11). Bien qu'initialement étudiée pour des codes 1/n avec $n = 2^N$, cette solution peut être appliquée dans le cas général ($n \neq 2^N$) dès que :

- seulement une partie de l'arbre est soit utilisée pour les valeurs paires de n,
- soient introduites les cellules qui traitent un nombre impair de variables pour les valeurs impaires de n [DIA 74a].

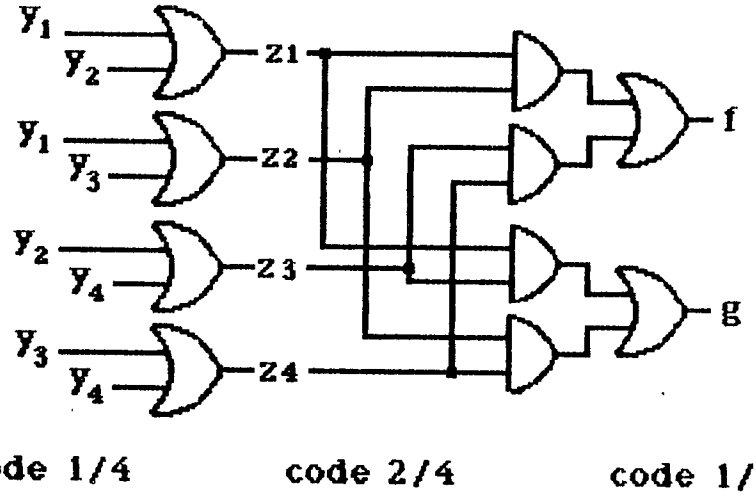



Figure 11 : Cellule standard pour un contrôleur du code 1/4

Des nombres plus importants de transistors et portes logiques sont nécessaires à la deuxième solution qu'ils étaient à la proposition d'Anderson. Une telle solution est justifiée pour les applications où l'automatisation du dessin est envisagée; le contrôleur peut être totalement assemblé à partir des cellules antérieurement dessinés (bibliothèque des cellules, par exemple). Si $N \geq 7$ (code 1/128), la première solution, celle d'Anderson, peut présenter des problèmes à cause du nombre important d'entrées aux portes OU.

Il faut remarquer aussi que la solution proposée par Anderson ne peut pas être appliquée à la construction des contrôleurs des codes 1/7 et 1/3. Ces codes sont des cas particuliers , pas entièrement résolus.

Pour le code 1/3, David a suggéré une solution basée sur l'utilisation d'un circuit séquentiel, qui est présentée en [DAV 78]. Selon Reddy [RED 74], un contrôleur TSC combinatoire, où ne sont utilisées que des portes OU et ET, n'existe pas. Dans ce même papier, il présente une proposition pour le contrôleur du code 1/7 : il fait une conversion intermédiaire du code 1-parmi-7 à un code 3-parmi-6, pour lequel existe un circuit contrôleur.

*II.2.3.2.3. Contrôleurs m-parmi-n ($m \neq 1$, $n \neq 2m$)*

Dans le cas général d'un code m-parmi-n quelconque, la méthode la plus simple est la traduction du code m/n à $1/C_n^m$ en utilisant des portes ET. Ce code $1/C_n^m$ peut être vérifié pour des contrôleurs déjà examinés. Le plus grand inconvénient de cette méthode est rapporté au nombre important des portes ET, chacune avec m entrées.

En [MAR 77], nous trouvons une proposition d'un circuit de contrôle pour les codes m/n reposée sur des fonctions majorité. Trois cas différents sont considérés dans cette proposition :

$A : 2m + 2 \leq n \leq 4m,\ m \geq 2$
$B : n = 2m + 1,\ m \geq 2$
$C : n > 4m,\ m \geq 2.$

Les contrôleurs du code m-parmi-n décrits en [MAR 77] ont la structure générale composée de trois sous-circuits C1, C2 et C3 (figure 12). Le circuit C1 a n entrées et z sorties, où :

z = 4 pour les codes m-parmi-(2m + 1),
z = 5 pour les codes 2-parmi-n, et
z = 6 pour les autres codes m-parmi-n.

C1 reçoit des entrées qui sont vecteurs du code m-parmi-n et produit des sorties codées en 1-parmi-z. C2 exécute la traduction d'un code 1-parmi-z en 2-parmi-4. Ce dernier forme l'ensemble des entrées d'un contrôleur 2-parmi-4, qui est le bloc final C3.

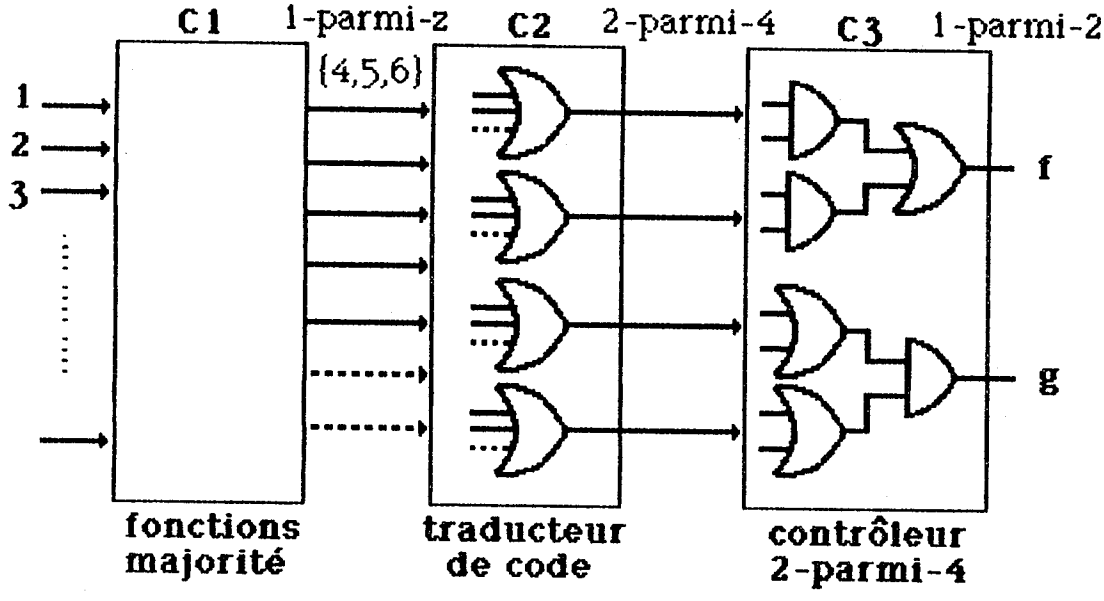



Figure 12 : Contrôleur du code m-parmi-n [MAR 77]

Cas A : $2m + 2 \leq n \leq 4m,\ m \geq 2$

Les bits d'entrée sont divisés en deux groupes , A et B, tels que $n_a = \lfloor n/2 \rfloor$ et $n_b = n - n_a = \lceil n/2 \rceil$. Les deux fonctions $f_1$ et $f_2$ sont définis ci-dessous :

$$f_1 = \Sigma\, T(a \geq i).\, T(b \geq m-i)\,,\ i=(1,\ldots,m-1),\ \text{impair}$$
$$f_2 = \Sigma\, T(a \geq i).\, T(b \geq m-i),\ i=(1,\ldots,m-1),\ \text{pair}.$$

Les bits de l'ensemble A sont divisés en deux sous-groupes, $A_1$ et $A_2$, tels que $n_{a1} = \lfloor n_a/2 \rfloor$ et $n_{a2} = n_a - n_{a1} = \lceil n_a/2 \rceil$ Les deux fonctions $f_3$ et $f_4$ sont définies comme suit :

$$f_3 = \Sigma\, T(a_1 \geq i).\, T(a_2 \geq m-i),\ i=(m-n_{a2},\ldots,n_{a1}),\ \text{impair}$$
$$f_4 = \Sigma\, T(a_1 \geq i).\, T(a_2 \geq m-i),\ i=(m-n_{a2},\ldots,n_{a1}),\ \text{pair}.$$

Les bits de l'ensemble B sont divisés en deux sous-groupes, $B_1$ et $B_2$, tels que $n_{b1}=\lfloor n_b/2 \rfloor$ et $n_{b2}= n_b - n_{b1}=\lceil n_b/2 \rceil$. Les deux fonctions sont définies ci-dessous :

$$f_5 = \Sigma\ T(b_1 \geq i).\, T(b_2 \geq m-i),\ i=(m-n_{b2},\ldots,n_{b1}),\ \text{impair}$$
$$f_6 = \Sigma\ T(b_1 \geq i).\, T(b_2 \geq m-i),\ i=(m-n_{b2},\ldots,n_{b1}),\ \text{pair}\ .$$

Le bloc C1 est réalisé par ces fonctions $f_i$, en étant i = 1, 2,....,6 comme défini ci-dessus, en forme de somme des produits (ET - OU). C2 est un traducteur du code 1-parmi-6 à 2-parmi-4, et ses entrées sont des sorties en provenance de C1. C3 reçoit, comme entrées, les sorties de C2 : il est un contrôleur du code 2-parmi-4 et génére les deux sorties générales du circuit , f et g. Ce circuit a sept couches logiques, au maximum.

On peut remarquer le cas spécial de m = 2, pour lequel $f_2$ est égal à zéro ; donc C1 n'aura que cinq sorties et C2 sera un traducteur du code 1-parmi-5 au 2-parmi-4.

Cas B : $n = 2m + 1,\ m \geq 2$

Les bits d'entrée sont divisés en deux groupes A et B, tels que $n_a = m$ et $n_b = m + 1$. Les bits du groupe B sont divisés en deux sous-groupes $B_1$ et $B_2$ tels que $n_{b1}=\lfloor n_b/2 \rfloor$ et $n_{b2}=n_b-n_{b1}=\lceil n_b/2 \rceil$. Les fonctions $f_1$ , $f_2$ , $f_3$ et $f_4$ sont définies comme suit :

$$f_1 = \Sigma\, T(a \geq i).\, T(b \geq m-i),\ i=(1,\ldots,n),\ \text{impair}$$
$$f_2 = \Sigma\, T(a \geq i).\, T(b \geq m-i),\ i=(1,\ldots,n),\ \text{pair}$$

$f_3 = \Sigma\ T(b_1 \geq i).\ T(b_2 \geq m-i),\ i=(m-n_{b2},\ldots,n_{b1})$, impair
$f_4 = \Sigma\ T(b_1 \geq i).\ T(b_2 \geq m-i),\ i=(m-n_{b2},\ldots,n_{b1})$, pair.

C1 est réalisé en forme de somme des produits avec des sorties $f_1$, $f_2$, $f_3$ et $f_4$. C2 fait la traduction du code 1-parmi-4 en 2-parmi-4. C3 est le même que dans le cas précédent.

Cas C : $n > 4m$, $m \geq 2$.

La conception d'un contrôleur pour ce cas-ci, où $n \geq 4m$ est décrite par une procédure. Le calcul des équations est dépendant des conditions particulières au début et des valeurs intermédiaires des variables ; donc nous allons laisser la description sous la forme de procédure. Elle est récursive et initialement appelée sous la forme suivante :

k = 1
S = (ensemble de toutes les entrées)
Appeler *Conception du Contrôleur* (n, s, k, $H_1$, $H_2$),
où $H_1$ et $H_2$ sont les sorties générales du contrôleur.

La procédure est décrite comme suit.

Procédure "*Conception du Contrôleur*"($N_S$, S, I, G(I), $G_2$(I)) ;

(1) Divisez S en deux groupes $A^I$ et $B^I$ tels que $n_a{}^I = \lfloor n_S/2 \rfloor$, $n_b{}^I = n_S - n_a{}^I = \lceil n_S/2 \rceil$. Soit j=m si $n_a{}^I = m$ sinon j = m-1. Alors,

$f_1{}^I = \Sigma\ T(a^I \geq i)\ .\ T(b^I \geq m-i),\ i=(1,\ldots,j)$, impair
$f_2{}^I = \Sigma\ T(a^I \geq i)\ .\ T(b^I \geq m-i),\ i=(1,\ldots,j)$, pair.

(2) Si $n_a{}^I > 2m$ alors

i) k = k + 1
ii) Appelez Conception du Contrôleur ($n_a$,$A^I$,k,$f_3$,$f_4$)
iii) allez à l' item (4).

(3) Si $n_a{}^I = m$ alors $f_3{}^I = f_4{}^I = 0$ ; sinon divisez l'ensemble $A^I$ en deux

sous-ensembles $A_1^I$ et $A_2^I$ tels que $n^I_{a1}=\lfloor n^I_a/2 \rfloor$, $n^I_{a2}=\lceil n^I_a/2 \rceil$ et,

$$f_3^I = \Sigma\, T(a_1^I \geq i)\,.\,T(a_2^I \geq m-i),\ i=(m-n^I_{a2},\ldots,n^I_{a1}),\ \text{impair}$$
$$f_4^I = \Sigma\, T(a_1^I \geq i)\,.\,T(a_2^I \geq m-i),\ i=(m-n^I_{a2},\ldots,n^I_{a1}),\ \text{pair}.$$

(4) Si $n_b^I > 2m$, alors

i) $k + 1$
ii) Appelez Conception du Contrôleur $(n_b, B^I, k, f_5, f_6)$
iii) Allez à l'item (6).

(5) Si $n_b^I = m$ alors $f_5^I = f_6^I = 0$ ; sinon divisez l'ensemble $B^I$, en deux sous-ensembles $B_1^I$ et $B_2^I$ tels que $n^I_{b1} = \lfloor n_b^I/2 \rfloor$, $n^I_{b2} = \lceil n_b^I/2 \rceil$, et

$$f_5^I = \Sigma\, T(b^I_1 \geq i)\,.\,T(b^I_2 \geq m-i),\ i=(m-n^I_{b2},\ldots,n^I_{b1}),\ \text{impair}$$
$$f_6^I = \Sigma\, T(b^I_1 \geq i)\,.\,T(b^I_2 \geq m-i),\ i=(m-n^I_{b2},\ldots,n^I_{b1}),\ \text{pair}.$$

(6) Soit C1(I) un circuit avec n entrées et dont les sorties forment l'ensemble $f_i^I$, $1 \leq i \leq 6$, défini ci-dessus. Les sorties du C1(I) sont traduites à un code 2-parmi-4 par un translateur du code C2(I). Ces sorties, en provenance du C2(I) sont testées par C3(I), lequel est un contrôleur du code 2-parmi-4. Les sorties du C3(I) sont f(I) et g(I).

(7) Retourner

(8) Fin.

Un contrôleur du code 3-parmi-7 dessiné à partir des règles proposées par [MAR 77] est présenté (figure 13). Ce code est inclu dans le cas A, à partir duquel les équations ont été prises.

### II.2.4. Codes de Berger

#### II.2.4.1. Généralités

Les codes de Berger sont codes séparables. Un mot à longueur n codé à $I = n - k$ bits d'information et k bits de contrôle, où $k = \lceil \log_2 n - k + 1 \rceil$.

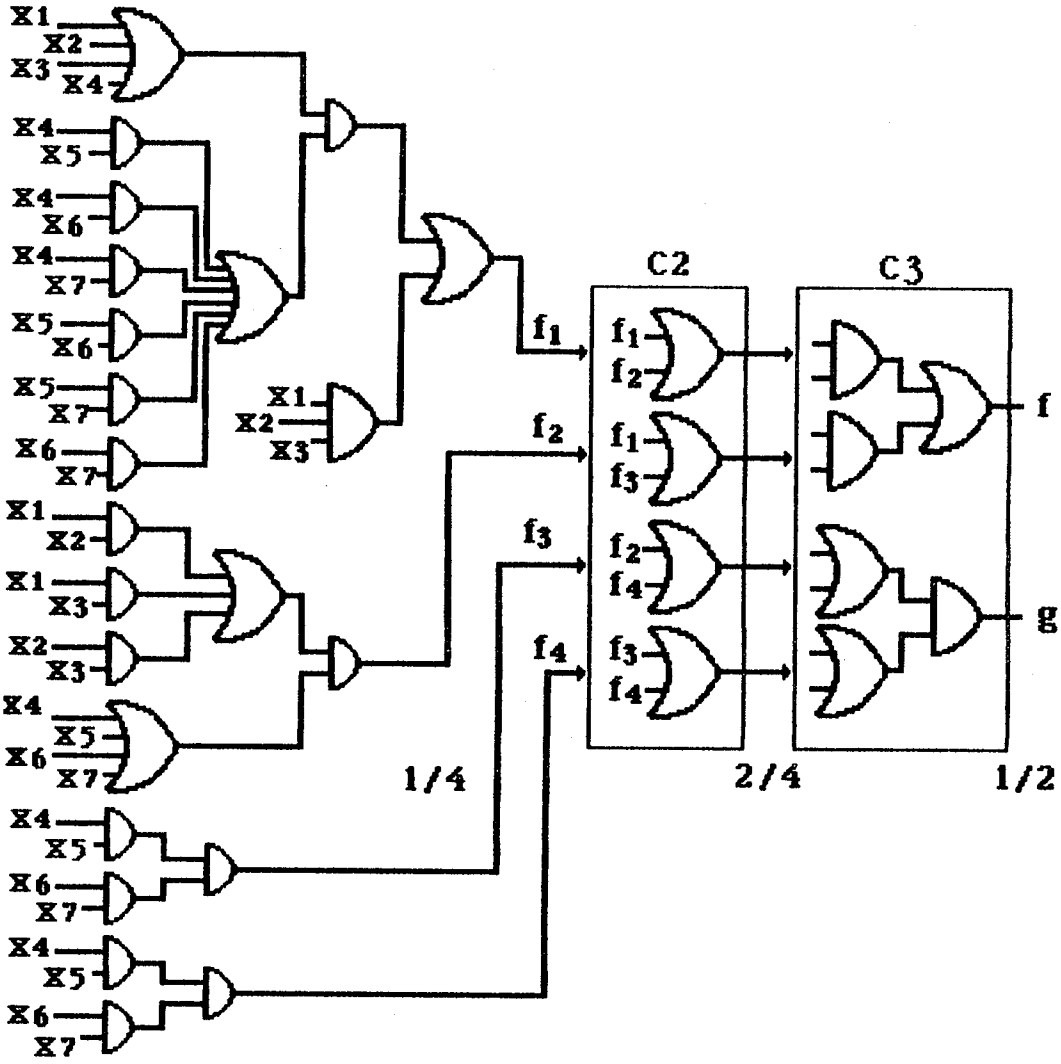



Figure 13 : Un contrôleur du code 3-parmi-7 [MAR 77]

**Définition D10**
La concaténation de I bits d'information et k bits de contrôle, où les k bits de contrôle sont obtenus par le calcul du numéro binaire correspondant au nombre de uns parmi les I bits d'information, ce numéro en étant complémenté bit à bit, i.e., tous les zéros sont changés par un et tous les uns par zéro, est un mot du code de Berger.

D'autre part, on peut obtenir aussi les k bits de contrôle à travers le calcul du numéro binaire correspondant au nombre de zéros parmi les I bits d'information [BER 61].

Les codes de Berger nécessitent un nombre plus important de bits redondants (k) que les codes m-parmi-n, mais ils sont les codes séparables optimaux pour la détection de fautes unidirectionnelles - c'est celui qui utilise le moins de bits de contrôle pour un nombre donné de bits

d'information [CRO 78].

Un code de Berger est à longueur maximale si le nombre de bits d'information est $I = 2^k-1$, un tel code est complet puisque les k bits de contrôle prennent les $2^k$ configurations possibles. Les longueurs courantes d'information étant des multiples ou des puissances de 2, les codes de Berger correspondants ne sont pas à longueur maximale et sont donc incomplets.

II.2.4.2. Circuits de contrôle

Le schéma général d'un contrôleur pour un code séparable peut être composé d'un circuit combinatoire N qui reçoit les I bits d'information comme entrées et génère le complément binaire des bits de contrôle aux sorties. Un contrôleur double-rail E est utilisé pour comparer les k bits de contrôle avec les sorties de N. La figure 14 donne un exemple de ce schéma général [MAR 78].

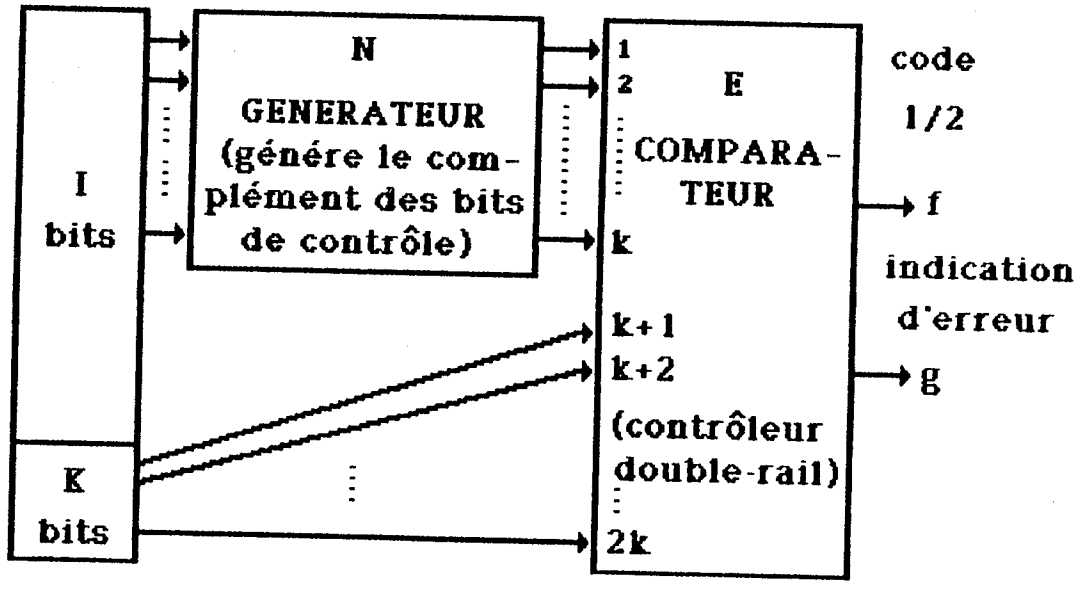



Figure 14 : Le schéma général d'un contrôleur du code de Berger

Dans le schéma exemplifié, le circuit générateur N produit le numéro binaire des uns parmi les n-k bits d'information. Il est composé à partir d'un ensemble de modules d'additionneurs et de demi-additionneurs à 2 bits, qui exécutent l'addition en parallèle des bits d'information. Toutes les combinaisons sont disponibles à chaque paire d'entrées dans l'ensemble des mots du code de Berger. Une faute produira un mot non codé aux entrées d'E et ainsi une erreur détectable sera produite aux sorties générales du

contrôleur. Ce schéma peut être utilisé seulement pour les codes de Berger de longueur maximale, i.e., ceux pour lesquels la longueur des bits d'information est $2^k$-1. Pour un code de Berger de longueur non-maximale, il faut définir un code équivalent, où les bits de contrôle prennent toutes les $2^{nk}$ combinaisons possibles. Donc, quelques modifications peuvent se faire nécessaires dans le circuit de génération des bits de contrôle [ASH 76].

La figure 15 donne un exemple de réalisation de la cellule de base du circuit générateur (additionneur).

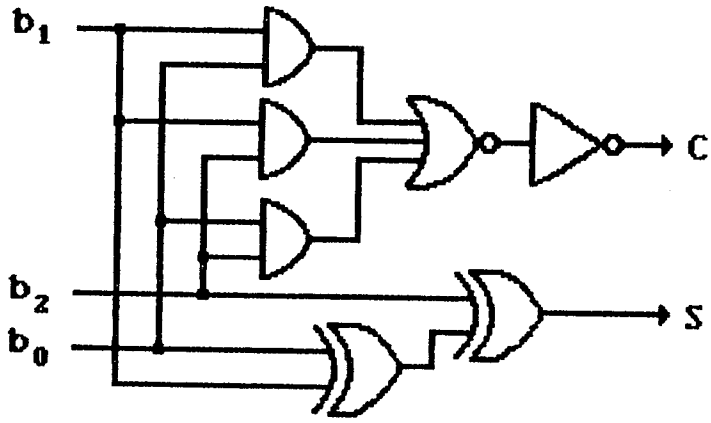



Figure 15 : Cellule de base pour le bloc N

Une deuxième solution du circuit de contrôle utilise la translation du code de Berger en un code $1/2^l$, qui est lui-même contrôlé à l'aide des circuits classiques des codes 1/n. L'inconvénient de cette méthode est le nombre important de transistors dans la matrix ET du PLA qui exécute le décodage des $2^l$ combinaisons du code de Berger [CRO 78 ].

## II.2.5. Codes de Berger modifiés

### II.2.5.1. Généralités

Ces codes ont été considérés par [DON 82] et [MAK 82].

Nous continuerons à utiliser la même symbologie employée dans la description des codes de Berger. Nous ajoutons les symboles $I_0$ et $I_1$ pour représenter respectivement le nombre de "0"s et le nombre de "1"s parmi les I bits d'information.

**Définition D11**
Les codes de Berger modifiés sont constitués par des ensembles de bits où il y a I bits d'information et k bits de controle, divisés en deux groupes : C1 et C2 . Le premier groupe, C1, est obtenu par un des quotients $I_0$ module (m+1) ou $I_1$ module (m+1), où 1< m< I, et le deuxième correspond à la codification des bits C1 en un parmi les codes m-parmi-n, Berger ou double-rail.

Pour le premier groupe de bits de contrôle, C1, $[\log_2 (m+1)]$ bits sont nécessaires et le nombre de bits du deuxième groupe, C2 , est dépendant du code choisi.

Conceptuellement, m+1 peut être n'importe quel entier, mais dans le cas que $m+1= 2^J$, l'implémentation du circuit du contrôleur sera la plus simple [DON 82].

## II.2.5.2. Circuits de contrôle

Le circuit de contrôle pour le code de Berger présenté à la section II.2.4.2. peut être facilement changé dans un contrôleur de code de Berger modifié.

Il est constitué de deux parties principales, comme montré à la figure 16. Le circuit CC1 vérifie les bits d'information à travers la génération du complément du premier groupe de bits de contrôle C1 (par le circuit N1) suivie de la comparaison de ce résultat avec C1 (par le circuit N2). Le deuxième niveau de codification de C1 et C2 est vérifié par le circuit CC2.

Soit C1 le groupe de bits de contrôle défini par :

$$C1 = (2^J - 1) - (I_1 \text{ module } (m+1)),$$

où J est le nombre de bits de C1. Alors, C1 est le complément, bit par bit, de $I_1$ module (m+1). Dans ce cas, le circuit N1 est un générateur de module. Le circuit N2 est un contrôleur de code double-rail, et les J sorties du circuit N1 seront comparés avec le groupe C1 du mot de code par le contrôleur N2. Le deuxième niveau de codification , C2, est testé par le circuit CC2, dont la structure dépend du code utilisé. Alors C1 et C2 peuvent former soit un code double-rail soit un autre code de Berger, par exemple. Si C1 et C2 forment un code double-rail, CC2 sera égal à N2. S'il n'y a pas d'erreur, C1 et

C2 composent un mot de code et nous avons C1' = C2, f= f' , et g= g'.

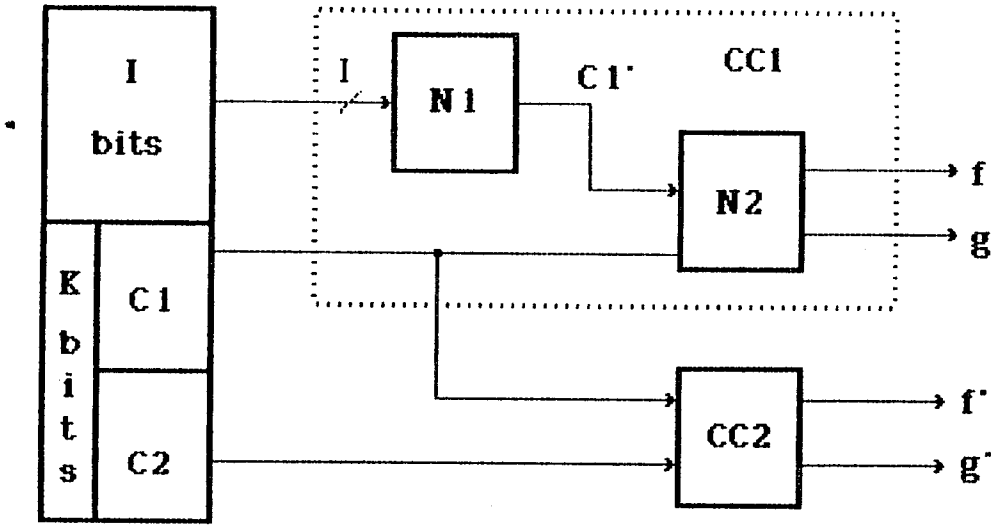



Figure 16 : Structure générale du contrôleur pour le code de Berger modifié

## II.3. CONCLUSION

Les principaux codes classiques ont été présentés, ainsi que des exemples de circuits de contrôle associés.

Le choix des codes à utiliser doit être fait au niveau de la conception du système, pour les informations produites par le bloc fonctionnel, selon la capacité de détection d'erreurs souhaitée, des hypothèses de pannes considérées et même de l'architecture interne du bloc fonctionnel. Si l'on considère des hypothèses de pannes et certaines structures où des erreurs multiples (par exemple) peuvent se produire aux sorties, il faut évidemment utiliser un code ayant la capacité de détecter des erreurs multiples afin d'assurer 100% de sûreté en termes de détection.

Les codes que nous venons de présenter ont des capacités diverses de détection d'erreurs, listées ci-dessous :

| code | erreurs détectées |
|---|---|
| parité | simples |
| duplication | multiples |
| complémentation (double-rail) | multiples |

| | |
|---|---|
| m-parmi-n | unidirectionnelles |
| Berger | unidirectionnelles |
| Berger modifié | unidirectionnelles*. |

Les codes de parité ne détectent que des erreurs simples mais sont à redondance minimale. Les codes de duplication et double-rail peuvent détecter des erreurs multiples mais résultent en une redondance matérielle importante. Pour la détection des erreurs unidirectionnelles, nous pouvons choisir parmi les codes m-parmi-n, Berger ou Berger modifié. Le premier groupe donne des codes optimaux pour la détection de ce type d'erreurs, mais n'étant pas séparables, peuvent exiger des circuits de transcodage de taille assez importante. Les deux derniers, Berger et Berger modifiés, sont séparables, mais les circuits de contrôle nécessaires peuvent être de redondance matérielle importante.

Les circuirs de contrôle proposés ici comme exemple, semblent être les plus convenables pour la plupart des cas (nombre moyen de bits d'information). Ils peuvent être utilisés comme proposition initiale pour la conception d'un contrôleur SCD, comme nous le vérifierons dans le prochain chapitre. Dans ce cas, il reste à examiner uniquement la topologie du circuit contrôleur en fonction de sa propriété "strongly code disjoint", ainsi qu'à éliminer quelques redondances, si possible.

D'autres propositions de circuits de contrôle et d'autres codes sont donnés dans les réferences indiquées au long du chapitre.

# CHAPITRE 3:

## CONCEPTION DES CONTROLEURS

## STRONGLY CODE DISJOINT (SCD)

# CONCEPTION DES CONTROLEURS STRONGLY CODE DISJOINT (SCD)

## III.1. DEFINITIONS DE BASE : CIRCUITS SFS ET CONTROLEURS SCD

D'après les définitions des circuits TSC, nous avons vu que ceux-ci devraient détecter toutes fautes survenant. Cette condition est difficile à accomplir, surtout lorsqu'on considère des hypothèses de pannes analytiques - elle est beaucoup plus raisonnable si elle est alliée à un modèle logique, comme l'était celui des collages. Il peut exister des défauts qui ne causent pas de changements à la fonction éxécutée par le circuit, et n'étant pas dangereuses, nous n'avons pas besoin de les éviter. Ces idées sont l'origine des concepts de redondance, qui seront vus par la suite.

**Définition D12**
Un circuit G est redondant pour un défaut f si la fonction qu'il réalise n'est pas modifiée en présence du défaut f [BRE 76].

La dernière définition peut être raffinée, étant donné que G peut être redondant pour l'ensemble X ou seulement pour le code d'entrée A ($A \subseteq X$). Le deuxième est plus restrictif que le premier.

**Définition D13**
Un circuit G qui réalise une fonction $G(x, \varnothing)$ est redondant pour un défaut f et pour l'ensemble des vecteurs d'entrée X si :

$$\forall x \in X,\ G(x, f) = G(x, \varnothing).$$

**Définition D14**
Un circuit G qui réalise une fonction $G(a, \varnothing)$ est redondant pour un défaut f et pour le code d'entrée A si :

$$\forall a \in A,\ G(a, f) = G(a, \varnothing).$$

Nous remarquons qu'un circuit redondant pour une faute f, ne garde pas la propriété "self-testing", puisqu'il y a des défauts non détectés.

Pourtant un problème plus grave, associé à ces défauts pour lesquels le circuit est redondant, existe; c'est le fait qu'une deuxième faute ajoutée à la première non détectée peut provoquer des vecteurs du code erronés aux sorties. Par exemple : dans un circuit satisfaisant la propriété "fault-secure" pour F, il pourrait exister une faute $f_1 \in F$ et le circuit serait redondant pour $f_1$. Donc, le circuit continue à fonctionner et un deuxième défaut $f_2 \in F$ peut survenir. Dans ce cas, le défaut présent dans le circuit est la séquence $\langle f_1, f_2 \rangle$ lequel peut ne pas appartenir à F ; par conséquent, il n'est plus certain que la propriété "fault secure" sera gardée.

Afin que ce problème soit surmonté, les circuits "strongly fault secure" sont définis [SMI 78]. Ils sont "fault secure", pour $f_1$ et pour chaque sous-séquence initiale et "self-testing" pour la combinaison des défauts de la séquence $\langle f_1, f_2, \ldots f_n \rangle$. Ainsi, l'ensemble de défauts F inclut aussi des combinaisons de fautes ou de séquences.

**Définition D15**
Pour une séquence de défauts $\langle f_1, f_2, \ldots f_n \rangle$ soit k le plus petit entier pour lequel :

$$\exists\, a \in A,\ G(a, U f_j) \neq G(a, \emptyset) \quad \text{où } j=1,\ldots,k.$$

Si un tel k n'existe pas, posons k = n+1. Alors G est "strongly fault secure" (SFS) pour la séquence $\langle f_1, f_2, \ldots f_n \rangle$ si $\forall\, a \in A$ :

$$\underline{\text{soit}}\ G(a, U f_j) = G(a, \emptyset)$$

$$\underline{\text{soit}}\ G(a, U f_j) \notin B, \qquad \text{où } j=1,2,\ldots,k.$$

**Définition D16**
Un circuit G est "strongly fault secure" (SFS) pour un ensemble de défauts F s'il est "strongly fault secure" pour chaque séquence constituée d'éléments appartenant à l'ensemble F.

Avec ces dernières définitions, nous obtenons des circuits, qui ne devront pas nécessairement détecter ou indiquer tous les défauts. Il faut rendre sûre la correction des sorties, donc la propriété "TSC goal".

L'hypothèse H1 (énoncé à la section II.1.) étant respectée, on peut démontrer que chaque circuit SFS accomplit le "totally self-checking goal" [SMI 78].

Il est possible de vérifier que tout circuit TSC est aussi SFS. C'est un cas particulier pour lequel k=1 (définition D15). Par conséquent, la définition d'un circuit TSC est plus restrictive que celle des SFS. En conclusion, les circuits SFS constituent la plus large classe de circuits combinatoires qui accomplissent le TSC GOAL.

**Définition D17**
Un circuit est "strongly redundant" pour une séquence de défauts $\langle f_1, f_2, \ldots, f_n \rangle$ et pour l'ensemble de vecteurs d'entrée X (respectivement pour le code d'entrée A), si le circuit est redondant pour les n séquences $\langle f_1 \rangle, \langle f_1, f_2 \rangle, \ldots, \langle f_1, f_2, \ldots f_n \rangle$ et pour l'ensemble de vecteurs d'entrée X (resp. pour le code

d'entrée A).

On utilise l'ordre des défauts dans la séquence $\langle f_1, f_2, \ldots, f_n \rangle$ pour définir l'ordre d'occurrence des défauts; donc $f_1$ survient le premier, $f_2$ le deuxième, etc... Il faut remarquer qu'un circuit redondant pour une séquence $\langle f_1, f_2, \ldots, f_i, f_j, \ldots, f_n \rangle$ peut ne pas l'être pour une autre séquence $\langle f_1, f_2, \ldots, f_j, f_i, \ldots, f_n \rangle$. Ce n'est pas le cas d'un circuit SFS pour une classe d'hypothèses de pannes comme défini ci-dessous [NIC 83].

**Définition D18**
Un circuit G est SFS pour une classe Cf d'hypothèses de pannes si pour toutes les séquences de fautes $f_i$ appartenant à Cf qui peuvent survenir,

soit il existe k tel que :

- G est "strongly redundant" pour la séquence de défauts $\langle f_1, f_2, \ldots, f_{k-1} \rangle$ ; et
- $\forall$ a $\in$ A soit $G(a, \langle f_1, f_2, \ldots, f_k \rangle) = G(a, \varnothing)$ ou $G(a, \langle f_1, f_2, \ldots, f_k \rangle) \notin B$ ; et
- $\exists$ a $\in$ A tel que $G(a, \langle f_1, f_2, \ldots, f_k \rangle) \notin B$

soit

- G est "strongly redundant" pour toutes les séquences de défauts.

Dans la Classe I ne sont pas considérés des défauts multiples, mais si redondances existent, le résultat peut être un défaut multiple. Le groupe de fautes n'appartient pas à la classe considérée mais chacune des fautes individuelles appartient à cette classe.

Dans la théorie classique, un contrôleur TSC est défini comme un circuit TSC et "code disjoint". Mais, à partir du moment où nous admettons des redondances, il faut définir un nouveau contrôleur qui peut atteindre le "TSC goal" même devant la possibilité d'occurrence de ces redondances.

On appelle B, le code d'entrée du contrôleur (code de sortie du bloc fonctionnel) et C le code de sortie du contrôleur.

**Définition D19**
Un contrôleur G est redondant pour un défaut f, pour un code d'entrée B et pour un code de sortie C si :

$\forall$ b $\in$ B, G(b, f) $\in$ C et
$\forall$ b $\notin$ B, G(b, f) $\notin$ C.

**Définition D20**
Un contrôleur G est "strongly redundant" pour une séquence de défauts $\langle f_1, f_2,.....,f_n\rangle$ , pour un code d'entrée B, et pour un code de sortie C s'il est redondant pour toutes les n sous-séquences $\langle f_1\rangle$, $\langle f_1,f_2\rangle$, $\langle f_1,f_2,f_3\rangle$,...,$\langle f_1,f_2,.....,f_n\rangle$ (c'est à dire que le contrôleur est "code disjoint" en présence de chacune des sous-séquences), pour le code d'entrée B et pour le code de sortie C.

Nous retrouvons ici la même observation que celle déjà expliquée pour les circuits SFS : le contrôleur redondant pour une séquence de défauts $\langle f_1,.....,f_i,f_j,....,f_n\rangle$ n'est pas nécessairement redondant pour la séquence $\langle f_1,.....,f_j,f_i,....,f_n\rangle$.

Maintenant nous pouvons définir les contrôleurs "strongly code disjoint" avec lesquels un système peut accomplir le "TSC goal" certaines conditions étant respectées, même en présence de défauts [NIC 84c]. Ils sont définis pour une séquence de défauts, un ensemble et après une classe générique de défauts.

**Définition D21**
Avant l'occurrence de défauts, le circuit G est "code disjoint. Pour une séquence de défauts $\langle f_1,f_2,.....,f_n\rangle$ , soit k le plus petit entier pour lequel il existe un vecteur b ∈ B tel que :

$$G(b, U f_j) \notin C \quad \text{où } j=1,...,k.$$

S'il n'existe pas un tel k posons k=n+1. Alors le circuit G est "strongly code disjoint" pour la séquence $\langle f_1,f_2,.....,f_n\rangle$ si :

$$\forall\, b \notin B, \forall\, m \in \{1,2,.....,k-1\}, G(b, U f_j) \notin C \quad \text{où } j=1,...,m.$$

**Définition D22**
Un circuit G est "strongly code disjoint" pour un ensemble de défauts F, s'il est "strongly code disjoint" pour chaque séquence de défauts dont les éléments appartiennent à l'ensemble F.

**Définition D23**
Un contrôleur est "strongly code disjoint" (SCD) pour une classe Cf d'hypothèses de pannes s'il est "code disjoint" et pour toutes les séquences de fautes $f_i$ appartenant à Cf qui peuvent survenir,

<u>soit</u> il existe k tel que

- le contrôleur est "strongly redundant" pour la séquence de

défauts $\langle f_1, f_2, f_3, \ldots, f_{k-1} \rangle$, et

- $\exists\, b \in B \mid G(b, \langle f_1, f_2, \ldots, f_k \rangle \notin C$,

soit

- le contrôleur est "strongly redundant" pour toutes les séquences de défauts.

Pour un système composé par un bloc fonctionnel et un contrôleur, l'hypothèse d'occurrence de défauts est enoncée ensuite.

**Hypothèse H2**
Après l'occurrence d'un défaut dans le bloc fonctionnel, il s'écoule un laps de temps suffisant pour que tous les vecteurs du code d'entrée A soient appliqués au bloc fonctionnel, avant qu'un deuxième défaut survienne au bloc fonctionnel ou au contrôleur. Après l'occurrence d'un défaut dans le contrôleur, il s'écoule un laps de temps suffisant pour que tous les vecteurs du code d'entrée B soient appliqués au contrôleur, avant qu'un deuxième défaut survienne dans le contrôleur ou dans le bloc fonctionnel.

En [NIC 83] , il est démontré qu'un système composé de :
- un circuit SFS et
- un contrôleur SCD

accomplit le but de "totally self checking goal", l'hypothèse H2 étant assurée.

Les contrôleurs classiques avaient les propriétés "code disjoint", "self testing" et "fault secure". Après le changement des définitions, il faut revoir les propriétés :

- Propriété "code disjoint": tous les contrôleurs doivent être "code disjoint" afin d'assurer la correspondance entre les entrées hors code et les sorties hors code aussi. Mais la définition des circuits "code disjoint" garantit cette propriété seulement pendant qu'il n'y a pas eu de faute dans le contrôleur, i.e., pendant le fonctionnement normal. A partir du moment où l'on considére la possibilité d'occurrence de défauts pour lesquels le circuit est redondant, la propriété "strongly code disjoint" devient nécessaire. De plus, cette dernière propriété assure la détection de la première faute survenue pour laquelle le circuit n'est pas redondant.

- Propriété "self-testing": les contrôleurs SCD n'ont pas besoin d'être "self-testing". Toutes les fautes pour lesquelles le circuit est redondant restent non détectées sans problème pour la qualité TSC du système.

- Propriété "fault-secure": au niveau des sorties des contrôleurs il faut pas que tous les vecteurs de sortie soient corrects, i.e., étant re... la propriété "code disjoint", le vecteur de sortie produit ne doit pas ... obligatoirement celui programmé.

Ainsi, la propriété "strongly code disjoint" suffit aux contrôleurs qui seront utilisés dans des systèmes supposés atteindre le "TSC" ... l'hypothèse H2 étant considérée.

D'autre définitions, encore, peuvent caractériser des contrôleurs ... comme celles présentée par la suite.

**Définition D24**
Avant l'occurrence de défauts, le circuit G est "code disjoint". Pour ... séquence de défauts $\langle f_1, f_2, \ldots, f_n \rangle$, soit k le plus petit entier pour lequel existe un vecteur b c B tel que :

$$G(b, U f_j) \notin C \quad \text{où } j=1,2,\ldots,k;$$

si un tel k n'existe pas, posons k=n+1. Alors le circuit G est "strongly ... disjoint" (SCD) pour la séquence $\langle f_1, f_2, \ldots, f_n \rangle$ si :

$$\forall b \notin B, \forall m \in \{1,2,\ldots,k\} : G(b, U f_j) \notin C \quad \text{où } j=1,2,\ldots,m$$

**Définition D25**
Le circuit est SCD pour un ensemble de défauts F, s'il est SCD (selon la définition D24) pour chaque séquence de défauts dont les éléments appartiennent à l'ensemble F.

**Définition D26**
Un contrôleur est SCD pour une classe Cf d'hypothèses de pannes s'il est "code disjoint" et si pour chaque séquence de défauts $\langle f_1, f_2, \ldots, f_n \rangle$ due à la classe Cf qui peut survenir,
soit il existe k tel que :

- le contrôleur est "strongly redundant pour la séquence de défauts $\langle f_1, f_2, \ldots, f_{k-1} \rangle$ ,
- $\forall b \notin B, G(b, \langle f_1, f_2, \ldots, f_k \rangle \notin C$.

soit

- le contrôleur est "strongly redundant" pour toutes les séquences de défauts.

On peut remarquer que la définition D26 (et respectivement les définitions D24 et D25) est plus forte que D23 (respectivement D21 et D22)

puisqu'elle concerne une propriété pas vraiment nécessaire aux contrôleurs: ils sont "code disjoint" pour $\forall$ b $\in$ B et $\forall$ b $\notin$ B même en présence d'une faute $f_k$ pour laquelle le circuit n'est pas redondant.

En raison de sa rigueur, la définition D26 peut être utilisée avec une hypothèse moins forte que H2 dans la composition d'un système qui atteint le "TSC goal" [JAN 84c].

**Hypothèse H3**
Entre l'occurrence de deux défauts quelconques affectant le bloc fonctionnel, ou entre l'occurrence de deux défauts quelconques affectant le contrôleur, il s'écoule un laps de temps suffisant pour que tous les vecteurs du code d'entrée A soient appliqués.

Sous l'hypothèse H3, un défaut peut survenir au bloc fonctionnel (ou au contrôleur) avant qu'une faute existante dans le contrôleur (ou dans le bloc fonctionnel) ait été détectée. Le bloc fonctionnel étant "strongly redundant", pour la séquence de fautes $\langle f_1, f_2, \ldots, f_{m-1} \rangle$ et le contrôleur étant "strongly redundant" pour la séquence de fautes $\langle f_1', f_2', \ldots, f_{n-1}' \rangle$, ces défauts notés $f_m$ et $f_n'$, respectivement, peuvent survenir même si le bloc fonctionnel et le contrôleur ne sont pas redondants pour les séquences de défauts résultantes $\langle f_1, f_2, \ldots, f_m \rangle$ et $\langle f_1', f_2', \ldots, f_n' \rangle$, respectivement. Aucune autre faute ne sera acceptée.

**Proposition P1**
L'hypothèse H3 étant respectée, un système composé de :
- un circuit SFS et
- un contrôleur SCD (Définition D26)

atteint le "TSC goal".

**Démonstration de P1**
Initialement, il n'y a pas de faute dans le système ; donc, pour le bloc fonctionnel $\forall$ a $\in$ A, $G_b(a, \varnothing) \in B$ et pour le contrôleur $\forall$ b $\in$ B, $G_c(b, \varnothing) \in C$ et $\forall$ b $\notin$ B, $G_c(b, \varnothing) \notin C$; si des fautes ou des séquences de défauts sont survenues, les blocs sont redondants vis-à-vis de ces défauts, pour le bloc fonctionnel , $\forall$ a $\in$ A: $G_b(a, \langle f_1 f_2, \ldots, f_{m-1} \rangle = G_b(a, \varnothing)$ et pour le contrôleur $\forall$ b $\in$ B, $G_c(b, \langle f_1', f_2', \ldots, f_{n-1}' \rangle \in C$ et $\forall$ b $\notin$ B, $G_c(b, \langle f_1', f_2', \ldots, f_{n-1}' \rangle \notin C$. Le bloc fonctionnel est nommé $G_b$ et le contrôleur, $G_c$, pour abréger.

Une (nouvelle) faute peut se produire, affectant soit le bloc fonctionnel, soit le contrôleur. Ou, une faute peut survenir à un de ces blocs avant qu'une autre déjà arrivée dans l'autre bloc ait été détectée.

Une des trois situations peut en résulter : il y aura soit une faute dans le bloc fonctionnel, soit une faute dans le contrôleur, soit une faute dans le bloc fonctionnel (respectivement dans le contrôleur) et une autre dans le contrôleur (respectivement dans le bloc fonctionnel). Chacune de ces situations est analysée dans un des cas suivants.

Cas 1:

Une faute $f_m$, m=1,2,3,..., se produit dans le bloc fonctionnel. Avec l'hypothèse H3, toutes les entrées a ∈ A seront appliquées avant l'occurrence d'un nouveau défaut dans ce bloc. Selon la définition des circuits SFS, deux situations peuvent se produire :

a) $G_b$ est "strongly redundant" pour le défaut $f_m$ ou pour la séquence de défauts en résultant : pour tous a ∈ A et toutes les séquences de fautes $f_i$ appartenant à F, $G_b(a,\langle f_1,f_2,.....,f_m\rangle) = G(a, ø)$. Donc, le système ne produira pas de sorties hors code, ni de sorties erronées non plus.

b) Soit m=1, et $\exists$ a ∈ A | $G_b(a,f_1) \notin B$; soit m = 2,3,4,..... et $G_b$ est "strongly redundant" pour la séquence de défauts $\langle f_1,f_2,.....,f_{m-1}\rangle$ mais $\exists$ a ∈ A | $G_b(a, \langle f_1,f_2,.....,f_m\rangle) \notin B$ lequel rend sûr la détection de $f_m$. Par conséquent, la première sortie erronée du bloc fonctionnel sera un vecteur hors code. Jusqu'à ce moment, aucune faute n'avait affecté le contrôleur, ou il est "strongly redundant" pour la séquence déjà arrivée: il reçoit les sorties du bloc fonctionnel et produit :

$$\forall b \in B, G_c(b, ø) \in C$$

$$\forall b \notin B, G_c(b, ø) \notin C,$$

ce dernier permettant la détection du défaut $f_m$ (dans le bloc fonctionnel). Dans le cas où une séquence de fautes pour laquelle le contrôleur est redondant serait déjà produit, le symbole "ø" dans les expressions est substitué par $\langle f_1,f_2,.....,f_{n-1}\rangle$ .

Si la prochaine faute à apparaître dans le système survient au bloc fonctionnel, le comportement sera semblable à celui décrit ci-dessus. Cette faute survenant au contrôleur avant la détection de l'autre existante dans le

bloc fonctionnel résultera en un comportement comme celui décrit par le cas 3.

Cas 2:

Une faute $f_n'$, n=1,2,3,..., se produit dans le contrôleur. Par l'hypothèse H3, tous les vecteurs d'entrée a ϵ A seront appliqués au bloc fontionnel avant l'occurrence d'un nouveau défaut dans le contrôleur; comme une autre faute ne survient pas au bloc fonctionnel, ou il est "strongly redundant" pour la séquence déjà survenue, tous les b ϵ B seront reçus par le contrôleur. D'après la définition des contrôleurs SCD, deux situations peuvent résulter :

a) $G_c$ est "strongly redundant" pour la séquence de défauts, donc pour tous b ϵ B, $G(b,\langle f_1', f_2', \ldots\ldots, f_n'\rangle)$ ϵ C. Ainsi le système ne produit pas de vecteurs hors code, ni de sorties erronées.

b) soit m=1 et ∀ b ϵ B, $G_c(b_1,f_1)$ ϵ C ou $G_c(b_1,f_1) \notin$ C; soit m = 2,3,4,.... et $G_c$ est "strongly redundant" pour la séquence de défauts $\langle f_1', f_2', \ldots\ldots, f_{n-1}'\rangle$ et pour tous b ϵ C, $G_c(b_1,\langle f_1', f_2', \ldots\ldots, f_n'\rangle)$ ϵ C ou $G_c(b_1, \langle f_1', f_2', \ldots\ldots, f_n'\rangle) \notin$ C. Une sortie hors code ainsi générée, permet la détection du défaut.

Si la prochaine faute à apparaître dans le système survient au contrôleur, le comportement sera analogue à celui décrit ci-dessus. Si ce défaut survient au bloc fonctionnel avant la détection de l'autre dans le contrôleur, le comportement résultant est décrit par le prochain cas.

Cas 3:

Une faute survient au contrôleur avant qu'une autre, déjà arrivée au bloc fonctionnel ait été détectée, ou un faute survient au bloc fonctionnel avant qu'une autre déjà arrivée au contrôleur ait été détectée. Par l'hypothèse H3, tous les vecteurs d'entrée a ϵ A seront appliqués au bloc fonctionnel, avant l'occurrence d'une autre faute dans le même bloc où la faute précédente s'était produite. Selon les définitions, quatre situations peuvent en résulter :

a) le bloc fonctionnel et le contrôleur sont "strongly redundant" pour les séquences de défauts. Donc, pour les séquences de défauts en résultant, pour tous a ϵ A, $G_b(a,\langle f_1, f_2, \ldots\ldots, f_m\rangle) = G_b(a, \emptyset)$ et pour tous b ϵ B, $G_c(b,\langle f_1',$

$f_2', \ldots, f_n'\rangle \in C$. Par conséquent, le système ne produira pas de vecteurs hors code, ni de sorties erronées.

b) le contrôleur est "strongly redundant" pour la séquence de défauts mais le bloc fonctionnel n'en est pas. Alors, nous revenons au cas 1b.

c) le bloc fonctionnel est "strongly redundant" pour la séquence de défauts, mais le contrôleur n'en est pas. Alors, nous revenons au cas 2b.

d) le bloc fonctionnel et le contrôleur ne sont pas redondants pour les séquences de défauts. Une faute peut survenir soit, d'abord au bloc fonctionnel soit, d'abord au contrôleur. Analysons les situations possibles.

d' : une faute survient initialement au bloc fonctionnel. Une deuxième se produit dans le contrôleur avant que les vecteurs a $\in$ A aient été générés (sinon, le cas 1 s'applique). Cette situation est illustrée par le diagramme de temps présenté à la figure 17.

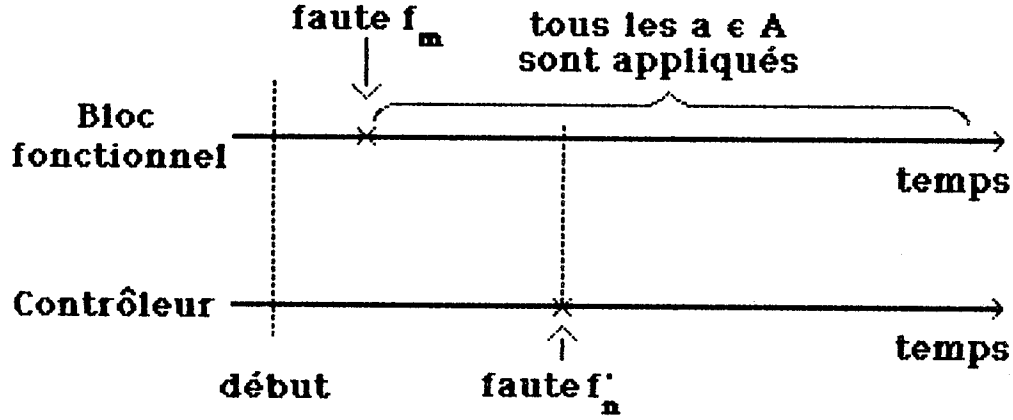



Figure 17 : Rapport de temps entre les occurrences de défauts dans le bloc fonctionnel et dans le contrôleur (cas d')

Soit $a_1$ le a $\in$ A pour lequel un b $\notin$ B est produit ; i.e., soit m = 1, $a_1 \in A$ -> $G_b(a_1, f_1) \notin B$ et soit m=2,3,4,..., $G_b$ est "strongly redundant" pour la séquence de défauts $\langle f_1, f_2, \ldots, f_{m-1}\rangle$ et pour $a_1 \in A$, $G_b(a_1, \langle f_1, f_2, \ldots, f_m\rangle) \notin B$. Deux situations sont possibles (d'1 et d'2).

d'1 : $a_1$ est généré avant l'occurrence du défaut dans le contrôleur.

Dans ce cas, un vecteur hors code sera produit comme entrée pour le contrôleur en raison de $f_m$, et sa détection est assurée par la définition SCD (définition forte ou faible).

d'2 : $a_1$ est généré après l'occurrence du défaut dans le contrôleur. Le contrôleur reçoit une sortie du bloc fonctionnel $G_b(a_1,\langle f_1,f_2,...,f_m\rangle) \notin B$ et la définition SCD forte est nécessaire: le contrôleur est "strongly redundant" pour la séquence $\langle f_1',f_2',...,f_{n-1}'\rangle$ et $\forall\, b \notin B$, $G_c(b, \langle f_1',f_2',...,f_n'\rangle) \notin C$, étant n = 2, 3,4,... Si n = 1, $\forall b \notin B$, $G_c(b, f_1') \notin C$.

d" : une faute se produit initialement dans le contrôleur. Une deuxième survient au bloc fonctionnel avant que tous les vecteurs a c A aient été appliqués (sinon, le cas 2 s'applique). Cette situation est illustrée par le diagramme de temps présenté à la figure 18.

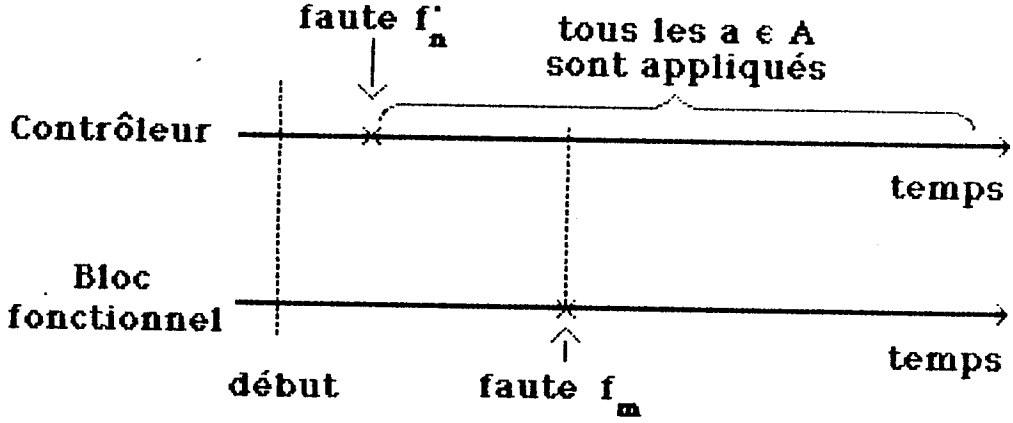



Figure 18 : Rapport du temps entre les occurrences de défauts dans le contrôleur et dans le bloc fonctionnel (cas d")

L'application de tous les vecteurs a ∈ A au bloc fonctionnel résulte en tous les b ∈ B étant appliqués aux entrées du contrôleur, pendant le fonctionnement normal. Soit $b_1$ le vecteur b ∈ B pour lequel $G_c \notin C$, étant $b_1$ généré par $a_1'$. Nous remarquons que $a_1'$ n'a pas de rapport avec $a_1$, mais il peut arriver que $a_1 = a_1'$. Deux situations sont possibles (d"1 et d"2).

d"1 : $a_1'$ est appliqué avant que la faute apparaisse dans le bloc fonctionnel. Donc $b_1$ est généré, et $G_c(G_b(a_1',\emptyset),\langle f_1',f_2',.....f_n'\rangle) \notin C$, lequel permet la détection du défaut $f_n'$ dans le contrôleur. Le symbole "ø" sera

substitué par une séquence $\langle f_1, f_2, \ldots, f_{m-1} \rangle$ si des défauts pour lesquels le bloc fonctionnel est redondant sont déjà survenus.

d''2 : $a_1'$ est appliqué après la survenance du défaut dans le bloc fonctionnel. Donc, $b_1$ n'est pas nécessairement généré. Le bloc fonctionnel est SFS, et $\forall$ a $\in$ A, $G_b(a_1, \langle f_1, f_2, \ldots, f_m \rangle) = G_b(a, \emptyset)$ ou $G_b(a_1, \langle f_1, f_2, \ldots, f_m \rangle) \notin B$. Si $G_b(a_1, \langle f_1, f_2, \ldots, f_m \rangle) = G_b(a, \emptyset)$ et si $a = a_1'$, $G_b(a_1', \langle f_1, f_2, \ldots, f_m \rangle) = b_1 \in B$. Dans cette situation, le défaut $f_n'$ dans le contrôleur est détecté (avec n'importe laquelle des définitions SCD) puisque $G_c(b_1, \langle f_1', f_2', \ldots, f_n' \rangle) \notin C$. Pendant que $G_b(a_1, \langle f_1, f_2, \ldots, f_m \rangle) = G_b(a, \emptyset)$ et $a = a_1'$, les défauts ne sont pas détectés car $G_c(G_b(a, \langle f_1, f_2, \ldots, f_m \rangle), \langle f_1', f_2', \ldots, f_n' \rangle) \in C$. Si $a = a_1$, $G_b(a_1, \langle f_1, f_2, \ldots, f_m \rangle) \notin B$ et la propriété SCD <u>forte</u> est nécessaire au contrôleur afin d'assurer que $G_c(G_b(a, \langle f_1, f_2, \ldots, f_m \rangle), \langle f_1', f_2', \ldots, f_n' \rangle) \notin C$. Cela signifie que des sorties hors code seront produites par le contrôleur (en présence de défaut) à partir des vecteurs hors code générés par le bloc fonctionnel.

Q.E.D.

Nous remarquons que la proposition P1 résulte en une condition suffisante mais pas nécessaire pour un certain nombre de situations car, en général, le laps de temps d'application des vecteurs du code d'entrée A au bloc fonctionnel est plus grand que celui demandé pour la détection des défauts dans le contrôleur. Nous expliquons: l'ensemble des vecteurs d'entrée indispensables au test du contrôleur est généré par l'application des vecteurs d'entrée au bloc fonctionnel. Si cet ensemble est obtenue par l'application d'une partie des vecteurs d'entrée du système (un sous-ensemble, donc), seulement ce laps de temps est nécessaire dans le cas de défaut uniquement dans le contrôleur. Si l'on tient compte de cette remarque, on peut énoncer une nouvelle hypothèse, à partir de H3. Cependant, elle ne sera pas utile du point-de-vue pratique une fois que l'on a considéré H3 pour n'importe quel type de défaut qui survient au système, en principe, sans l'attacher à un des blocs.

## III.2. CONCEPTION DES CONTROLEURS A PARTIR DES CELLULES

La conception d'un contrôleur nouveau peut devenir nécessaire si l'on considère les différents codes, le nombre d'entrées dépendant des lignes de sortie du bloc fonctionnel et probablement les changements topologiques en

fonction de l'arrangement des autres blocs du système. Cette conception sera une activité laborieuse il faudra analyser toutes les situations possibles de défauts qui peuvent survenir, suivies des changements topologiques correspondants nécessaires.

Ce travail peut être réduit significativement, à la mesure que l'on utilise une bibliothèque de cellules de contrôleurs. Il faut les concevoir une fois, étudier les possibilités d'assemblage pour ces cellules et l'utilisateur les assemblera selon son besoin - type de code, nombre d'entrées et topologie convenable [JAN 83b, JAN 84a].

Le but de cette section est l'étude de règles d'assemblage afin d'obtenir un contrôleur SCD à partir des cellules SCD. Plus précisément, les propriétés SCD d'une cellule sont définis vis-à-vis de :

- un ensemble ou une classe d'hypothèses de pannes ;

- un ensemble de vecteurs d'un code d'entrée (en général, tous les vecteurs du code qui appartiennent à l'ensemble des vecteurs d'entrée) ;

- une hypothèse d'occurrence de défauts concernant un circuit SFS et un contrôleur SCD (l'hypothèse H2, par exemple) afin d'assurer l'accomplissement du "TSC goal".

Ces propriétés devront être définies pour un contrôleur conçu à partir de cellules SCD. Tout d'abord, nous commenterons des aspects généraux. Ensuite nous étudierons la conception d'une cellule et des règles générales pour le dessin complet d'un contrôleur, vis-à-vis de la Classe I d'hypothèses de pannes.

## III.2.1. Cellules de base et problèmes d'assemblage

### III.2.1.1. Hypothèses de pannes

Quand des cellules sont assemblées en ayant pour but un contrôleur, des défauts peuvent affecter des cellules et/ou des lignes utilisées pour interconnecter des cellules. C'est-à-dire, ces fautes peuvent être :

F1. faute dans une cellule,
F2. faute affectant une seule interconnexion,
F3. court-circuit entre cellules,
F4. court-circuit entre interconnexions,
F5. court-circuit entre cellule et interconnexion.

Les défauts pouvant survenir dans une cellule sont étudiés pendant sa conception. Leur détection aux sorties de la cellule est assuré à ce niveau, et il ne faut pas revoir le cas durant l'assemblage des cellules.

Les fautes affectant une interconnexion dépendent de la classe d'hypothèses de pannes. Par exemple, si la Classe I est prise en compte, ces fautes s'agissent surtout de coupures. Ces défauts résultent en "collages", et peuvent être détectés facilement car avec l'application des entrées appartenant à l'ensemble des vecteurs du code d'entrée ces lignes prennent les deux valeurs possibles. Donc, un vecteur hors code sera rapidement produit aux sorties générales du contrôleur en raison de la condition de collage. Ces coupures des interconnexions peuvent être vues comme transférées logiquement aux entrées de la prochaine cellule et ce cas est aussi examiné pendant la conception d'une cellule.

Les courts-circuits entre des cellules devront être étudiées, la Classe I étant considérée. Cette vérification est faite à l'égard de la redondance du contrôleur pour le défaut résultant : le contrôleur étant redondant pour le court-circuit, ou ceci étant détecté par un des vecteurs du code, il ne pose pas de problèmes. La difficulté se rapporte aux courts-circuits pour lesquels le circuit n'est pas redondant vis-à-vis (seulement) des vecteurs hors code -par conséquent ils ne sont pas détectés pendant le fonctionnement normal du bloc fonctionnel et l'on devra empêcher qu'ils puissent survenir. Ceci sera examiné plus tard.

La détection de courts-circuits entre interconnexions peut être assurée si chaque paire de lignes internes prend à la fois des valeurs différentes, la technologie NMOS étant considérée. Si nous particularisons cette situation à la Classe I et si nous nous attachons uniquement aux cas des courts-circuits entre des lignes de même niveau (dans la structure en arbre), la détection est rendue sûre pour chaque paire de lignes internes voisines qui reçoivent les valeurs 01 ou 10 à la fois. D'autres situations comme des courts-circuits affectant des lignes qui appartiennent à différents niveaux dans l'arbre d'assemblage ou telles que l'ensemble complet des vecteurs d'entrée n'est pas reçu par le circuit, devront faire l'objet d'études spécifiques.

Dans le cas de courts-circuits entre cellules et interconnexions nous considérons les lignes internes aux cellules et non les lignes qui conduisent les signaux d'entrée et sortie. Ces dernières avaient déjà été examinées dans le cas précédent. Ces défauts pourront être détectés, en général, à condition que les lignes prennent des valeurs différentes pendant l'application des vecteurs du code d'entrée. Le même problème souligné auparavant (dans le cas de lignes appartenant à des cellules différentes)

des défauts non détectables malgré le fait que le circuit ne soit pas redondant vis-à-vis d'eux, peut aussi se produire dans la situation présente.

Ainsi, la plus grande difficulté rapportée à la conception des contrôleurs SCD qui reçoivent un ensemble complet de vecteurs d'entrée et qui sont considérés sous une hypothèse de fautes comme par exemple l'hypothèse H2, réside dans l'élimination des défauts non détectables par l'application des vecteurs du code mais pour lesquels le contrôleur n'est pas redondant. Nous rappelons que dans le cas des contrôleurs, le circuit est dit redondant pour une faute si la fonction n'est pas changée pour l'ensemble des entrées appartenant ou non au code. Ce cas de défauts non-détectables est illustré par les exemples qui suivent.

Dans la figure 19, un court-circuit est indiqué entre des sorties intermédiaires d'un contrôleur de parité pair, lesquelles prennent des valeurs égales durant le fonctionnement normal - donc la détection du défaut devient impossible. Un tel court-circuit empêche le contrôleur d'être SCD. Cependant, le problème peut être évité avec une modification du circuit de base ou d'une conception attentive (c'est-à-dire, qu'il sera surmonté par une implémentation physique convenable).

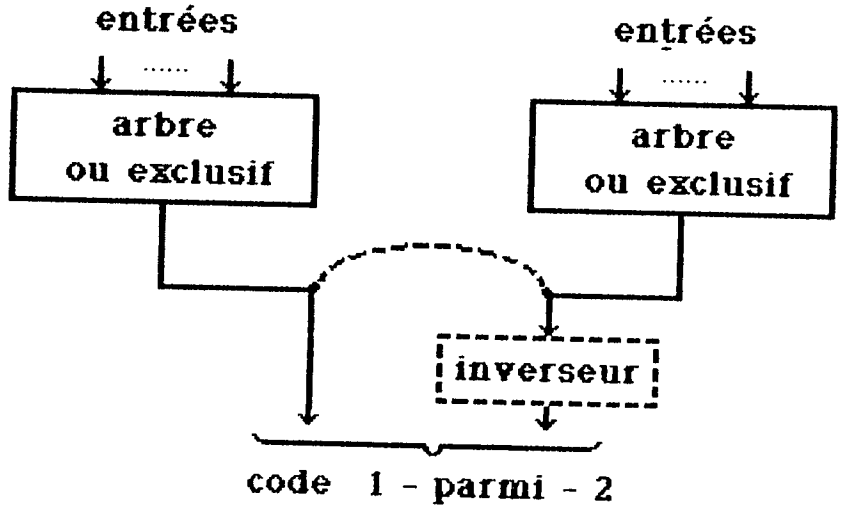



Figure 19 : Court-circuit indétectable dans le contrôleur à parité

Un autre exemple, représenté par la figure 20, est le court-circuit entre le point P et la sortie intermédiaire $f_2$ des cellules double-rail assemblées, même si isolées, elles sont SCD. Cette panne n'est pas détectable

avec les entrées du code, et la conception de la cellule devra être développée d'une telle façon que le court-circuit ne soit pas physiquement possible.

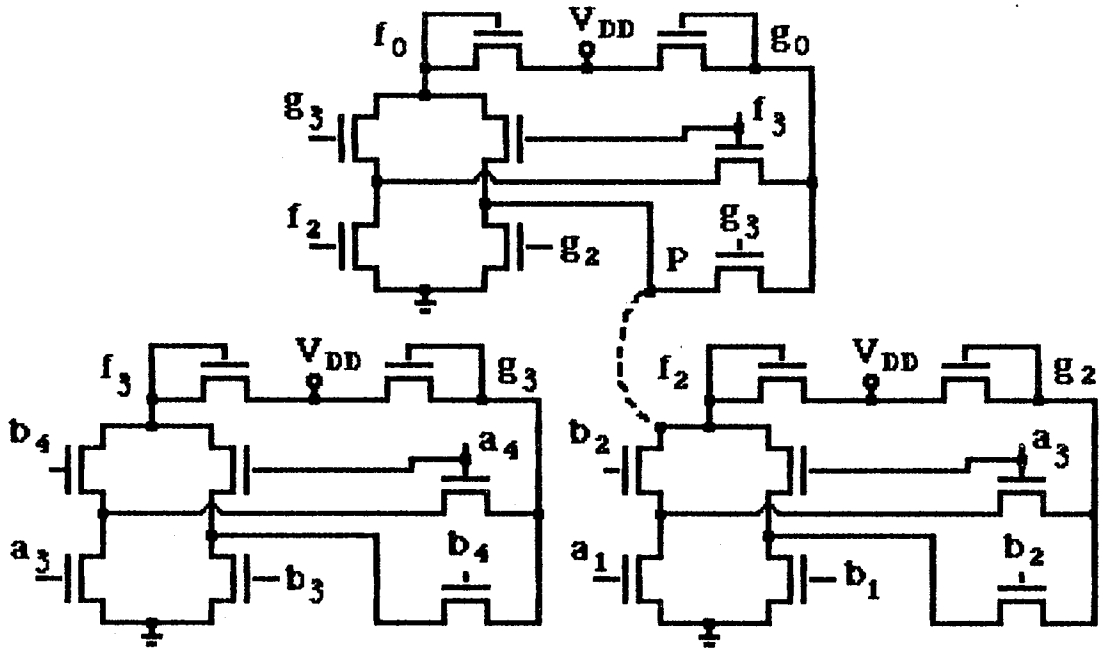



Figure 20 : Assemblage de cellules double-rail - le circuit résultant peut ne pas être SCD

D'autres défauts peuvent provoquer des oscillations dans le circuit ou donner des propriétés séquentielles indésirables. Leur occurrence doit être évité.

Il devient évident que l'assemblage de cellules SCD et la détectabilité de toute faute possible parmi les interconnexions avec l'application de l'ensemble des vecteurs du code n'assure pas la propriété SCD du contrôleur résultant. Il faut introduire des contraintes à la conception des cellules aussi bien qu'à leur assemblage afin d'empêcher les fautes en résultant non détectables comme celles exemplifiées, affectant les cellules, ou cellule et interconnexion. Si ces contraintes ne sont pas fournies explicitement à l'utilisateur d'une bibliothèque de cellules, il lui faudra du temps et des efforts assez importants pour tester toutes les possibilités de défauts introduites par l'assemblage des cellules pour qu'il soit sûr de la propriété SCD du contrôleur résultant. Nous reviendrons sur ces idées dans la section III.2.3.

III.2.1.2. Vecteurs d'entrée

Généralement, tous les vecteurs d'entrée sont nécessaires pour tester toutes les fautes qui peuvent survenir dans une cellule SCD. Mais ce fait n'implique pas qu'il faille appliquer tous les vecteurs d'entrée au contrôleur construit à partir de cellules indépendantes afin d'assurer la détection des défauts possibles.

Nous avons vu, dans la sous-section précédente III.1.1., qu'il y a cinq groupes de défauts. Chacun de ces groupes est testé par un ensemble de vecteurs du code selon la description ci-dessous :

- les fautes détectables dans les cellules sont testées par l'application de l'ensemble complet des vecteurs du code d'entrée de la cellule;

- les fautes affectant une interconnexion sont testées par l'application des deux valeurs logiques différentes à chacune de ces lignes;

- les courts-circuits entre cellules sont testés par les valeurs d'entrée du contrôleur qui forcent une des régions affectées à, comme conséquence d'un changement de valeur logique, provoquer une erreur aux sorties de la cellule respective. Si l'on ne trouve cette condition qu'en appliquant des vecteurs hors code, le court-circuit doit être évité par des changements topologiques;

- les courts-circuits entre des interconnexions sont testés par l'application des combinaisons d'entrée qui provoquent des valeurs différentes à chaque paire des lignes (01 ou 10);

- les courts-circuits entre cellule et interconnexion sont testés de la même façon que ceux entre deux cellules.

Donc, afin de tester tous les types de défauts listés, nous aurons besoin d'un groupe de vecteurs d'entrée composé par tous ceux qui sont nécessaires à chacun des items ci-dessus, lequel est un sous-ensemble de l'ensemble complet des vecteurs du code (ils peuvent être égaux ou non).

Il reste encore quelques remarques concernant l'ensemble des entrées du code à être appliquées en vue de la détection des défauts, en fonction des hypothèses de pannes considérées.

Plusieurs auteurs ([CAR 68], [CRO 78], parmi d'autres) ont affirmé que l'application de tout l'ensemble des entrées du code au contrôleur était nécessaire, comme garantie de détection des fautes dans les circuits TSC/"code-disjoint". Un tel but pouvait être nécessaire quand les modèles de représentation de fautes par des collages étaient utilisés. Du moment qu'un modèle réel est considéré et les redondances devenues possibles, la détection de tous les défauts pouvant survenir n'est plus toujours exigée.

Ceci est le cas des circuits SCD proposés par la suite. Contrairement aux contrôleurs TSC/"code-disjoint", dans lesquels toutes les fautes survenant devraient être détectées, dans les circuits SCD quelques pannes restent non-détectées et, de même, ils respectent encore les spécifications initiales. Nous faisons référence aux situations d'occurrence des défauts pour lesquels le circuit est redondant et qui ne seront pas détectés par l'application de l'ensemble complet des vecteurs d'entrée.

#### III.2.1.3. Hypothèses d'occurrence de pannes

D'une façon générale, nous considérons l'hypothèse de pannes H2 pour tout le système. Donc, en ce qui concerne le contrôleur complet, nous allons avoir une seule faute interne (parmi les cinq types classés de F1 à F5) et d'autres fautes ne peuvent pas survenir avant que l'ensemble des vecteurs du code soit appliqué. Cependant, dans certains cas des arrangements de cellules, d'hypothèses plus faibles peuvent être examinées. Ces possibilités seront analysées vis-à-vis des cas particuliers de contrôleurs et d'organisations internes bien définies dans la section des "Applications".

### III.2.2. Conception d'une cellule

Par la suite, nous examinerons une procédure pratique pour la conception d'une cellule de contrôleur SCD. Elle doit être regardée telle qu'une aide pour les premiers dessins de ces cellules, pas comme une méthode. Il est possible d'utiliser un circuit logique (de nature cellulaire) pour le dessin initial, une proposition classique d'un contrôleur TSC. Après avoir obtenu un premier essai topologique pour la cellule de base et éliminer les redondances évidentes, il faut analyser les défauts possibles et le comportement résultant du circuit dans le cas où ils arrivent. Des défauts comme des contacts ou pré-contacts défaillants ou des coupures des lignes d'alimentation causent des erreurs simples et peuvent être détectés par l'application des vecteurs du code. Les coupures des lignes d'alimentation provoqueront des erreurs unidirectionnelles, et leur

détection est assurée par l'utilisation d'un code convenable (avec des sorties complémentaires, par exemple) et si la parité d'inversion est la même pour les sorties des portes alimentées et les sorties générales.

Les types de fautes plus difficiles à analyser sont les courts-circuits, car ils peuvent avoir pour résultat la création de fonctions de sortie nouvelles. Dans la Classe I d'hypothèses de défauts, ces courts-circuits peuvent affecter soit des lignes d'aluminium, soit celles de diffusion. S'il s'agit de deux lignes d'alimentation et si elles sont du même type, le circuit est redondant pour le défaut résultant ; si elles sont de types différents, on suppose que le circuit est détruit. Si seulement l'une des deux est une ligne d'alimentation, la ligne de signal affectée par le court prend une valeur en permanence, condition détectable par l'application des combinaisons d'entrée. Enfin, étant toutes les deux des lignes de signal, trois possibilités peuvent survenir:

1. la fonction exécutée par le circuit n'est pas modifiée; donc il sera redondant pour ce défaut et il ne se pose pas de problème;

2. la fonction exécutée par le circuit est modifiée et cette situation peut être reconnue avec l'application de l'ensemble des vecteurs du code (i.e., le circuit est "self-testing" pour ce court-circuit);

3. il y a une modification de la fonction exécutée mais cette situation ne peut pas être reconnue avec l'application de l'ensemble des vecteurs du code; i.e., des nouveaux termes sont ajoutées aux équations correspondants à la description du circuit, lesquels peuvent toujours être simplifiés avec les valeurs normales d'entrée, par exemple.

La topologie où cette dernière alternative peut se produire, devra être modifiée soit par le déplacement des lignes considérées, soit par l'implémentation d'une des deux avec un autre matériel, ou par l'introduction d'une troisième ligne implémenté avec le même matériel entre les deux premières.

Dans le cas particulier des contrôleurs, il n'y a pas un grand nombre de propositions différentes de circuits optimisés pour la conception de chaque cellule, et leur arrangement final dépendra du nombre des lignes d'entrée et de l'espace disponible. Il faut encore examiner les possibilités des défauts crées par l'assemblage des cellules, mais cela fera l'objet de la

prochaine section. Tout d'abord, donnons un exemple de conception d'une cellule pour contrôleur double-rail [JAN 83a].

Prenons comme proposition de base pour la cellule, le circut logique suggéré par Carter et Schneider [CAR 68] (figure 6). Pour cette cellule, une proposition topologique est présentée à la figure 21. Le court-circuit signalé S1-S2, qui peut survenir dans l'implémentation réelle, n'est pas détectable avec l'application de l'ensemble des entrées du code [(01,10), (01,01), (10,10), (10,01)]. Le diagramme des transistors correspondant est montré par la figure 22 ; nous pouvons remarquer que les valeurs de sortie, sous l'occurrence du court-circuit 1, sont égales à celles qui seraient obtenues pendant le fonctionnement normal. Et pourtant, le contrôleur n'est pas redondant pour ce défaut par la définition D19, car à l'application des vecteurs d'entrée hors code [(00,11), (11,00)], les sorties sont des valeurs appartenant à l'ensemble des sorties dans le code. Donc, cette proposition n'est pas SCD.

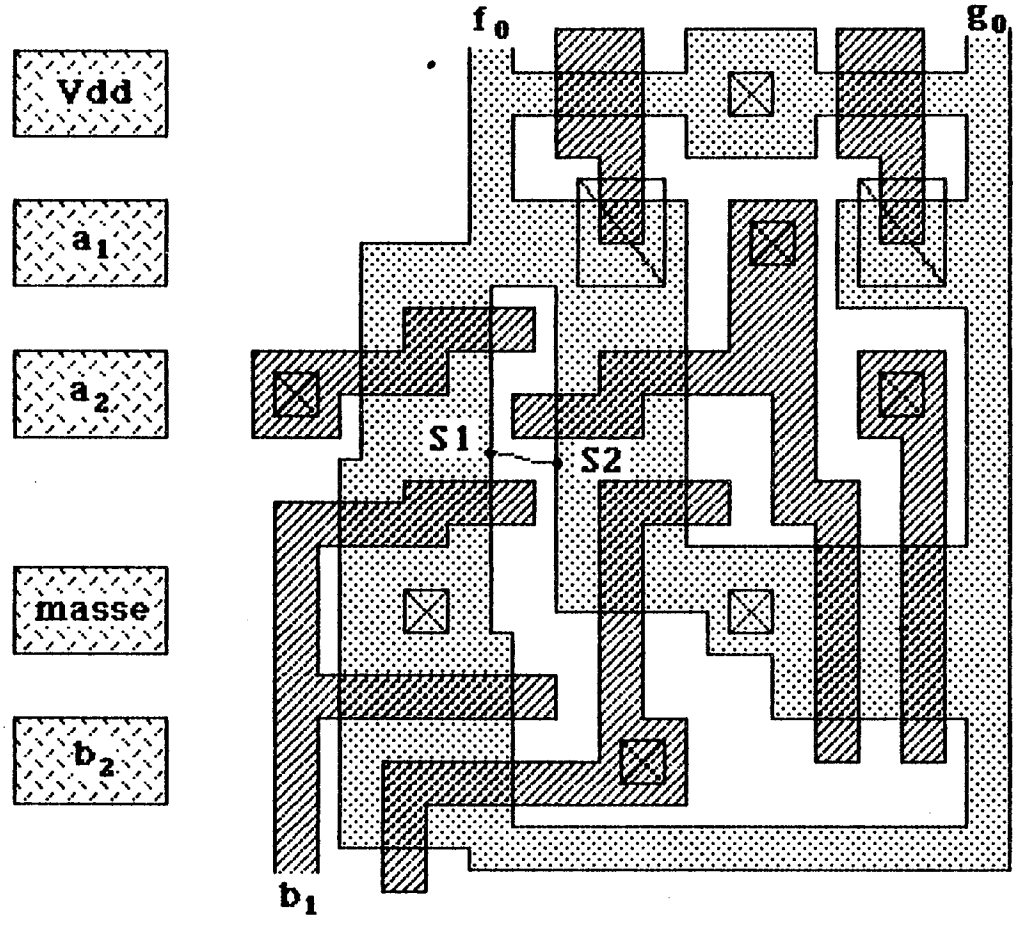



Figure 21 : Proposition topologique d'une cellule double-rail qui n'est pas SCD (technologie NMOS)

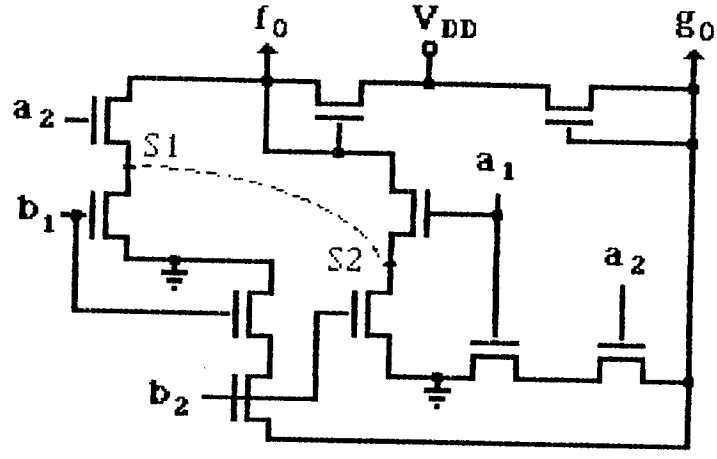



Figure 22: Circuit électrique avec la même topologie du dessin de la figure 21

Une proposition SCD est illustrée par la figure 23. Etant considérée la Classe I des hypothèses de pannes, le circuit est "code disjoint" pendant le fonctionnement normal et, si des défauts se produisent, soit le circuit est "strongly redundant" pour eux, soit le défaut ou la séquence de fautes est détecté. Une coupure de la ligne de diffusion ou un contact défaillant signalés "$f_r$" dans la figure 23 sont des défauts pour lesquels le circuit est redondant. Par ailleurs, un cas assez intéressant est celui de la séquence de fautes $\langle f_1, f_2, f_3 \rangle$ (signalé sur la même figure) pour laquelle le circuit est aussi redondant. A partir de cet exemple, la possibilité de suppression d'un des transistors affectés devient évidente. Des considérations analogues nous amènent à supprimer un deuxième transistor de la branche symétrique; par conséquent, nous aurons une cellule résultante avec 8 transistors [NIC 84c]. Des propositions topologiques pour ce dernier cas seront étudiées dans la section III.3.1.

### III.2.3. Règles de conception

Dans la sous-section précédente, nous avons examiné une succession d'actions pour le dessin de cellules SCD, commenté des propositions, mais nous n'avons pas établi de règles. Quelques règles peuvent être trop restrictives et même pas nécessaires à tous les types de contrôleurs de codes ; par contre, elles peuvent aider à la conception de nouvelles cellules pour des contrôleurs de codes différents, avec une autre organisation ou d'autres dimensions plus adaptées à des applications spécifiques. Une cellule bien conçue peut aussi rendre plus facile l'activité d'assemblage. Les

contraintes pour leur assemblage, données avec les propositions des cellules, aident l'utilisateur de cette bibliothèque à les organiser et à vérifier la propriété SCD du contrôleur résultant.

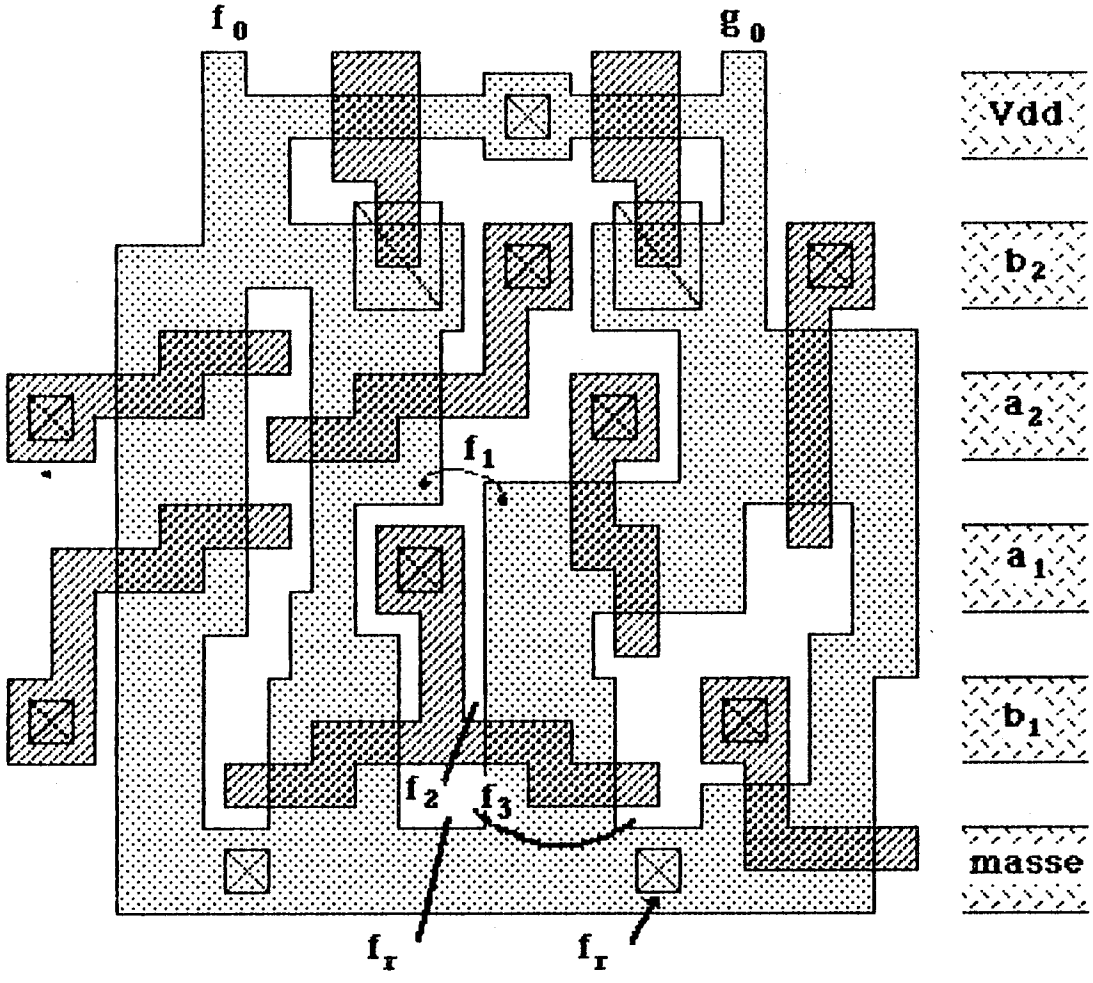



Figure 23: Implémentation SCD d'une cellule du code double-rail, pour la Classe I

Les règles présentées par la suite se rapportent à la topologie des cellules et des interconnexions; leur but principal est d'éviter l'occurrence des défauts ne détectables qu'avec l'application d'entrées hors code, et causés par un des cas possibles de courts-circuits décrits à la sous-section III.2.1.1.

Nous rappelons que la Classe I d'hypothèses de pannes est considérée. Cette classe n'inclut pas les courts-circuits entre deux lignes de polysilicium. On considère que ces défauts surviennent durant le processus de fabrication [COU 81]. Donc, cette caractéristique peut être utilisée afin de réduire des défauts possibles - voilà l'origine de la règle R1.

**Règle R1**
Toutes les interconnexions parmi des cellules sont faites avec des lignes en polysilicium. Les interconnexions référées ici sont des lignes pour les signaux d'entrée et de sortie de chaque cellule et ne concernent pas les lignes d'alimentation.

**Proposition P2**
L'assemblage de cellules éxécutée respectant la règle R1, élimine la possibilité des courts-circuits avec des lignes d'interconnexion (items F4 et F5, sous-section III.2.1.1.).

Les cas référés par la proposition P2 sont illustrés à la figure 24.

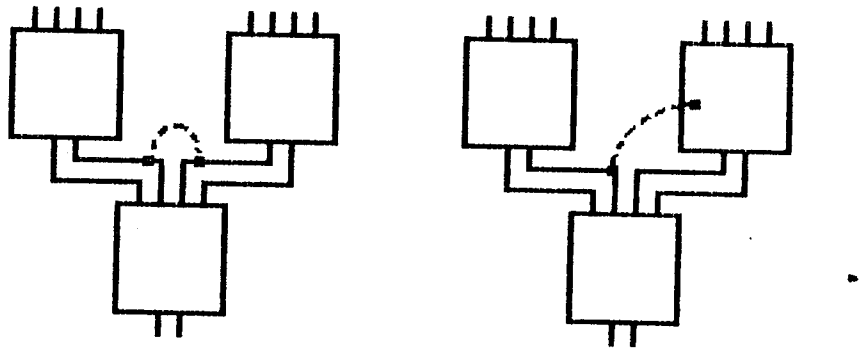
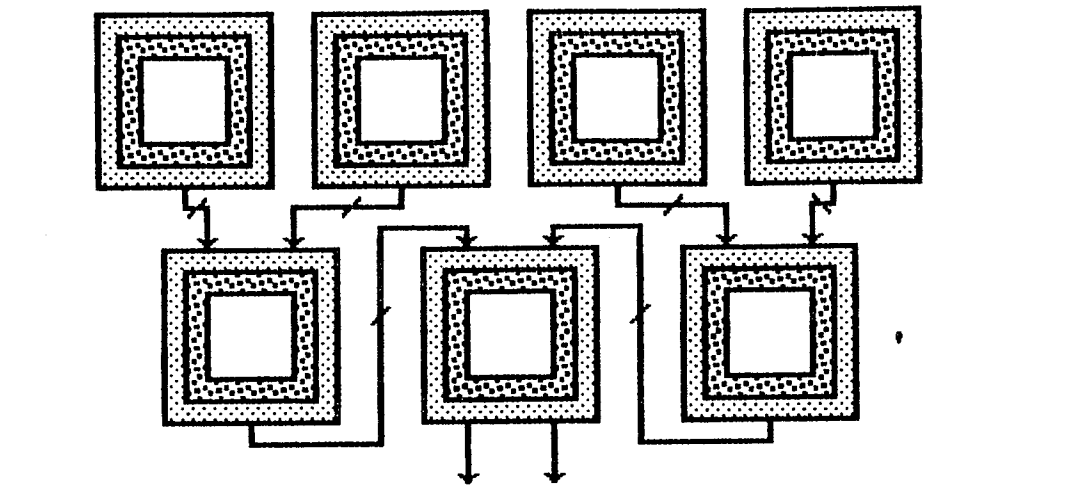


Figure 24 : Courts-circuits incluant des lignes d'interconnexion

Il faut encore examiner les possibilités de courts-circuits entre deux lignes internes à des cellules différentes, puisque ces pannes peuvent provoquer des défauts non détectables avec le code d'entrée (mais pour lesquels le contrôleur n'est pas redondant), ou des boucles de retour. Ces types de fautes ne sont pas acceptables dans des circuits SCD. On peut les éviter en modifiant la topologie de la cellule, comme suggéré par la prochaine règle.

**Règle R2**
Les régions limitrophes de chaque cellule implémentées en diffusion doivent correspondre à des lignes d'alimentation, $V_{SS}$ ou $V_{DD}$. Les lignes de métallisation placées près des limites de la cellule doivent être isolées par l'insertion d'une ligne d'alimentation en métal entre la ligne de signal et la limite de la cellule, si l'on considère qu'un court-circuit peut survenir avec une métallisation dans une autre cellule qui n'est pas une ligne d'alimentation (voir figure 25).




EXEMPLE D'ISOLATION D'UNE CELLULE

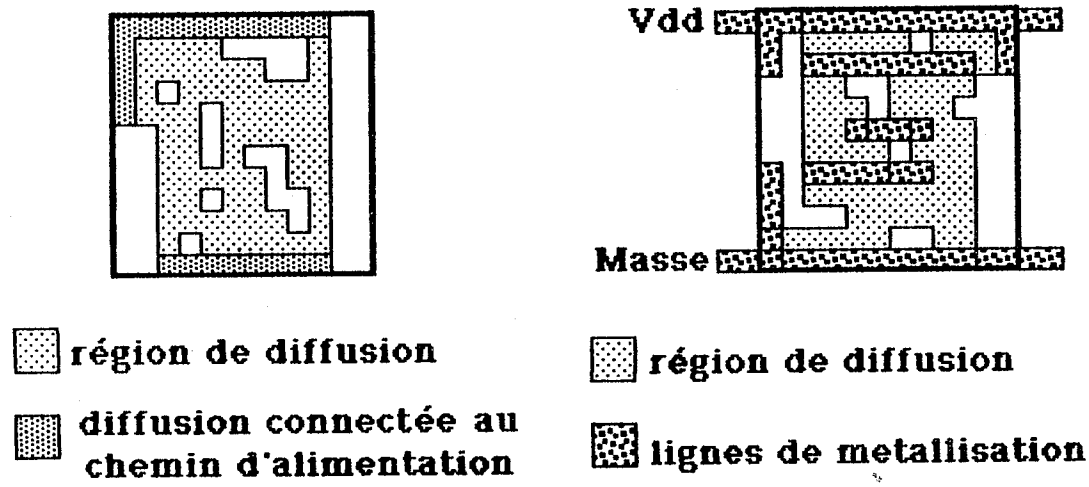



Figure 25 : Contraintes d'isolation selon la règle R2

**Proposition P3**
Tout assemblage de cellules SCD dont les dessins respectent la règle R2 et l'interconnexion est en accord avec la règle R1, résulte en un contrôleur SCD pour la Classe I d'hypothèses de pannes, étant assurée l'application de l'ensemble des vecteurs du code à chaque cellule.

A travers l'utilisation des règles R1 et R2, nous éliminons les types de courts-circuits inacceptables décrits par les items F3, F4 et F5 (sous-section III.2.1.1.), la Classe I étant considérée, car pour le cas des courts-circuits, leur détectabilité sera dépendante des vecteurs d'entrée reçus par chaque cellule et non des vecteurs d'entrée reçus par le contrôleur complet. Cela

est dû à l'hypothèse disant que le résultat d'un court-circuit avec une ligne d'alimentation est un collage (cette région garde en permanence la valeur de la ligne d'alimentation). Puisque F1 et F2 sont des défauts couverts grâce à l'utilisation des cellules SCD, le contrôleur complet est SCD.

Dans certains cas, la règle R2 peut se révèler trop stricte: quoiqu'il soit toujours possible de la respecter, elle peut susciter une augmentation importante aux dimensions de la cellule. Pourtant, cette règle peut être particularisée en fonction des différents codes et structures d'organisation interne des contrôleurs, selon le prochain item.

**Règle R3**

Il suffit qu'un des côtés limitrophes de cellules voisines, étant les deux implémentés en diffusion, soit constitué par une ligne d'alimentation, $V_{DD}$ ou $V_{SS}$. Les lignes de métallisation localisées près des limites de la cellule doivent être isolées par l'insertion d'une ligne d'alimentation en métal dans la région limitrophe d'une des cellules, si l'on considère qu'un court-circuit pourrait se produire entre celle-là et une ligne en métal (laquelle n'est pas un chemin d'alimentation) d'une autre cellule.

Cette règle signifie que: pour deux cellules assemblées face à face, seulement une des deux devra respecter la règle R2. Cette situation est illustrée par la figure 26.

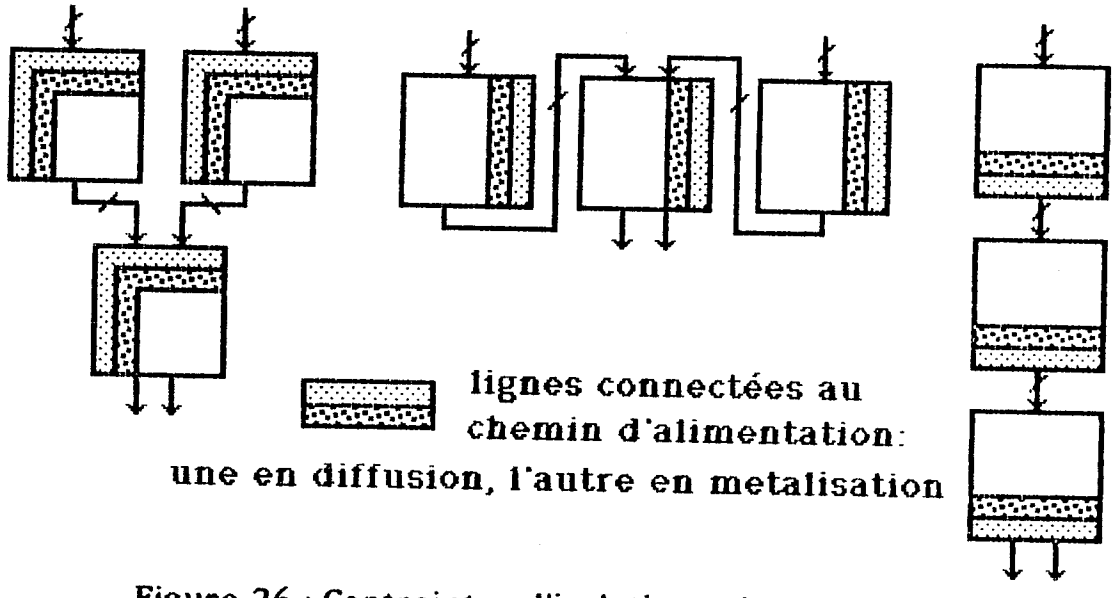



Figure 26 : Contraintes d'isolation selon la règle R3

**Proposition P4**

Quelque arrangement de cellules SCD rassemblées où les règles R3 et R1

sont respectées, si l'on assure que chaque cellule reçoit son ensemble de vecteurs du code, résulte en un circuit SCD pour la Classe I des hypothèses de pannes.

Comme dans le cas de la proposition P3, les courts circuits inacceptables inclus par les items F3, F4 et F5 (sous-section III.2.1.1.) sont éliminés par la propositions P4, la Classe I étant considérée. Cependant, nous remarquons que les cellules conçues selon la règle R3 (et qui ne respectent pas la règle R2) sont implicitement associées à contraintes d'arrangement - une fois dessinées, elles ne peuvent pas être rassemblées de n'importe quelle façon. Les conséquences des opérations exécutées avec les cellules pendant l'assemblage - rotations, translations, symétries, etc... - devront être analysées auparavant, vis-à-vis de la propriété SCD du circuit.

Nous avons expliqué, au début de cette sous-section, que les règles énoncées ici sont suffisantes mais pas nécessaires. D'autres propositions peuvent être données avec pour base d'autres règles moins strictes, lesquelles se rapportent à des cas spécifiques de contrôleurs. Elles sont présentées par la suite.

### III.2.3.3. Règles spécifiques

La prochaine proposition se réfère aux contrôleurs double-rail en cascade.

**Règle R4**

Les régions limitrophes de chaque cellule implémentées en diffusion doivent correspondre à des lignes d'alimentation ($V_{SS}$ ou $V_{DD}$) ou à des lignes d'entrée/sortie (signaux d'entrée ou de sortie de la cellule), sous la condition que les entrées et les sorties d'une même cellule ne se rencontrent pas du même côté. Les lignes de métallisation placées près des limites de la cellule, exceptées celles qui correspondent à des signaux d'entrée et de sortie des cellules, devront être isolées du bord par l'insertion d'une ligne d'alimentation en métal entre elles et la limite de la cellule, il peut survenir un court-circuit avec une métallisation d'une autre cellule qui n'est pas utilisée comme un chemin pour l'alimentation.

Le placement des signaux d'entrée et de sortie sur des côtés opposés d'une même cellule, aide aussi à l'arrangement en cascade, ce qui peut être vue comme un avantage additionnelle.

**Proposition P5**
Un contrôleur double-rail composé de cellules qui respectent la règle R4, assemblées en une cascade avec des niveaux linéairement progressifs (voir figure 27a.) et qui reçoit tout l'ensemble des vecteurs d'entrée, est un contrôleur SCD.

Un exemple d'une cascade où les niveaux n'augmentent pas linéairement est donné à la figure 27b. Il y a un mélange de cellules de différents niveaux, les unes à côté des autres.

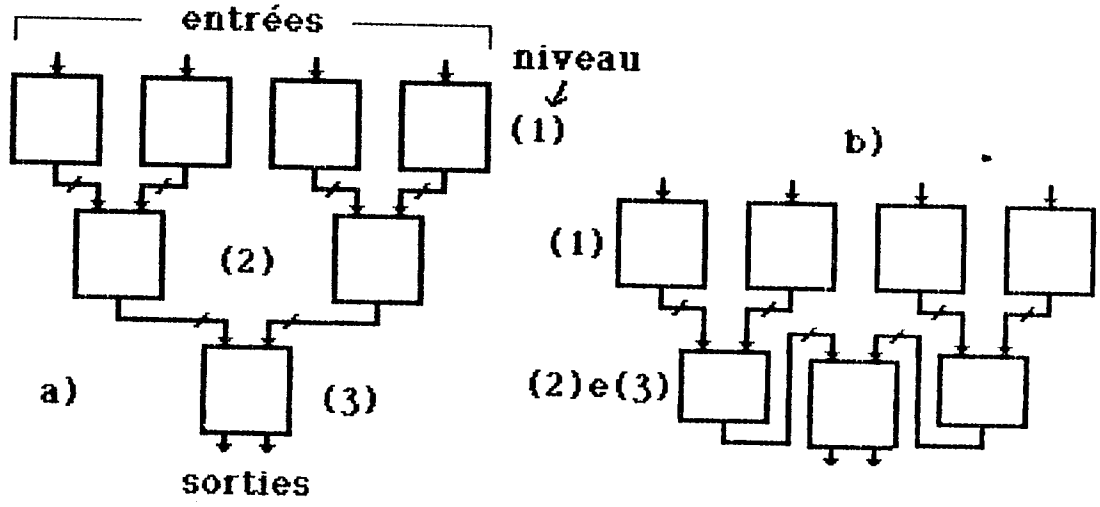



Figure 27: Organisation en cascade. a) cascade avec des niveaux linéairement progressifs; b) cascade avec des niveaux non linéairement progressifs.

A partir de la proposition P5, aucune des possibilités de défaut listées dans la sous-section III.2.1.1. - de F1 à F5 - est totalement éliminée. Mais, en raison de l'arrangement considéré et de l'hypothèse que nous avons , un contrôleur double-rail qui reçoit un ensemble complet de vecteurs du code, on peut assurer la détection des défauts survenants et pour lesquels le circuit n'est pas redondant.

Nous remarquons que les interconnexions des cellules ne sont pas obligatoirement en polysilicium, selon la dernière proposition, P5. Elles peuvent être implémentées aussi en diffusion ou aluminium. En revanche, afin d'assurer la détection de toute faute possible, l'ensemble des vecteurs de test appliqué aux entrées du contrôleur devra inclure les valeurs qui détectent ces courts-circuits parmi les interconnexions. Le code d'entrée complet est suffisant mais pas nécessaire, dans ce cas. Dans la figure 28

sont illustrés des cas où les cellules reçoivent chacune leur ensemble complet de vecteurs d'entrée, pourtant ce n'est pas le cas du contrôleur. Nous pouvons facilement vérifier qu'il y a des courts-circuits parmi les lignes d'interconnexion qui resteraient non-détectéses malgré le fait que le contrôleur ne soit pas redondant pour eux, et que chaque cellule reçoive le code d'entrée complet. Celà nous amène à conclure que les réductions dans l'ensemble des vecteurs d'entrée nécessaires au test du contrôleur afin d'obtenir un nombre plus petit de vecteurs d'entrée doit prendre en compte chaque structure particulière d'assemblage.

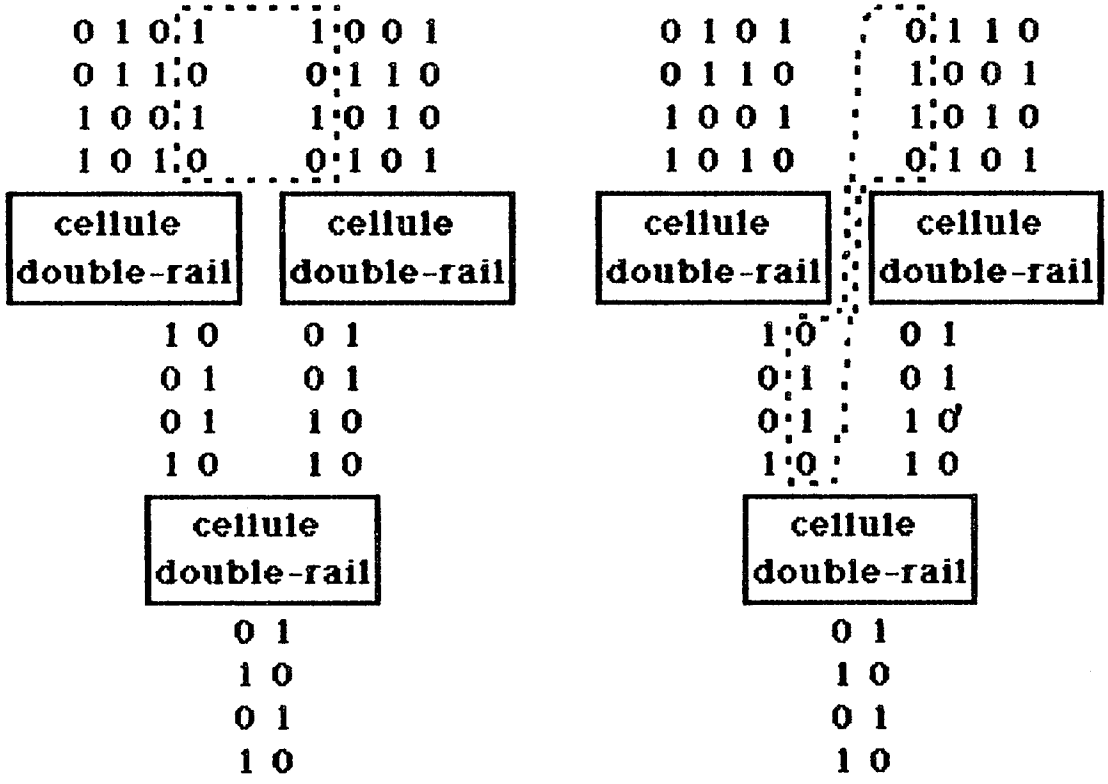



Figure 28 : Ensemble de vecteurs du code d'entrée réduit, incapable de détecter des fautes inacceptables

La proposition P5 peut aussi être appliquée pour les contrôleurs de parité pair. Dans le cas de contrôleurs de parité impair, il faut modifier le circuit logique - ceci est examiné dans la sous-section concernant les contrôleurs de parité (III.3.2.).

## III.3. APPLICATIONS : ETUDE DE CAS

Dans l'étude qui suit nous proposons la conception des contrôleurs SCD

conçus à partir de cellules SCD rassemblées convenablement car ces circuits SCD composent la plus large classe des contrôleurs disponible [NIC 83].

Les exemples sont présentés en technologie NMOS et la Classe I d'hypothèses de pannes est toujours considérée.

## III.3.1. Contrôleur double-rail

### III.3.1.1. Cellule de base

Dans la figure 29, nous proposons le schéma d'une cellule de contrôleur double-rail pour k = 2, c'est-à-dire, deux bits d'information (deux paires complémentaires). C'est la version avec 8 transistors, dérivée de la proposition analysée dans la sous-section III.2.2..

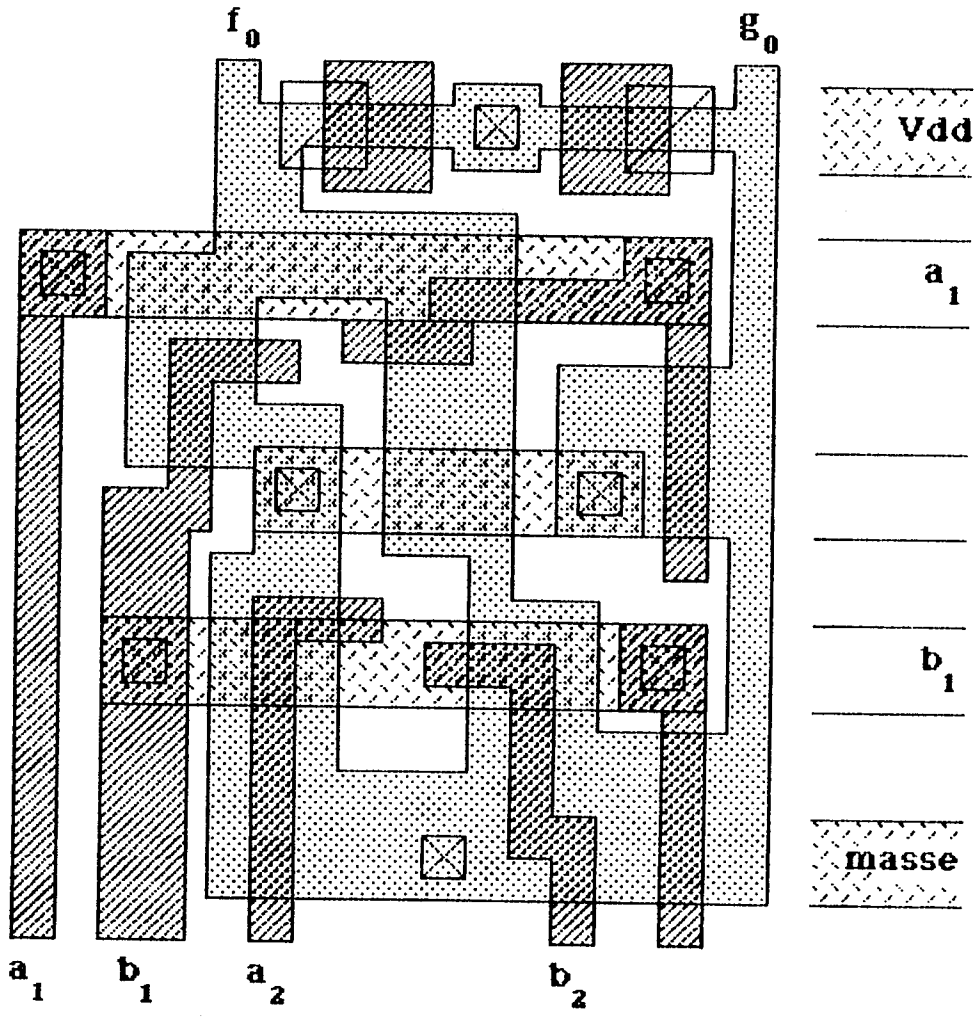



Figure 29: Proposition d'une cellule de contrôleur double-rail (SCD)

Prenant en compte toutes les fautes possibles dans la Classe I, ce circuit :

- pendant le fonctionnement normal, est "code disjoint" parce qu'il produit toujours des sorties hors code pour les entrées hors code qu'il reçoit;

- en présence de chaque séquence de fautes appartenant à la Classe I, soit il est "strongly redundant", soit la séquence est détectée avec un des vecteurs du code d'entrée qui produit une sortie hors code.

Donc, ce circuit est "strongly code disjoint" (SCD) pour la Classe I d'hypothèses de pannes et pour l'ensemble des vecteurs d'entrée {(01,01), (01,10), (10, 01), (10,10)}.

Cette proposition topologique correspond au circuit électrique montré à la figure 30 et ne peut pas être représenté par des portes logiques conventionnelles. Le circuit électrique est repris de [NIC 84c].

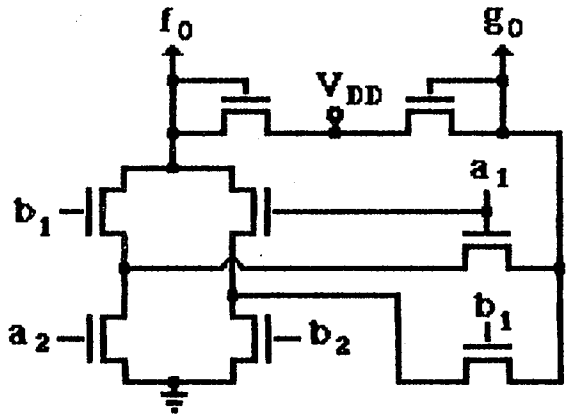



Figure 30 : Circuit électrique pour la cellule proposée

III.3.1.2. Contrôleur double-rail

La figure 31 présente une proposition d'assemblage pour un contrôleur double-rail de trois cellules, à partir de la cellule de base analysée dans la sous-section précédente. Cela fait, donc, un contrôleur pour $k = 4$.

Cette implémentation résultante a été étudié vis-à-vis de toutes les fautes possibles dans la Classe I, et il y a, au moins, un court-circuit non

détectable qui peut survenir: il est signalé $s_2$- $f_1$ dans la figure. Quoique son occurrence soit peu probable car il y a deux lignes de polysilicium entre ces deux points en diffusion, il faut la considérer. Et le problème peut être facilement corrigé par un petit changement dans la topologie de la cellule, lequel consiste en une solution intermédiaire entre les règles R3 et R4.

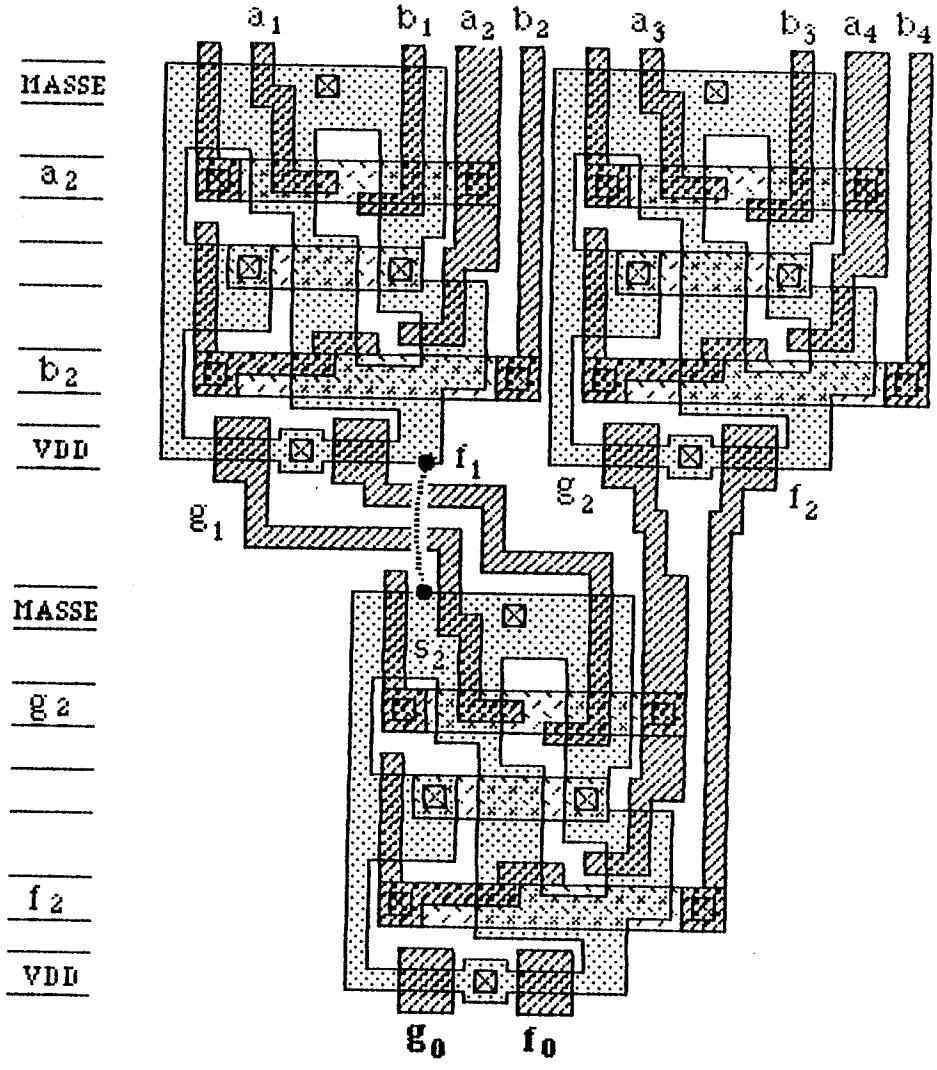



Figure 31 : Contrôleur double-rail (k = 4) assemblé à partir de 3 cellules

La nouvelle cellule conçue selon une combinaison des règles R3 et R4, dont les restrictions d'interconnexion demandent l'application de la règle R1, est montrée à la figure 32.

La possibilité du court-circuit non détectable et pour lequel le circuit n'était pas redondant, entre $s_2$ et $f_1$ est éliminée et d'autres problèmes ne sont pas introduits. Avec la même modification, les courts-circuits possibles

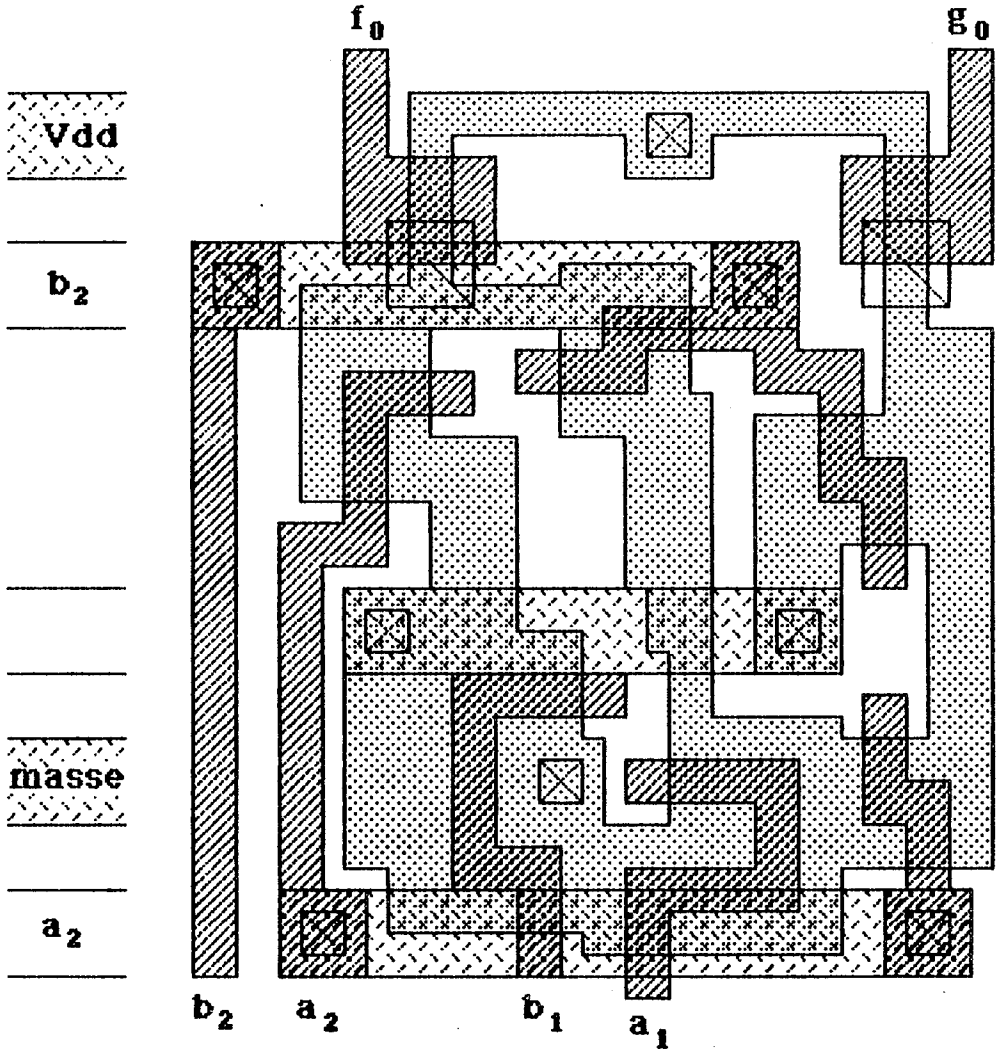



Figure 32 : Cellule SCD conçue par une solution intermédiaire entre les règles R3 et R4

auparavant - $g_1$. $g_0$ et $f_1$. $g_0$ sont aussi écartés. Bien qu'ils peuvent être détectés avec des vecteurs appartenant au code d'entrée, avant leur détection ils peuvent causer des valeurs indéterminées aux sorties ou des oscillations (puisqu'ils résultent en boucles de retour). Toutes les autres fautes possibles (dans la Classe I) peuvent être détectées, ou le circuit est redondant pour elles. Par conséquent, le circuit proposé est SCD vis-à-vis de la Classe I d'hypothèses de pannes et pour l'ensemble des vecteurs du code d'entrée. La règle R1 étant assurée, ces cellules sont rassemblées selon l'arrangement indiqué par la figure 33, pour k = 4. Le contrôleur résultant est SCD.

L'utilisation des règles proposées associée à un assemblage convenable comme moyen de conception, élimine les situations possibles où nous

pouvions avoir des défauts non détectables pour lesquels le circuit n'était pas redondant.

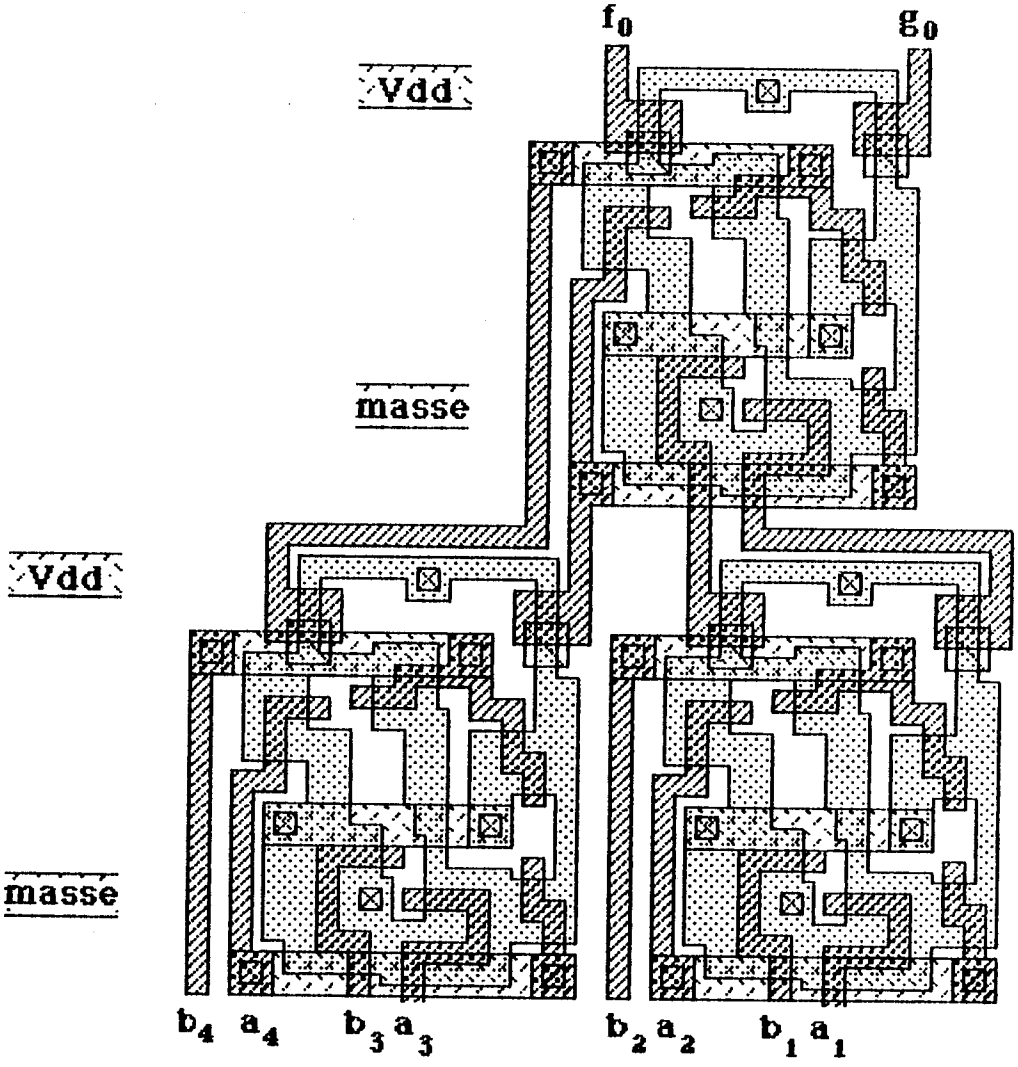



Figure 33 : Contrôleur double-rail SCD pour k = 4 (3 cellules)

La figure 34 représente un contrôleur double-rail de 7 cellules pour 16 bits codés - donc, 8 paires complémentaires. Ces cellules sont réunies à travers une structure en arbre. Le circuit résultant a été réalisé dans le CMP 84 - Circuit Multi-Projet 84 français et du point du vue logique, son fonctionnement correspond aux specifications intiales. Des défauts physiques n'ont pas été simulés.

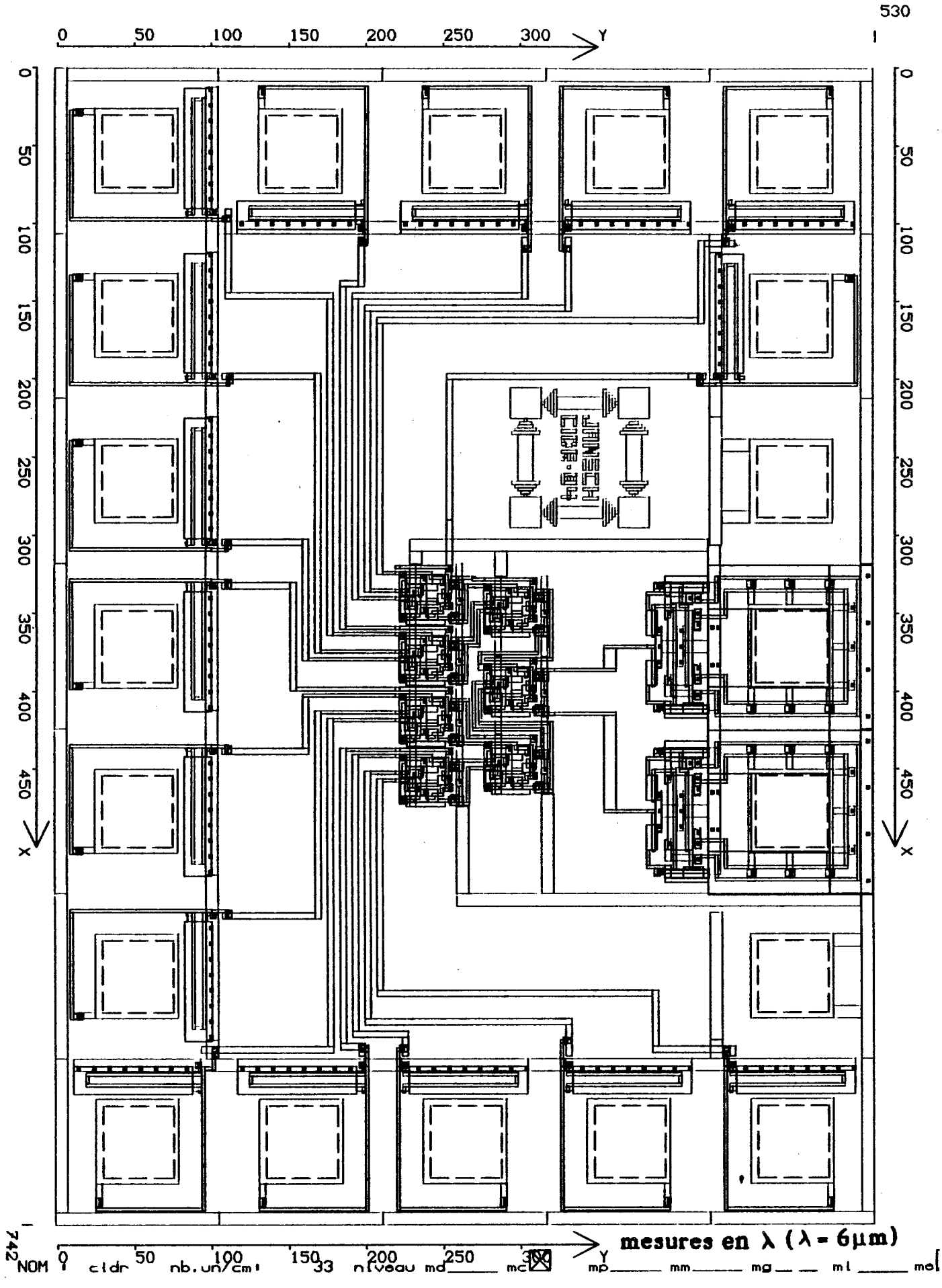



**Figure 34 : Contrôleur double-rail pour 8 paires complémentaires d'entrée**

### III.3.1.3. Hypothèses de pannes pour un contrôleur multi-cellulaire (double-rail)

Un système composé d'un circuit SFS et d'un contrôleur SCD - et en particulier un contrôleur double-rail multi-cellulaire conçu selon la définition SCD (D23) - atteint le "TSC goal", étant considéré l'hypothèse H2. L'assemblage de cellules SCD résulte en un contrôleur qui aura le même comportement que d'autres conçus à partir d'une solution récursive équivalente.

Pour le cas spécifique du contrôleur double-rail, on peut admettre encore d'autres pannes additionnelles (par rapport à l'hypothèse H2) dans des situations bien définies. Nous rappelons que selon la première partie de H2, après l'occurrence d'un défaut affectant le contrôleur, il s'écoule un laps de temps suffisant pour que tout le code d'entrée B soit appliqué et d'autres pannes ne surviennent pas ni dans le bloc fonctionnel, ni dans le contrôleur.

Dans les structures cellulaires rassemblées en arbre, on peut avoir un défaut additionnel (ou même plus) survenant avant la détection du premier, dès qu'ils se localisent soit dans des cellules de même niveau, soit dans des cellules de branches différentes de l'arbre; par conséquent, les sorties d'une cellule affectée ne seront pas entrées d'une autre cellule affectée. L'occurrence de fautes diffusées parmi plusieures cellules sans restriction de localisation est admise dans les circuits double-rail dont les cellules "strongly code disjoint" sont conçues selon la définition forte (D26).

Des exemples pour les situations décrites ci-dessus sont présentés par la figure 35.

La deuxième partie de l'hypothèse H2 fait référence aux fautes affectant le bloc fonctionnel : après l'occurrence de l'une d'elles, il s'écoule un laps de temps suffisant pour l'application de toutes les entrées du code et d'autres fautes ne surviennent pas, ni dans ce bloc, ni dans le contrôleur.

Dans n'importe quel arrangement des cellules double-rail, plusieurs fautes peuvent être acceptées aux entrées, dès qu'elles ne se localisent pas de forme multiple sur une seule paire complémentaire d'entrée.

Ces situations sont illustrées par la figure 36.

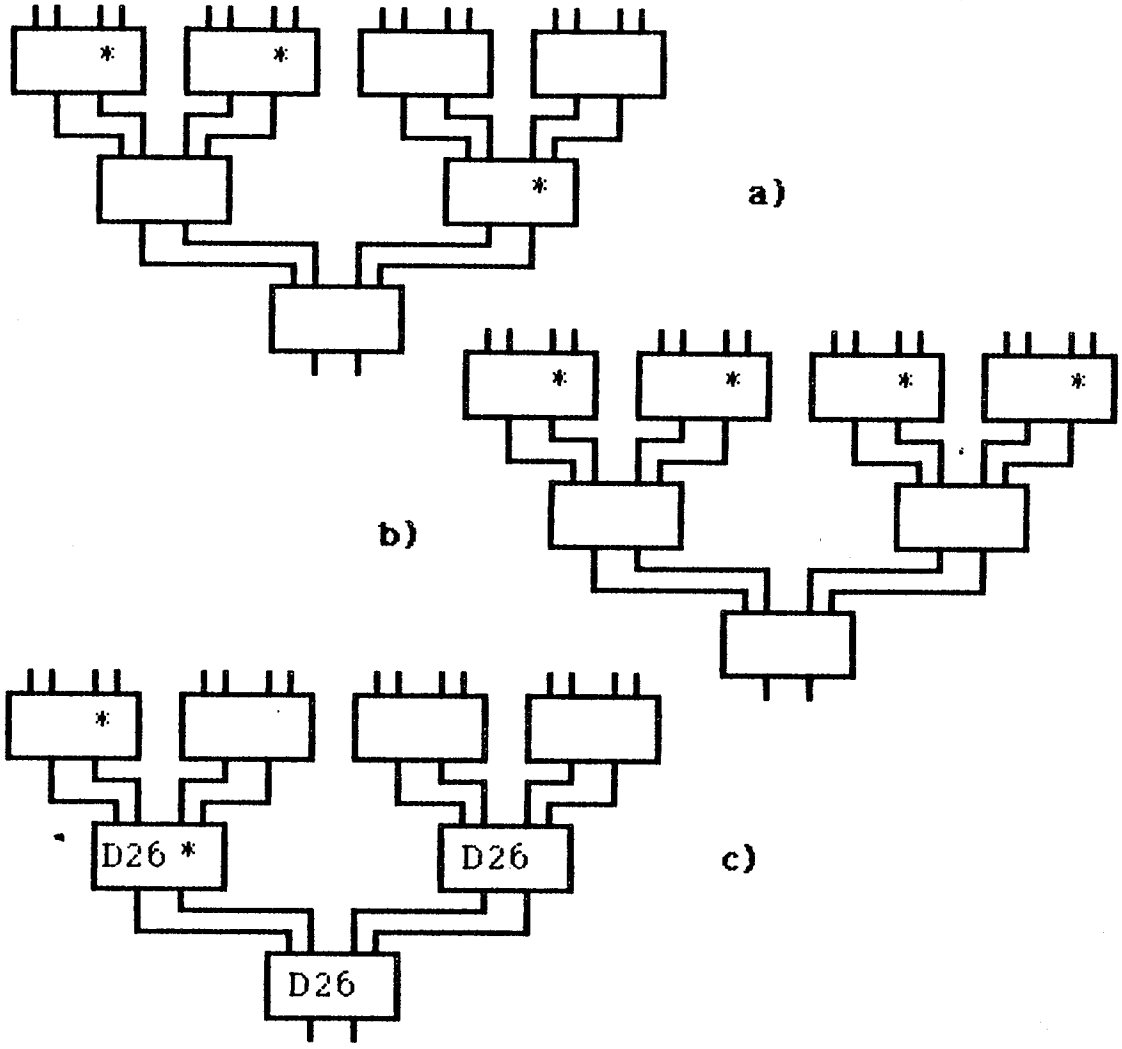



Figure 35 : Plusieurs cellules affectées par défauts. a) cellules de différentes branches ou du même niveau; b) plusieurs cellules du même niveau; c) cellules de niveaux différents dans la même branche - excepté le premier niveau les autres doivent respecter la définition D26 des contrôleurs SCD

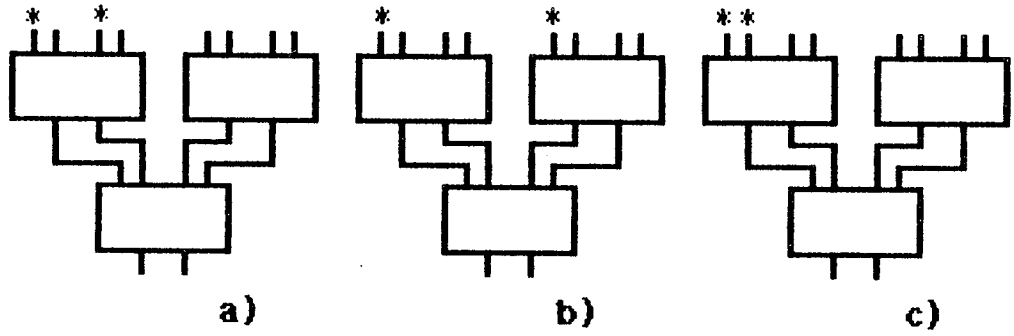



Figure 36: Plusieurs entrées erronées. a) cas admis: paire d'entrées différentes de la même cellule étant affectées; b) cas admis: entrées de cellules différentes étant affectées; c) cas non admis: une seule paire d'entrée étant affectée par des fautes multiples - les défauts peuvent être masqués

## III.3.2. Contrôleur de parité

### III.3.2.1. Cellule de base

Les considérations et hypothèses utilisées dans cette sous-section pour la conception sont les mêmes déjà référées dans le cas du contrôleur double-rail.

La figure 37 présente une proposition pour une cellule ou-exclusive. Les entrées A et B sont perpendiculaires aux lignes d'alimentation afin de permettre plus facilement leur composition en cascade. De plus, ses limites latérales sont arrangées d'une telle façon qu'un court-circuit entre des cellules voisines irait affecter au moins une ligne d'alimentation ou deux entrées. Donc, concernant les diffusions, la règle R3 est respectée. Les court-circuits possibles entre des lignes d'aluminium (entrées d'une même cellule ou de cellules différentes) ne posent pas de problèmes dans l'arbre ou-exclusive car ils sont facilement détectables par l'application du code d'entrée.

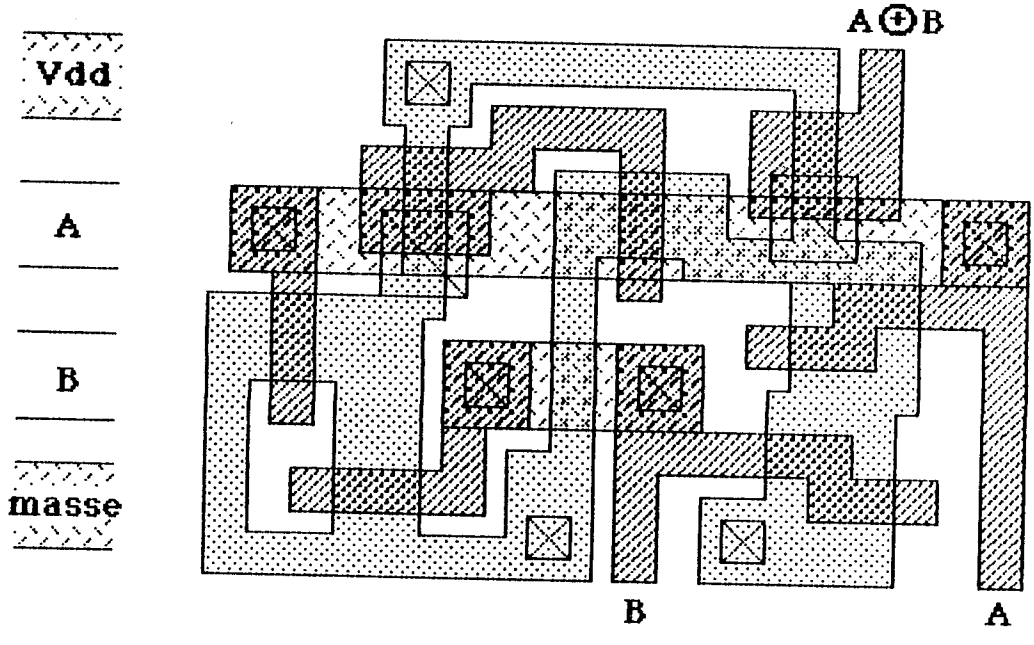



Figure 37 : Proposition pour une cellule ou-exclusive

Le circuit électrique correspondant à la cellule ou-exclusive suggérée est illustré dans la prochaine figure (38).

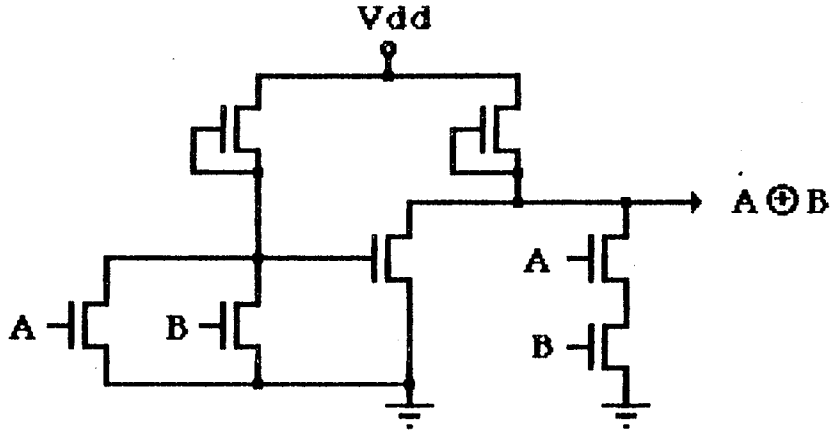



Figure 38 : Circuit électrique pour une cellule ou-exclusive

III.3.2.2. Contrôleur de parité

Ce contrôleur reçoit, comme des entrées, l'information codée en parité, laquelle est divisée en deux groupes. Chacun de ces groupes est analysé par un arbre ou-exclusif. Un segment d'un arbre ou-exclusif est illustré par la figure 39, afin d'exemplifier l'assemblage des cellules. Pour un code de parité pair il faut inverser la sortie d'un des arbres pour qu'elle compose un code 1-parmi-2 avec l'autre sortie. Ensemble, elles forment le code de sortie du contrôleur.

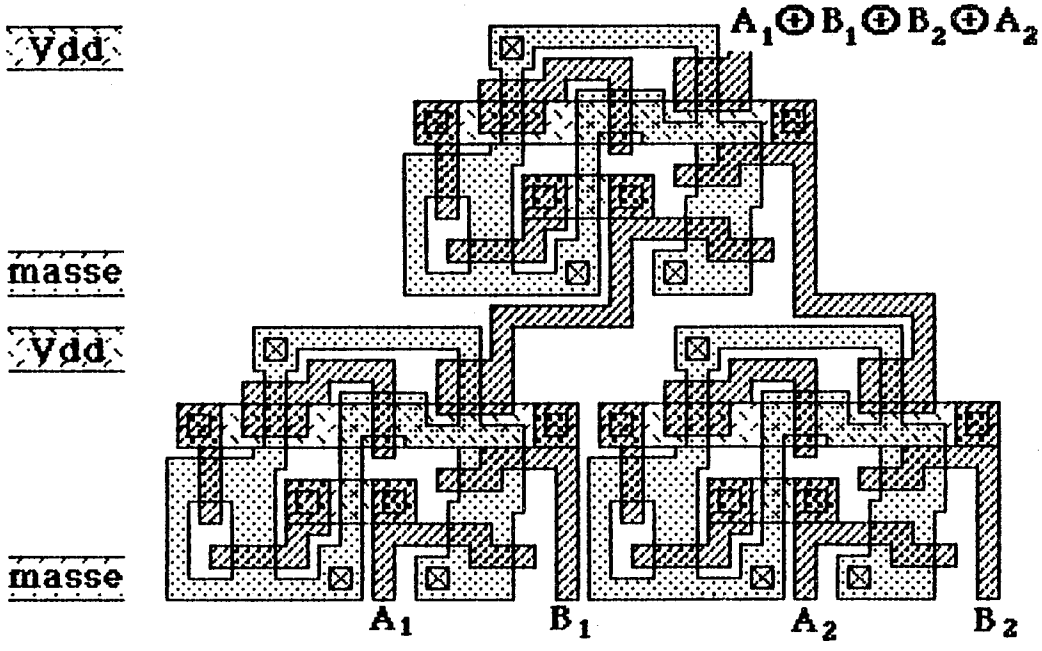



Figure 39: Contrôleur de parité SCD (4 bits)

Si le contrôleur reçoit les entrées de deux vecteurs codées en parité, on peut réunir les deux paires de sorties 1-parmi-2 en une cellule de contrôleur double-rail, afin d'avoir une seule paire de lignes de sortie. Cette situation est présentée par la figure 40 où sont utilisées: les cellules ou-exclusives proposées dans la sous-section III.3.2.1., la cellule double-rail comme proposée à la sous-section III.3.1.2. et deux inverseurs additionnels. L'information reçue aux entrées est consitutée par des vecteurs de 8 bits (chacun) codés en parité. Ses entrées sont nommées $e_1,e_2,.....,e_8$ et $e_1'$, $e_2'$,.....,$e_8'$, et le circuit produit un code 1-parmi-2 aux sorties (s et $\bar{s}$).

Ce circuit a été réalisé dans le CMP 84 et son fonctionnement du point du vue logique correspond aux specifications initiales. Des défauts physiques n'ont pas été simulés.

Une faute pouvant être à l'origine de problèmes dans ce contrôleur à parité est le court-circuit entre les sorties des deux arbres ou - exclusifs utilisés pour l'analyse du même vecteur d'entrée, signalée dans la figure 41. Quand il n'y a pas de fautes aux entrées du contrôleur (i.e., dans les lignes d'information en provenance du bloc fonctionnel), les valeurs des points signalés seront égales - donc, le défaut résultant n'est pas détectable par l'application de l'ensemble des vecteurs du code d'entrée. Dans le cas particulier du circuit présenté à la figure 40, nous avons utilisé des lignes en polysilicium. Ainsi, la possibilité d'occurrence de ce court-circuit est éliminée dans la classe d'hypothèses de pannes considérée.

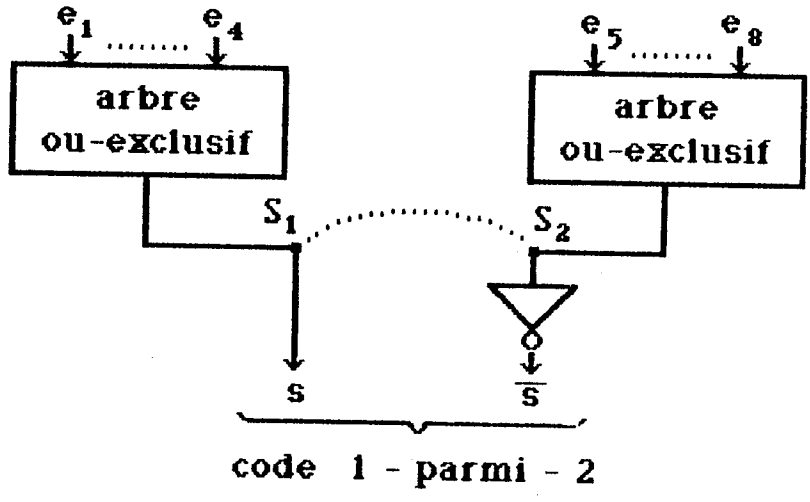



Figure 41 : Court-circuit non détectable ($s_1$ - $s_2$)

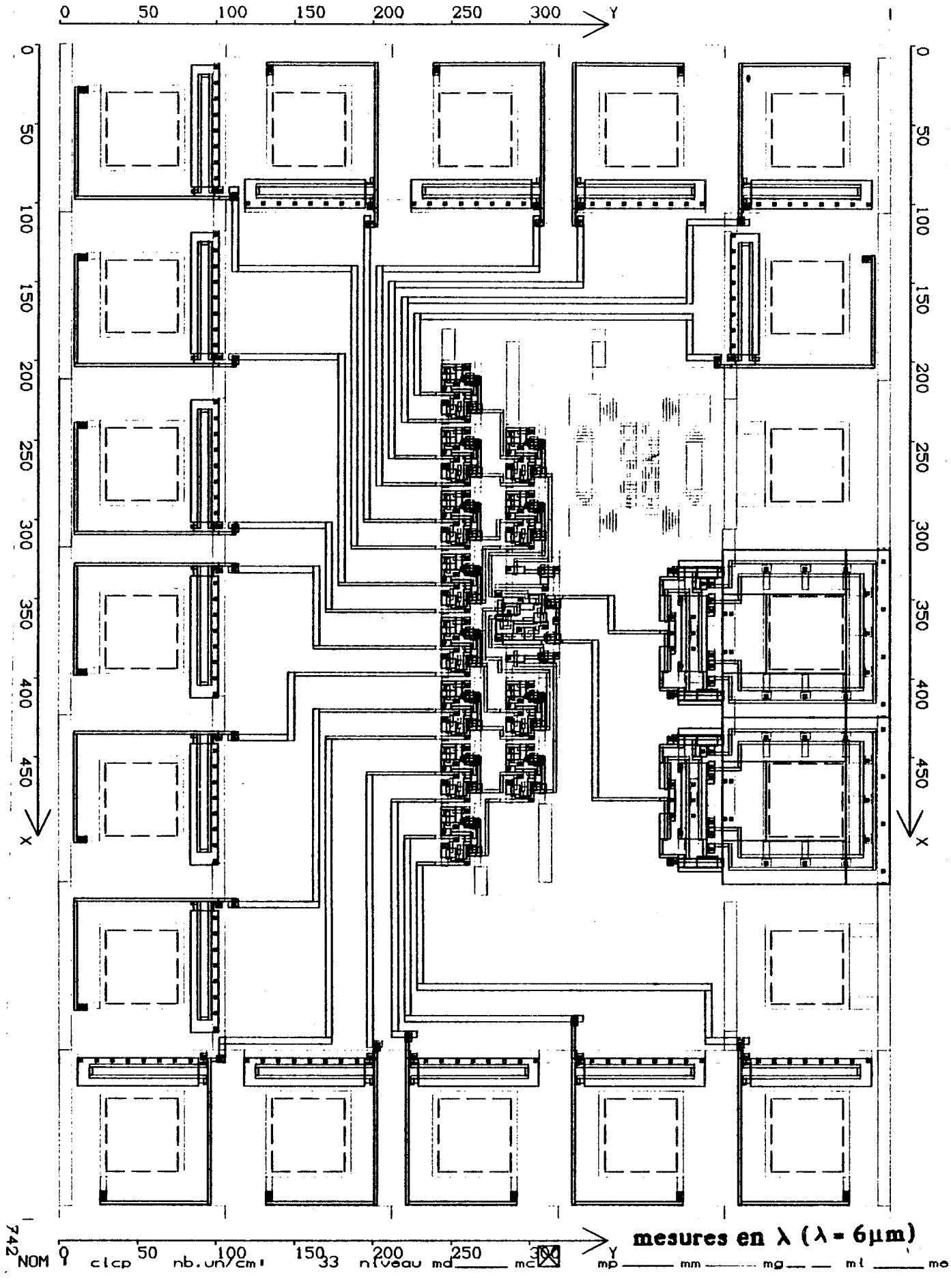



**Figure 40: Contrôleur de parité pour 16 entrées**

Il est aussi possible d'éviter l'occurrence de cette panne à travers un petit changement du circuit de base. Nous avons:

$$s = e_1 \oplus e_2 \oplus e_3 \oplus e_4$$

et

$$\overline{s} = (\overline{e_5 \oplus e_6 \oplus e_7 \oplus e_8})$$

mais

$$\overline{s} = (\overline{e_5 \oplus e_6 \oplus e_7 \oplus e_8}) = e_5 \oplus e_6 \oplus e_7 \oplus \overline{e_8}\,.$$

selon les propriétés de la fonction ou-exclusive. Ces considérations résultent en un circuit dont la structure de blocs correspondante est illustrée par la figure 42, laquelle élimine la possibilité de défaut que nous venons de décrire.

### III.3.2.3. Hypothèses de pannes pour le contrôleur multi-cellulaire (parité)

Le contrôleur de parité résultant d'un assemblage adéquat de cellules ou-exclusives et double-rail SCD, est aussi SCD dans l'ensemble. Donc, utilisé comme le compagnon d'un circuit SFS, ils constituent un système qu'atteint le "TSC goal", étant considérée l'hypothèse H2.

Les codes de parité assurent la détection d'une erreur simple - un nombre pair d'erreurs s'annulent mutuellement. Ainsi, dans un contrôleur de parité, une seule erreur peut survenir: dans le contrôleur ou à l'information codée qui se présente aux entrées.

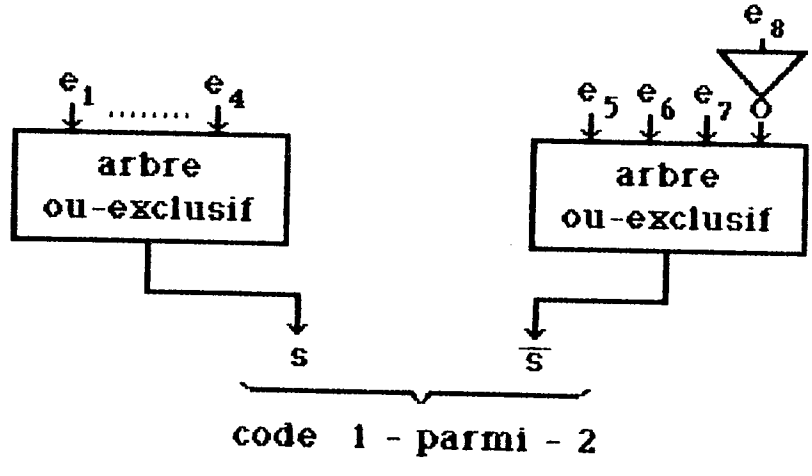



Figure 42 : Contrôleur de parité pair où le défaut non détectable a été éliminé

Pourtant, dans le cas spécifique du contrôleur montré à la figure 40, où l'on utilisait une cellule double-rail pour l'analyse des sorties des arbres ou-exclusifs (lesquels reçoivent des vecteurs indépendants), deux erreurs ou deux pannes sont admises dès qu'elles ne se produisent pas sur le même vecteur d'entrée ni dans un des arbres qui vérifie un seul vecteur d'entrée, respectivement.

L'hypothèse d'occurrence de deux défauts que l'on vient de considérer est peut être assez particulière; ce que l'on veut clarifier est que, malgré le fait qu'un seul contrôleur soit utilisé pour deux vecteurs codés, on admet une faute en chaque vecteur. Il ne s'agit pas vraiment d'une modification de l'hypothèse H2, mais de nuances concernant les considérations des limites de chaque bloc fonctionnel et contrôleur.

Les diverses possibilités de défauts pour ce cas sont montrées dans la figure 43. Des fautes survenant concomitantement avec une autre dans la cellule double-rail ne sont détectées si cette cellule est conçue selon la définition SCD forte (D26).

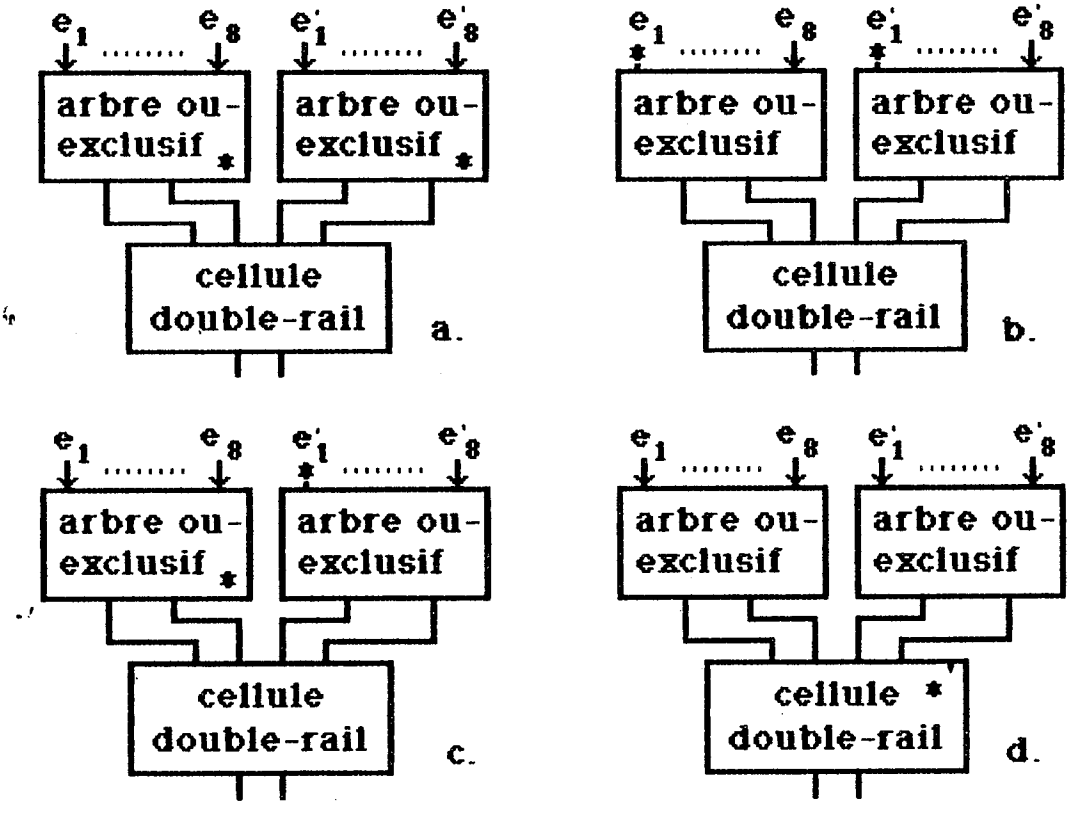



Figure 43 : Situations de fautes multiples admises. a) une faute dans chaque arbre ou-exclusif; b) une faute dans chaque vecteur d'entrée; c) une faute dans un des arbres ou- exclusifs et une autre dans le vecteur d'entrée non associé au même arbre; d) une faute dans la cellule double-rail

## III.4. STRUCTURES MULTICONTROLEURS

### III.4.1. Considérations générales

Dans quelques applications, nous pouvons envisager l'utilisation d'une structure multicontrôleur, où nous avons plusieurs contrôleurs dont les sorties sont vérifiées par un contrôleur général, comme illustrée par la figure 44.

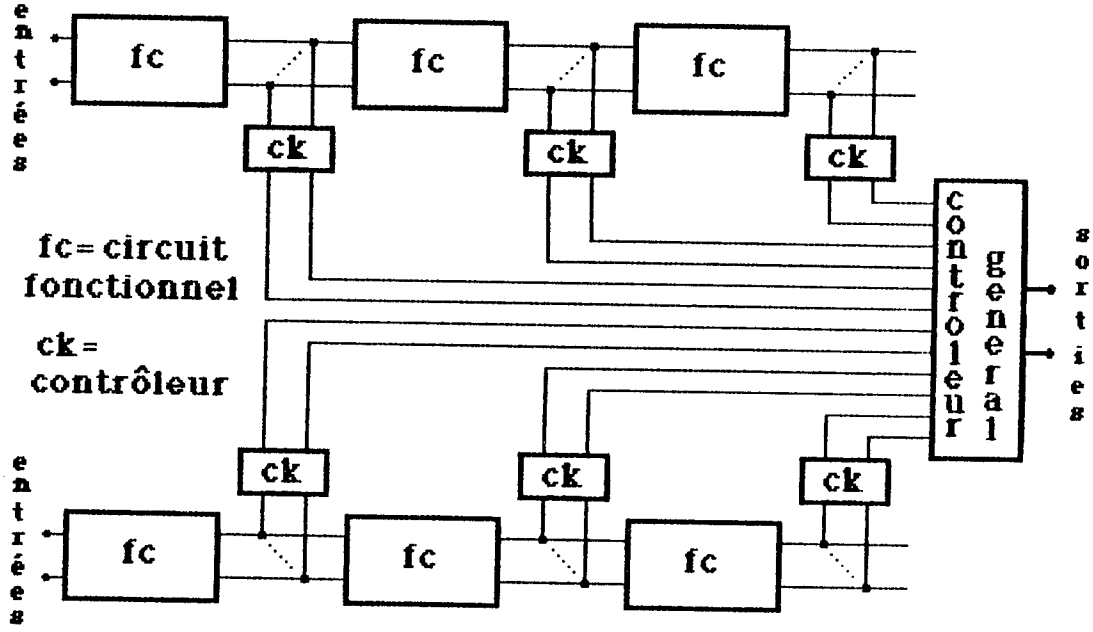



Figure 44 : Une structure multicontrôleur

Dans cette structure, chacun des contrôleurs (construit à partir de cellules ou pas) reçoit les sorties d'un circuit fonctionnel, ces sorties étant utilisées par d'autres blocs en cascade ou non, les analyse et a ses sorties à lui vérifiées par un contrôleur général. Il n'y a que les sorties de ce contrôleur général du point-de-vue extérieur au système; les sorties appartenant à chacun des contrôleurs internes sont analysées en particulier seulement un cas d'erreur détectée aux sorties générales, afin de localiser le bloc affecté.

La contrôlabilité et l'observabilité de chacun des blocs doivent être examinées pendant la conception du système. L'ensemble des vecteurs de test nécessaire à chaque circuit fonctionnel devra être fourni par son prédécesseur, afin d'assurer la détection des fautes.

Mais concernant les hypothèses de pannes, il en faut d'autres pour la structure multicontrôleur car il s'agit en fait d'une situation différente de l'approche cellulaire, comme nous l'expliquerons par la suite.

Le problème principal associé à une telle structure est que, accepté la cascade des blocs fonctionnels, une faute survenant dans un d'entre eux peut affecter les sorties des suivants, et le résultat peut se diffuser parmi plusieurs contrôleurs. Cette situation peut être résolue par l'utilisation d'un contrôleur classique qui ait des sorties complémentaires (pour ceux qui analysent directement les sorties des blocs fonctionnels), ces sorties étant vérifiées par un contrôleur général double-rail. Le système proposé est celui de la figure 45.

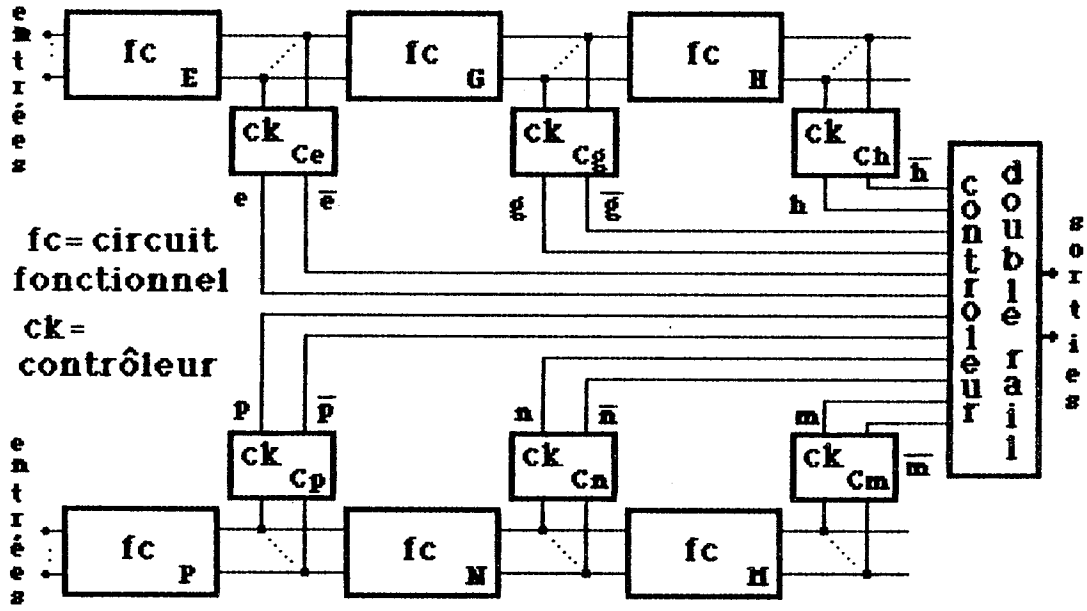



Figure 45 : Proposition d'organisation pour un multicontrôleur

Par la suite, nous étudierons les hypothèses de défauts possibles, prenant en compte les considérations initiales ci-dessus mentionnées.

### III.4.2. Hypothèses d'occurrence de pannes

Pour la formulation des hypothèses, nous pouvons considérer une des deux configurations suivantes: soit le système est divisé en deux parts : une fonctionnel et l'autre un multicontrôleur, soit le système est vu comme un ensemble de sous-systèmes (chacun composé d'un bloc fonctionnel et de

son contrôleur associé) vérifiés par un contrôleur général. Les deux configurations amènent aux mêmes conclusions.

Nous allons, tout d'abord, définir les expressions et termes utilisés par la suite.

On considère l'ensemble de blocs fonctionnels et de contrôleurs comme un système. Celui-ci est composé de plusieurs sous-systèmes (bloc fonctionnel + contrôleur associé) et d'un contrôleur général. Chaque sous-système est une paire 'bloc fonctionnel-contrôleur", le contrôleur analysant les sorties du bloc fonctionnel. Quand on se réfère à la partie fonctionnelle, c'est pour indiquer l'ensemble des blocs fonctionnels ; les contrôleurs ou, plus précisément, les contrôleurs associés sont ceux qui vérifient les sorties de chacun des blocs fonctionnels. La figure 46 montre les schémas correspondants à ces divisions.

On appelle A l'ensemble des entrées du code du système considéré; donc, a c A sont les vecteurs du code reçus par l'ensemble des blocs fonctionnels. Dans les deux configurations illustrées par la figure 46, les valeurs internes générées sont égales - elles sont groupées différemment.

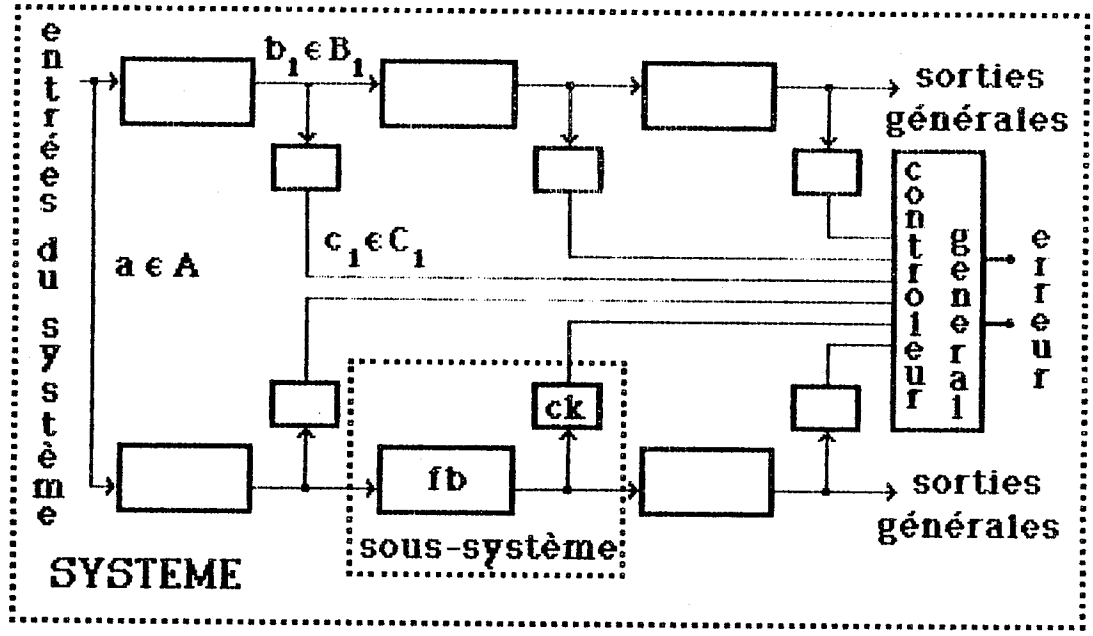
Pour le cas de l'ensemble de sous-systèmes / contrôleur général, pendant le fonctionnement normal (donc, "sans faute") chacun des blocs fonctionnels produit des sorties $b_i \in B_i$, lesquelles sont analysées par chacun des contrôleurs respectifs. Ces derniers produisent des sorties $c_i \in C_i$, dont la concaténation $(c_1.c_2.c_3.....,c_i) \in C$ constitue l'ensemble d'entrées du contrôleur général. Le contrôleur général donne l'indication d'erreur à travers son code de sortie.

Pendant le fonctionnement normal, les vecteurs du code appliqués au système et selon l'arrangement des blocs fonctionnels, devront assurer que tous les blocs en cascade, aussi bien que les contrôleurs, reçoivent les vecteurs nécessaires à leur test.

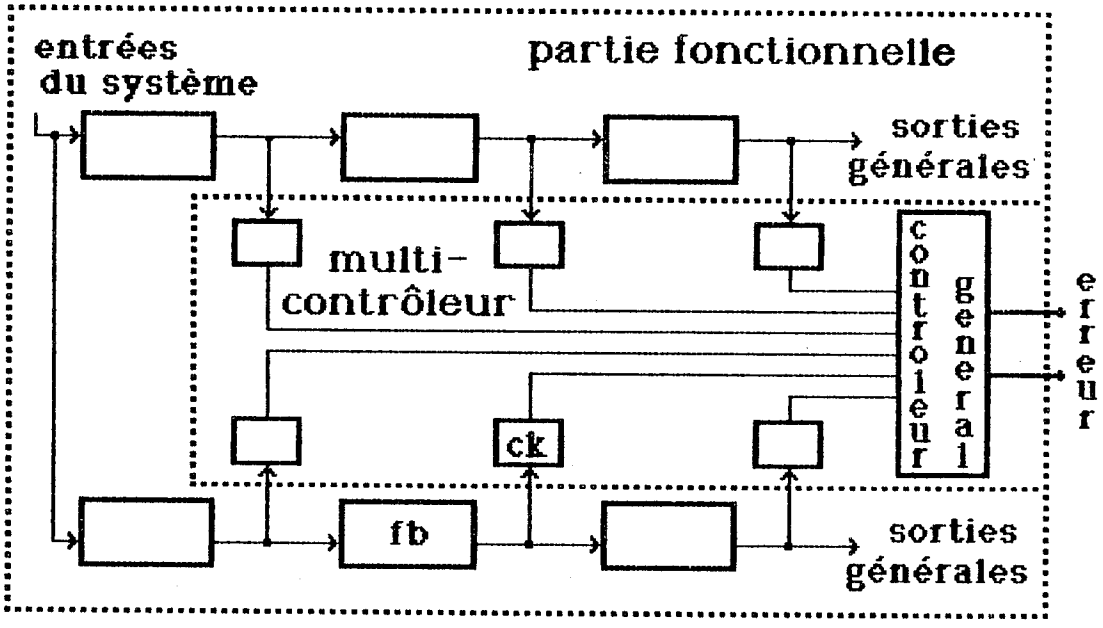
**Hypothèse H4**

Entre l'occurrence de deux fautes quelconques affectant le contrôleur général, il s'écoule un laps de temps suffisant pour que le code d'entrée C (où $(c_1.c_2.c_3.....,c_i) \in C$) soit appliqué et d'autres défauts ne surviennent pas dans les sous-systèmes. Entre l'occurrence de deux fautes quelconques affectant un sous-système, il s'écoule un laps de temps suffisant pour que le code A soit appliqué aux entrées externes du système et d'autres fautes

ne surviennent pas au contrôleur.




a.




b.

Figure 46: Les deux manières de diviser le système et la nomenclature correspondante. a) ensemble de sous-systèmes et le contrôleur général; b) une partie fonctionnelle + un multicontrôleur

Nous considérons que l'occurrence de défauts dans chacun des

sous-systèmes suit soit l'hypothèse H2 soit H3, selon la conception du contrôleur associé (si la propriété SCD respectée est celle définie par D23 ou D26, respectivement). H4, comme énoncée ci-dessus, convient à la structure présentée par l'item a (figure 46). L'hypothèse correspondante au schéma de l'item b sera énoncée plus tard.

**Proposition P6**
Etant considérée l'hypothèse H4, un système composé de :
- blocs fonctionnels SFS,
- contrôleurs associés SCD,
- un contrôleur général SCD,

atteint le "TSC goal".

**Démonstration de P6**

Initialement, soit il n'y a pas de faute dans le système soit, s'il y en a, le système est "strongly redundant" pour ces fautes ou séquences de défauts. Ainsi, un défaut peut arriver à n'importe quel bloc du système, dans le contrôleur général ou dans un des sous-systèmes. Nous considérons l'hypothèse H2 pour l'occurrence de défauts affectant chaque sous-système.

Cas 1:

Une faute $f_g$'' se produit dans le contrôleur général, dont la fonction éxécutée est appelé $G_g$. L'hypothèse H4 assure que toutes les entrées c e C seront appliquées au contrôleur général avant qu'un deuxième défaut affecte ce même bloc ou un des sous-systèmes. S'il n'y a pas l'occurrence de défauts dans les sous-systèmes, les entrées c e C sont effectivement reçues par le contrôleur durant cette période, et deux situations peuvent résulter :

a) le contrôleur général est "strongly redundant" pour la faute ou séquence de défauts, et pour tous c e C, $G_g(c,\langle f_1'',f_2'',.....,f_g''\rangle) \in C'$, où C' est l'ensemble de vecteurs de sortie du contrôleur général. Par conséquent, le contrôleur général ne produit pas d'indication d'erreur, ni de vecteurs erronés.

b) le contrôleur général est "strongly redundant" pour la séquence de défauts $\langle f_1'',f_2'',.....,f_{g-1}''\rangle$, mais $\exists\ c \in C \mid G_g(c,\langle f_1'',f_2'',.....,f_{g-1}''\rangle) \notin C'$ et la faute est détectée.

Si un nouveau défaut survient au système dans le contrôleur général, le comportement sera similaire à celui décrit ci-dessus.

Cas 2:

Une faute $f_m$ se produit dans un des sous-systèmes ou des fautes se produisent dans des sous-systèmes différents. Par l'hypothèse H4, d'autres fautes n'apparaîtront pas dans un même sous-système ni dans le contrôleur général, avant que le code d'entrée A soit appliqueé. Cette supposition considére un laps de temps plus grand que l'hypothèse H2 car on demande l'application du code d'entrée A avant l'occurrence d'une nouvelle faute dans le sous-système - l'application du code d'entrée B (un sous-ensemble de A peut être plus réduit) est suffisant pour la détection des pannes dans le contrôleur. Donc, la proposition d'un système composé de circuit SFS avec contrôleur SCD sous l'hypothèse H2 démontré en [NIC 83a] est considérée comme valable pour assurer qu'une panne pour laquelle le sous-système n'est pas redondant résultera en un vecteur hors code aux sorties, i.e., une sortie c e C est générée. Enfin, dans les conditions ici admises, il est certain qu'une paire d'entrées hors code correspondant au sous-système affecté se présentera au contrôleur général, quand le code d'entrée A est appliqué au système.

Les entrées du contrôleur général (double-rail, selon la définition du système) sont l'ensemble des paires de sorties en provenance des contrôleurs associés. Il suffit qu'une paire d'entrées ne soit pas complémentée pour que leur concaténation résulte en un vecteur hors code aux entrées du contrôleur double-rail. Celui-ci n'étant pas sans faute, produira l'indication d'erreur correspondante. Par conséquent, le défaut dans le sous-système (ou les défauts dans les sous-systèmes) est détecté.

Q.E.D.

Du point-de-vue théorique, lorsque l'on considère l'hypothèse H2 pour l'occurrence de défauts dans chacun des sous-systèmes, on peut remettre en question la suffisance et la nécessité de l'hypothèse H4, employée avec les systèmes composés de plusieurs blocs fonctionnels SFS et contrôleurs SCD, le "TSC goal" étant envisagé. D'après la démonstration de la proposition P5, sa suffisance devient claire. Toutefois, s'il y a une seule faute dans un des contrôleurs associés, lequel peut être testé par un ensemble d'entrées $b_n$ e B généré par un sous-ensemble de A, il est évident qu'il ne serait pas nécessaire d'appliquer tous les a e A. Mais en fait, en utilisant un système comme défini ci-dessus,il n'y a pas moyen d'établir une différence entre les défauts qui surviennent aux blocs fonctionnels ou aux contrôleurs associés, et les seules entrées contrôlables sont les entrées du système, nommées a e A. De toute manière, puisque les défauts peuvent se produire presque

"disséminés" dans le système, on peut les avoir dans un des contrôleurs et dans un des blocs fonctionnels d'un autre sous-système à la fois, des sous-ensembles différents de A seront nécesaires à tout moment. Donc, afin d'avoir une hypothèse réaliste, nous ne pouvons pas demander un laps de temps plus petit que celui d'application des entrées a e A.

Vérifions, maintenant, les changements de l'hypothèse concernant un système selon le schéma présenté à la figure 46b - une partie fonctionnel + un multicontrôleur. Pendant le fonctionnement normal, chacun des blocs fonctionnels produit des sorties $b_i \in B_i$ - ces sorties concaténées constituent les entrées du multicontrôleur. Ainsi, le multicontrôleur reçoit $(b_1.b_2.b_3...b_i) \in B$ comme des entrées et ses sorties générales donnent l'indication d'erreur. A l'intérieur du système, les vecteurs nécessaires au test de chacun des blocs fonctionnels et du multicontrôleur doivent être assurés.

**Hypothèse H4'**
Entre l'occurrence de deux fautes quelconques affectant le multicontrôleur, il s'écoule un laps de temps suffisant pour que le code d'entrée B soit appliqué et d'autres fautes ne se produisent pas dans la partie fonctionnelle. Entre l'occurrence de deux fautes quelconques affectant la partie fonctionnelle, il s'écoule un laps de temps suffisant pour que le code d'entrée A soit appliqué au système, et d'autres fautes ne surviennent pas au multicontrôleur.

La composition d'un système basé sur ce modèle est suggérée par la prochaine proposition.

**Proposition P7**
Etant assurée l'hypothèse H4', un système composé de :
- une partie fonctionnel SFS et
- un multicontrôleur SCD

atteint le "TSC goal".

Nous n'allons pas démontrer cette proposition car il ne s'agit que d'une division thèorique différente du même système présenté par la proposition P6.

### III.4.3. Extention des concepts des contrôleurs SCD pour des applications hors ligne

Nicolaïdis et Courtois [NIC 84b] proposent l'utilisation du système

présenté au début de cette section pour des applications hors-ligne; nous profiterons de ses propriétés autotestables pour le test en fonctionnement, et pour la maintenance de l'équipement, et davantage de l'organisation du système pour la localisation des pannes.

Concernant l'emploi hors-ligne, chacun des sous-systèmes devient une unité remplaçable; mais nous pouvons avoir aussi plusieurs blocs et contrôleurs dans la composition d'une de ces unités. Ainsi une unité remplaçable est caractérisée comme un bloc séparable composé d'une partie fonctionnelle (ou plusieurs parties fonctionnelles) et de son contrôleur associé (ou de leurs contrôleurs associés); cette unité reçoit un ensemble d'entrées externes et génère les sorties fonctionnelles externes correspondantes aussi bien que l'indication d'erreur produite par son contrôleur interne.

La figure 47 donne des exemples d'unités remplaçables.

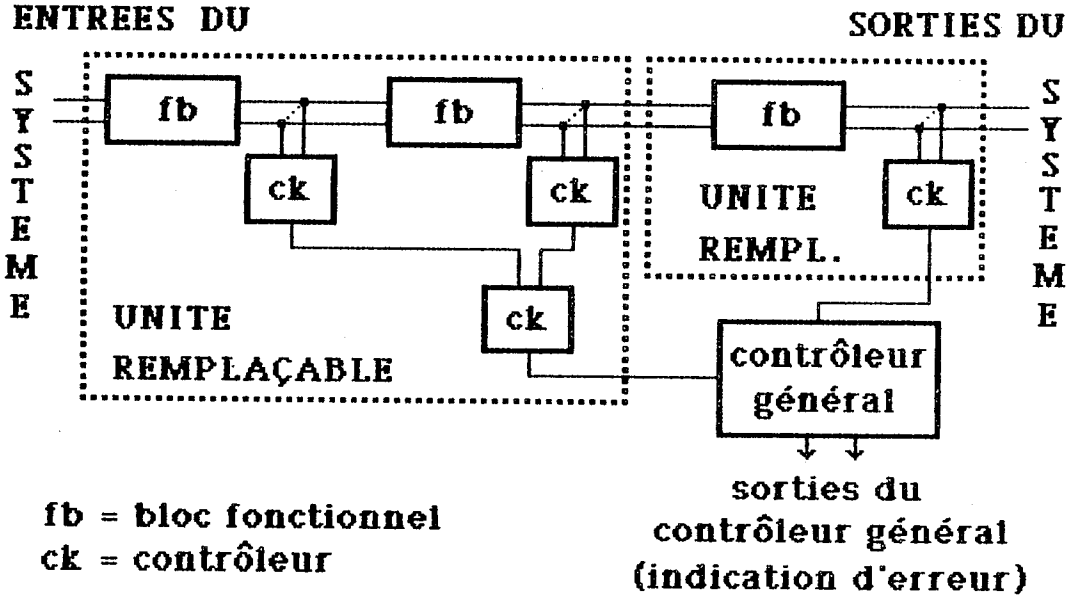



Figure 47 : Système composé de deux unités remplaçables et un contrôleur général

La méthode suggérée en [NIC 84b] particularisée par une application dans l'aviation, utilise les propriétés autotestables pour détecter des défauts pendant une mission, l'hypothèse H4 étant considérée (le terme sous-système est substitué par "unité remplaçable") pour l'occurrence de pannes dans le système, et H2 pour chaque unité remplaçable entre deux missions quelconques, il y a une période de test hors-ligne pendant laquelle de deux à plusieurs procédures de test peuvent être exécutées.

Durant cette période, une procédure de test consiste à appliquer tous les vecteurs du code appartenant à l'ensemble A (par conséquent tous b c B nécessaires à la vérification du multicontrôleur seront générés); les deux procédures de base sont les suivantes:

TEST 1- exécuté juste après la conclusion d'une mission
TEST 2- exécuté juste avant le début d'une nouvelle mission.

Si une faute est détectée pendant l'exécution de la procédure TEST 1, un service de maintenance suit; la localisation du défaut est possible au niveau de l'unité remplaçable par l'analyse des sorties des contrôleurs de chacune de ces unités.

Plusieurs TEST2 peuvent être réalisés entre le TEST1 et une nouvelle mission. La détection d'une faute ou d'une séquence de fautes par cette procédure résultera en un service de maintenance, une reconfiguration du système ou même la suspension de l'utilisation du système, selon la méthode prévue et le laps de temps disponible.

Afin d'assurer que les procédures de test hors-ligne seront effectives, il faut bien définir les hypothèses de pannes qui leur sont associées. Nous formulons des hypothèses dérivées des précédentes, adaptées à la terminologie des test hors-lignes. Les propositions suivantes n'ont pas besoin d'être démontrées car les propriétés des blocs considérés sont les mêmes que les cas précédents; donc, leurs démonstrations sont encore valables.

**Hypothèse H5**
Entre le TEST1 et le TEST2 (ou entre deux procédures de TEST2 pendant la même période de test hors-ligne), seulement une faute (appartenant à un ensemble de défauts si des ensembles sont considérés,ou appartenant à une classe de pannes si des classes sont prises en compte) peut survenir, soit affectant chaque unité remplaçable, soit le contrôleur général.

**Proposition P8**
Etant considéré l'hypothèse H5, un système composé de:
- des unités remplaçables conçues avec des circuits SFS et des contrôleurs SCD, et
- un contrôleur général SCD,

atteint le "TSC goal" pendant une mission (c'est-à-dire, à partir du dernier TEST2 de la dernière opération hors-ligne jusqu'au prochain TEST1 de la prochaine opération hors-ligne).

Nous remarquons que l'hypothèse H5 est énoncée en vue d'un système composé de la même façon que celui utilisé sous l'hypothèse H4. En outre, nous pouvons vérifier que H5 admet l'occurrence de pannes concomitentes dans des unités remplaçables différentes, entre deux tests. La démonstration de la proposition P8 est analogue à celle présentée pour P6, avec la terminologie échangée.

L'hypothèse associée à la même situation, donnée par [NIC 84b], est fondée sur la même structure que celle du système que nous avons examinée avec H4': " Entre les procédures TEST1 et TEST2, une seule faute (appartenant à l'ensemble de défauts si des ensembles sont considérés, ou appartenant à la classe de pannes, si des classes sont considérées) peut se produire affectant soit la partie fonctionnelle soit le contrôleur". Nous pouvons observer que cette hypothèse est un peu plus restrictive que H5 car un seul défaut est accepté dans tout le système. C'est une condition suffisante mais pas nécessaire. Dans les deux cas, la conception a été faite à partir de circuits SFS et de contrôleurs SCD.

Si le contrôleur général a la propriété SCD forte (D26), il est possible d'utiliser le système sous une hypothèse plus large (plus faible) - pour les périodes de mission et de test hors-ligne.

**Hypothèse H6**
Entre l'occurrence de deux fautes affectant le contrôleur général, il s'écoule un laps de temps suffisant pour l'application du code d'entrée C. Entre l'occurrence de deux fautes quelconques affectant une unité remplaçable, il s'écoule un laps de temps suffisant pour l'application du code d'entrée A aux entrées externes du système.

**Proposition P9**
Etant respectée l'hypothèse H6, un système composé de:

- des unités remplaçables conçues à partir de circuits SFS et de contrôleurs SCD, et
- un contrôleur général SCD (par la définition D26),

atteint le "TSC goal" durant une mission (à partir du dernier TEST2 de la période de test hors-ligne précédente jusqu'au prochain TEST1 de la prochaine opération hors-ligne).

**Hypothèse H7**
Entre le TEST1 et le TEST2 (ou entre deux procédures de TEST2 de la même période d'opération hors-ligne), une seule faute affectant le contrôleur général peut se produire. Entre le TEST1 et le TEST2 (ou entre deux

procédures de TEST2 de la même période d'opération hors-ligne), une seule faute peut survenir affectant chacune des unités remplaçables.

Les fautes référées ci-dessus appartiennent à un ensemble ou à une classe de défauts, selon le cas considéré.

**Proposition P10**
Etant respectée l'hypothèse H7, un système composé de:
- des unités remplaçables conçues avec des circuits fonctionnels SFS et contrôleurs SCD, et
- un contrôleur général, lequel est SCD selon la définition D26 (définition forte),

atteint le "TSC goal" pendant une mission (à partir du dernier TEST2 de la période de test hors-ligne précédente jusqu'au prochain TEST1 de la prochaine opération hors-ligne).

## III.5. CONCLUSION

Dans ce chapitre, nous avons étudié la conception des contrôleurs combinatoires - "strongly code disjoint", ainsi que des hypothèses de pannes qui peuvent survenir dans les systèmes dans lesquels ils sont utilisés, en applications en ligne et hors ligne.

L'analyse de chaque nouvelle proposition topologique des ces contrôleurs pouvant ne pas être très facile pour le concepteur qui n'est pas familiarisé avec les principles du dessin SCD, peut être évitée si une bibliothèque de cellules pré-conçues est utilisée. La plupart des contrôleurs classiques presentés dans le chapitre II ou les blocs qui composent le circuit de contrôle peuvent être assemblés à partir de cellules de base directement. Mais l'assemblage des cellules pré-dessinées est une activité qui eventuelment introduit aussi des fautes non détectables mais pour lesquels le contrôleur n'est pas redondant - une condition inacceptable donc, à être eliminée, comme nous avons examiné dans le chapitre present.

Des cas particuliers, des contrôleurs pour des codes double-rail et à parité, ont éte examinés en détail. Les règles générales données auparavant, peuvent être particularisées (moins de contraites) dans chaque cas de contrôleur, selon le comportement résultant des fautes apparues par conséquent.

L'étude de la redondance du circuit vis-à-vis des fautes qui

surviennent nos ammène à une conclusion additionnelle interessante: c'est la possibilité de réduire le nombre des éléments électriques de base (par conséquent à une réduction possible des dimension physiques du même) du contrôleur. Cependant ces réductions ne seront pas toujors faites par des raisons comme: a) soit les redondances résultent d'une conception automatisée et l'analyse des cas particuliers n'étant pas dangereux (ne présentent pas de problèmes subséquement) ne sera pas réalisé; b) soit les redondances ne peuvent pas être eliminées.

En vue des hypothèses de fautes, nous avons consideré la technologie NMOS, et c'est donc à cette technologie qui s'addressent les règles et les dessins présentés. Les conclusions deveront être revues en fonction des applications en CMOS et technologies à multi-interconnexions.

A la fin du chapitre, nous avons examiné des hypothèses de pannes qui peuvent survenir quand des ensembles de contrôleurs SCD - ici nommés multi-contrôleurs - sont utilisés pour détecter des pannes pendant le fonctionnement en ligne aussi bien qu'hors ligne. C'est une possibilité interessante quand sa sûreté de fonctionnement est indispensable, car la redondance materielle est réduite à un type de circuits de test.

# CHAPITRE 4:

## CONTROLEURS STRONGLY LANGUAGE DISJOINT (SLD)

# CONTROLEURS STRONGLY LANGUAGE DISJOINT (SLD)

## IV.1. DEFINITIONS DE BASE: MACHINES SEQUENTIELLES ET LANGAGES

Tout d'abord, nous introduisons des définitions de base pour des langages et machines séquentielles. Elles sont reprises de [DEN 78].

**Définition D27**
Un vocabulaire V est un ensemble fini de symboles que représentent des objets atomiques ou indivisibles utilisés dans la construction des phrases. La séquence de k symboles en V, $\partial = v_1, v_2, \ldots\ldots, v_k$ où $v_i \in V$, est appellée chaîne de longueur k dans le vocabulaire V. En supprimant normalement les virgules, on écrit $\partial = v_1 v_2 \ldots\ldots v_k$. L'ensemble de toutes les chaînes en V est nommé $W = \cup W_k$, où $k = 0, 1, \ldots, \infty$, la chaîne vide inclue.

**Définition D28**
La concaténation de deux chaînes $\partial$ et $\Phi$ est la séquence de symboles composée par l'extension de la séquence $\partial$ avec la séquence de symboles $\Phi$, donc :

$$m(\partial, \Phi) = v_1 v_2 \ldots\ldots v_m . u_1 u_2 \ldots u_n = \partial.\Phi$$

où $\partial = v_1 v_2 \ldots\ldots v_m$ et $\Phi = u_1 u_2 \ldots u_n$.

Si $\partial = \Phi.\Psi$, on dit que la chaîne $\Phi$ est un préfixe de $\partial$, ou un préfixe propre si $\Psi$ n'est pas une chaîne vide. D'une façon similaire, $\Psi$ est un suffixe de $\partial$, ou un suffixe propre de $\Phi$ s'il n'est pas vide.

La concaténation de symboles et chaînes selon des procédures prédéfinies résultera en un langage. Ces procédures définies composent une part d'une grammaire.

**Définition D29**
Une grammaire formelle est une quadruple G = (N, T, P, Σ) telle que :

N est un ensemble fini de symboles non terminaux
T est un ensemble fini de symboles terminaux
N et T sont disjoints : $N \cap T = \emptyset$
P est un ensemble fini de productions
Σ est le symbole de phrase : Σ ¢ (NUT).

Chaque production en P est une paire ordonnée de chaînes (ß,¥) où :

$ß = \Phi A \Psi$
$¥ = \Phi \partial |$

et dans lequel $\partial, \Phi$ et $\Psi$ peuvent être des chaînes vides en $(N \cup T)^*$ et A est Σ ou un caractère non terminal. On écrit en général une procédure (ß,¥) comme ß->¥.

**Définition D30**
Soit G une grammaire formelle. Une chaîne de symboles en $(N \cup T)^* \cup \{\Sigma\}$ est vue comme une forme de phrase. Si $\beta \rightarrow \varphi$ est une production de G et $\partial = \Psi\beta\Psi$ et $\partial' = \Psi\varphi\Psi$ sont des formes de phrases, on dit que $\partial'$ est dérivé immédiatement de $\partial$ en G. Ce rapport est indiqué par $\partial \Rightarrow \partial'$. Si $\partial_1, \partial_2, \ldots, \partial_n$ est une séquence de formes de phrases telle que $\partial_1 \Rightarrow \partial_2 \Rightarrow \ldots \Rightarrow \partial_n$, on dit que $\partial_n$ est dérivable de $\partial_1$ et la relation est représentée par $\partial_1 \Rightarrow \partial_n$. La séquence $\partial_1, \partial_2, \ldots, \partial_n$ est dite la dérivation de $\partial_n$ à partir de $\partial_1$ selon G.

**Définition D31**
Le langage L(G) généré par une grammaire formelle G est l'ensemble de chaînes terminales dérivables de $\Sigma$ :

$$L(G) = \{ \partial \in T^* \mid \Sigma \Rightarrow \partial \}$$

Si $\partial \in L(G)$, on dit que $\partial$ est une chaîne, une phrase, ou un mot dans le langage généré par G.

Un langage généré par une grammaire formelle peut être reconnue par une machine d'états finis, qui est l'automate le plus simple. Une machine d'états finis, connue aussi comme une "machine séquentielle", manipule des séquences des symboles d'entrée et sortie, et son comportement est représenté par une séquence d'états. Les sorties de ces machines peuvent dépendre de transitions et entrées ou uniquement de l'état. Ces deux cas sont définis différemment.

**Définition D32**
Une machine d'états finis affectée par transitions, ou un automate de Mealy, est un sixtuplet $M = (Q, X, Z, \delta, w, q_0)$, dont :

Q est un ensemble fini d'états internes
X est un vocabulaire fini d'entrées
Z est un vocabulaire fini de sorties
$\delta$ est une fonction de transition d'état $\delta : Q \times X \rightarrow Q$
w est une fonction de sortie $w : Q \times X \rightarrow Z$
$q_0 \in Q$ est l'état initial.

**Définition D33**
Une machine d'états finis affectée par états, ou automate de Moore, est un sixtuplet $M = (Q, X, Z, \delta, w, q_0)$ où

Q est un ensemble fini d'états internes
X est un vocabulaire fini d'entrée
Z est un vocabulaire fini de sortie

$\delta$ est un fonction de transition d'état $\delta : Q \times X \rightarrow Q$
w est une fonction de sortie $w : Q \rightarrow Z$
$q_0 \in Q$ est l'état initial.

Il est possible de transformer un automate de Mealy en un de Moore et vice-versa. Il est aussi possible d'établir des relations d'équivalence entre machines et états. Ensuite, sont présentées des définitions complémentaires liées à ces concepts.

**Définition D34**
Deux machines $M_1$ et $M_2$ sont équivalentes si et uniquement si :

(1) Leurs vocabulaires d'entrée et leurs vocabulaires de sortie sont les mêmes : $X_1 = X_2$, $Z_1 = Z_2$.

(2) Pour chaque stimulus, $M_1$ et $M_2$ produisent des réponses identiques. Donc si $x_1(t) = x_2(t)$, $t \geq 1$, alors $z_1(t) = z_2(t)$, $t \geq 1$.

Si $M_1$ et $M_2$ sont équivalentes, nous écrivons $M_1 \approx M_2$.

**Définition D35**
Soit M la machine d'états finis avec fonction de transition $\delta : Q \times X \rightarrow Q$. Si $\delta(q, x) = q'$, on dit que l'état q' est le x-successeur de l'état q, dénoté $q \rightarrow q'$. Si une chaîne de symboles d'entrée $\partial = i(1)\, i(2) \ldots i(t)$ emmène M de l'état $q = q(0)$ à l'état $q' = q(t)$, i.e., si $q(0) \rightarrow q(1) \rightarrow \ldots \rightarrow q(t)$ on dit que l'état q' est le $\partial$-successeur de l'état q, représenté par $q \Rightarrow q'$. Sous ces conditions, $q(0)\, q(1) \ldots\ldots q(t)$ est appelée une séquence d'états admissible pour $\partial$.

**Définition D36**
Deux états $q_a$ et $q_b$ d'une machine de Mealy $M = (Q, X, Z, \delta, w, q_1)$ sont des états équivalents si et uniquement si les machines $M_a = (Q, X, Z, \delta, w, q_a)$ et $M_b = (Q, X, Z, \delta, w, q_b)$ sont équivalentes. Un énoncé analogue s'applique aux machines de Moore. Si $q_a$ et $q_b$ sont des états équivalents, nous écrivons $q_a \approx q_b$.

Cette définition peut être énoncée d'un façon simplifiée (applicative) comme ci-dessous.

**Définition D37**
Deux états $q_a$ et $q_b$ d'une machine d'états finis M sont équivalents si et

uniquement si :

1a. Mealy : pour tous les $x \in X$, $w(q_a,x) = w(q_b,x)$ ; i.e., les sorties pour des transitions correspondentes de ces deux états sont identiques.
1b. Moore : $w(q_a) = w(q_b)$ ; i.e., les symboles de sortie pour les deux états sont identiques.
2. Pour tous les $x \in X$, $\delta(q_a,x) \approx \delta(q_b=x)$ ; i.e., les x-successeurs des deux états sont eux-mêmes des états équivalents.

Ces définitions présentées au préalable seront utilisées dans le texte qui suit comme base pour les concepts et expressions rapportés au domaine de machines et langages.

## IV.2. CIRCUITS "SEQUENTIALLY SELF-CHECKING"

Une machine séquentielle est définie par le sixtuplet $M = (Q, X, Z, \delta, w, q_0)$. Q est l'ensemble des états internes. X et Z sont les alphabets d'entrée et de sortie, respectivement. $\delta$ et w sont la fonction de transition et la fonction de sortie, respectivement. Et $q_0$ est l'état initial. Nous allons considérer comme modèle la machine de Mealy, mais les idées présentées peuvent être facilement appliquées au modèle équivalent de Moore, une fois que la transformation d'un modèle à l'autre, et vice-versa, est possible.

En présence d'une panne f, une machine $M = (Q,X,Z,\delta,w,q_0)$ devient $M^f = (Q^f,X,Z^f,\delta^f,w^f,q_0)$. Pendant le fonctionnement normal (avant l'occurrence des défauts), $\delta(i,q)$ est l'état atteint lorsque l'on applique la séquence i à partir de l'état q et w(i,q) est la séquence de sortie obtenue dans les mêmes conditions. S'il y a un défaut f qui intervient dans l'état q de M, il est supposé que $M^f$ atteint immédiatement l'état $q^f$. Ainsi, $\delta^f(\lambda,q) = q^f$, $q \in Q$, $q^f \in Q^f$ et $w^f(i,q)$ est la séquence de sortie obtenue.

Tout d'abord nous allons citer les principales publications concernant les circuits sequentiellment sûres, et ensuite nous expliquerons l'approche utilisé ici.

DIAZ [DIA 74b] a défini une machine séquentielle TSC comme en étant "self-checking" et "fault secure". Selon Diaz, une machine séquentielle est "self-checking" si $\forall f \in F, \exists i \in I_M, \exists q \in Q \mid w^f(\delta^f(i,q)) \in O_M$; et elle est "fault

secure" si $\forall f \in F, \forall i \in I_M, \forall q \in Q$ soit $w^f(\delta^f(i,q)) = w(\delta(i,q))$ soit $w^f(\delta^f(i,q)) \notin O_M$. Une telle définition n'assure pas que l'état interne à partir duquel la détection du défaut est possible, sera atteint.

La structure interne de cette machine est décrite par un modèle de Moore et utilise des états codés. Ces états sont testés en fonction de son appartenance au code ou non (propriété combinatoire). La machine est composée de trois parts. La première reçoit des entrées double-rail et les traduit en un code 1-parmi-n. La deuxième éxécute la fonction "prochain état" et la mémorisation de l'état. La fonction sortie est calculée par le troisième bloc qui reçoit les états codés comme entrées : ce bloc est, en fait, un contrôleur combinatoire TSC de ces entrées et donne un résultat combinatoire de cette analyse en un code double-rail. Donc, la sortie et le contrôle de la machine sont mélangés. Par exemple, si nous supposons une seule paire de sorties, 01 et 10 seraient des valeurs possibles de sortie et 00 et 11 iraient coder une erreur. Une telle machine séquentielle est représentée dans la figure 48. Il faut encore remarquer que les pannes sont modélisées par des collages.

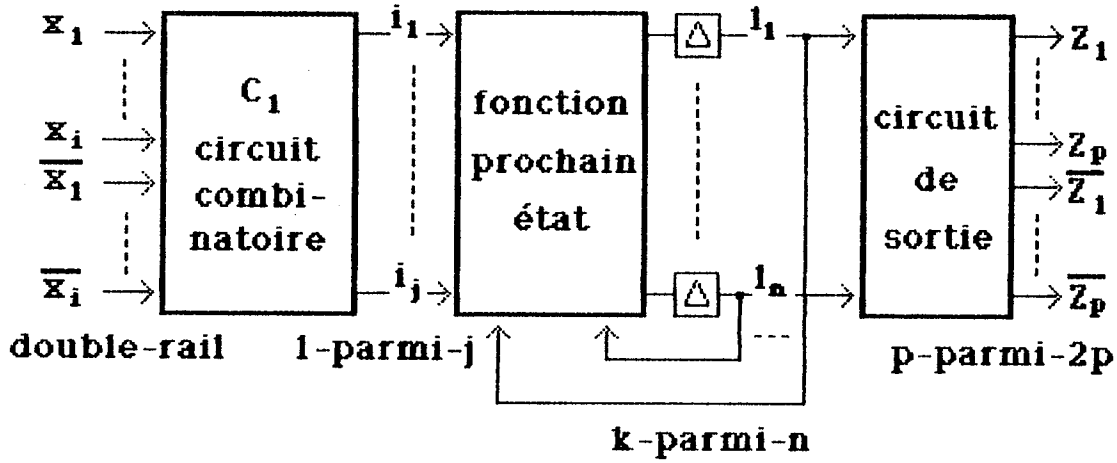



Figure 48: Proposition de Diaz pour le système séquentiel

ÖZGUNER en [ÖZG 77] a proposé une structure similaire à celle (ci-dessus) de Diaz. Il utilise le modèle de Mealy pour la machine et un contrôleur lui est incorporé seulement quand des sorties double-rail sont envisagées.

En [DIA 79], une nouvelle définition des circuits autocôntrolables est donnée en éliminant le problème précédent de ne pas atteindre l'état

interne à partir duquel la détection du défaut serait possible. Selon cette nouvelle définition, une machine séquentielle est autocontrôlable pour un ensemble de défauts F si $\forall f \in F, \forall q \in Q, \forall i \in I_q$, il existe une séquence d'entrée i telle que $w^f(\delta^f(i,q))$ n'appartient pas au code de sortie. Le modèle de collage est considéré pour les fautes. Les contrôleurs, aussi dans ce cas, sont combinatoires et vérifient le code des états.

D'autres papiers dans ce même domaine ont étudié le dessin de machines séquentielles sûres, en utilisant toujours des modèles de collages et des sorties codées.

TOHMA et alli. [TOH 71] ont défini un système sûr pour des défauts symétriques et ont montré que la codification k-parmi-n des états conduit à une réalisation plus facile des machines séquentielles sûres. Le circuit de sortie est combinatoire et vérifie le code des variables d'état.

CHUANG et DAS [CHU 78] ont proposé une machine fonctionnelle séquentielle avec des états codés en code de Berger, et le contrôleur simplement vérifie ce code.

Ces arrangements peuvent être généralisés par des systèmes comme montré à la figure 49. En fait, un des deux contrôleurs représenté est utilisé - on montre les deux places possibles.

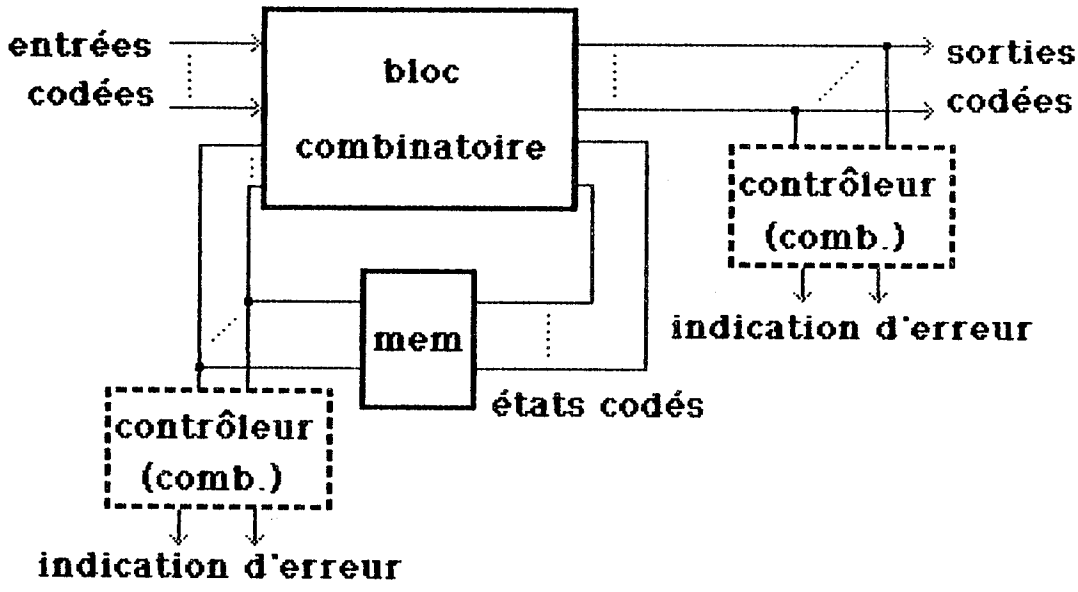



Figure 49: Version étendue des machines autocontrôlables selon Diaz, Özguner et autres

TAKAOKA et IBARAKI [TAK 73] ont étudié le dessin des circuits SC et FS à partir d'une approche algébrique; cependant cette étude n'a pas une connexion directe avec d'autres faites sur les systèmes FS et SC.

MEYER et SUNDSTROM [MEY 75] ont regardé la conception des machines séquentielles avec le diagnostic de fautes sans restriction, pendant le fonctionnement, par l'utilisation d'une machine inverse.

Si l'on suppose les sorties et les états codés avec des codes de duplication, le schéma représenté à la figure 50 devrait être obtenu. VIAUD et DAVID [VIA 80] ont montré que dans ce cas, le contrôle des états n'est pas demandé : la surveillance des sorties (qui concaténées forment un code à duplication) est suffisante. Nous remarquons à nouveau que le contrôleur est combinatoire. Une machine autocontrôlable construite avec l'adjonction de la machine inverse allait aussi avoir besoin d'un contrôleur combinatoire seulement.

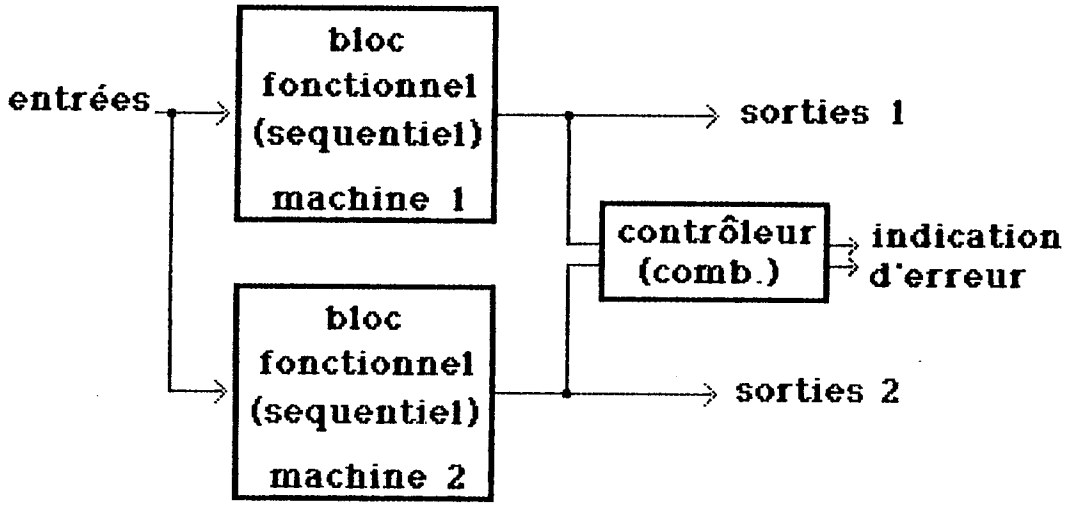



Figure 50: Schéma d'une machine SC basée en duplication

La structure que nous allons considérer est celle de la description suivante. Les séquences d'entrée sont définies par un langage d'entrée donné. Les séquences de sortie sont définies par un langage de sortie donné. Egalement, le contrôleur reçoit un langage d'entrée déterminé (le langage de sortie du système fonctionnel séquentiel), et produit sorties qui appartiennent à un langage de sortie. Ces sorties indiquent une erreur si elles n'appartiennent pas au langage de sortie. Les propriétés nécessaires aux contrôleurs dépendent des caractéristiques de sortie du système fonctionnel : elles sont définies comme en étant séquentielles, mais le contrôleur peut être particularisé pour donner des sorties combinatoires

appartenant à un code. La structure de base est décrite par la figure 51.

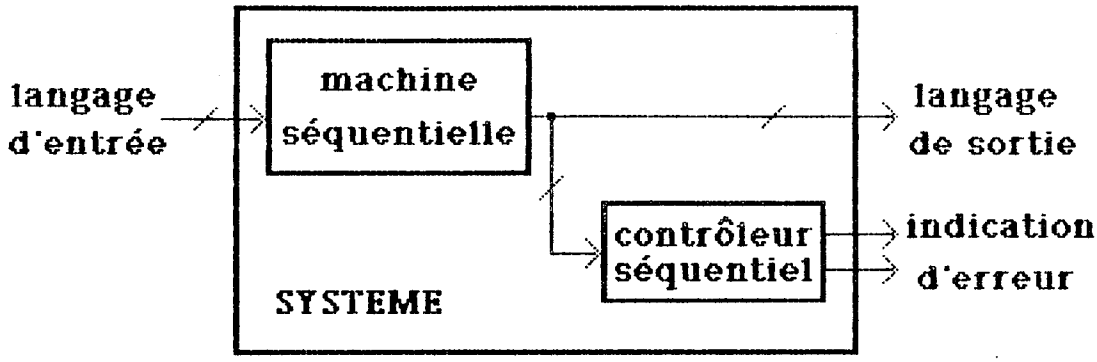



Figure 51: Système séquentiellement autocontrolable

Soit $i = i_1 . i_2$ la concaténation des $i_1$ et $i_2$. La séquence d'entrée $i_1$ est appelée préfixe de i et $i_2$ est le suffixe de i. Nous noterons P(i) et S(i) les ensembles de préfixes et suffixes de i, respectivement. Les définitions qui suivent ont été prises de [VIA 80].

**Définition D38**
Le langage d'entrée de M, appelé $I_M$, est l'ensemble de toutes les séquences d'entrée i applicables à partir de l'état initial $q_0$ (donc, $I_M$ définit le fonctionnement normal).

**Définition D39**
Le langage de sortie de M, appelé $O_M$, est l'ensemble des séquences de sortie $O_m$ que l'on peut obtenir à partir de l'état $q_0$ et défini de la façon suivante :

$$O_m = \{ w(i,q_0) : i \in I_M \}.$$

Soit $I_q$ l'ensemble des séquences d'entrée applicables en operation normale à partir de l'état q: $I_q = \{i_2 : i_1 . i_2 \in I_M \text{ et } \delta(i_1,q_0)=q$.

$I_q^{\infty} \in I_q$ est l'ensemble des séquences de longueur non bornée.

Le plus petit préfixe de $i_2$ pour lequel une séquence de sortie n'appartenant pas au langage de sortie de M est obtenue en présence du défaut f, est noté $i_{2m}$. Donc, nous avons :

$$w(i_1, q_0) . w^f(i_{2m}, q) \notin O_M.$$

**Définition D40**
Une machine est "sequentially fault secure" (SeFS) pour une panne f, un état q, et une séquence d'entrée $i_2 \in I_q$ si :

$$w^f(i_2', q) = w(i_2', q), \forall\, i_2' \in P(i_{2m}),\ i_2' \neq i_{2m}, \text{ si } i_{2m} \text{ existe;}$$
$$\forall\, i_2' \in P(i_2), \text{ autrement.}$$

On peut établir un rapport entre cette définition et celle donnée pour les circuits "fault secure", aux systèmes combinatoires. La sortie devra être égale à celle obtenue pendant le fonctionnement normal, ou une indication d'erreur devra être produite - pour les circuits combinatoires c'est un mot hors code; pour les circuits séquentiels , cette indication est une séquence non comprise dans le langage de sortie. Dans les deux cas, le mot non codé et le symbole qui fait que la séquence devienne non appartenant au langage de sortie, causent la détection du défaut.

**Définition D41**
Une machine est "sequentially self-testing" (SeST) pour une panne f, un état q, et une séquence d'entrée $i_2 \in I_q$ si : $\forall\, i_1$ tel que $\delta(i_1, q_0) = q$ et tel que $i_1 . i_2 \in I_M$ : $w(i_1, q_0) . w^f(i_2, q) \notin O_M$.

Nous remarquons que, en ce qui concerne les circuits "sequentially self-testing", la définition n'est pas donnée en termes de "...il y en a au moins un..." (cas combinatoire) une fois que la propriété est définie pour un état donné et une séquence d'entrée appliquée à partir de ceci - la longueur de cette séquence (le préfixe $i_{2m}$ qui produira l'indication d'erreur) est un paramètre variable selon les défauts, état et séquence considérés. Cette considération réunie à la propriété "sequentially fault secure" supprime la possibilité de ne pas arriver à l'état q à cause du défaut f.

La définition suivante est facilement compréhensible si analysé ensemble avec la figure 52.

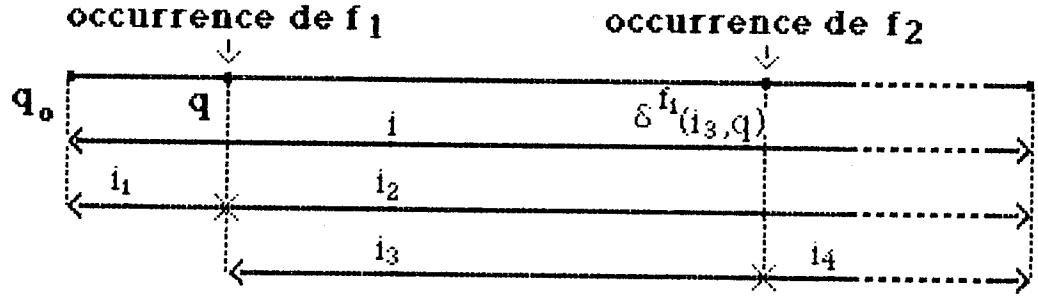



Figure 52: Diagramme de temps pour la définition D42

**Définition D42**
Une machine est "sequentially self-checking" (SeSC) pour un ensemble F de pannes si : $\forall f_1 \in F, \forall q \in Q, \forall i_2 \in I_q^{\infty}$ ,

soit

a. la machine est à la fois SeST et SeFS pour $(f_1, q, i_2)$ ;

soit

b. la machine est SeFS pour $(f_1, q, i_2)$ et $\forall f_2 \in F$ et $\forall i_2 \in S(i_2)$ ($f_2$ survenant après $f_1$), soit la propriété a., soit la propriété b. est vérifiée pour la panne $f_1 \cup f_2$ prenant la place de $f_1$, $\delta^{f_1}(i_3, q)$ tel que $i_2 = i_3 . i_4$ prenant la place de q, et $i_4$ à la place de $i_2$.

Nous remarquons que la définition D42 est récursive. Pendant que la machine reste uniquement SeFS pour les fautes qui arrivent, d'autres fautes peuvent subsequement se produire et la sortie est toujors correcte (jusqu'à sa détection, par la propriété SeST, item a.).

Avant de commenter la définition ci-dessus, nous allons donner les concepts d'un circuit redondant transférés au domaine séquentiel.

**Définition D43**
Un circuit M est redondant pour une panne f, un état q et une séquence d'entrée $i_2 \in I_q$, si $\forall i_2' \in P(i_2)$, $w^f(i_2', q) = w(i_2', q)$.

**Remark Rk1**
Dans un circuit redondant pour un défaut f, un état q, et une séquence d'entrée $i_2 \in Iq$, le préfixe $i_{2m}$ n'existe pas. Nous rappelons que $i_{2m}$ est le préfixe le plus petit tel que $w(i_1 . q_0) . w^f(i_{2m}, q) \notin O_M$.

**Définition D44**
Un circuit est redondant pour un défaut f ou une séquence de défauts $\langle f_1, f_2, \ldots, f_n \rangle$ et un langage d'entrée $I_M$ s'il est redondant pour tous les états q et pour toutes les séquences d'entrée $i_2 \in I_q^{\infty}$ .

Les circuits séquentiels "strongly redundant" sont définis d'une façon semblable aux circuits combinatoires "strongly redundant".

**Définition D45**
Un circuit est "strongly redundant" pour une séquence de défauts $\langle f_1, f_2, \ldots,$

$f_n$>, pour un ensemble d'états Q, et pour un ensemble des séquences d'entrée, si le circuit est redondant pour les n séquences <$f_1$>,<$f_1$, $f_2$>,.....,<$f_1$, $f_2$,....., $f_n$>, pour l'ensemble d'états q ϵ Q, et pour l'ensemble des séquences d'entrée applicables.

La définition récursive D42 (pour une machine SeSC)) comprend des propriétés de redondance, i.e., la propriété énoncée en b. décrit le comportement d'un circuit qui est redondant pour un ensemble de défauts. Si $f_1$ n'est pas détecté, un nouveau défaut $f_2$ peut apparaitre et être détecté ou le circuit est aussi redondant pour cette nouvelle faute, l'état q et pour la séquence d'entrée définie, laquelle est maintenant $i_1 . i_2 . i_3 . i_4$. Cela signifie qu'après l'occurrence d'une séquence de défauts appartenants à F non détectée (défauts pour lesquels le circuit est redondant) la machine reste "sequentially fault-secure" et lorsque elle est "sequentially self-testing" pour le défaut résultant <$f_1$, $f_2$,....., $f_n$>, la détection va se passer avant qu'une autre faute f c F n'apparaisse. Ce concept est la généralisation de la notion "strongly fault secure", énoncé par SMITH et METZE [SMI 78] pour des circuits combinatoires.

La propriété a. donne des conditions correspondantes aux circuits "totally self-checking" pour une faute $f_1$.

Toutes les définitions établies pour des circuits combinatoires peuvent être obtenues à partir des définitions établies pour des circuits séquentiels puisque il y a une correspondance entre les variables de ces deux classes. L'état q existant pour les circuits séquentiels et qui est une conséquence des évènements passés, n'a pas de signification auprès du cas combinatoire, comme il n'y a pas de notion de mémoire dans ce dernier. Donc, s'il n'y a pas "le changement de l'état q", cet état est toujours le même, et il n'est pas nécessaire de le mentionner. Et, puisqu'il n'y a pas de changement d'état, l'ordre d'application des différentes combinaisons d'entrée n'est pas importante - la notion de séquence d'entrée utilisée pour les circuits séquentiels est substituée par un ensemble de mots d'entrée codés dans les circuits combinatoires. Les notations listées ci-dessous sont employées comme correspondantes, en vue de l'obtention des définitions des circuits combinatoires à partir des séquentiels :

| | | |
|---|---|---|
| circuits séquentiels | ---> | circuits combinatoires |
| w(i, q) | ---> | G(a, ø) |
| $w^f$(i, q) | ---> | G(a, f) |
| P(i) | ---> | A |

séquence d'entrée i ---> ensemble de tous les vecteurs d'entrée

$i_{2m}$ ---> $\exists\, a \in A \mid G(a, f) \notin B$.

Cette notation étant utilisée, il est possible de retrouver la définition d'un circuit redondant, pour le cas combinatoire, à partir de la définition D43. Avec les substitutions données ci-dessus, nous aurons : "un circuit G est redondant pour une faute f, et pour les vecteurs d'entrée $a \in A$, si :

$$\forall\, a \in A,\ G(a, f) = G(a, \varnothing)".$$

Nous remarquons qu'un circuit redondant pour un code d'entrée A et une faute f est redondant pour un défaut f, pour n'importe quel état q, et pour toutes les séquences $i_2 \in I_q$ si l'alphabet d'entrée appartient à A. Mais l'inverse n'est pas vérifiée.

Les circuits "sequentially self-checking" ont été définis en prenant en compte un ensemble de défauts F. Afin d'éviter le besoin d'une liste de fautes, la définition des circuits "sequentially self-checking" pour une classe générale d'hypothèses de pannes est donnée.

**Définition D46**
Un circuit est "sequentially self-checking" (SeSC) pour une classe Cf des hypothèses de pannes si, pour toutes les séquences de pannes $f_i$ appartenant à Cf qui peuvent survenir, pour tous les états $q \in Q$, pour toutes $i_2 \in I_q$,

soit

a) le circuit est à la fois SeST et SeFS pour $(f_i, q, i_2)$ ;

soit

b) le circuit est SeFS pour $(f_1, q, i_2)$ et : $\forall\, f_2 \in F$ et $\forall\, i_2 \in S(i_2)$ ($f_2$ survenant après $f_1$), soit la propriété a. ou la propriété b. est vérifiée pour la panne $f_1 \cup f_2$ prenant la place de $f_1$, $\delta^{II}(i_3, q)$ tel que $i_2 = i_3 \,.\, i_4$ prenant la place de q, et $i_4$ à la place de $i_2$.

Cette dernière définition peut être réduite pour la situation combinatoire, en utilisant les notations correspondantes données précédemment. Quand les circuits SeFS et SeST ont été définis, nous avions remarqué qu'il n'y avait pas une équivalence complète entre ces définitions et celles de circuits FS et SF, respectivement, si bien qu'elles peuvent être utilisées dans ce sens dès que les restrictions particulières à chaque cas sont respectées. Cette définition est aussi récursive comme D42 - donc,

d'autres fautes peuvent se produire si la machine est seulement SeFS pour les défauts anterieurs.

La définition D46 devient : "Un circuit est "strongly fault secure" (SFS) pour une classe Cf d'hypothèses de pannes si, pour toutes les séquences de pannes $f_i$ appartenant à Cf qui peuvent arriver, pour tout a c A,

soit

a. le circuit est à la fois ST et FS pour $f_i$ ;

soit

b. le circuit est seulement FS pour $f_i$ et cette faute n'est pas détectée; pour $\forall$ $f_2$ e F, $\forall$ a e A, soit la propriété a. ou soit la propriété b. est vérifiée : $\langle f_1, f_2 \rangle$ prenant la place de $f_1$, $G(a,\langle f_1, f_2 \rangle)$ prenant la place de $G(a,\langle f_1 \rangle)$, et toutes les entrées a e A étant appliquées."

Cette définition n'est pas présentée, dans un style égal à la définition originale SFS (D18), tout de même elle peut être considérée comme sa correspondante. Elles expriment des propriétés similaires, bien qu'en utilisant un formalisme différent. David et Thevenod-Fosse [DAV 78] avaient proposé une définition similaire pour les circuits combinatoires, d'ailleurs, ennoncé comme suit: "Un circuit est "strongly fault secure" pour un ensemble de fautes F si, pour chaque f en F, soit : a) le circuit est "totally self-checking"; soit b) le circuit est "fault secure" et si une nouvelle faute se produit, pour le défaut multiple obtenu, soit la propriété a. soit la propriété b. est vrai".

Pour des circuits séquentiels, la propriété TSC assure que la première sortie erronée de la machine M produit une séquence de sortie non appartenant au langage de sortie $O_M$.

L'hypothèse qui peut être utilisée afin d'accomplir le TSC goal, est énoncée ci-dessous.

**Hypothèse H8**

Entre l'occurrence de deux défauts quelconques appartenant à l'ensemble F ou à Cf, il s'écoule un laps de temps suffisant pour que, $\forall$ q e Q, $\forall$ $i_1$ tel que $i_1 . i_2$ e $I_M$ et $\delta(i_1,q_0) = q$, toutes les $i_{2m}$ pour chaque faute f ou séquence de défauts soient appliquées.

**Proposition P11**
L'hypothèse H8 étant assurée, un circuit "sequentially self-checking "(SeSC) accomplit le but de "totally self-checking goal".

**Demonstration de P11**

Initialement, il n'y a pas de faute dans le circuit, donc : $\forall$ $i_1$ tel que $\delta(i_1, q_0) = q$ et tel que $i_1 . i_j \in I_M$, $w(i_1, q_0) . w(i_j, q) \in O_M$. Aussi $\forall$ $i_j \in I_q$, $w(i_j,q)$ est correct. Ou, si des défauts ont déjà eu lieu, le circuit est redondant pour ces défauts et $\forall$ $i_j' \in P(i_j)$, $w^{<f1,f2,....,f\,m-1>}(i_j',q)= w(i_j',q)$.

Si une (nouvelle) faute $f_m$ se produit, l'hypothèse H8 assure que pour $\forall q \in Q$, pour $\forall$ $i_1$ tel que $i_1 . i_2 \in I_M$ et $\delta(i_1,q_0)=q$, toutes $i_{2m}$ pour chaque défaut f ou séquence de défauts seront appliquées avant l'occurrence d'autres fautes. Si des fautes n'ont pas encore eu lieu, soit m=1. Deux cas peuvent arriver :

a) Le circuit est SeST pour la séquence de défauts $\langle f_1,f_2,....,f_m\rangle$ et état ; donc $i_{jm} \in I_q$ étant appliquée, $w^{<f1,f2,....,fm>}(i_{jm},q) \notin O_M$, et la faute est détectée. Jusqu'à la détection, l'exactitude des sorties du circuit était assurée par la propriété SeFS.

b) Le circuit est seulement SeFS pour $\langle f_1,f_2,....,f_m\rangle$ et la séquence de défauts n'est pas détectée puisque $\forall$ $i_j' \in P(i_j)$, $w^{<f1,f2,....,fm>}(i_j',q) = w(i_j',q)$. Le circuit est redondant pour la séquence $\langle f_1,f_2,....,f_m\rangle$ et une nouvelle faute $f_{m+1}$ peut se produire. Dans ce cas, le comportement du circuit sera décrit soit pour l'item a., soit pour l'item b., selon les possibilités de détection. En vue de cette analyse, la nouvelle faute sera donnée par $\langle f_1,f_2,....,f_m,f_{m+1}\rangle$, $i_j^* \in S(i_j)$ prenant la place de $i_j$ et $\delta^{<f1,f2,....,fm>}(i_i^*,q)$ où $i_j= i_1^* . i_j^*$ prenant la place de q.

Q.E.D.

**Remark Rk2** - Remarque à l'hypothèse H8
Le but de l'hypothèse H8 est d'assurer la détection du premier défaut ou d'un autre nouveau pour lequel le circuit n'est pas redondant, avant qu'une nouvelle faute ait lieu, semblablement au but de l'hypothèse H1 pour les circuits combinatoires. Une autre façon de formuler l'hypothèse H8 est de contraindre l'apparition des Séquences de Transition Détectrices, selon les

définitions qui suivent.

Ces définitions sont prises de [DAV 79].

**Définition D47**
Soient $M_1$et $M_2$ deux machines séquentielles avec des ensembles d'entrées identiques. Un état $q_{i1}$ de $M_1$ et un état $q_{j2}$ de $M_2$ sont dits i-compatible ("input-compatible") si $M_1$ peut être dans l'état $q_{i1}$, en sachant que $M_2$ est dans l'état $q_{j2}$, et vice-versa, et en sachant qu'au moins un vecteur d'entrée à été appliqué aux deux machines.

Si une machine M sans panne devrait être à l'état $q_i \in Q$, la machine $M^f$ avec panne ne peut être à n'importe quel état de $Q^f$. $CS_i^f \subseteq Q^f$ indique l'ensemble des états i-compatible avec $q_i$.

**Définition D48**
Une séquence de transition TS est défini par un état $q_i$ et une séquence d'entrée i. Nous notons : TS = $q_i$ i la concaténation de deux séquences de transition $TS_1 = q_1 i_1$ et $TS_2 = q_2 i_2$ est définie, si $\delta(q_1,i_1) = q_2$, par $TS_1 . TS_2 = q_1 i_1 . q_2 i_2 = q_1 i_1 i_2$.

**Définition D49**
Une séquence de transition TS=$q_i$i est une Séquence de Transition Détectante (DTS) pour une faute f si la faute f est détectée quand TS est appliquée à $M^f$, quel que soit l'état $q_j^f \in CS_j^f$.

A partir de ces dernières définitions, l'hypothèse H8 pourra être formulée comme suit: "Entre l'occurrence de deux défauts quelconques appartenant à l'ensemble F ou à Cf, il s'écoule un laps de temps suffisant pour que, $\forall$ q $\in$ Q, $\forall$ $i_1$ tel que $i_1 . i_2 \in I_M$ et $\delta(i_1,q_0) = q$, si DTS existe, au moins une DTS pour chaque faute ou séquence de défauts est appliquée."

Nous remarquons que, quand $i_{2m}$ existe, $i_2 \in I_q$, une séquence q $i_{2m}$ peut être une DTS. Et aussi, que l'hypothèse H8 ne peut pas être formulée exactement comme l'hypothèse H1 puisque le nombre des séquences possibles appartenant à $I_M$ peut être infini. Cela n'est pas le cas pour les circuits combinatoires, où le nombre des vecteurs d'entrée possibles

corresponde aux vecteurs a ϵ A.

Dans la prochaine section, nous étudierons les contrôleurs à être utilisés par les systèmes séquentiels.

## IV.3. CONTROLEURS "STRONGLY CODE DISJOINT"

Les contrôleurs "strongly language disjoint" définis dans [JAN 84] sont différents des contrôleurs trouvés dans d'autres papiers concernant la conception des machines séquentielles autocontrolables. Les contrôleurs "strongly language disjoint" vérifient une propriété séquentielle, donc ils sont définis comme des contrôleurs séquentiels. Avant, des contrôleurs combinatoires étaient utilisés pour la conception des machines séquentielles autocontrolables, comme presenté dans la section précedante.

Au contraire des cas décrits précedement (dans la section IV.2), les contrôleurs "strongly language disjoint" (SLD) sont contrôleurs séquentiels, pour être utilisés associés à un bloc fonctionnel séquentiel, selon le schéma général représenté précédemment dans la figure 51. Par exemple, prenons un bloc fonctionnel lequel a une sortie à deux bits. Pendant le fonctionnement normal, la sortie correspond à l'alternance des valeurs contenant un nombre impair ou pair de zéros. Ce langage de sortie peut être formellement décrit par l'expression: $O_M=(00+11)(01+10)^*$. Une séquence sans faute peut être : 00, 01, 11, 10, 11, 10, ... Pour une sortie du circuit fonctionnel ainsi défini, le contrôleur doit être un circuit séquentiel.

La propriété "language disjoint" peut être introduite comme une généralisation de la propriété "code disjoint" [VIA 80].

**Définition D50**
Une machine M est "language disjoint" si :

$$\forall\, i \in I_M,\ w(i, q_0) \in O_M \text{ et}$$

$$\forall\, i \notin I_M,\ w(i, q_0) \notin O_M.$$

Ensuite, comme pour la présentation des systèmes combinatoires, le contrôleur est défini selon VIAUD - DAVID.

**Définition D51**
Un circuit est un contrôleur "sequentially self-checking" s'il est à la fois "sequentially self-checking" et "language disjoint".

Si comparée avec les définitions données pour les circuits combinatoires, cette définition est intermédiaire entre les contrôleurs "totally self-checking" et les contrôleurs "strongly code disjoint" (convenablement transposés à l'univers séquentiel) puisqu'elle considére la possibilité d'occurrence de fautes redondantes en vue de la propriété "self-testing", mais ne considère pas les mêmes fautes en fonction de la propriété "language disjoint". Plus précisément, une telle définition correspondrait à une de contrôleurs combinatoires SFS-CD. Nous avons déjà observé que la propriété "fault secure" n'est pas nécessaire aux contrôleurs tandis que la "code disjunction" devra être examinée en présence des fautes dans le contrôleur. Ainsi, les contrôleurs SLD sont définis [JAN 84b].

Dans les définitions suivantes, nous appelons respectivement $I_C$ et $O_C$ les langages d'entrée et de sortie du contrôleur, et w(i,q) la séquence de sortie obtenue à partir de q ϵ Q du contrôleur. Bien qu'un langage soit défini pour être obtenu aux sorties du contrôleur, ce langage peut être dégénéré dans un code de sortie (par exemple, un code double-rail). Dans la suite, la propriété séquentielle est considérée pour garder la généralité des définitions.

**Définition D52**
Un contrôleur C est redondant pour une faute f, un langage d'entrée $I_C$ et un langage de sortie $O_C$ si :

$$\forall\, i \in I_C\,,\ w^f(i,q) \in O_C \text{ et}$$
$$\forall\, i \notin I_C\,,\ w^f(i,q) \notin O_C.$$

La corrélation de cette définition avec sa correspondante donnée pour les contrôleurs combinatoires (D19) est immédiate. Nous ajoutons à la liste précédente, les substitutions qui suivent :

| | | |
|---|---|---|
| langage | ---> | code |
| $i \in I_C$ | ---> | $b \in B$ |
| $O_C$ | ---> | C |

et en faisant les remplacements nécessaires, nous obtenons la définition: "un contrôleur G est redondant pour une faute f et pour un code d'entrée B et pour un code de sortie C si : $\forall\, b \in B$, $G(b, f) \in C$ et $\forall\, b \notin B$, $G(b, f) \notin C$."

**Définition D53**
Un contrôleur est "strongly redundant" pour une séquence de défauts $\langle f_1, f_2, \ldots, f_n \rangle$, pour un langage d'entrée $I_C$ et pour un langage de sortie $O_C$ s'il est

redondant pour toutes les n sous-séquences de défauts $\langle f_1\rangle, \langle f_1,f_2\rangle, \langle f_1,f_2,f_3\rangle, \ldots\ldots, \langle f_1,f_2 \ldots\ldots,f_n\rangle$ et pour un langage d'entrée $I_C$ et pour un langage de sortie $O_C$.

Maintenant, il est possible d'énoncer les définitions analogues pour les contrôleurs SLD appartenant à un système séquentiel : le contrôleur "strongly language disjoint" (contrôleur SLD). Nous gardons le même style utilisé par [NIC 83] pour les circuits combinatoires - sont pris en compte d'abord une séquence de fautes, ensuite un ensemble de fautes et après une classe d'hypothèses de pannes.

**Définition D54**
Avant l'occurrence de défauts, le contrôleur est "language disjoint". Pour une séquence de défauts $\langle f_1,f_2,\ldots\ldots,f_n\rangle$, un état q, et une séquence d'entrée $i_2$ e $I_C$, soit k le plus petit entier pour lequel $\delta(i_1,q_0) = q$ et tel que :

$$i_1 . i_2 \in I_C : w(i_1,q_0) . w^{\langle f1,f2,\ldots,fk\rangle}(i_2,q) \notin O_C.$$

S'il n'existe pas un tel k, posons k = n+1. Alors C est "strongly language disjoint" (SLD) pour la séquence de fautes si :

$$\forall\, i \notin I_C, \forall\, m \in \{1,2,\ldots,k-1\}, w(i_1,q_0).w^{\langle f1,f2,\ldots,fm\rangle}(i,q) \notin O_C.$$

**Définition D55**
Le contrôleur C est "strongly language disjoint" (SLD) pour un ensemble de fautes F si C est SLD pour chaque séquence de défauts dont les éléments appartiennent à l'ensemble F.

**Définition D56**
Un contrôleur est SLD pour une classe Cf d'hypothèses de pannes s'il est "language disjoint" et si pour chaque séquence de défauts $f_i$ appartenant à la classe Cf :
soit il existe k tel que :

* le contrôleur est "strongly redundant" pour la séquence $\langle f_1, f_2,\ldots\ldots,f_{k-1}\rangle$ et ,
* $\forall\, i_2$ tel que $\delta(i_1,q_0)=q$ et tel que $i_1 . i_2 \in I_C$ : $w(i_1,q_0). w^{\langle f1,f2,\ldots\ldots,fk\rangle}(i_2,q) \notin O_C$ ;

soit

le contrôleur est "strongly redundant" pour toutes les séquences de défauts.

A partir de la définition ci-dessus, les substitutions nécessaires faites, il est possible de rejoindre la définition D23 donnée pour les circuits combinatoires : "un contrôleur est SCD pour une classe Cf d'hypothèses de pannes s'il est "code disjoint" et si, pour chaque séquence de défauts $f_1$ appartenant à la classe Cf :
soit il existe k tel que :

* le contrôleur est "strongly redundant" pour la séquence $\langle f_1, f_2, \ldots, f_{k-1} \rangle$ et,
* $\exists$ b $\in$ B tel que G (b, $\langle f_1, f_2, \ldots, f_k \rangle$) $\notin$ C ;

soit

* le contrôleur est "strongly redundant" pour toutes les séquences de défauts".

Afin de composer un système à partir d'un bloc fonctionnel et d'un contrôleur, une supposition concernant l'occurrence de pannes dans chacun de ces composants, doit être faite. Cela sera l'hypothèse correspondante à H2, utilisée aux systèmes combinatoires.

**Hypothèse H9**
Entre l'occurrence de deux défauts quelconques affectant le bloc fonctionnel, il s'écoule un laps de temps suffisant pour que, $\forall$ q $\in$ Q, $\forall$ $i_1$ tel que $i_1 \cdot i_2 \in I_M$ et $\delta(i_1, q_0) = q$, toutes les $i_{2m}$ pour chaque faute f ou séquences de défauts qui peuvent se produire dans le bloc fonctionnel soient appliquées à ce bloc, avant qu'un deuxième défaut survienne à lui même ou au contrôleur. Entre l'occurrence de deux défauts quelconques affectant le contrôleur, il s'écoule un laps de temps suffisant pour que, $\forall$ q $\in$ Q, $\forall$ $i_1$ tel que $i_1 \cdot i_2 \in I_C$ et $\delta(i_1, q_0) = q$, toutes les $i_{2m}$ pour chaque faute f ou séquences de défauts qui peuvent se produire dans le contrôleur soient appliquées à ce bloc, avant qu'un deuxième défaut survienne au bloc fonctionnel ou à lui même.

**Proposition P12**
L'hypothèse H9 étant assurée, un système composé de :

- un circuit SeSC,
- un contrôleur SLD

accomplit le but de "totally self-checking goal".

**Démonstration de P12**

Initialement, il n'y a pas de faute dans le système. Conformément à l'hypothèse H9, deux cas peuvent arriver : soit une faute peut apparaître

dans le contrôleur soit dans le bloc fonctionnel, mais pas dans les deux blocs simultanément.

Cas 1:

Une faute $f_m$ se produit dans le bloc fonctionnel. Pour chaque $q \in Q$, $\forall$ $i_1$ tel que $i_1 . i_2 \in I_M$ et $\delta(i_i,q_0) = q$, toutes $i_{2m}$ pour chaque faute sont appliquées avant l'occurrence d'autre défaut dans ce bloc ou dans le contrôleur.

a) Le circuit est SeST pour la séquence de défauts $\langle f_1, f_2,.....,f_m \rangle$ et pour l'état ; donc $i_j \in I_M$ étant appliquée, $w(i_i,q_0) . w^{\langle f1,f2,.....,fm \rangle}(i_j,q) \notin O_M$, et la faute est détectée. Jusqu'à la détection, l'exactitude des sorties du circuit était assurée par la propriété SeFS.

b) le circuit est seulement SeFS pour $\langle f_1,f_2, ....,f_m \rangle$, pour l'état et pour la séquence d'entrée, et la séquence de défauts n'est pas détectée puisque $\forall$ $i_j' \in P(i_j)$ , $w^{\langle f1,f2,.....,fm \rangle}(i_j',q)=w(i_j',q)$. Le circuit est redondant pour la séquence $\langle f_1,f_2,.....,f_m \rangle$ et une nouvelle faute peut se produire, dans le bloc fonctionnel ou dans le contrôleur. En vue de cette analyse, il faut retourner au début de la démonstration. La nouvelle faute, si placée dans le bloc fonctionnel, sera donnée par $\langle f_1,f_2,.....,f_m,f_{m+1} \rangle$, $i_j^* \in S(i_j)$ prend la place de $i_j$ et $\delta^{\langle f1,f2,.....,fm \rangle}(i_j^*$, q) où $i_j = i_1^* . i_j^*$, prend la place de q. Si la nouvelle faute affecte le contrôleur, la situation est décrite par le prochain cas.

Cas 2:

Une faute $f_n'$ se produit dans le contrôleur. C'est assurée l'application de toutes les séquences $i_e : i_k . i_e \in I_C$ qui peuvent détecter des défauts dans le contrôleur, avant l'occurrence d'autres défauts dans le contrôleur ou dans le bloc fonctionnel. Les séquences d'entrée du contrôleur sont produites par les séquences appliquées au bloc fonctionnel, mais la longueur de ces séquences peut être plus courte que celle des séquences appliquées au bloc fonctionnel. D'après la définition D56, nous savons que :

a) il est appliqué $i_k$ tel que $\delta(i_k,q_0) = q$ et $i_k . i_e \in I_C$: $w(i_k,q_0) . w^{\langle f1',f2',....,fn' \rangle}(i_1,q) \notin Oc$, et le défaut est détecté; ou

b) le contrôleur est "strongly redundant" pour la séquence de défauts $\langle f_1', f_2', \ldots, f_n' \rangle$ et une autre faute peut se produire, dans le bloc fonctionnel ou dans le contrôleur. Dans ce cas, l'analyse est reprise du début de cette démonstration. Si le nouveau défaut affecte le controleur, il sera donné par $\langle f_1', f_2', \ldots, f_n', f_{n+1}' \rangle$, $i_e' \in S(i_e)$ prend la place de $i_e$ et $\delta^{\langle f1', f2', \ldots, fn' \rangle}(i_k^*, q)$, où $i_e = i_k^* . i_e^*$, prend la place de q.

Q.E.D.

Nous pouvons vérifier les propriétés des contrôleurs SLD à partir de leur définition :

*"Strongly language disjoint" - Tous les contrôleurs, par définition, sont "language disjoint". Mais, comme nous avons considéré la possibilité d'occurrence de fautes pour lesquelles le circuit est redondant, cette propriété a été étendue à la "strongly language disjunction", qu'assure la correspondance des séquences d'entrée appartenant au langage d'entrée à des séquences de sortie appartenant au langage de sortie, et aussi des séquences d'entrée non appartenant au langage d'entrée à des séquences de sortie non appartenant au langage de sortie, même en présence de défauts. La propriété SLD assure aussi la détection de la première faute pour lequel le contrôleur n'est pas redondant avec l'application des séquences définies par le langage d'entrée.

*"Sequentially self-checking" - Cette propriété n'est pas vérifié mais le but TSC du système est conservé, une fois que les défauts pour lesquels le circuit est redondant restent non détectés. Il a été remarqué, ci-dessus, que la propriété SLD assure la détection du premier défaut pour lequel le contrôleur n'est pas redondant, avec l'application d'une séquence appartenant au langage d'entrée.

*"Sequentially fault-secure" - En général, cette propriété n'est pas nécessaire aux contrôleurs parce que les valeurs de sortie ne seront pas utilisés subséquemment. Il faut seulement rendre sûr que les sorties appartenant au langage de sortie ne résultent que de l'application de bonnes entrées (i. e., valeurs appartenant au langage d'entrée) e que des entrées non définies par le langage d'entrée produiront toujours des sorties non appartenant au langage de sortie; mais ce comportement est déjà assuré par la propriété SLD.

Alors, la seule propriété nécessaire aux contrôleurs SLD est la "strongly language disjunction".

Des définitions plus fortes peuvent être données pour les contrôleurs SLD, qui constituent la généralisation des définitions fortes pour les contrôleurs SCD [NIC 83].

**Définition D57**
Avant l'occurrence de défauts, le contrôleur C est "language disjoint". Pour une séquence de défauts $\langle f_1,f_2,...,f_n\rangle$, pour un état q, et pour une séquence d'entrée $i_2 \in I_q$, soit k le plus petit entier pour lequel $\delta(i_1,q_0) = q$ et tel que:

$$i_1 . i_2 \in I_C : w(i_1, q_0) . w^{\langle f1,f2,...,fk\rangle}(i_2, q) \notin O_C.$$

S'il n'existe pas un tel k, posons k = n. Alors C est "strongly language disjoint" (SLD) pour la séquence de défauts si :

$$\forall i_2 | i_1 . i_2 \notin I_C, \forall m \in (1,2,...,k),\ w(i_1,q_0) . w^{\langle f1,f2,...,fk\rangle}(i_2,q) \notin O_M.$$

Cette dernière définition peut être étendue afin de caractériser un contrôleur SLD pour une classe de défauts, dans le même style utilisé par la définition D26.

**Définition D58**
Un contrôleur est SLD pour une classe Cf d'hypothèses de pannes s'il est "language disjoint" et si pour chaque séquence de fautes $f_i$ appartenant à Cf qui peuvent survenir, pour un état q, et pour une séquence d'entrée $i_2 \in I_q$,
soit il existe k tel que :

* le contrôleur est "strongly redundant" pour la séquence de défauts $\langle f_1,f_2,.....,f_{k-1}\rangle$, et
* $\forall i_1 . i_2 \notin I_C : w(i_1, q_0) . w^{\langle f1,f2,.....,fk\rangle}(i_2, q) \notin O_C$ ;

soit

* le contrôleur est "strongly redundant" pour les séquences de défauts.

Cette dernière définition (D58) est plus restrictive que la D56, donnée aussi pour les contrôleurs SLD, prennant en compte une classe Cf d'hypothèses de pannes. La définition D56 rend sûr le "code disjunction" en présence des fautes pour lesquelles le circuit était redondant aussi bien que la détection de la première pour laquelle le circuit n'était pas redondant ; par la définition D58, il est assuré que, même si un défaut pour lequel le circuit n'est pas redondant survient, la propriété "language disjoint" est gardée. Cette propriété n'est pas réellement nécessaire aux contrôleurs, mais elle peut nous amener à utiliser un système séquentiel sous une hypothèse moins forte que H9, et qui accompli aussi le "totally

self-checking goal".

**Hypothèse H10**
Entre l'occurrence de deux défauts quelconques affectant le bloc fonctionnel, ou entre l'occurrence de deux défauts quelconques affectant le contrôleur, il s'écoule un laps de temps suffisant pour que, $\forall$ q $\epsilon$ Q, $\forall$ $i_1$ tel que $i_1 \cdot i_2 \epsilon I_M$ et $\delta(i_1, q_0) = q$, toutes les $i_{2m}$ pour chaque faute f ou séquence de défauts qui peuvent survenir au système séquentiel soient appliquées au bloc fonctionnel.

Cette hypothèse est celle correspondant à l'H3 pour les circuits séquentiels puisqu'on peut admettre l'occurrence de défauts dans le bloc fonctionnel (ou dans le contrôleur) avant que la faute produite dans le contrôleur (ou dans le bloc fonctionnel) ait été détectée.

**Proposition P13**
L'hypothèse H10 étant assurée, un système composé de :
- un circuit SeSC,
- un contrôleur SLD (défini par D58),

accomplit le "totally self-checking goal".

**Démonstration de P13**
Initialement, il n'y a pas de faute dans le système ; donc pour le bloc fonctionnel et pour des séquences d'entrée $i_j$ : $i_i \cdot i_j \epsilon I_M$, $w_M(i_1,q_0) \cdot w_M(i_j,q) \epsilon O_M$, et dans le contrôleur pour des séquences d'entrée $i_e$ : $i_k \cdot i_e \epsilon I_C$, $w_C(i_k,q_0) \cdot w_C(i_e,q) \epsilon O_C$. La séquence d'entrée du contrôleur ($\epsilon I_C$) est la séquence de sortie du bloc fonctionnel ($\epsilon O_M$).

Une faute peut se produire , affectant soit le bloc fonctionnel soit le contrôleur. Ou une faute peut survenir à un de ces blocs avant q'une autre déjà arrivée dans l'autre bloc ait été détectée.

Cas 1:

Une faute $f_m$ se produit dans le bloc fonctionnel. D'autres défauts n'étant pas encore arrivés, posons m=1. Il est assuré l'application des séquences $i_j \epsilon P(i_j)$ avant l'occurrence d'une autre faute dans ce bloc. Deux cas peuvent arriver :

a) le circuit est SeST pour cette séquence de défauts $\langle f_1,f_2,....,f_n\rangle$ et pour l'état ; étant appliquée $i_j \in I_q$, $w(i_1,q_0)$. $w^{\langle f1,f2,....,fm\rangle}(i_j, q) \notin O_M$ , donc le défaut est détecté. Jusqu'à ce moment, l'exactitude des sorties du circuit était garantie par la propriété SeSF ;

b) le circuit est seulement SeFS pour $\langle f_1,f_2,....,f_m\rangle$, pour l'état et pour la séquence d'entrée, et la séquence de défauts n'est pas détectée puisque $\forall\ i_j' \in P(i_j')$ , $w^{\langle f1,f2,....,fm\rangle}(i_j',q) = w(i_j',q)$. Le circuit est redondant pour la séquence $\langle f_1,f_2,....,f_m\rangle$ et une nouvelle faute peut se produire dans le bloc fonctionnel ou dans le contrôleur. En vue de cette analyse, il faut retourner au début de cette démonstration. La nouvelle faute étant placée dans le bloc fonctionnel, elle sera donnée par $\langle f_1,f_2,....,f_m,f_{m+1}\rangle$ , $i_j^* \in S(i_j)$ prenant la place de $i_j$ et $\delta^{\langle f1,f2,....,fm\rangle}(i_j^*,q)$, ou $i_j$-$i_j^*$. $i_j^*$, prenant la place de q.

Cas 2:

Une faute $f_n'$ se produit dans le contrôleur. D'autres défauts n'étant pas encore arrivés, posons n=1. C'est assurée l'application de toutes les séquences $i_e \in P(i_e)$ au contrôleur, ces séquences étant capables de détecter les fautes de ce bloc, avant l'occurrence d'autre défaut affectant le contrôleur ou le bloc fonctionnel. D'après la définition D58, nous savons :

a) il est appliqué $i_k$ tel que $\delta(i_k,q_0) = q$ et $i_k$ . $i_e \in I_C$ : $w(i_k,q_0)$ . $w^{\langle f1',f2',....fn'\rangle}(i_e , q) \notin O_C$ , et le défaut est détecté ; ou

b) le contrôleur est "strongly redundant" pour la séquence de défauts $\langle f_1',f_2',.....,f_n'\rangle$ et une autre faute peut se produire, dans le bloc fonctionnel ou dans le contrôleur. Dans ce cas, l'analyse est reprise dès le début de cette démonstration. Si le nouveau défaut affecte le contrôleur, il sera donné par $\langle f_1',f_2',.....,f'_{n+1}\rangle$ , $i_e^* \in S(i_e)$ prenant la place de $i_e$ et $\delta^{\langle f1',f2',....fn'\rangle}(i_k^*,q)$ où $i_e$ - $i_k^*$. $i_e^*$ prenant la place de q.

Cas 3:

Une faute survient au contrôleur avant qu'une autre déjà arrivée au bloc fonctionnel ait été détectée ou une faute survient au bloc fonctionnel avant qu'une autre déjà arrivée au contrôleur ait été détectée. D'après

l'hypothèse H10, les séquences d'entrée $i_{2m} \in P(i_2)$ sont appliquées au bloc fonctionnel avant l'occurrence d'une autre faute dans le même bloc où s'était produite la faute précédente. Selon les définitions, quatre situations peuvent arriver:

a) le bloc fonctionnel est SeFS et le contrôleur est "strongly redundant" pour les séquences de défauts. Donc, pour toutes les séquences de défauts appartenant à la classe définie,

$$\forall i_j : i_i \,.\, i_j \in I_M,\ w_M(i_i,q_0) \,.\, w_M^{\langle f1,f2,.....,fm\rangle}(i_j,q) \in O_M \text{ et}$$
$$\forall i_e : i_k \,.\, i_e \in I_C \,.\, w_C(i_k,q_0) \,.\, w_C^{\langle f1',f2',.....,fn'\rangle}(i_j,q) \in O_C .$$

Alors le système ne produira pas de séquences de sortie non appartenant au langage de sortie, mais des séquences de sortie fausses ne seront pas générées.

b) le contrôleur est "strongly redundant" pour les séquences de défauts mais le bloc fonctionnel n'en est pas. Alors, nous revenons au cas 1.

c) le bloc fonctionnel est "sequentially fault secure" pour la séquence de défauts mais le contrôleur n'en est pas. Alors, on revient au cas 2.

d) le bloc fonctionnel et le contrôleur ne sont pas redondants pour les séquences de défauts. Une faute peut survenir soit, d'abord au bloc fonctionnel soit, d'abord au contrôleur. Nous analysons les situations possibles. Les figures étant pareilles à celles présentées dans la démonstration de la proposition P1, sont omises.

d' : une faute survient initialement dans le bloc fonctionnel. Une deuxième se produit dans le contrôleur avant que les séquences $i_j \in P(i_j)$ aient été appliquées (sinon, le cas 1 peut être appliqué).

Soit $i_{jm}$ le préfixe le plus petit de $i_j$ pour lequel une séquence de sortie non appartenant au langage de sortie est produite au bloc fonctionnel, en présence du défaut. Les deux situations décrites ci-dessous sont possibles (d'1 et d'2).

d'1 : $i_{jm}$ est appliquée avant l'occurrence du défaut dans le contrôleur. Dans ce cas, une séquence non appartenant au langage d'entrée du contrôleur sera produite à cause de $f_m$, et sa détection est assurée par la propriété SLD (selon les définitions forte ou faible).

d'2 : $i_{jm}$ est appliquée après l'occurrence du défaut dans le contrôleur. Le bloc fonctionnel est SeFS pour la séquence de défauts $\langle f_1, f_2, \ldots, f_{m-1} \rangle$, et pour $i_{jm} \in I_M$ , $w_M(i_{jm}, q_0)$ . $w_M{}^{\langle f1, f2, \ldots, fm \rangle}(i_j, q) \notin O_M$, lequel garanti la détection de la séquence de défauts. Le contrôleur, où la faute n'était pas encore détecté, et qui reçoit les sorties non appartenant à $O_M$ (donc $i_e \notin I_C$), est "strongly redundant" pour la séquence de défauts $\langle f_1', f_2', \ldots, f'_n \rangle$ et $\forall i_k$ . $i_e \notin I_C$, $w(i_k, q_0)$ . $w^{\langle f1', f2', \ldots, fn' \rangle}(i_e, q) \notin O_C$.

Alors, la production d'une sortie non appartenant au langage de sortie du contrôleur permettant la détection du défaut, est un résultat de la détection du défaut dans le bloc fonctionnel et de sa propriété "language disjoint", même en face d'une séquence de fautes pour laquelle le contrôleur n'est pas redondant.

d" : Une faute se produit initialement dans le contrôleur. Une deuxième survient au bloc fonctionnel avant que la séquence $i_j \in P(i_j)$ ait été appliquée (sinon, le cas 2 peut être appliqué).

En fonctionnement normal, l'application d'une séquence $i_j \in P(i_j)$ au bloc fonctionnel résulte en une séquence $i_e \in P(i_e)$ aux entrées du contrôleur. Soit $i_{em}$ le plus petit préfixe de $i_e$ pour lequel une séquence de sortie non appartenant au langage de sortie est produite par le contrôleur, en présence de défaut. Nous remarquons que $i_{em}$ n'a pas de rapport direct avec $i_{jm}$, mais ils peuvent éventuellement avoir la même longueur. Deux situations sont possibles (d"1 et d"2).

d"1: $i_{em}$ survient avant qu'une faute apparaisse dans le bloc fonctionnel. Donc, une séquence non définie par le langage de sortie est générée aux sorties du contrôleur, et permet la détection du défaut $f_n'$ affectant le contrôleur.

d"2 : $i_{em}$ est appliquée après la survenance du défaut dans le bloc fonctionnel. Donc, $i_{em}$ n'est pas nécessairement générée. Le bloc fonctionnel étant SeFS, produit une séquence de sortie égale à celle qui serait générée en fonctionnement normal ou elle n'appartient pas au langage de sortie. Si la sortie du circuit fonctionnel est celle du fonctionnement normal et $i_{em}$

arrive, alors $w(i_k,q_0)$ . $w^{<f1',f2',....,fn'>}(i_{em},q) \notin O_C$ - le défaut $f_n'$ du contrôleur est détecté avec n'importe quelles définitions SLD.

Si $i_{jm}$ survient avant, le bloc fonctionnel produit une séquence de sortie non appartenant au langage de sortie, et la propriété SLD forte est nécessaire au contrôleur afin d'assurer que lui aussi produise une séquence de sortie non appartenant au langage de sortie.

Q.E.D.

Le rapport parmi les différents contrôleurs, obtenu à partir des définitions proposées, est présenté dans la figure 53. Dans ce schéma, on peut remarquer que les contrôleurs "sequentially self-checking" (SeSC) et aussi "sequentially self-testing" (SeST) (partiellement décrits par l'item a. de la définition D46), forment le groupe le plus restrictif. Ensuite, nous avons le groupe des contrôleurs SeSC mais qui ne sont pas SeST (item b. de la définition D46). Les contrôleurs définis par D58 peuvent être SeST et/ou SeFS ou seulement "strongly language disjoint", même en présence de défauts pour lesquels le circuit n'est pas redondant. C'est pourquoi ils sont un sous-groupe des contrôleurs SLD décrits par D56, qui est la plus large classe de contrôleurs.

Tous les contrôleurs définis pour des circuits combinatoires et séquentiels, avec les propriétés concernants chacun, sont représentés dans la figure 54. Les groupes combinatoires sont dessinés avec des traits fins. Aucun des ensembles séquentiels n'est totalement inclus par un autre parmi des ensembles combinatoires, à cause de la propriété "language disjoint" (qui est plus large que la propriété "code disjoint").

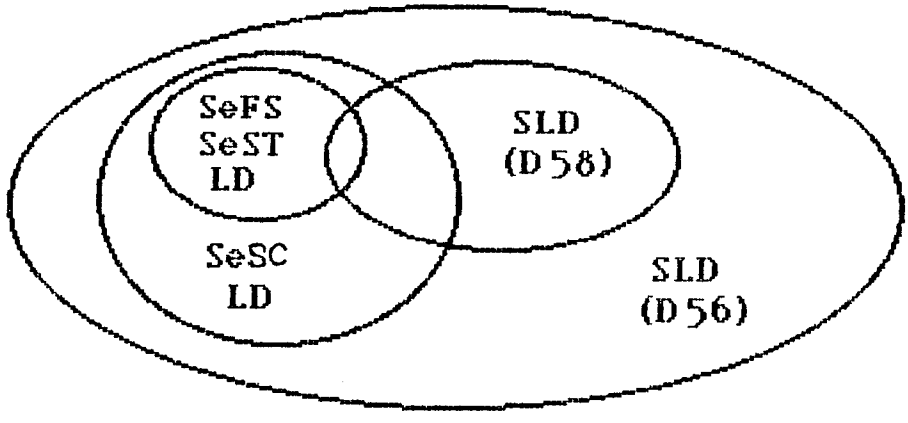



Figure 53 : Les contrôleurs dans l'univers séquentiel

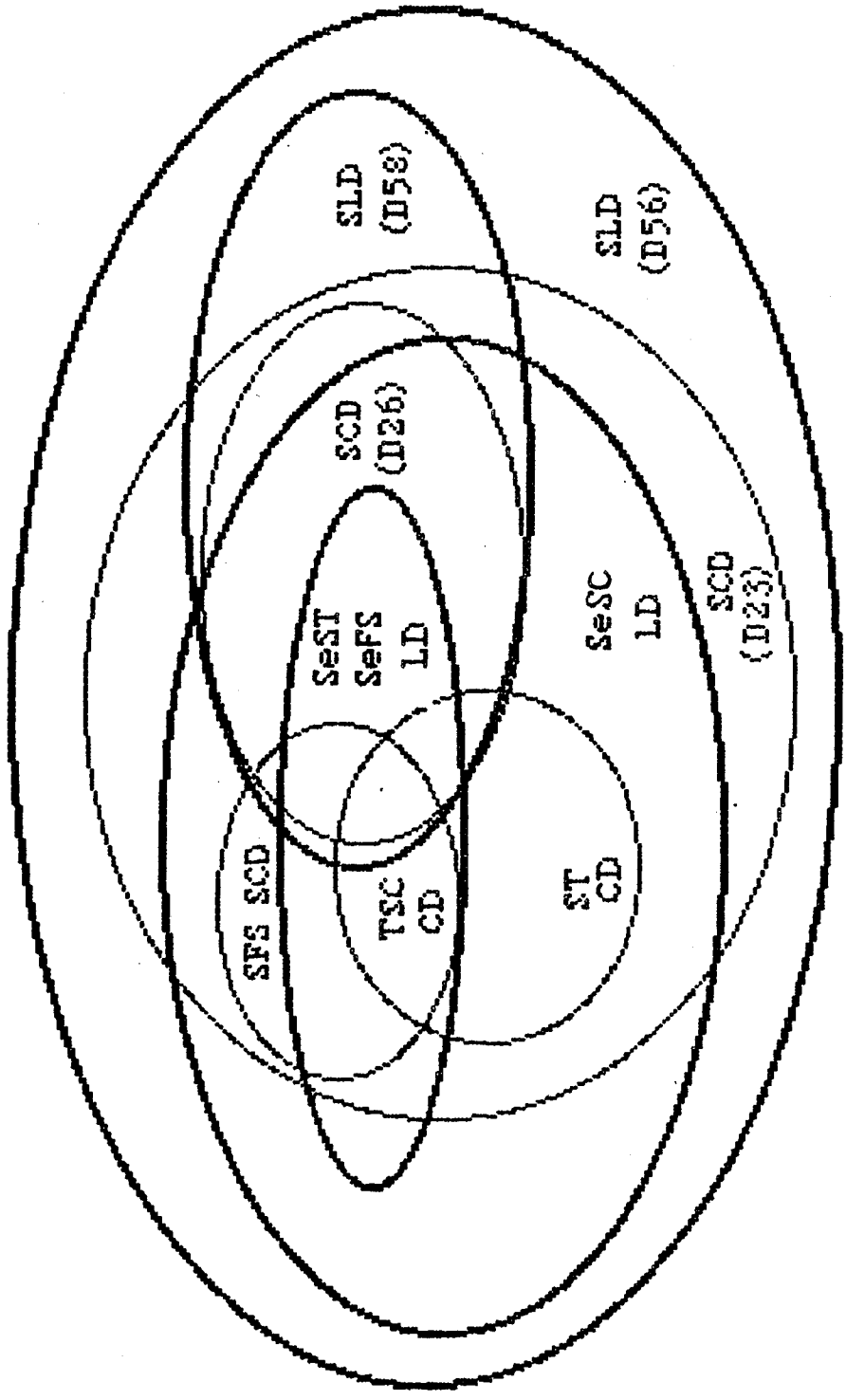



Figure 54 : L'univers des contrôleurs

## IV.4. CONCEPTION DES CONTROLEURS SLD

### IV.4.1. Langages d'entrée pour des contrôleurs séquentiels

A notre connaissance, il n'y a pas encore de langages classiques définis pour des contrôleurs séquentiels. Apparemment, le type de langage le plus convenable pour cette application est le type 3, sous lequel sont groupés celles générées par des grammaires régulières. Les productions d'une grammaire régulière linéaire à droite (par exemple) peuvent être listées par: A --> aB, A --> a et $\Sigma$ --> $\lambda$ , où A et B sont des symboles non-terminaux, a est un symbole terminal, $\Sigma$ est un symbole de phrase et $\lambda$ est la chaîne vide. Si L est un langage en un vocabulaire V et généré par une grammaire régulière, elle peut être décrite par une expression régulière et reconnue par un accepteur d'états finis. Cet accepteur d'états finis peut correspondre à une machine de Mealy ou de Moore.

Des accepteurs d'états finis non-déterministes qui correspondent à des langages réguliers (ou ensemble régulier, ou langage d'états finis) peuvent être facilement conçus. Donc, il est intéressant de connaître les trois théorèmes ci-dessous, qui sont repris de [DEN 78] :

**Théorème T1:** Pour chaque accepteur d'états finis $M_n$, on peut construire un accepteur d'états finis déterministe $M_d$ tel que $L(M_d) = L(M_n)$ .

**Théorème T2:** Pour toute grammaire linéaire à droite G, on peut construire un accepteur d'états finis M tel que L(M) = L(G).

**Théorème T3:** Chaque accepteur d'états finis reconnaît un langage qui peut être décrit par une expression régulière.

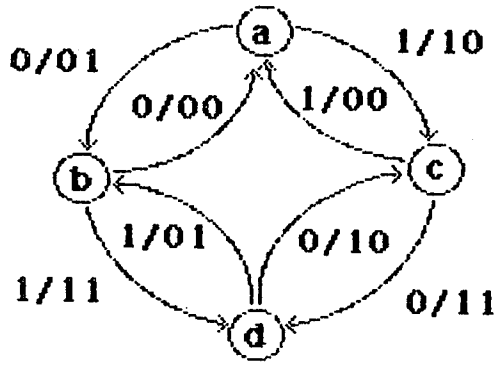
A partir de ces théorèmes, nous pouvons conclure qu'il est toujours possible de concevoir un accepteur d'états finis déterministe correspondant à un langage linéaire. Prenant en compte les circuits "sequentially self checking", le prochain problème à résoudre est la conception d'un contrôleur SLD pour ce langage régulier.

### IV.4.2. Définition d'un langage et de son accepteur

Viaud et David, en [VIA 80], utilisent comme exemple un langage de sortie obtenu d'un circuit fonctionnel, pour l'analyse duquel il faut un

contrôleur avec des propriétés séquentielles. Dans cette référence, la solution est développée jusqu'au schéma fonctionnel général avec les tableaux d'états respectifs. Nous utilisons ce même exemple et étudierons les possibilités d'implémentation physique interne aux blocs. La machine est définie comme suit.

Soit $\partial$ la séquence de symboles dans un vocabulaire binaire {0,1}. Nous considérons une machine initialement sans faute $M_1$, qui reçoit des séquences d'entrée définies comme $\partial$ et, dans le fonctionnement normal, produit une chaîne de sortie constituée par des vecteurs avec des nombres pairs ou impairs de zéros alternés, i.e., $O_{M1} = ((00+11)(01+10))^*$. Cette chaîne nommée $O_{M1}$, sortie d'un circuit fonctionnel, est le langage d'entrée d'un contrôleur "strongly language disjoint". Le contrôleur vérifie la correction de la séquence et produit des vecteurs de sortie appartenant à un code double-rail ; donc, $O_C$ = {10,01} s'il n'y a pas de faute dans la machine $M_1$. Dans le cas contraire, s'il y a une faute dans la machine $M_1$ ou même dans le contrôleur, des sorties $O_C$ = {00,11} seront produites. La machine $M_1$ peut être représentée par l'automate et le sixtuplet ci-dessous (figure 55).




$M_1 = (Q_1, X_1, Z_1, \delta_1, w_1, q_0)$

où

$Q_1 = \{ a,b,c,d \}$

$X_1 = \{ 0,1 \}$

$Z_1 = \{ 00,01,10,11 \}$

$q_0 = a$

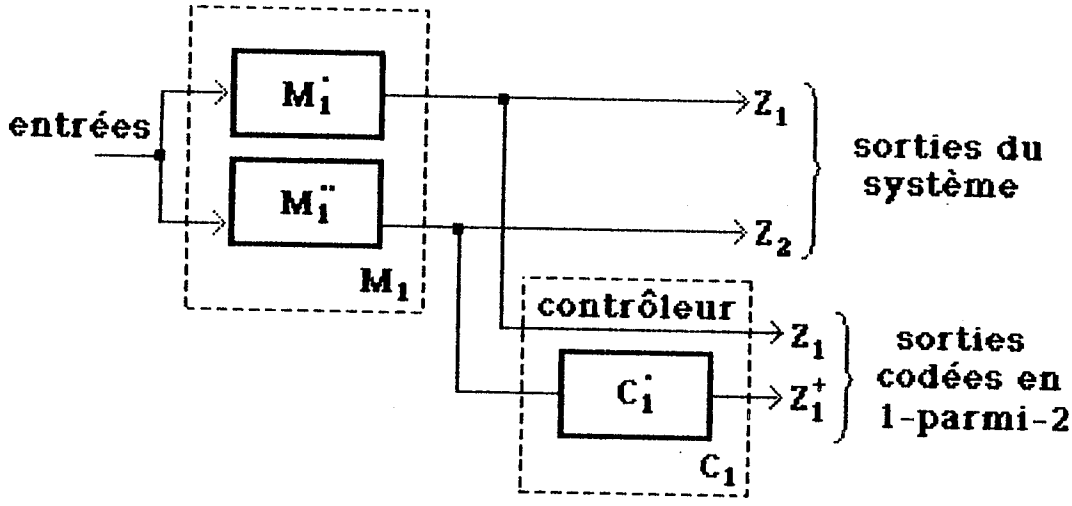
| $\delta(i,q)$ | a | b | c | d |
|---|---|---|---|---|
| 0 | b | a | d | c |
| 1 | c | d | a | b |

| w(i,q) | a | b | c | d |
|---|---|---|---|---|
| 0 | 01 | 00 | 11 | 10 |
| 1 | 10 | 11 | 00 | 01 |

Figure 55 : Automate et sixtuplet correspondants à la machine proposée

La solution de [VIA 80] propose des machines parallèles pour le bloc fonctionnel : chacune produit une sortie indépendante. Ces sorties sont

processées partiallement par le contrôleur, qui donne les sorties générales combinatoires codées en 1-parmi-2. Une telle structure est utilisée sous l'hypothèse d'occurrence de défauts mutuellement-exclusive parmi les blocs considérés (chaque machine parallèle composante + le contrôleur). Le schéma général suggéré par Viaud et David se trouve dans la figure 56.




a) la machine de la figure 55 et son contrôleur

| $\delta_{M_i'}$ | ab | cd |
|---|---|---|
| 0 | ab | cd |
| 1 | cd | ab |

| $\delta_{M_i''}$ | ac | bd |
|---|---|---|
| 0 | bd | ac |
| 1 | ac | bd |

| $\delta_{C_i}$ | ab | cd |
|---|---|---|
| 0 | $\beta$ | $\alpha$ |
| 1 | $\beta$ | $\alpha$ |

| $Z_1 = w_{M_i'}$ | ab | cd |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 0 |

| $Z_2 = w_{M_i''}$ | ac | bd |
|---|---|---|
| 0 | 1 | 0 |
| 1 | 0 | 1 |

| $Z_1^+ = w_{C_i}$ | $\alpha$ | $\beta$ |
|---|---|---|
| 0 | 1 | 0 |
| 1 | 0 | 1 |

b) tableaux de prochain état et sortie correspondent aux sous-machines de $M_i$ et au contrôleur

Figure 56: Description du système basé sur des machines parallèles

On peut regarder avec intérêt le système proposé ci-dessus, car il ne doit pas résulter en une augmentation trop importante par les circuits du

contrôleur, en principe. Nous remarquons qu'il s'agit d'une machine de deux états. L'utilisation de cette proposition suppose qu'une faute n' affecte pas à la fois ni les deux machines $M_1'$ et $M_1''$ ni une des machines et le contrôleur. Cette hypothèse de pannes peut être associée à H9 déjà formulée, mais une condition additionnelle se fait nécessaire : les défauts qui apparaissent dans la machine $M_1$ ne devront pas modifier les sorties de plus d'une des sous-machines (comme $M_1'$ et $M_1''$, par exemple). Il faut assurer, par des règles, que des erreurs n'affecteront pas, à la fois, les sorties des deux sous-machines. L'analyse des situations de défaut entre les différentes blocs peut être faite selon une procédure analogue à celle déroulée par l'assemblage de cellules des contrôleurs combinatoires (Section III.2.). On peut remarquer, aussi que la période nécessaire pour la détection de défauts dans les deux approches est à peu près la même (avec des machines composées ou avec une seule machine) puisque on utilise dans les deux cas des ensembles d'entrée identiques. Donc, les mêmes hypothèses sont considérées, indépendamment de la structure interne.

### IV.4.3. Considérations sur l'implémentation physique

La machine suggérée ci-dessus aussi bien que d'autres qui respectent la définition des contrôleurs "strongly language disjoint", pourrait être conçue à partir d'une proposition logique traditionnelle (i.e. en utilisant des flip-flops et de la logique combinatoire, ou d'autres approches), ou à partir des structures régulières, comme des PLA's et registres, par exemple.

Mais d'une façon générale, les "flip-flops" ne semblent pas être convenables pour la conception des contrôleurs SLD. Quelques types de "flip-flops" (comme les S-R et J-K) sont inadéquats, car des combinaisons d'entrée avec des valeurs indéterminées ("don't care") sont autorisées ; cette condition peut cacher des collages ou des fautes dans le langage d'entrée, par exemple. La sûreté de propagation de certains défauts aux sorties des "flip-flops" est aussi un problème. En plus de ces inconvénients, les "flip-flops" sont des cellules de mémoire qui utilisent un nombre important de transistors et nécessitent une logique de programmation associée. Donc, ils ne semblent pas être suffisamment intéressants pour une analyse plus détaillée.

Une autre approche traditionnelle en circuits séquentiels est celle de circuits asynchrones, éxécuté à partir des portes logiques simples et conçus par le mode fondamental (en éliminant les courses). Donc, nous supposons qu'une table primitive de flux est construite, avec le graphique de flux

correspondant, et que les états sont stables.

L'emploi de la cellule universelle - CUSA (Cellule Universelle pour Séquences Asynchrones) [DAV 77] comme bloc de base pour la conception du circuit du dernier paragraphe pourrait nous mener à une solution pratique puisque cette cellule peut être programmée. Mais cette cellule, du point-de-vue logique, n'est pas ni "self-testing" ni "code disjoint". Donc une autre méthode doit être utilisée.

La conception du circuit complet, à partir des tables de transition, peut conduire à une solution correcte mais probablement très épuisante du point-de-vue du concepteur car :

a) à chaque nouveau langage d'entrée (qui est dépendant du circuit SeSC) correspond un nouveau circuit du contrôleur ;
b) ces nouveaux dessins peuvent demander un nombre important de transistors et l'analyse des propriétés SLD devient difficile et longue à faire.

## IV.4.4. Conception à base de structures régulières

### IV.4.4.1. Structure interne et propriétés

Le contrôleur SLD, en étant un circuit séquentiel, peut être décomposé dans des blocs combinatoires et de mémorisation, comme présenté à la figure 57.

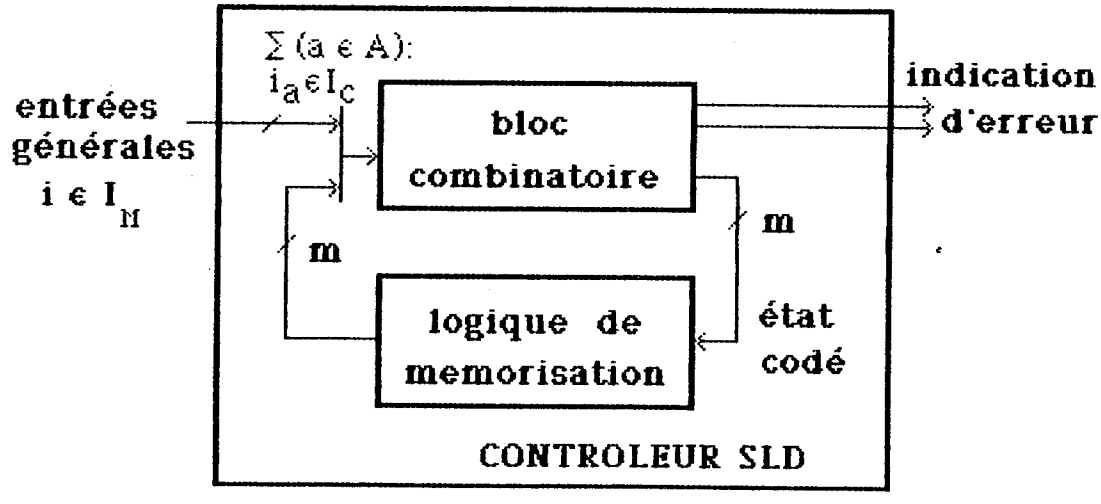



Figure 57 : Structure de base du système séquentiel

Selon les définitions, les sorties du contrôleur appartiennent à un langage de sortie. Ce langage peut être restreint en un code double-rail. En étant sorties d'une machine séquentielle, elles sont calculées en fonction des états et des entrées (dans le cas d'un modèle de Moore, elles sont uniquement fonction des états). Dans la suite, nous considérerons ce système particulier avec une paire de sorties codées en double-rail.

Les entrées du contrôleur sont définies par un langage et viennent du bloc fonctionnel. Elles sont reçues par la logique combinatoire que calcule la fonction prochain état - produisant des états codées, et la fonction sortie - qui donne l'indication d'erreur. La logique de mémorisation fonctionne comme une boucle de retour pour le bloc combinatoire qui avec les entrées générales composent les données pour les successives fonctions prochain-état.

Si l'on prend en compte la conception de chaque bloc en séparé, il est possible d'implémenter le contrôleur défini au début de cette section avec des structures régulières comme des PLAs, registres, etc... Les deux premiers cas représentés dans la figure 58, produisent des comportements pareils du système ; ils ne différent que par le modèle utilisé dans la définition de la machine d'états. Les registres 1 et 2 sont habilités en différentes phases d'horloge - cette condition assure que les boucles de retour n'iront pas provoquer des courses et les registres peuvent être sensibles au niveau. Alors, si des cellules maître-esclave sont employées pour le registre, et si les lignes d'entrée gardent des valeurs stables durant un laps de temps suffisamment long pour la propagation dans la logique combinatoire (et avec des valeurs stables aux sorties), un seul registre sera nécessaire - dans ce cas là, nous aurons le schéma du système montré à l'item c de la figure 58. Dans toutes ces propositions, la logique combinatoire peut être éxécutée par de PLAs. Les propriétés nécessaires pour chacun des blocs composants sont analysées par la suite.

Nous considérons un système dont la structure est celle de l'item c (figure 58). Il est utilisé un modèle de Mealy, avec un registre maître-esclave, et des entrées stables (celles qui viennent du bloc fonctionnel).

Afin d'étudier la structure interne des contrôleurs SLD et leurs propriétés nécessaires, nous considérerons le contrôleur comme un bloc qui reçoit les entrées définies par un langage (ces entrées proviennent du bloc fonctionnel). Le contrôleur étant un accepteur d'états-finis, est "language disjoint" pour un langage d'entrée, et l'hypothèse H9 peut être reformée.

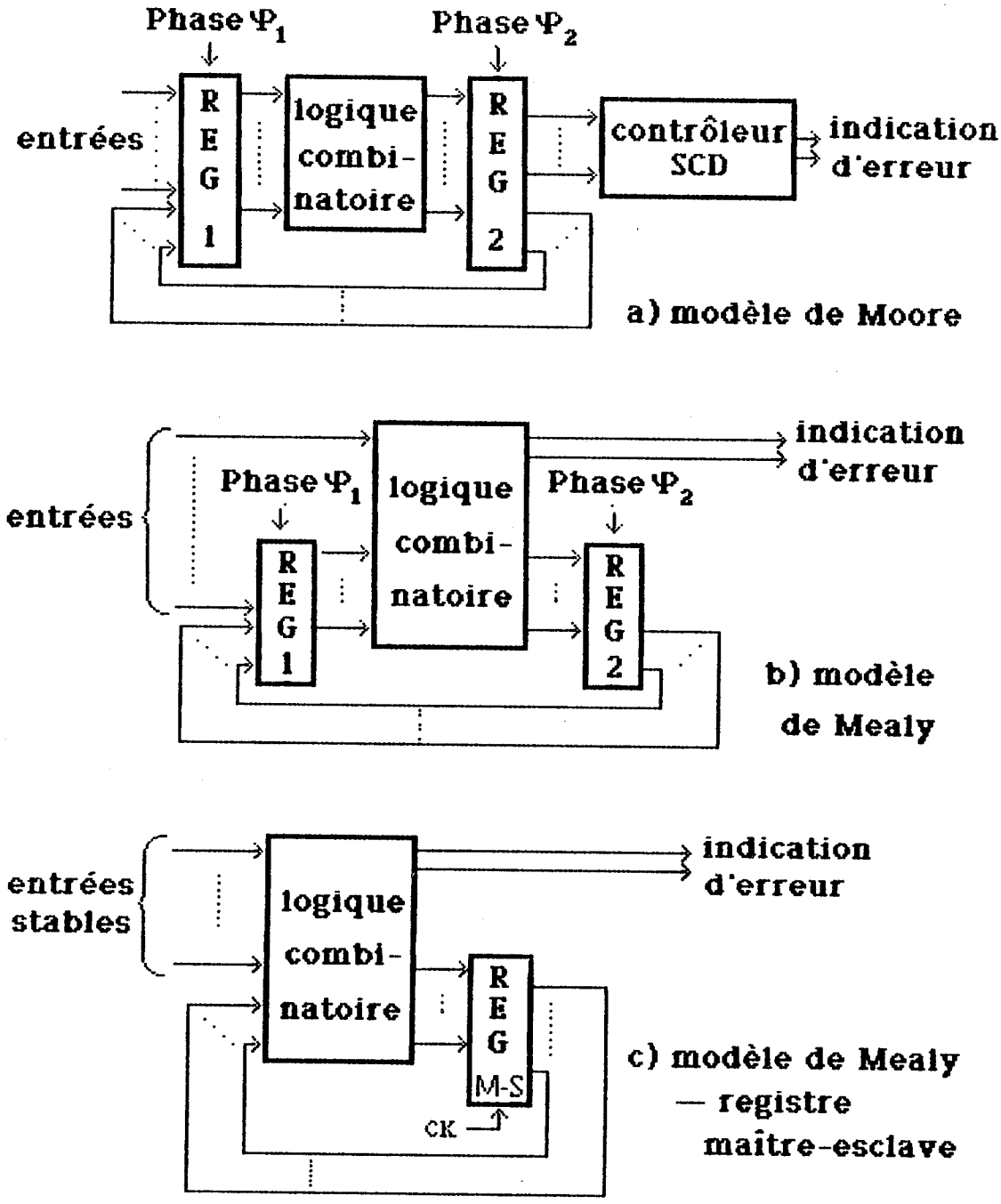



Figure 58 : Blocs généraux pour des solutions conçues avec des structures régulières. a) Modèle de Moore; b) Modèle de Mealy; c) Modèle de Mealy, avec un registre maître- esclave

L'accepteur LD est tel qu'il reconnaît un langage selon le fait que : une entrée appartenant à une séquence d'entrée i apparaît quand l'état présent appartient à un ensemble d'états.

Soit B l'ensemble des vecteurs de code qui se forment par la

concaténation d'une entrée x ∈ X avec un état q ∈ Q. Une entrée est un vecteur codé si et uniquement s'il y a une des séquences d'entrée i ∈ $I_C$ qui correspond à ce vecteur. L'ensemble des états étant codé, on peut donner des exemples de vecteurs qui n'appartient pas au code (y ∉ B) :

- une entrée $x_1$ ∈ X qui n'apparaît jamais dans les séquences i ∈ $I_C$, quand l'état est $s_j$,
- n'importe quelle entrée $x_k$ ∈ X quand l'état q ∉ Q.

Maintenant, l'hypothèse H9 reformée est énoncée.

**Hypothèse H9'**
Entre l'occurrence de deux défauts quelconques affectant le bloc fonctionnel, il s'écoule un laps de temps suffisant pour que, ∀ q ∈ Q, ∀ $i_1$ tel que $i_1 . i_2$ ∈ $I_M$ et $\delta(i_1,q_0) = q$ ; toutes les $i_{2m}$ pour chaque faute f ou séquences de défauts qui peuvent se produire dans le bloc fonctionnel soient appliquées à ce bloc, avant qu'un autre défaut survienne à l'accepteur déterministe d'états-finis (contrôleur). Entre l'occurrence de deux défauts quelconques affectant l'accepteur déterministe d'états-finis, il s'écoule un laps de temps suffisant pour que tous les vecteurs du code B soient appliqués au contrôleur (i.e., les séquences i ∈ $I_C$ appliquées aux contrôleurs sont telles que tous les vecteurs du code B sont appliqués), avant qu'un autre défaut survienne au bloc fonctionnel.

Dans les prochaines sous-sections (IV.4.4.1.1. et IV.4.4.1.2.), sont données des propositions rapportées aux propriétés internes des contrôleurs SLD.

### *IV.4.4.1.1. Conception en blocs de contrôleurs SLD*

**Proposition P14**
Pour la conception en blocs d'un contrôleur SLD avec une structure comme celle présentée à la figure 59 (modèle de Mealy), la propriété "strongly language disjoint" (comme énoncée par la définition D56) sera assurée s'il se compose de :

- une logique combinatoire SFS pour la fonction prochain état,
- une logique combinatoire SCD pour la fonction de sortie d'indication d'erreur (code double-rail), et
- une logique de mémorisation SFS/SCD*,

l'hypothèse H9' étant assurée, et chaque faute ne pouvant affecter q'un seul bloc à la fois.

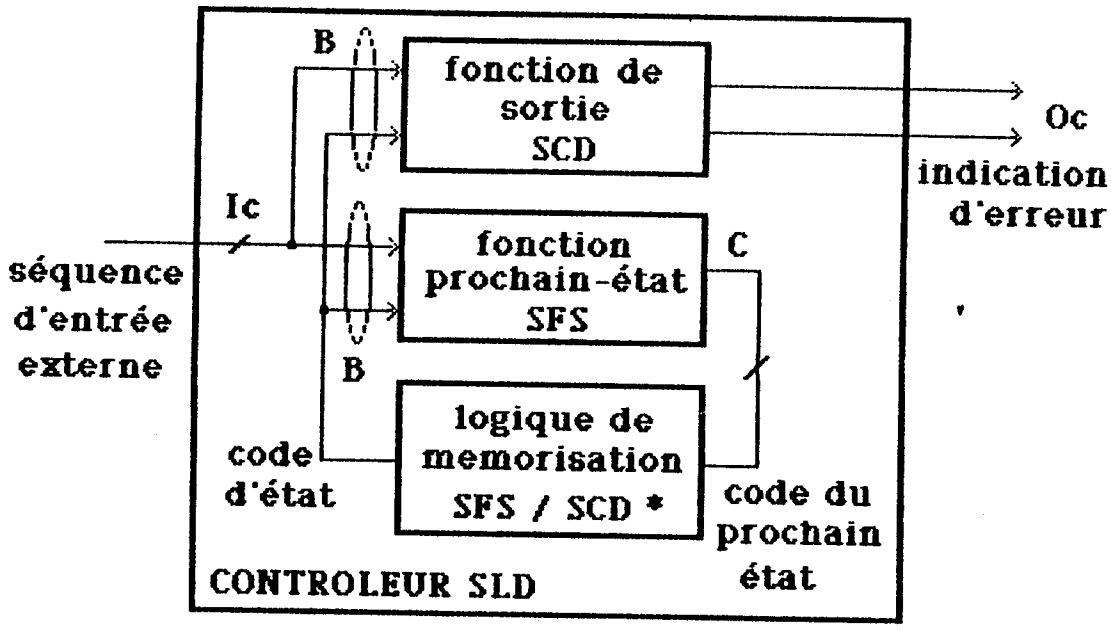



Figure 59 : Proposition pour la structure interne du contrôleur SLD (P14 - modèle de Mealy)

La fonction de sortie est SCD vis-à-vis du code d'entrée B (concaténation de l'entrée externe avec le code de l'état) et du code de sortie $O_C$ et la logique de mémorisation est SCD et SFS vis-à-vis du code C; toutes les deux se rapportent à la définition large (faible) des contrôleurs combinatoires SCD (D23). La fonction prochain état est SFS pour le code de sortie C.

L'astérisque est associé à la notion SFS/SCD* en signalant que des propriétés combinatoires sont utilisées pour des circuits où des variables de temps interviennent aussi bien. Avant que l'habilitation de charge et le signal d'horloge ne soient pas valables, les sorties du circuit ne changent pas. Quand le circuit est habilité par charge/horloge, les sorties devront devenir la copie des valeurs d'entrée, ou elles devront être un vecteur hors code (donc la propriété SFS est nécessaire). Les entrées en dehors du code doivent produire aussi des sorties en dehors du code, même si des fautes pour lesquelles le circuit est redondant sont déjà survenues. Ainsi, nous gardons l'astérisque pour signaler le délai requis entre l'application des entrées et la production des sorties (en dépendance de la période d'horloge et de la propagation interne des signaux). On suppose ques les lignes des circuits d'horloge et de charge ne sont pas affectées par des défauts - ou que ces circuits sont testés séparément.

La restriction selon laquelle on a des blocs internes isolés ceux-ci

n'étant pas affectés par une faute commune, peut être enlevée à condition que d'autres circuits et/ou propriétés soient ajoutés. Ces possibilités sont vérifiées en d'autres propositions présentées après la démonstration de P14.

**Demonstration de P14**

Par la suite, nous montrerons qu'un circuit conçu selon la proposition P14, aura le comportement défini comme correspondant aux contrôleurs SLD. Par l'hypothèse H9', il n'arrive pas de défaut aux entrées générales avant l'écoulement d'un laps de temps suffisant pour la détection d'une faute apparue dans le contrôleur - si le défaut n'est pas détecté, c'est parce que le contrôleur est redondant pour lui.

Au début, avant l'occurrence d'un défaut, le circuit est "language disjoint", donc $\forall\, i \in I_C,\ w(i,q_0) \in O_C$ et $\forall\, i \notin I_C,\ w(i,q_0) \notin O_C$. Si une faute ou une séquence de défauts $\langle f_1,f_2,..,f_n\rangle \in Cf$ survient au circuit, nous montrerons qu'il peut en résulter deux situations :

a) il n'y a pas encore eu de défauts ou le circuit est "strongly redundant" pour $\langle f_1,f_2,.....,f_n\rangle$ et $i_2$ étant appliquée telle que $\forall\, i_1 \mid \delta(i_1,q_0) = q$ et $i_1 \,.\, i_2 \in I_C : w(i_1,q_0) \,.\, w^{\langle f1,f2,....,fk\rangle}(i_2, q) \notin O_C$;

b) le contrôleur est "strongly redundant" pour les séquences de défauts. Dans ce cas, si une faute se produit dans la séquence d'entrée appliquée au contrôleur, la détection est assurée par la propriété "language disjoint", car $\forall\, i \notin I_C,\ w(i, q_0) \notin O_C$.

Maintenant nous allons analyser un circuit composé où sont combinées des propriétés SCD et SFS. Les défauts qui se produisent, peuvent affecter soit le bloc combinatoire soit la logique de mémorisation, mais pas les deux à la fois. Ces cas seront examinés ensuite.

Cas 1:

Une faute $f_k$ se produit dans le bloc de calcul de la fonction sortie. Ceci étant SCD, deux situations peuvent en résulter :

a) le bloc est un contrôleur "strongly redundant " pour la séquence de défauts ;

b) soit il n'y avait pas encore de défauts, soit le bloc est un contrôleur "strongly redundant" pour la séquence de fautes résultante $\langle f_1, f_2, \ldots, f_{k-1} \rangle \in Cf$ et il existe un vecteur de code $b \in B$ tel que $G(b, \langle f_1, f_2, \ldots, f_k \rangle) \notin O_C$.

Dans la première situation, le défaut ne sera pas détecté mais le bloc reste CD, donc le contrôleur, LD. Dans la deuxième, l'hypothèse H9' assure qu'il est appliqué un vecteur du code B pour lequel l'erreur est indiquée, et détectée (nous remarquons que ce vecteur codé est sûrement appliqué si le défaut n'affecte pas d'autres blocs).

Cas 2:

Une faute $f_k'$ se produit dans le bloc de la fonction prochain état. Ceci étant SFS, deux situations peuvent en résulter :

a) le bloc est "strongly redundant" pour la séquence de fautes ;

b) soit il n'y avait pas encore de défauts, soit le bloc est "strongly redundant" pour la séquence de fautes résultante $\langle f_1', f_2', \ldots, f_k' \rangle \subset Cf$, et : pour tous les vecteurs d'entrée $b \subset B$ soit $G(b, \langle f_1', f_2', \ldots, f_k' \rangle) = G(b, \emptyset)$ soit $G(b, \langle f_1', f_2', \ldots, f_k' \rangle) \notin C$, C étant le code de sortie des états. Aussi, un vecteur $b \in B$ sera appliqué tel que $G(b, \langle f_1', f_2', \ldots, f_k' \rangle) \notin C$.

Dans la première situation, le défaut ne sera pas détecté mais le bloc reste FS, donc le contrôleur, LD. Dans la deuxième, pendant que des sorties appartenant au code sont produites, elles ne sont pas erronées. La détection du défaut se doit à la production d'un mot non codé (C) aux sorties du bloc, celles-ci étant reçues et transférées comme une valeur non appartenant au code B par la logique de mémorisation (grâce à la propriété SCD) au circuit de sortie. Nous remarquons que la propriété SFS est nécessaire pour assurer l'apparition du b tel que $G(b, \langle f_1', f_2', \ldots, f_k' \rangle) \notin C$. Si ce bloc n'était pas SFS, l'état qui, combiné avec une entrée, permet la détection pourrait ne pas apparaître.

Cas 3:

Une faute $f_k''$ se produit dans le bloc de mémorisation. Ceci étant SFS/SCD* , deux situations peuvent en résulter :

a) le bloc est "strongly redundant" pour la séquence de défauts ;

b) soit il n'y avait pas encore de défauts soit le circuit est "strongly redundant" pour la séquence de défauts résultante $\langle f_1'', f_2'', \ldots, f_{k-1}'' \rangle \in Cf$, et : pour tous les $c \in C$ soit $G(c, \langle f_1'', f_2'', \ldots, f_k'' \rangle) = G(c, \emptyset)$ soit $G(c, \langle f_1'', f_2'', \ldots, f_k'' \rangle) \notin C$, C étant l'ensemble des vecteurs de sortie utilisés pour le codage des états (on suppose que les ensembles de vecteurs d'entrée et de sortie de la logique de mémorisation sont identiques). Et aussi, un $c \in C$ sera appliqué aux entrées tel que $G(c, \langle f_1'', f_2'', \ldots, f_k'' \rangle) \notin C$.

Dans la première situation, la faute ne sera pas détectée et le bloc reste FS et le contrôleur, LD. Dans la deuxième, pendant que des sorties codées sont produites, elles ne sont pas erronées. La détection du défaut se doit à la production d'une valeur non codée (C) aux sorties du bloc, celle-ci étant reçue par le circuit de la fonction sortie. Ce circuit est "code disjoint", donc il donne l'indication d'erreur. Nous remarquons que la propriété SFS est nécessaire pour assurer que les sorties appartenant au code sont correctes puisqu'un code erroné pourrait conduire à un état erroné - par conséquent, le $c \in C$ qui assure la détection pourrait ne pas apparaître.

Q.E.D.

Nous remarquons que l'hypothèse H9' n'est pas nécessaire pour la détection de fautes dans ce bloc. Il faut assurer que la séquence d'entrée fasse passer le contrôleur par tous les états en Q, comme condition suffisante.

Le contrôleur présenté à la figure 59, a la structure d'un modèle de Mealy car la fonction de sortie est calculée à partir des variables d'entrée et d'état. Dans le cas du modèle de Moore, la séquence d'entrée n'est pas fournie directement au bloc de sortie. Donc, les propriétés internes nécessaires proposées en P14 ne sont pas suffisantes puisque des vecteurs hors code $b \notin B$ peuvent être reçus par le bloc de fonction prochain état (SFS), causés par une erreur à l'entrée $I_C$. Le comportement non défini dans ce cas n'était pas un problème quand le modèle de Mealy était considéré.

**Remarque R63**
Si le modèle de Moore est utilisé pour la machine proposée en P14, les propriétés SCD et SFS sont nécessaires au bloc de fonction prochain état.

Le bloc de fonction prochain état, selon le modèle de Moore étant

SFS/SCD, si un vecteur hors code est appliqué, il produira une sortie hors code (un état invalide, non codé), par la propriété SCD. Cette valeur sera propagée à travers la logique de mémorisation (qui est elle aussi SCD) au bloc de fonction sortie. Ce dernier produit une indication d'erreur (un vecteur de sortie hors code) par la propriété SCD. La proposition qui correspond à ce modèle peut être formulée comme suit .

**Proposition P15**

Pour la conception en blocs d'un contrôleur SLD avec une structure comme celle présentée à la figure 60 (modèle de Moore); la propriété "strongly language disjoint" (comme énoncée par la définition D56) sera assurée s'il est composé de :

- une logique combinatoire SFS/SCD pour la fonction prochain état ,
- une logique combinatoire SCD pour la fonction de sortie d'indication d'erreur (code double-rail), et
- une logique de mémorisation SFS/SCD*,

l'hypothèse H9' étant assurée, et chaque faute ne pouvant affecter q'un seul bloc à la fois.

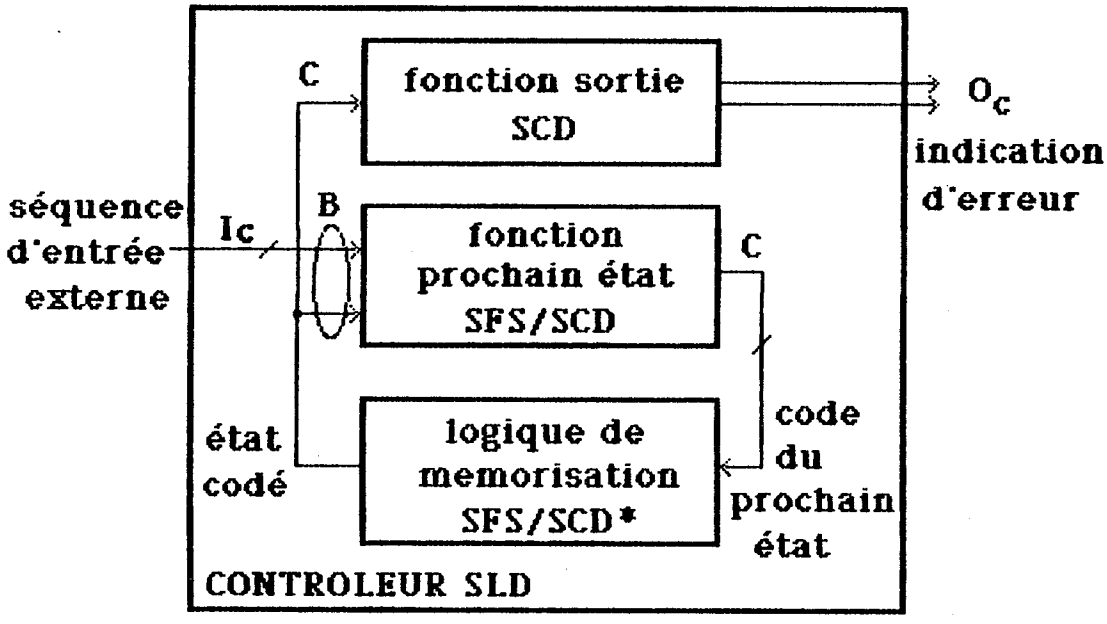



Figure 60 : Proposition pour la structure interne d'un contrôleur SLD (P15 - modèle de Moore)

La fonction de sortie est SCD vis-à-vis des codes d'entrée C et de sortie $O_C$; la logique de mémorisation est SCD et SFS vis-à-vis du code C. La fonction prochain état est SCD et SFS pour le code d'entrée B et pour le code de sortie C. Pour tous ces blocs, on utilise la définition large (faible) des

contrôleurs combinatoires SCD (D23).

La proposition P15 devient évidente si l'on regarde la remarque Rk3, expliquée avant cette proposition.

**Remarque Rk4**
L'indication d'erreur (qui résulte d'une faute produite dans le bloc fonctionnel) donnée par un contrôleur SLD conçu selon le modèle de Moore, sera disponible plus tard (délai maximal =1 période d'horloge) que celle obtenue à partir d'un contrôleur SLD conçu selon le modèle de Mealy.

La remarque Rk4 est une conséquence des paragraphes précédents - la machine de Mealy "examine" directement les entrées au bloc de fonction de sortie tandis que dans la machine de Moore, l'information d'entrée est processée avec l'état avant d'être emmenée au bloc de fonction sortie. Cette condition devra être observée si les résultats qui viennent du bloc fonctionnel sont utilisés immédiatement, et résultent en changements de valeurs dans d'autres circuits par exemple.

Un contrôleur "strongly language disjoint" avec une organisation interne comme montrée aux figures 59 et 60, peut être implementé avec des éléments de base différents. A la fin de cette section, nous analysons l'emploi d'un PLA pour l'exécution de la logique combinatoire SFS ou SFS/SCD et de registres pour la logique de mémorisation. La logique SCD peut être exécutée aussi par PLAs ou par l'assemblage de cellules de base de contrôleurs SCD, comme celles exemplifiées en [JAN 84a] et [JAN 84c]. La logique combinatoire SFS ou SFS/SCD peut aussi être implementée avec de la logique anarchique dès que les propriétés nécessaires sont ajoutées. Mais, dans ce cas, la conception est plus difficile, car toutes les conditons devront être considérées en fonction d'une disposition particulière.

Si d'autres architectures internes sont souhaitées, elles peuvent être utilisées, dès que les propriétés nécessaires sont assurées. C'est le cas, par exemple, d'un arrangement où il n'y a pas d'isolation interne parmi les blocs. Des modules et/ou des propriétés supplémentaires sont demandées. Nous en examinerons plusieurs par la suite.

**Proposition P16**
La propriété "strongly language disjoint" sera assurée à un contrôleur SLD conçu avec une structure interne en blocs (selon la figure 61), s'il est composé de :

- une logique combinatoire SFS/SCD,

- une logique de mémorisation SFS*,
- un contrôleur interne SCD qui vérifie le codes des états aux entrées de la logique de mémorisation,

l'hypothèse H9' étant assurée et chaque faute ne pouvant affecter qu'un seul bloc à la fois.

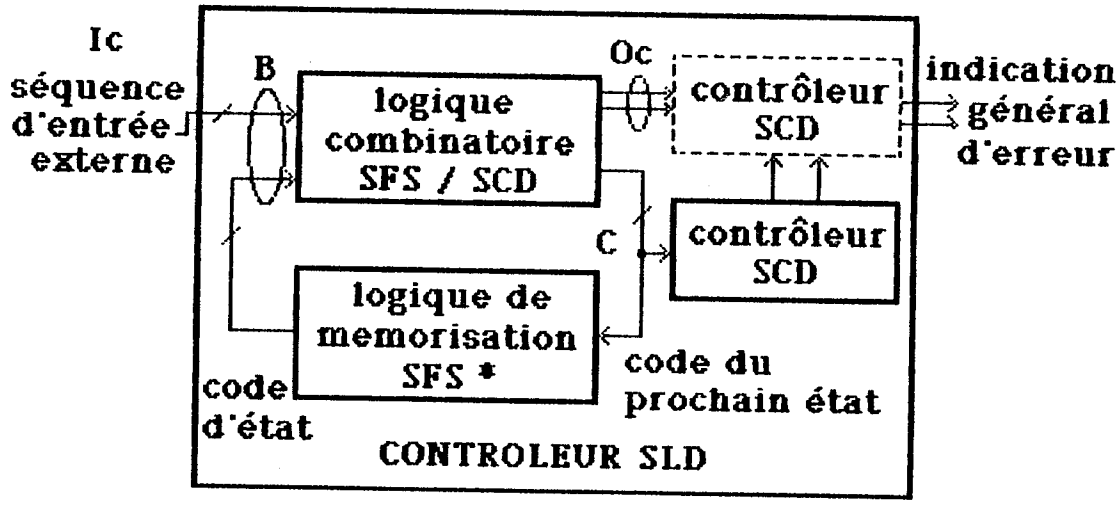



Figure 61 : Structure interne d'un contrôleur SLD selon la proposition P16

La logique combinatoire est SCD vis-à-vis du code d'entrée B et des codes de sortie C et $O_C$, et SFS vis-à-vis du code d'entrée B et du code de sortie C. Le contrôleur1 est SCD pour le code C. Pour tous les deux, on utilise la définition large (faible) des contrôleurs combinatoires SCD (D23). On peut utiliser aussi un contrôleur général SCD, placé à l'intérieur, afin de réduire les sorties à une seule paire. Ce dernier sera SCD pour les codes d'entrée $O_C$ et pour celui en provenance du contrôleur1. La logique de mémorisation est SFS pour le code C.

**Démonstration de P16**

Ensuite, nous montrons qu'un circuit conçu selon la proposition P16 aura le comportement correspondant à un contrôleur défini comme SLD. Nous considérons, par l'hypothèse H9', qu'il n'y a pas eu de fautes affectant les entrées générales avant l'écoulement d'un laps de temps suffisant pour la détection d'une faute existant dans le contrôleur ou, le contrôleur est redondant pour les fautes déjà apparues.

Au début, avant l'occurrence d'un défaut, le circuit est "langage disjoint". Si une faute ou une séquence de défauts $\langle f_1, f_2, \ldots, f_i \rangle \in Cf$ survient au circuit, deux situations peuvent en résulter:

a) il n'y a pas encore eu de défauts ou le circuit est "strongly redundant" pour $\langle f_1,f_2,....,f_{k-1}\rangle$ et $i_2$ étant appliquée telle que $\forall\, i_1 | \delta(i_1,q_0) = q$ et $i_1 \,.\, i_2 \in I_C : w(i_1,q_0).\, w^{\langle f1,f2,....,fk\rangle}(i_2,q) \notin O_C$.

b) le contrôleur est "strongly redundant" pour les séquences de défauts. Dans ce cas, si une faute se produit dans la séquence d'entrée appliquée au contrôleur, la détection est assurée par la propriété "language disjoint", car $\forall\, i \notin I_C\,,\, w(i_1,q_0) \notin O_C$.

Maintenant, nous analysons des circuits composés où sont combinés des propriétés SCD et SFS.

Les défauts qui se produisent peuvent affecter soit le bloc combinatoire soit la logique de mémorisation, mais pas les deux à la fois. Ces cas seront examinés ensuite.

Cas 1:

Une faute $f_k$ se produit dans le bloc combinatoire, et ceci uniquement est affecté par cette faute. Ce bloc est SFS pour le code d'entrée comme défini précédemment, et pour le code de sortie correspondant aux états. Il est SCD vis-à-vis du code d'entrée B et des codes de sortie $O_C$ ou C, ou pour tous les deux. Deux situations peuvent se produire :

a) le bloc est "strongly redundant" pour la séquence de défauts ; ou

b) soit il n'était pas survenu de défauts auparavant, soit le bloc était "strongly redundant" pour la séquence de défauts résultante $\langle f_1,f_2,...,f_{k-1}\rangle \in$ Cf et pour tous b ∈ B soit $G(b,\langle f_1,f_2,....,f_k\rangle) = G(b,\emptyset)$ ou $G(b,\langle f_1,f_2,....,f_k\rangle) \notin C$ puis que le bloc est SFS. (Ces sorties sont celles qui codent les états - $O_C$ n'est pas concerné). Si un vecteur hors code c ∉ C est produit, le contrôleur1 SCD détecte le défaut. L'application de b tel que $G(b,\langle f_1,f_2,....,f_k\rangle) \notin C$ est assurée par la propriété SFS et par l'hypothèse H9'.

Dans la première situation, le défaut ne sera pas détecté mais le bloc reste FS et CD, donc le contrôleur LD. Dans la deuxième, pendant que des vecteurs du code C sont produits, ils ne sont pas erronés. Le premier vecteur en dehors du code C aux sorties de ce bloc sera détecté par le contrôleur1 SCD.

Cas 2:

Une faute $f_k'$ se produit dans le bloc de mémorisation, et ceci uniquement est affecté par cette faute. Ce bloc est SFS*. Deux situations peuvent avoir lieu:

a) le bloc est "strongly redundant" pour la séquence de défauts ;

b) soit des défauts ne sont pas survenus auparavant, soit le bloc est "strongly redundant" pour la séquence de fautes résultante $\langle f_1', f_2', \ldots, f_{k-1}' \rangle \in Cf$, et pour tous $c \in C$ soit $G(c, \langle f_1', f_2', \ldots, f_k' \rangle) = G(c, \emptyset)$ soit $G(c\langle f_1', f_2', \ldots, f_k' \rangle) \notin B$.

Dans la première situation, le défaut ne sera pas détecté mais le bloc reste "fault secure", donc le contrôleur, LD. Dans la deuxième, pendant que des vecteurs du code sont produits, ils ne sont pas erronés. La détection du défaut se doit à la génération d'un mot hors code (B) aux sorties du bloc ; ce mot est reçu comme une entrée hors code par la logique combinatoire. La logique combinatoire, étant SCD, assure qu'un vecteur en dehors de code ($O_C$ ou C) sera produit à ses sorties, en donnant l'indication d'erreur. Nous remarquons le besoin de la propriété SFS pour assurer l'apparition du $c \in C$ tel que $G(c, \langle f_1', f_2', \ldots, f_{k-1}' \rangle) \notin B$. Si ce bloc ne fût pas SFS, l'état que concaténé à une entrée donnée permet la détection pourrait ne pas apparaître.

Q.E.D.

**Proposition P17**

La propriété "strongly language disjoint" sera assurée au contrôleur SLD conçu avec une structure interne en blocs, s'il est composé de :

- une logique combinatoire "SFS/SCD/ (D26), et
- une logique de mémorisation SFS/SCD*,

l'hypothèse H9' étant assurée et chaque faute n'affectant qu'un seul bloc à la fois.

La logique combinatoire est SCD pour le code d'entrée B et le code de sortie $O_C$ (et C) et elle est SFS pour le code d'entrée B et le code de sortie C. La logique de mémorisation est SCD et SFS vis-à-vis du code C. La propriété SCD du bloc combinatoire considère la définition forte (D26).

La structure interne du contrôleur composé selon la proposition P17 est montrée à la figure 62.

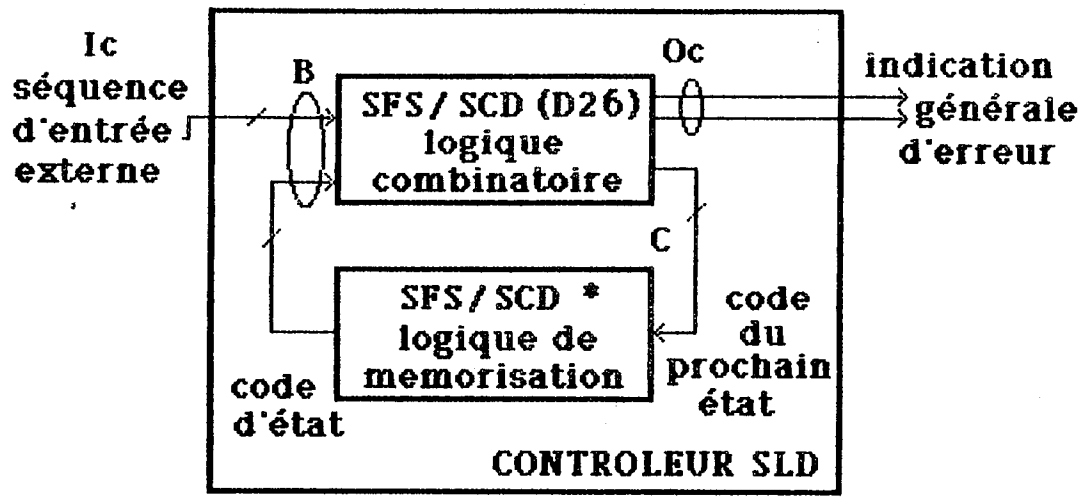



Figure 62 : Structure interne du contrôleur SLD selon la proposition P17

**Démonstration de P17**

La proposition P17 est dérivée de P16 par la suppression du contrôleur1. Donc, nous allons démontrer cette proposition uniquement pour la situation où il était nécessaire , et que sa substitution par le renforcement de la propriété SCD du bloc combinatoire ainsi que l'addition de la propriété SCD au bloc de mémorisation est également efficace.

Si une faute affecte le bloc combinatoire, soit sa sortie sera correcte soit elle sera un vecteur hors code, par la propriété SFS. On peut remarquer que, si la sortie erronée est une des lignes du code $O_C$ au lieu de C, la détection est immédiate. Mais étant C le code de sortie affecté, cette valeur sera reçue par le bloc de mémorisation et transférée aussi comme un vecteur hors code, car ce bloc est SCD. Aux entrées de la logique combinatoire, le code B composé par l'entrée externe et ce vecteur hors code en provenance du bloc de mémorisation sera une valeur non codée. Puisqu'on considère qu'il y a déjà une faute dans la logique combinatoire, la propriété SCD forte lui est nécessaire pour qu'une sortie hors du code $O_C$ soit produite.

Q.E.D.

En raison des boucles de retour existantes dans les circuits séquentiels, les exigences concernant l'effet de quelques défauts sont différentes de celles considérées pour les circuits combinatoires. C'est le cas des défauts affectant des blocs différents. Dans les circuits combinatoires, les sorties d'un seul bloc étaient affectées , et transmises à l'indication d'erreur

générale. Dans les circuits séquentiels, au début seulement, les sorties d'un des blocs seront affectées aussi, mais si cette valeur erronée est utilisée comme entrée d'un des blocs affectés, ce dernier produira des sorties non prévisibles, voire erronées. L'isolation entre les blocs élimine l'hypothèse de défauts affectant deux blocs.

**Proposition P18**
La propriété "strongly language disjoint" sera assurée au contrôleur SLD conçu avec une structure interne en blocs comme montré à la figure 63, s'il est composé de :

- une logique combinatoire SFS/SCD,
- une logique de mémorisation SFS*,
- des contrôleurs internes SCD qui vérifient le code des états aux entrées des deux blocs ci-dessus,

l'hypothèse H9' étant assurée.

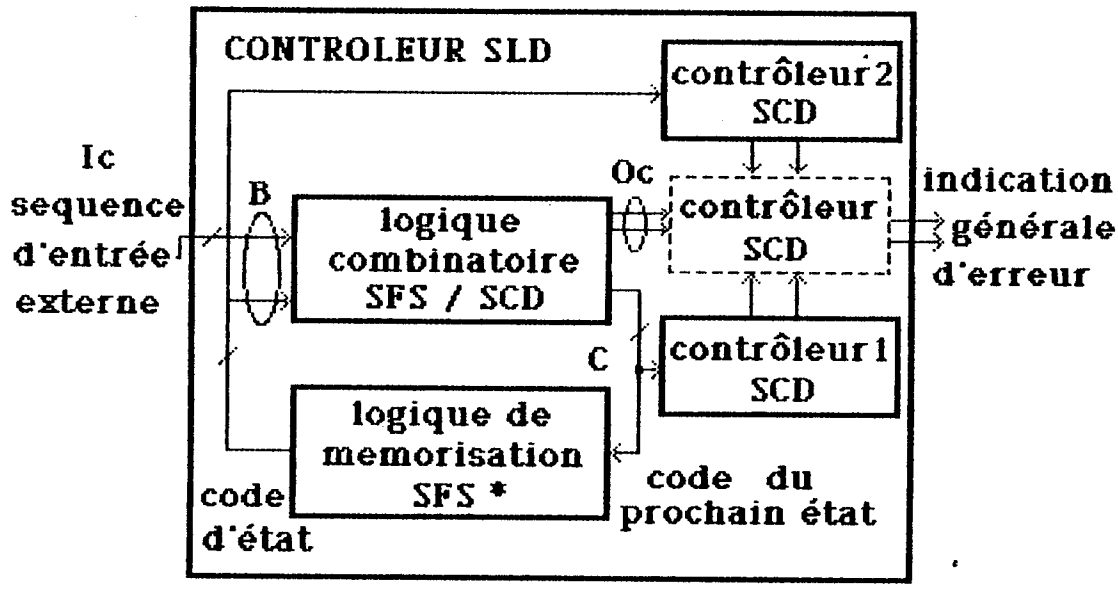



Figure 63 : Structure interne du contrôleur SLD selon la proposition P18

La logique combinatoire est SCD vis-à-vis du code d'entrée B et des codes de sortie C et $O_C$, et est SFS pour le code d'entrée B et le code de sortie C. Le contrôleur1 et le contrôleur2 sont SCD pour le code C. Pour ces blocs, la définition SCD large (faible), D23, est considérée. Un contrôleur additionnel peut être utilisé pour réduire le nombre des lignes de sortie; il sera SCD vis-à-vis du code $O_C$ et des codes de sortie du contrôleur1 et du contrôleur2. La logique de mémorisation est SFS pour le code C.

**Démonstration de P18**

La démonstration de la proposition P18 est similaire à celle de P16 concernant les cas 1 et 2. Mais une troisième situation peut se produire puisque des défauts affectant à la fois les logiques combinatoires et de mémorisation sont admises. Pour les cas de défauts qui n'affectent qu'un des blocs composants, il faut regarder la démonstration de P16. Ici, nous analysons seulement le troisième cas.

Cas 3:

Une faute $f_k$ se produit affectant deux blocs. Dans ce cas, la détection du défaut est assuré d'une façon similaire aux cas 1 et 2(à la démonstration de P16), sauf quand la détection était assurée par la propriété SCD de la logique combinatoire puisqu'elle aussi peut être affectée par le défaut. Dans ce cas, on peut avoir une double faute (aux entrées et à l'intérieur du bloc combinatoire) et la production d'un vecteur hors code (en dehors de $O_C$ et de C) ne sera plus assurée. Le contrôleur2 étant implémenté comme montré à la figure 63, la détection est assurée.

Q.E.D.

**Proposition P19**

La propriété "strongly language disjoint" sera assurée au contrôleur SLD conçu avec une structure interne en blocs comme montré à la figure 64, s'il est composé de :

- une logique combinatoire SFS/SCD,
- une logique de mémorisation SFS/SCD (D26)*, et
- un contrôleur interne qui vérifie le code des états aux entrées de la logique combinatoire,

l'hypothèse H9' étant assurée.

La logique combinatoire est SCD vis-à-vis du code d'entrée B et des codes de sortie $O_C$ et C, et est SFS pour le code d'entrée B et le code de sortie C. Le contrôleur2 est SCD pour le code C. Tous les deux considérent la définition large (faible), D23, des contrôleurs. La propriété SCD de la logique de mémorisation fait référence au code C et est définie par la définition forte des contrôleurs combinatoires SCD (D26). Un contrôleur général peut être utilisé pour réduire le nombre des sorties ; ceci sera SCD pour $O_C$ et pour le code de sortie du contrôleur2.

**Démonstration de P19**

La démonstration de la proposition P19 est similaire aux précédentes -

de P16 à P18. Il n'est pas difficile de remarquer que le contrôleur1 de P18 a été substitué par la propriété SCD forte vis-à-vis de C ajoutée à la logique de mémorisation. Avec la structure proposée, pour des défauts affectant les blocs combinatoires et de mémorisation à la fois, il en résultera le comportement décrit comme suit.

Cas 3:

Une faute $f_k$ affectant les deux blocs se produit. La détection dans ce cas est assurée d'une façon similaire aux cas 1 et 2 de P16, à l'exception d'une situation: quand sont affectées à la fois les lignes que produisent le code de sortie C et la logique de mémorisation. Une sortie non codée C peut être présentée aux entrées de la logique de mémorisation laquelle, étant SCD par la définition D26 (définition forte), assure une sortie hors code même en présence de défauts dans ce bloc. Cette sortie hors code sera détectée par le contrôleur2 SCD.

Q.E.D.

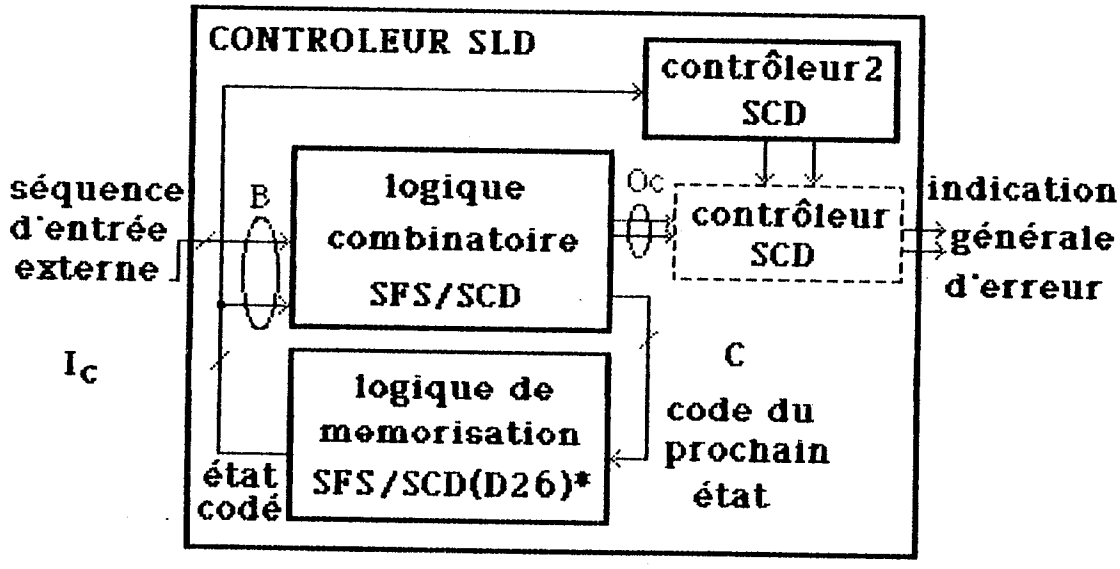



Figure 64 : Structure interne du contrôleur SLD selon la proposition P19

**Proposition P20**

La propriété "strongly language disjoint" sera assurée au contrôleur SLD conçu avec une structure interne en blocs comme montré à la figure 65, s'il est composé de :

- un bloc combinatoire SFS/SCD, étant la propriété SCD définie par la définition D26,

- une logique de mémorisation SFS*,
- un contrôleur interne qui vérifie le code des états aux entrées de la logique de mémorisation (contrôleur1, SCD),

étant assurée l'hypothèse H9'.

La structure interne du contrôleur composé selon la proposition P20 est montrée à la figure 65.

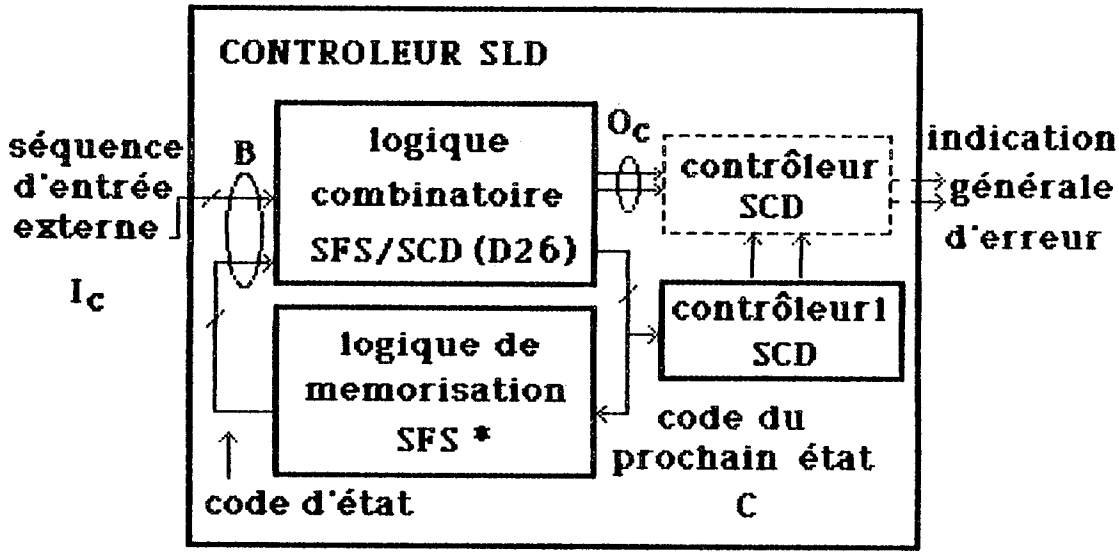



Figure 65 : Structure interne du contrôleur SCD selon la proposition P20

La logique combinatoire est SCD vis-à-vis du code d'entrée B et des codes de sortie C et $O_C$. La propriété SCD forte est considérée. Cette logique est aussi SFS pour les codes B (d'entrée) et C (de sortie). Le contrôleur1 est SCD pour le code C, et la définition SCD large (faible) est considérée. Un contrôleur additionnel SCD peut être utilisé afin de réduire le nombre des lignes de sortie, et il sera SCD pour le code $O_C$ et pour le code de sortie du contrôleur1. La logique de mémorisation est SFS pour le code C.

**Démonstration de P20**

Cette proposition, P20, est démontrée d'une manière analogue à la proposition P18 (ou P16), à l'exception de la situation où une faute affecte à la fois les logiques combinatoire et de mémorisation. Ainsi, nous faisons référence au cas 3, pour lequel cette démonstration est différente des autres qui correspondent aux cas précédents.

Cas 3:

Une faute $f_k$ affectant les deux blocs se produit. A partir de la

démonstration du cas 3 pour la proposition P18, il devient évident que si la définition SCD forte est utilisée pour le circuit combinatoire, ses propriétés additionnelles remplacent le contrôleur2.

Q.E.D.

**Proposition P21**

La propriété "strongly language disjoint" sera assurée au contrôleur SLD conçu avec une structure interne en blocs comme montré à la figure 66, s'il est composé de :

- un bloc combinatoire SFS/SCD, la propriété SCD étant déterminée par la définition D26,
- une logique de mémorisation SFS/SCD, la propriété SCD étant déterminée par la définition D26,

étant assurée l'hypothèse H9'.

La strucutre du contrôleur composé selon la proposition P21 est montré à la figure 66. La logique combinatoire est SCD pour le code d'entrée B et pour les codes de sortie C et $O_C$. La logique de mémorisation est SCD pour le code C. Pour toutes les deux, la définition forte des contrôleurs combinatoires SCD (D26) est considérée. La logique combinatoire est SFS pour le code B et la logique de mémorisation est SFS vis-à-vis du code C.

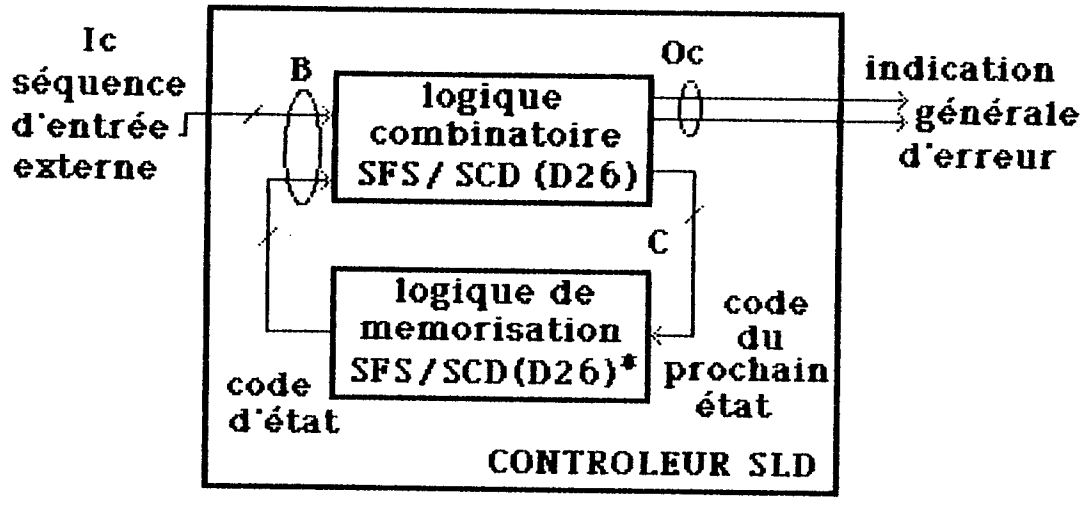



Figure 66 : Structure interne du contrôleur SLD selon la proposition P21

**Démonstration de P21**

Cette proposition est démontrée d'une façon analogue à P19, sauf pour les cas où les sorties C sont affectées par le défaut. Mais, à partir de cette

même démonstration (celle de P19), on peut vérifier que la propriété SCD forte (D26) adoptée pour le bloc combinatoire remplace le besoin antérieur du contrôleur2.

Q.E.D.

Nous remarquons que cette dernière proposition donne des propriétés suffisantes mais pas nécessaires, puisque la même structure peut être utilisé pour un contrôleur SLD, selon la définition SLD forte. Ceci sera démontré à la proposition P25.

*IV.4.4.1.2. Conception en blocs des contrôleurs SLD (définition forte)*

Concernant les cas où seulement une faute se produit soit dans le bloc fonctionnel (lequel résulte en un vecteur hors code aux entrées du contrôleur) soit dans le contrôleur, chacune parmi les propositions P14 - P21 assure la propriété "language disjoint", comme il a été démontré. Mais, cette propriété ne sera peut être plus assurée si une séquence non définie par le langage de sortie du bloc fonctionnel est produite (à cause d'une faute dans ce bloc) avant qu'un défaut déjà survenu au contrôleur ait été détecté. Dans un cas pareil, les contrôleurs SLD définis par la définition forte devront être utilisés.

Les contrôleurs SLD selon la définition D58 (la définition forte) peuvent être conçus comme ayant une structure interne dont les propriétés de chaque composant sont combinatoires. A l'exemple de la modification proposée pour l'hypothèse H9 (donnant H9'), nous reformulons l'hypothèse H10 qui s'applique aux cas où les contrôleurs SLD selon la définition forte sont considérés.

**Hypothèse H10'**
Entre l'occurrence de deux défauts quelconques affectant le bloc fonctionnel, il s'écoule un laps de temps suffisant pour que : $\forall\, q \in Q$, $\forall\, i_1$ tel que $i_1 . i_2 \in I_M$ et $\delta(i_1,q_0) = q$, toutes les $i_{2m}$ pour chaque faute f ou séquences de défauts qui peuvent se produire dans le bloc fonctionnel soient appliquées à ce bloc. Entre l'occurrence de deux fautes quelconques affectant l'accepteur déterministe d'états-finis, il s'écoule un laps de temps suffisant pour que tous les vecteurs du code B soient appliqués au contrôleur (i.e. les séquences $i \in I_C$ appliquées au contrôleur sont telles que tous les vecteurs $b \in B$ sont appliquées).

Dans les prochaines propositions (P22-P25) nous n'allons démontrer que les cas où les deux défauts surviennent, comme il a été expliqué avant. Pour les autres situations où il y a une seule faute soit dans le contrôleur soit à ses entrées, nous allons indiquer la proposition de base de laquelle la structure présentée se dérive par le renforcement avec des modules additionnels et des propriétés plus fortes.

**Proposition P22**

Pour la conception d'un contrôleur SLD défini par D58 (définition forte), la propriété "strongly language disjoint" sera assurée s'il se compose de:

- une logique combinatoire SFS pour la fonction prochain état,
- une logique combinatoire SCD (D26) pour la fonction de sortie,
- une logique de mémorisation SFS/SCD*,

l'hypothèse H10' étant assurée et chaque faute ne pouvant affecter qu'un seul de ces blocs à la fois.

La proposition P22 considère le modèle de Mealy pour la machine et elle est illustrée par la figure 67.

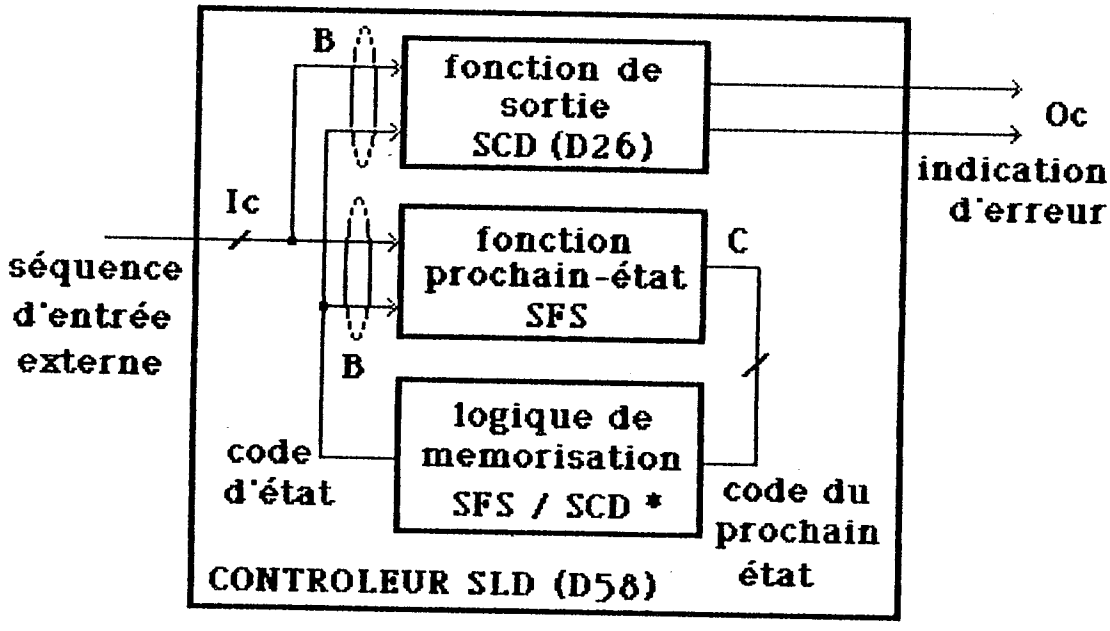



Figure 67 : Proposition pour la structure interne d'un contrôleur SLD (D58)
P22 - modèle de Mealy

La propriété SCD de la fonction de sortie se réfère au code B et à la définition forte des contrôleurs combinatoires SCD (D26). La propriété SCD de la logique de mémorisation se réfère au code C et à la définition large

(faible) des contrôleurs combinatoires SCD (D23). Ce bloc est aussi SFS pour le code C. La fonction prochain état est SFS pour le code d'entrée B et pour le code de sortie C.

**Démonstration de P22**

Cette proposition est dérivée de P14. Pour les cas d'une seule faute soit au contrôleur soit à ses entrées, la démonstration est égale à celle de P14. La séquence erronée présentée aux entrées du contrôleur combinée au code d'état aux entrées de la logique combinatoire sera un vecteur hors code pour le contrôleur. Le défaut produit dans le contrôleur et les différentes conséquences selon le bloc affecté, peuvent être décrits par un des cas suivants.

Cas 1:

Une faute $f_k$ se produit dans le bloc de fonction sortie. Ceci étant SCD selon la définition D26, assure la production de sorties hors code correspondent aux entrées hors code qu'il reçoit, même en présence d'une faute dans ce bloc. Ces entrées hors code résultent de la concaténation d'un vecteur d'entrée externe erronée avec l'état présent. Alors, le défaut est détecté immédiatement.

Cas 2:

Un défaut $f_k'$ se produit dans le bloc de fonction prochain état. Si le défaut aux entrées survient avant $f_k'$, il sera détecté par le bloc de fonction sortie. Mais, si la première faute est celle du bloc fonction prochain état ($f_k'$), un état hors code pourra être produit aux sorties de ce bloc puisqu'il est SFS. Cette valeur sera transférée par la logique de mémorisation et détectée par la fonction de sortie.

Cas 3:

Une faute $f_k''$ survient au bloc de mémorisation. Comme dans le cas 2, si le défaut aux entrées se produit avant $f_k''$, il sera détecté par le bloc de fonction de sortie. Mais si la première faute est à la logique de mémorisation ($f_k''$), la propriété SFS assure que: soit la sortie est correcte, soit elle est un vecteur hors code, qui sera détectée par la fonction sortie.

Q.E.D.

**Proposition P23**

A la conception en blocs d'un contrôleur SLD défini par D58 et selon la figure 68 (modèle de Moore), la propriété "strongly language disjoint" sera assurée s'il est composé de :

- une logique combinatoire SFS/SCD (D26) pour la fonction prochain état,
- une logique combinatoire SCD (D26) pour la fonction sortie, et
- une logique de mémorisation SFS/SCD (D26)*,

l'hypothèse H10' étant considérée et chaque faute pouvant affecter seulement un bloc à la fois.

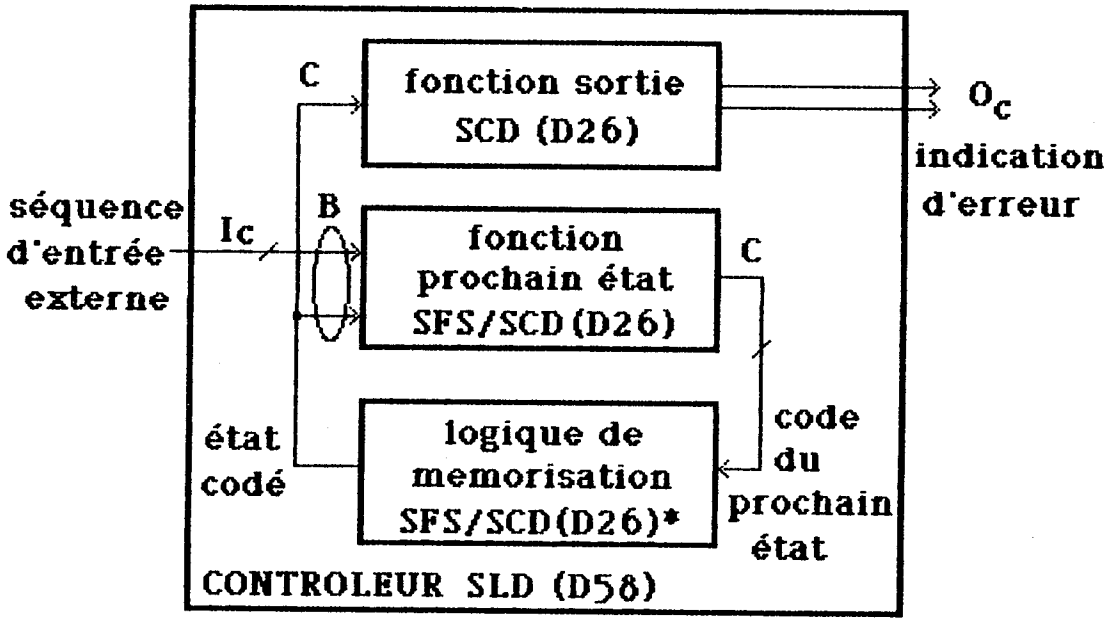



Figure 68 : Proposition pour la structure interne d'un contrôleur SLD (D58) P23 - modèle de Moore

La propriété SCD de la fonction de sortie et de la logique de mémorisation se réfère au code C. La logique de mémorisation est aussi SFS pour le code C. La fonction prochain état est SCD et SFS vis-à-vis du code d'entrée B et du code de sortie C. Pour tous ces blocs, la définition forte des contrôleurs SCD est utilisée.

**Démonstration de P23**

Cette proposition est dérivée de P15. Pour les cas d'une seule faute soit au contrôleur soit à ses entrées, la démonstration est égale à celle de P15. Pour une séquence erronée présentée aux entrées du contrôleur et une faute existante dans le contrôleur, les conséquences possibles selon le bloc

affecté sont décrites par un des cas qui suivent.

Cas 1:

Une faute $f_k$ se produit dans le bloc de fonction de sortie. Ceci étant SCD selon la définition D26, produit des sorties hors code correspondantes aux entrées hors code reçues, même en présence d'une faute dans ce bloc. Ces sorties en dehors du code reçues par le bloc de fonction sortie sont résultantes de l'application du vecteur d'entrée externe (hors code) aux blocs de fonction prochain état et de logique de mémorisation qui sont SCD, donc produisent également des sorties hors code. Ainsi, le défaut est détecté tout de suite.

Cas 2:

Une faute $f_k'$ survient au bloc de fonction prochain état. Ce bloc, étant SCD par D26, assure qu'au vecteur hors code présenté à ses entrées correspond une sortie hors code, même avec la présence d'un défaut dedans; cette sortie appliquée à la logique de mémorisation résultera aussi en un vecteur hors code (par la propriété SCD), lequel sera signalé par le bloc de fonction sortie.

Cas 3:

Un défaut $f_k''$ survient dans le bloc de mémorisation. Si une faute se produit aux entrées, le bloc de fonction prochain état reçoit un vecteur hors code et génère des sorties hors code (par la propriété SCD); celles-ci seront transférées aussi comme une valeur hors code par la logique de mémorisation puisqu'elle est SCD selon la définition D26. Ce vecteur en dehors du code reçu par la fonction sortie résulte en une indication d'erreur.

Q.E.D.

**Proposition P24**

La propriété "strongly language disjoint" d'un contrôleur SLD défini par D58 (définition forte) et conçu en blocs, sera assurée s'il est composé de :

- un bloc combinatoire SFS/SCD, étant la propriété SCD déterminée par la définition D26,
- une logique de mémorisation SFS*,
- un contrôleur interne que vérifie le code des états aux entrées de la logique de mémorisation,

l'hypothèse H10' étant assurée.

La structure interne du contrôleur dont les blocs composants sont décrits par la proposition P15 est montrée à la figure 69.

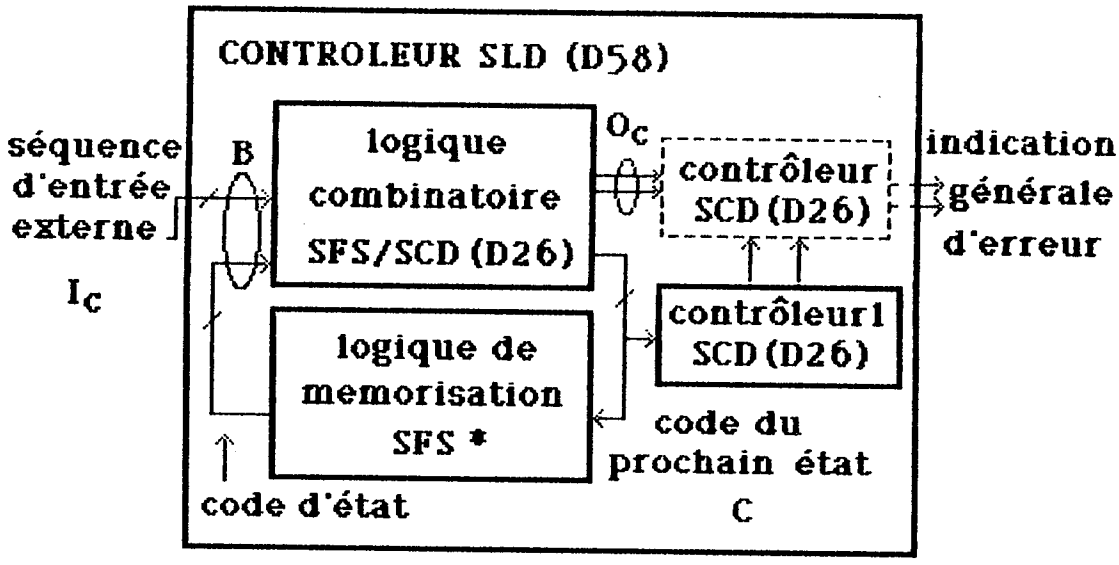



Figure 69: Structure interne du contrôleur SLD (D58) selon P24

La propriété SCD du bloc de la logique combinatoire se réfère au code d'entrée B et aux codes de sortie C et $O_C$. Il est SFS pour le code d'entrée B et le code de sortie C. Le contrôleur1 est SCD vis-à-vis du code C. Pour tous ces blocs, la propriété SCD forte (D26) des contrôleurs combinatoires est considérée. Un contrôleur additionnel peut être utilisé pour réduire le nombre des lignes de sortie. Celui-ci sera SCD pour le code $O_C$ et pour le code de sortie du contrôleur1.

**Démonstration de P24**

Cette proposition est dérivée de P18. Pour le cas d'une seule faute soit au contrôleur soit à ses entrées, la démonstration est égale à celle de P18. La séquence erronée présentée aux entrées du contrôleur concaténée au code de l'état aux entrées de la logique combinatoire sera un vecteur hors code pour le contrôleur.Le défaut survenant au contrôleur et les différentes conséquences selon le bloc affecté, peuvent être décrits par un des cas suivants.

Cas 1:

La faute $f_k$ dans le contrôleur se produit dans la logique combinatoire.

Ce bloc est SCD (par la définition forte) vis-à-vis du code d'entrée B et des codes de sortie C (code des états) et $O_C$ (indication d'erreur). Cette propriété assure que $\forall$ y $\in$ B, $G(y,<f_1,f_2,.....,f_k>) \notin O_C$ , donc une indication d'erreur sera générée aux sorties du circuit.

Cas 2:

La faute $f_k$ dans le contrôleur affecte la logique de mémorisation. La logique de mémorisation reçoit des entrées correctes puisque des sorties de la logique combinatoire en dehors du code seraient détectées par le contrôleur1 SCD. Grâce à la propriété SFS, si des sorties erronées sont produites, elles sont des sorties hors code - par conséquent, elles seront détectées par la logique combinatoire qui est SCD (D26).

Cas 3:

Un défaut affectant, à la fois, des lignes des blocs combinatoires et de mémorisation survient. Les sorties de, seulement, un des blocs est affectée à la fois - donc cette erreur sera détectée par le contrôleur1 interne ou signalée par les sorties d'indication d'erreur du bloc combinatoire.

Q.E.D.

**Proposition P25**

La propriété "strongly language disjoint" en une conception en blocs d'un contrôleur SLD défini par D58 (définition forte), sera assurée s'il se compose de :

- une logique combinatoire SFS/SCD (D26),
- une logique de mémorisation SFS/SCD (D26)",

étant considérée l'hypothèse H10'.

La structure interne du contrôleur composé selon la proposition P25 est montrée à la figure 70. La logique combinatoire est SCD vis-à-vis du code d'entrée B et des codes de sortie C et $O_C$. Ce bloc est SFS pour le code d'entrée B et le code de sortie C. Les propriétés SCD et SFS de la logique de mémorisation se réfèrent au code d'entrée C et au code de sortie B. La propriété forte des contrôleurs SCD est considerée pour les deux blocs.

**Démonstration de P25**

Cette proposition dérive de P24, le contrôleur1 étant substitué par la propriété additionnelle SCD (D26) de la logique de mémorisation, laquelle

assure la propagation des vecteurs hors code (C) même s'il y a une faute qui affecte la logique de mémorisation

Q.E.D.

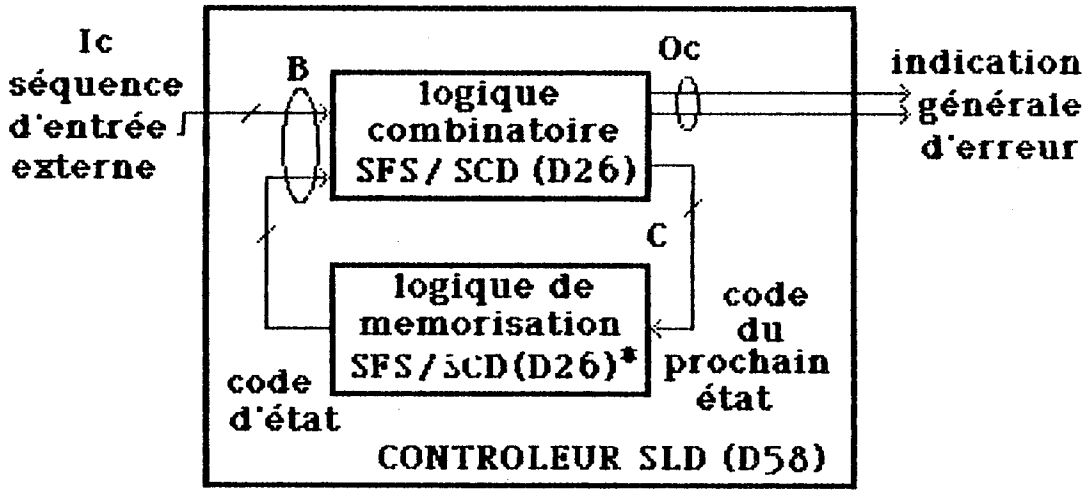



Figure 70 : Structure interne du contrôleur SLD selon la proposition P25

IV.4.4.2. Conception du bloc combinatoire avec des PLAs

Les propositions P14 à P25 sont basées sur la conception de blocs combinatoires qui reçoivent des entrées appartenant ou non à un code d'entrée B, produisent des sorties appartenant ou non aux codes de sortie C et $O_C$, et qui ont des propriétés SFS, SFS/SCD, ou SFS/SCD(D26). Nous examinons, ci-dessous, la conception de ces blocs à partir de PLAs.

En [NIC 84a] sont présentées des règles et des hypothèses de pannes pour la conception de PLAs SFS pour les cas d'erreurs simples, unidirectionnelles et multiples. L'analyse est basée sur des hypothèses de pannes de bas niveau et leur construction est détaillée dans cette référence. Nous reproduisons ici les conclusions de cette étude afin d'en utiliser aussi les règles pour notre circuit.

La structure NOR-NOR et la technologie NMOS sont considérées. La condition que le PLA soit "non-concurrent" n'est pas nécessaire. Les modèles initiaux d'implémentation pour les première et deuxième matrices, respectivement (71a. et 71b.) sont illustrés par la figure 71. Les entrées sont notées $I_0$, $I_1$,....., $I_k$ et les sorties produites, $B_1$, $B_2$,.....,$B_n$.

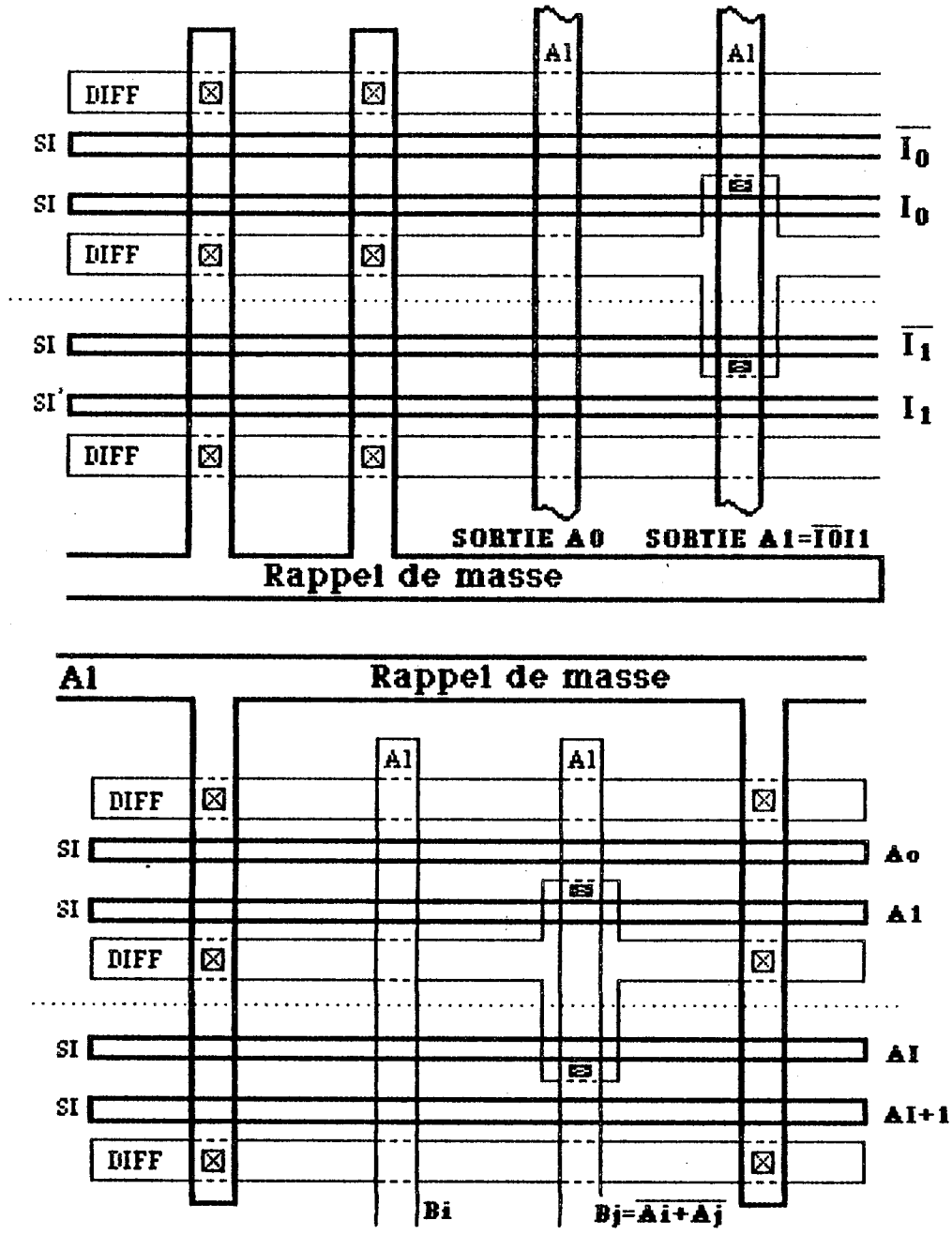



Figure 71: Implémentation du PLA. a) première matrice; b) deuxième matrice.

a) PLAs SFS pour un code détectant des erreurs simples

Le PLA est divisé en trois blocs: les entrées primaires, les monômes (colonnes), et les sorties primaires (lignes). Chacun de ces blocs est vérifié directement par un contrôleur de parité; par conséquent le degré de divergence maximal du bloc pour l'ensemble de ses lignes est égal à 1. Le degré de divergence maximal d'un bloc fonctionnel pour un ensemble de

lignes internes $I_1$, $I_2$,....., $I_k$ est égal au maximum des degrés de divergences effectifs des lignes $I_1$, $I_2$,....., $I_k$ (ceux-ci étant le nombre maximal des sorties primaires du circuit qui peuvent être connectées avec la ligne $I_i$ par l'intermédiaire des chemins sensibilisés par un vecteur d'entrée a ϵ A.

En raison de l'arrangement inhérent du PLA, les coupures et les courts-circuits des lignes d'alimentation peuvent produire des erreurs unidirectionnelles multiples affectant plusieurs sorties primaire. Cette possibilité donc est enlevée des hypothèses de pannes.

A l'implémentation du PLA, deux lignes d'entrée, deux monômes ou deux sorties ne peuvent pas assumer la même valeur en permanence. Ainsi n'importe quelle séquence de défauts, des courts circuits compris, sera détectés par le contrôleur de monômes, et le circuit ne sera pas "strongly redundant" pour la séquence.

Les monômes qui ne sont pas vérifiés directement par le contrôleur devront être examinés. Les courts-circuits entre deux lignes affectant une sortie qui ne sont pas détectables, devront être éliminés (par l'introduction d'autre lignes entre elles, par exemple).

Il faut aussi ajouter des circuits pour la génération des bits additionnels utilisés dans la codification de parité des monômes et des sorties. L'addition de ces circuits introduira des monômes qui ne sont pas testés directement par le contrôleur des monômes. Dans ce cas là, il est nécessaire d'introduire des lignes afin d'éviter des courts-circuits entre ces nouveaux monômes et d'autres qui pourraient résulter en des sorties multiples erronées. Le PLA augmenté est montré à la figure 72.

**Proposition P26**

Un PLA conçu selon la procédure décrite ci-dessus (à l'item a.) est "strongly fault secure" (SFS) pour des erreurs simples (les coupure de lignes d'alimentation étant enlevées des hypothèses de pannes).

Cette proposition est évidente à partir de l'explication donnée à l'item a. La démonstration complète est présentée en [NIC 84a].

b) PLAs SFS pour un code détectant des erreurs unidirectionnelles

En raison de l'organisation des PLAs, tous les chemins entre les lignes d'une même matrice et les sorties du circuit ont la même parité d'inversion.

Ainsi, le circuit est testé en utilisant le contrôle des entrées primaires et un code de sortie non ordonné généralisé.

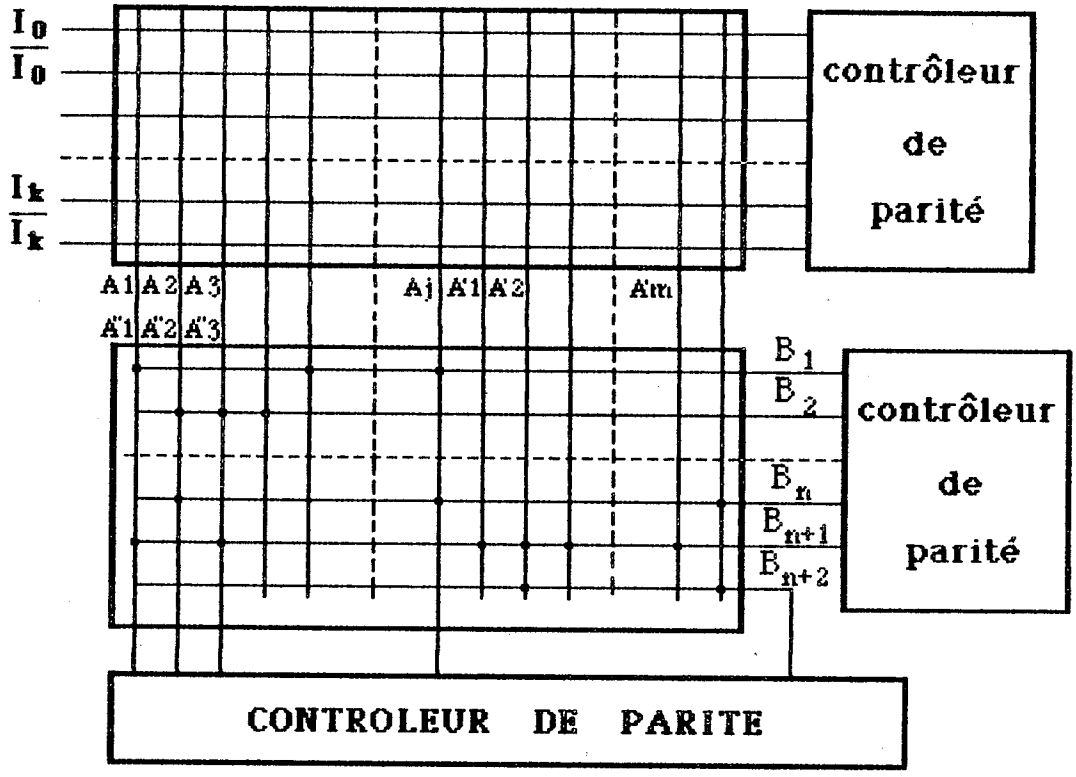



Figure 72 : PLA augmenté

Les lignes d'alimentation de la première matrice alimentent des monômes - pour ceux-ci, la parité d'inversion du chemin jusqu'aux sorties primaires est égale à 1. Les lignes d'alimentation de la deuxième matrice alimentent les sorties du PLA - pour celles-ci, la parité d'inversion est égale à zéro. Dans cette situation, la propagation des erreurs causés par des coupures des lignes d'alimentation est assurée.

Les courts-circuits ne posent pas de problèmes si l'on considère la classe de pannes et la structure du PLA suggérées : des courts-circuits entre des entrées et des monômes, ou entre des monômes et des sorties n'iront pas se produire puisque des matériaux non court-circuitables sont utilisés pour chaque groupe.

Les courts-circuits entre des lignes qui alimentent des portes avec des parités d'inversion différentes ne se produiront pas parce que chacune des matrices a la parité d'inversion uniforme (comme il a été vérifié en cas de

coupures).

Les sorties additionnelles pour les codes non ordonnés devront aussi être générées par le PLA.

**Proposition P27**
Un PLA conçu selon la procédure décrite ci-dessus (item b.) est "strongly fault secure" (SFS) pour les erreurs unidirectionnelles.

La proposition P27 peut être vérifiée directement par la justification présentée avant la proposition. La démonstration complète est donnée en [NIC 84a].

c) PLAs SFS pour un code détectant des erreurs multiples

Deux PLAs séparés sont nécessaires pour un code à duplication, afin d'éviter la propagation d'erreurs jusqu'aux deux groupes de sorties du PLA. Ce problème ne va pas se produire si un code double-rail ou un code de Berger est utilisé. Donc, le PLA est construit comme dans les cas précédents, avec un contrôleur interne, et des sorties appartenant à un code donné.

**Proposition P28**
Un PLA conçu selon une des méthodes que l'on vient d'énoncer (item c.) est "strongly fault secure" (SFS) pour des erreurs multiples.

La démonstration de la proposition P28 est donnée en [NIC 84a].

Donc, les propositions ci-dessus étant respectées, nous aurons des PLAs "strongly fault secure" pour des erreurs simples, unidirectionnelles ou multiples, respectivement.

Parmi les propositions pour la conception en blocs des contrôleurs SLD présentées dans la sous-section IV.4.4.1., nous trouvons des blocs combinatoires ayant les propriétés : SFS, SFS/SCD, SFS/SCD vis-à-vis d'un seul code de sortie ($O_C$ ou C), ou SFS/SCD (D26). Les propositions P26 à P28 donnent une méthode de conception pour des PLAs SFS, selon [NIC 84a], mais il faut ajouter des propriétés pour que ces PLA puissent être utilisés.

Afin d'avoir des PLAs SFS qui soient aussi SCD, deux conditions additionnelles sont suffisantes :

1) toutes les entrées devront être programmées, i.e., toutes les entrées

appartenant au code d'entrée produisent des sorties appartenant au code de sortie et toutes les entrées hors code produisent des sorties hors code ;

2) l'analyse des séquences de défauts qui peuvent survenir (en fonction de la propriété "strongly redundant") doit considérer le concept de redondance vis-à-vis des contrôleurs, donc la fonction en peut pas changer ni pour des entrées appartenant au code ni pour celles hors code. Par conséquent, les fautes ou séquences de défauts qui seraient détectables uniquement avec l'application de vecteurs hors code ne peuvent pas avoir la chance de se produire (cette condition peut demander des changements du circuit initial). La proposition suivante se réfère aux PLAs SFS/SCD.

**Proposition P29**
Une PLA conçu selon la proposition P26 (ou P27 ou P28), lequel est programmé pour, en recevant des entrées dans le code, produire des sorties aussi dans le code, et en recevant des entrées hors code produire des sorties hors code, et où les fautes ou séquences de fautes qui ne pourraient être détectées que par des vecteurs hors code sont empéchées de survenir (par des changements topologiques), est un PLA SFS/SCD pour des erreurs simples (ou unidirectionnelles, ou multiples).

**Démonstration de P29**
L'hypothèse H9' d'occurrence de pannes étant considérée, les défauts se produisent soit aux entrées, soit dans le circuit , mais ils n'affectent pas les deux à la fois. Le cas des erreurs simples est considéré dans cette preuve; des extensions pour les cas d'erreurs unidirectionnelles et multiples peuvent être obtenues d'une façon analogue. Dans ce cas, les propositions P27 et P28 prennent la place de P26, et les codes employés pour les sorties doivent être conformes aux types d'erreurs respectives (par exemple, des codes de Berger pour des erreurs unidirectionnelles et double-rail, pour des erreurs multiples).

La première partie de la proposition P29 a été expliquée au moment de la présentation des propositions P26, P27 et P28. Ainsi, nous commençons à partir de PLAs SFS vis-à-vis des erreurs considérées.

Deux ensembles d'entrées peuvent être appliquées au PLA: des entrées codées et celles hors code.

Le PLA est SFS pour l'ensemble des entrées appartenant au code: donc, quand une de ces entrées est reçue par le PLA, même en présence d'un défaut dans le PLA, la sortie sera correcte (une sortie dans le code) ou elle

sera un vecteur hors code, lequel permet la détection du défaut.

Le PLA est SFS pour l'ensemble des entrées hors code. Par l'hypothèse H9', des défauts ne peuvent pas se produire dans le système fonctionnel et dans le contrôleur à la fois - donc, si nous avons un vecteur hors code aux entrées du PLA (lequel provient d'un bloc déjà affecté) il n'y aura pas une autre faute (pour laquelle le contrôleur est redondant) dans le PLA. Par la définition SFS, la sortie sera correcte - dans ce cas cela signifie la production de la sortie programmée, une sortie en dehors du code.

La restriction selon laquelle tous les défauts qui peuvent survenir dans le PLA doivent être détectables par des entrées appartenant au code, ou le PLA doit être redondant vis-à-vis d'eux, des entrées codées ou des entrées hors code étant appliquées, assure qu'en recevant une entrée hors code le PLA ne produit pas une entrée dans le code, même en présence de défauts dans le PLA (pour lesquels il est redondant).

Q.E.D.

Les PLAs qui doivent être SFS/SCD pour un des codes de sortie ($O_C$, par exemple) et seulement SFS pour l'autre code (C dans le cas exemplifié) sont conçus selon un des items suivants :

a) nous pouvons faire un seul PLA SFS/SCD, avec des propriétés additionnelles vis-à-vis du code C donc, cela n'est pas inconvenant;

b) nous séparons les fonctions qui sont SFS et SFS/SCD pour construire les PLAs correspondants selon les propositions précédentes. Dans le bloc combinatoire résultant, pourtant, nous n'avons pas besoin d'isoler les deux PLAs.

La conception des PLAs SFS/SCD (D26) est plus compliquée que les précédentes. Dans les circuits où la propriété SCD forte est nécessaire, les entrées hors code produisent des sorties hors code, même s'il y a, dans le circuit, une faute pour laquelle il n'est pas redondant. En fait, un défaut localisé dans le PLA peut masquer un vecteur erroné aux entrées. Donc, il faut soit ajouter des circuits pour la vérification des entrées avant qu'elles soient utilisées par le PLA SCD, soit introduire une forme de duplication interne permettant de détecter ces fautes arrivant à l'intérieur du circuit. Les deux solutions résultent en un accroissement de la surface par rapport au circuit obtenu , avec la proposition du PLA SFS/SCD.

IV.4.4.3. Conception de la logique de mémorisation

La logique de mémorisation du système suggéré est un registre de n cellules, où n est le nombre de bits utilisé pour la codification des états. Ce registre reçoit l'information du bloc combinatoire, et au moment où le signal d'horloge est activé, il transfère les mêmes valeurs à ses sorties, n'éxécutant ancune logique de transformation. Par hypothèse, l'entrée reçue est correcte; si elle ne l'est pas, le bloc ne fait que propager l'erreur. Mais ce deuxième cas n'a pas besoin d'être considéré à côté d'une existence possible d'un défaut dans le registre en raison des hypothèses de pannes considérées, dans le cas où une logique de mémorisation SCD/SFS est utilisée.

S'il y a une faute dans le registre, soit les fautes produites sont égales à celles qui seraient obtenues en fonctionnement normal soit elles devront être hors code, ceci étant la propriété "fault secure". Donc, ce registre doit être SFS.

Dans [NIC 84a], un registre repris à l'unité opérationnelle du microprocesseur 68000 est analysé, concernant la propriété SFS. Il est montré que ce registre, en raison de son implémentation et parce qu'il reçoit toutes les combinaisons possibles d'entrée, est "strongly fault secure". Les conditions fondamentales pour cette propriété (SFS) sont assurées puisque :

a) la caractéristique fonctionnelle intrinsèque du registre (indépendance de chaque paire entrée/sortie) assure qu'il y a un seul chemin pour chaque entrée-sortie, donc le défaut survenu est propagé comme une erreur simple à la sortie correspondante;

b) des erreurs multiples unidirectionnelles causées par des coupures dans les lignes d'alimentation sont détectées car ces lignes après avoir été utilisées pour toutes les cellules du registre ont leurs extrémités connectées aux contrôleurs; ceux-ci signalent immédiatement le défaut s'ils cessent de recevoir l'alimentation;

c) les courts-circuits entre deux lignes d'alimentation du même type connectées par des chemins à deux sorties primaires différentes, lesquels pourraient être l'origine de séquences de fautes résultant en erreurs multiples, n'arriveront pas car les lignes de $V_{SS}$ et $V_{DD}$ sont implémentées alternées.

Il y a encore la possibilité de court-circuit entre une ligne de signal et une ligne de sortie de cellules différentes d'un même registre. Soit cette situation ne causera pas d'erreurs, soit une seule erreur sera produite aux sorties pour chaque vecteur d'entrée permettant la détection. Ainsi, les lignes d'entrée étant activées avec toutes les valeurs possibles, comme avec l'utilisation d'un code de parité, par exemple; ce défaut sera détecté à travers la production d'une sortie hors code (avec une erreur simple).

Dans le cas où le code d'entrée ne possède pas toutes les combinaisons possibles d'entrée, comme pour les codes 1-parmi-n par exemple, l'analyse individuelle des cellules implémentées est nécessaire. Un défaut peut affecter une paire de cellules, et il n'est, peut être, détectable qu'avec une des combinaisons qui n'est pas appliquée. Donc, il faut vérifier quelles sont ces combinaisons. A partir du dessin des cellules, on liste les défauts pouvant survenir entre en deux cellules, lesquels seraient détectables avec les valeurs non appliquées pendant le fonctionnement normal. La possibilité d'occurrence de ces fautes devra être éliminée par des changements du "layout" de la cellule, ou par l'utilisation de matériaux non court-circuitables, ou par le déplacement des lignes affectées. Si l'on prend l'exemple d'un code 1-parmi-n, une combinaison qui ne sera pas reçue par deux cellules voisines est 11 ;

si n = 4, nous aurons donc 0001 ou
0010 ou
0100 ou
1000 aux entrées.

La logique de mémorisation telle qu'elle est décrite ci-dessus n'est pas suffisante pour les cas où la propriété SCD forte est demandée. Dans ce cas il va falloir que des circuits additionnelles soient ajoutés (pour la duplication de certains morceaux ou pour le contrôle des entrées - comme nous l'avons vu dans le cas des PLAs) afin d'assurer la détection de la double faute.

### IV.4.4.4. Procédure générale de conception

Une procédure qui peut être suivie pour la construction d'un contrôleur SLD, est donnée par la suite. La structure interne peut être choisi parmi celles décrites par les propositions P14 à P25. Les entrées du contrôleur sont des sorties d'un bloc fonctionnel SeSC, lesquelles sont décrites par un langage régulier. Une paire de lignes codées en double-rail constituent les sorties du contrôleur.

En voilà les étapes :

1) Dessiner un automate déterministe qui reconnaît le langage reçu (comme nous l'avons vu à la section IV.4.2.). Cet automate est LD.

2) Définir quelles sont les hypothèses de pannes admises et le type respectif d'erreurs qui peuvent se produire : simple, unidirectionnelle, ou multiple. Donc, un code ayant la capacité correspondante de détection devra être choisi. Si la structure interne considérée utilise des contrôleurs internes SCD, il est intéressant à employer un code pour lequel le contrôleur est déjà connu - dans le cas contraire il faudra le concevoir aussi.

3) Définir le code des états en fonction du nombre d'états de l'automate et du code choisi.

4) Ecrire les équations logiques pour la fonction prochain état. Si le PLA est SCD, toutes les combinaisons d'entrée seront spécifiées pendant la programmation pour assurer que toutes les entrées codées (hors code) correspondent à des sorties codées (hors code).

5) Construire le(s) PLA(s) qui exécute(nt) la(les) logique(s) combinatoire(s) selon les propositions présentées à la sous-section IV.4.4.2. Les différences internes des PLAs concernant les erreurs qui peuvent se produire - simples, unidirectionnelles ou multiples - ont été données, respectivement, par les propositions P26, P27 et P28.

6) Les registres sont construits comme nous l'avons vu dans la section précédente.

Cette procédure est utilisable pour les cas des machines décrites, aussi bien par le modèle de Mealy que celui de Moore. Pourtant, nous pouvons remarquer des différences concernant les niveaux de difficulté, parfois distinctes. La fonction de sortie éxécutée par un contrôleur SCD, lequel analyse le code en provenance de la logique de mémorisation avec le code des entrées - pour une machine de Mealy - et seulement les sorties du bloc de mémorisation - pour le modèle de Moore - peut être plus simple dans ce deuxième cas. Dans le modèle de Mealy, il n'est pas sûr que la concaténation du code des états avec les vecteurs d'entrée résultera en un code standard. La conception originale d'un contrôleur SCD pour ce nouveau code peut introduire des problèmes difficiles à résoudre.

## IV.5. CONCLUSION

Les systèmes séquentiels sûrs - le cas le plus général parmi les circuits

sûrs - ont été examinés dans ce chapitre. Tout d'abord nous avons vérifié la généralité de leurs définitions qui peuvent être utilisées afin d'obtenir les définitions correspondentes pour les circuits et contrôleurs combinatoires.

Les sorties obtenues de ces systèmes qui suivent une propriété sequentielle, sont testées par de contrôleurs egalement séquentiels. Dans les machines séquentielles sûres utilisées auparavant, les sorties étaient combinatoires, et le code des états et/ou des sorties était vérifié. Ici, les contrôleurs "strongly language disjoint" ont été définis. Ils vérifient une langage de sortie et gardent la propriété "language disjoint" même en présence de pannes pour lesquelles le circuit est redondant.

La conception de ces contrôleurs est étudiée ensuite. Nous avons montré la possibilité de les construire à partir de structures régulières et/ou de circuits dont la conception avait été étudiée precedement. Plusieures possibilités sont examinées, en utilisant des blocs de base avec des propriétés diverses dans chaque cas. Les propositions présentées peuvent être utilisées comme départ pour des configurations synchrones ou asynchrones, puisque nous n'avons pas particularisé les conclusions pour aucun de ces cas. Dans le cas synchrone, des procedures additionnelles pour le contrôle de la ligne d'horloge du contrôleur sont nécessaires, mais le problème est plus simple que pour le cas asynchrone, en général.

La conception de blocs fonctionnels "sequentially self-checking" par des procedures similaires à celles suggerées par des contrôleurs SLD et l'utilisation de structures régulières de base reste à être étudiée. Des applications, en particulier, dépendentes de la technologie (comme la conception des PLAs et registres) et présentées ici en fonction de la technologie NMOS, doivent être étendues pour la technologie CMOS.

# Chapter 5:

## Conclusion

Nous avons étudié dans les chapitres précédents, les divers types de contrôleur:

- les "totally self-checking" qui ont été proposés conformes à la théorie classique;

- les "strongly code disjoint" qui composent la plus large classe des contrôleurs utilisés dans des systèmes combinatoire;

- et enfin, nous avons défini les contrôleurs "strongly language disjoint", nécessaires aux systèmes générant des sorties avec une propriété séquentielle.

Nous allons revenir sur chacun des groupes séparément par la suite.

Deux aspects de la théorie classique nous intéressent en particulier. Le premier se réfère aux codes employés pour l'information aux sorties du bloc fonctionnel (donc, les codes d'entrée des contrôleurs) car ils continuent toujours à être utilisés. Le choix en vue de leur capacité de détection est fonction des hypothèses de pannes considérées et des erreurs qui peuvent se produire parmi ces lignes d'information. Le deuxième aspect concerne les circuits logiques proposés par divers auteurs dans la théorie classique : nous avons remarqué qu'ils peuvent nous servir de base pour le début de l'activité de conception. Il est aussi possible d'améliorer ces propositions (parfois, on peut arriver à diminuer le nombre de transistors) ou peut être de trouver une solution dans les cas où les circuits TSC n'existaient pas.

Mais, en ce qui concerne la conception des circuits et leur implémentation, nous quittons la théorie classique pour passer aux contrôleurs nommés "Strongly Code Disjoint". Définis par Nicolaidis et Courtois, ils sont associés à des hypothèses de pannes analytiques (réelles), étant reconnu que le modèle de collage logique n'est pas représentatif des fautes possibles, en fait. Nous avons, donc, examiné la conception des contrôleurs SCD, et plus précisément, de ses aspects cellulaires. La modularité est aujourd'hui un des buts dans l'architecture des circuits. Par conséquent, une bibliothèque de cellules des contrôleurs est intéressante pour la construction des nouveaux, adaptés aux besoins de chaque système (type de code, nombre de bits, arrangement selon l'espace disponible). Il faut, pour cela, respecter certaines contraintes dans la conception des cellules ainsi que pour leur assemblage, de même pour l'architecture du circuit du contrôleur. Les règles et les propositions présentées ici assurent la propriété SCD du circuit résultant et peuvent être une aide assez utile au concepteur qui commence dans ce domaine. Ensuite, une fois acquise

l'expérience initiale, il peut facilement reconnaître les aspects qui sont restrictifs ou inconvenables et pas toujours nécessaires, pouvant ainsi proposer d'autres dessins, meilleurs. Deux cas particuliers ont été analysés en guise d'exemple : des contrôleurs double-rail et de parité.

Une nouvelle hypothèse d'occurrence de pannes dans un système a été développée. Nous analysons le cas de pannes concomitantes dans le bloc fonctionnel et aussi dans le contrôleur. Ceci doit respecter la propriété SCD forte. Des exceptions à ce cas ou des situations propres aux contrôleurs double-rail et de parité ont été examinées dans les sections d'applications particulières à chacun des types.

Cette étude concernant les contrôleurs de code a été complémenté avec l'analyse d'une structure multicontrôleur, définie vis-à-vis des sytèmes où les fonctions sont partagées parmi plusieurs blocs fonctionnels - chacun d'eux étant vérifié par un contrôleur. Les hypothèses d'occurrence de pannes dans ce système sont examinées pour des applications en ligne aussi bien qu'hors ligne.

Enfin, les contrôleurs "strongly language disjoint" ont été définis - ils composent la plus large classe de contrôleurs, lesquels associés à des circuits "sequentially self-checking" composent des sytèmes séquentielles qui atteignent le "TSC goal", sous certaines hypothèses de pannes. Ils vérifient des entrées qui suivent une propriété séquentielle et assurent la propriété "language disjoint" même quand les défauts, pour lesquels le circuit est redondant, se sont déjà produits. Le circuit n'étant pas redondant vis-à-vis du défaut survenu, la propriété "language-disjoint" est assurée jusqu'à la détection. Les contrôleurs séquentiels définis auparavant assuraient cette propriété uniquement jusqu'à l'occurrence du premier défaut.

Les propriétés déterminées par les définitions sont respectées sous certaines hypothèses. Dans le cas général, on suppose l'application d'un ensemble de séquences pendant le laps de temps qui sépare l'occurrence de deux défauts. Il faut garantir la correction des sorties jusqu'au moment de détection de la faute (ou séquence de défauts). En fait, les hypothèses utilisées pour le cas séquentiel demandent l'application des séquences pour lesquels la machine éxécute toutes les transitions possibles de son diagramme d'états, ainsi les fautes sont détectées, n'important pas l'état durant lequel elles sont survenues. Toute faute pour laquelle le contrôleur n'est pas redondant, doit être détectée en une de ces transitions possibles.

Au début, une machine ayant ces caractéristiques peut sembler

difficile à concevoir, mais nous avons démontré que l'on peut utiliser dans sa structure interne, des blocs autotestables examinés précédemment, que l'on sait construire. Plusieurs propositions suggèrent l'emploi des blocs combinatoires SFS, SCD ou SFS/SCD pour la conception des contrôleurs SLD (définis selon les définitions faible ou forte), sans aucun préjudice vis-à-vis de leurs propriétés séquentielles.

La conception des blocs de logique combinatoire dans les contrôleurs SLD peut être faite par les mêmes techniques examinées dans le chapitre III, concernant les contrôleurs SCD. Cependant, nous avons aussi étudié la possibilité de les concevoir avec l'utilisation de PLAs. Des propositions d'ajustement des PLAs pour le cas SCD ont été énoncées. Reste à vérifier si les dessins obtenus ne peuvent pas être réduits, tout en gardant les propriétés nécessaires.

Nous n'avons pas considéré le cas des pannes qui affectent le circuit d'horloge, ni des lignes de signal de "charge" dans les registres ou circuits de mémorisation. Nous avons supposé qu'elles n'y surviennent pas, afin de simplifier l'analyse. Pourtant cette hypothèse peut être enlevée dès que des procédures additionnelles de vérification sont ajoutées. Pour le signal d'horloge, par exemple, les circuits de génération seraient dupliqués, les valeurs des deux étant comparées après son utilisation par les divers blocs. Cela évite que des erreurs multiples généralisées se produisent à cause d'une panne dans la ligne d'horloge, et ne soient pas détectées. Des observations analogues peuvent être faites concernant les lignes de signal de charge. De même, il est intéressant d'avoir ces circuits isolés électriquement des autres utilisés pour les fonction logiques.

Enfin, à propos des contrôleurs SLD, il faut aussi remarquer qu'un dessin totalement cellulaire recursive - comme celui des contrôleurs SCD double-rail et des arbres de parité - n'est pas réalisable dans le cas séquentiel. Nous aurons, au moins, les deux types de blocs de base participant à la composition du tout - le combinatoire et la logique de mémorisation. Des extensions des idées ici examinées pour les contrôleurs SLD peuvent aussi être étudiées en vue de la conception des circuits fonctionnels SeSC, car leur dessin au niveau physique n'a pas encore été défini.

Toutes les conclusions et les observations faites dans ce document concernant la conception des contrôleurs en général, les considèrent comme des structures isolées des autres blocs composants du système, il n'y a donc pas de fautes affectant à la fois un contrôleur et une ligne quelconque voisine. Les règles et les propositions ont toujours considéré la technologie

NMOS et devront être revues pour la technologie CMOS, et, pour les cas d'interconnexions à multiniveaux, en fonction des hypothèses de pannes propres à chacune des situations.

# REFERENCES

[AND 71] ANDERSON D.A.
"Design of self-checking digital networks using coding techniques". Urbana, Coordinated Science Laboratory / Université d'Illinois, Septembre 1971. (Rapport n°527). 133p.

[AND 73] ANDERSON D.A.
"Design of totally self-checking check circuits for m-out-of-n codes". IEEE Transactions on Computers. New York, IEEE, 22(3) : 263-9, Mars 1973.

[ASH 76] ASHJAEE J.M., REDDY S.M.
"On totally self-checking checkers for separable codes". The 6th Annual International Conference on Fault-Tolerant Computing, Pittsburgh, Juin 21-23, 1976. Digest of Papers. New York, IEEE,1976. p.151-6.

[BER 61] BERGER J.M.
"A note on error detection codes for asymmetric channels". Information and Control. New York, 4(1) : 68-73, Mars 1961. p.68-73.

[BRE 76] BREUER M.A., FRIEDMAN A.D.
"Diagnosis and reliable design of digital systems". London, Pitman Publishing Limited,1976.

[CAR 68] CARTER W., SCHNEIDER P.
"Design of dinamically checked computers". In: IFIPS Congress, Edinburgh,1968. Information Processing 68. Amsterdam, North Holland, 1969. v.2, p.878-83.

[CHU 78] CHUANG H., DAS S.
"Design of fail-safe sequential machines using separable codes". IEEE Transaction on Computers. New York, IEEE, 27(3) : 249-52, Mars 1978.

[COU 81] COURTOIS B.
"Failure mechanisms, fault hypotheses and analytical testing in LSI NMOS (HMOS) circuits". VLSI 81, Université d'Edinburgh, Academic Press, 1981.

[CRO 78] CROUZET Y.
"Conception de circuits à large échelle d'intégration totalement autotestables". Toulouse, Université Paul Sabatier de Toulouse,

1978. (Thèse n.35).

[DAV 77] DAVID R., FOSSE P.
"Totally self-checking modular asynchronous circuits". The 7th Annual International Conference on Fault-Tolerant Computing, Los Angeles, Juin 28-30, 1977. Digest of papers. New-York, IEEE ,1977. p.193 (version reduite)

[DAV 78a] DAVID R.
"A totally self-checking 1-out-of-3 checker". IEEE Transactions on Computers. New York, IEEE, 27(6) : 570-2, Juin 1978.

[DAV 78b] DAVID R., THEVENOD/FOSSE P.
"Design of totally self-checking asynchronous modular circuits". Design automation and fault tolerant computing. New York, 2(4): 271-87, Octobre 1978.

[DAV 79] DAVID R., THEVENOD-FOSSE P.
"Detection transition sequences : application to random testing of sequential circuits". The 9th Annual International Conference on Fault-Tolerant Computing, Madison, Juin 20-22, 1979. Digest of papers. New-York, IEEE, 1979, p.121-4.

[DEN 78] DENNING P., DENNIS J., QUALITZ J.
"Machines, languages and computation". Englewood Cliffs, Prentice-Hall, 1978.

[DIA 74a] DIAZ M.
"Conception de systèmes totalement auto-testables et à pannes non-dangereuses". Toulouse, Université Paul Sabatier de Toulouse, 1974. (Thèse n° 618).

[DIA 74b] DIAZ M.
"Design of totally self-checking and fail safe sequential machines". The 4th Annual International Symposium on Fault-Tolerant Computing, Urbana, Juin 1974. Digest of papers. New-York, IEEE, 1974. p.3/19-24.

[DIA 79] DIAZ M., AZEMA P., AYACHE J.
"Unified design of self-checking and fail-safe combinational circuits and sequential machines". IEEE Transactions on Computers. New York, IEEE, 28(3) : 276-81, Mars 1979.

[DON 82] DONG H.
"Modified Berger codes for detection of unidirectional errors". The 12th Annual International Symposium on Fault-Tolerant Computing", Sta. Monica, Juin 22-24, 1982. Digest of Papers. New York, IEEE, 1982. p. 317-20.

[GAI 83] GAITANIS N., HALATSIS C.
"A new design method for m-out-of-n TSC codes". IEEE Transactions on Computers. New York, IEEE, 32(3) : 273-83, Mars 1983.

[GAL 80] GALIAY J., CROUZET Y., VERGNIAULT M.
"Physical versus logical fault models MOS LSI Circuits". IEEE Transactions on Computers. New York, IEEE, 29(6) : 527-31, Juin 1980.

[JAN 83a] JANSCH I., COURTOIS B.
"Design of checkers based on analytical fault hypotheses (preliminary report)". Grenoble, IMAG-TIM3/INPG, Mars 1983. (Rapport RR n°379). 56p.

[JAN 83b] JANSCH I., COURTOIS B.
"Design of checkers based on analytical fault hypotheses ". In: III Simposio Brasileiro de Microeletrônica, Sao Paulo, Juillet 26-28, 1983. Textes des Conférences. Sao Paulo, USP, 1983.

[JAN 84a] JANSCH I., COURTOIS B.
"Design of SCD checkers based on analytical fault hypotheses". In: ESSCIRC'84, Edinburgh, Septembre 19-21, 1984. Proceedings. Edinburgh, CEP Consultants Ltd., 1984. p.109-12.

[JAN 84b] JANSCH I., COURTOIS B.
"Strongly language disjoint checkers". Grenoble, IMAG-TIM3/INPG, Octobre 1984. ( Rapport RR n°468). 83p.

[JAN 84c] JANSCH I., COURTOIS B.
"SCD checkers, cellular checkers and multi-checkers structures". Grenoble, IMAG-TIM3/INPG, Novembre 1984. (Rapport RR n°476).

[MAK 82] MAK G.P., ABRAHAM J.A., DAVIDSON E.S.
"The design of PLAs with concurrent error detection".The 12th Annual International Symposium on Fault-Tolerant Computing, Sta Monica, Juin 22-24, 1982. Digest of papers. New York, IEEE, 1982.

p. 303-10.

[MAR 77] MAROUF M.A., FRIEDMAN A.D.
"Efficient design of self-checking checkers for m-out-of-n codes". The 7th Annual International Conference on Fault-Tolerant Computing, Los Angeles, Juin 28-30, 1977. Digest of papers. New-York, IEEE ,1977. p.143-9.

[MAR 78] MAROUF M.A., FRIEDMAN A.D.
"Design of self-checking checkers for Berger codes". The 8th Annual International Conference on Fault-Tolerant Computing, Toulouse, Juin 21-23, 1978. Digest of papers. New-York, IEEE, 1978. p.179-84.

[MEY 75] MEYER J., SUNDSTROM R.
"On-line diagnosis of unrestricted faults". IEEE Transactions on Computers. New York, 24(5) : 468-75, Mai 1975.

[NIC 83a] NICOLAIDIS M., COURTOIS B.
"Design of self-checking systems based on analytical fault hypotheses". Grenoble, IMAG/INPG, Mars 1983. (Report RR n°353). 66p.

[NIC 83b] NICOLAIDIS M., JANSCH I., COURTOIS B.
"Strongly code disjoint checkers". Grenoble, IMAG-TIM3/INPG, Novembre 1983. (Report RR n°404). 21p.

[NIC 84a] NICOLAIDIS M.
"Conception de circuits intégrés autotestables pour des hypothèses de pannes analytiques". Grenoble, INPG, Janvier 1984. (Thèse).

[NIC 84b] NICOLAIDIS M., COURTOIS B.
"Design of self-checking NMOS (HMOS) integrated circuits" In: Design for tactical avionics maintainability, AGARD Conference. Bruxelles, Mai 7-10, 1984. Proceedings (n°361). Neuilly-sur Seine, OTAN (NATO), 1984. p.13-1/13-20.

[NIC 84c] NICOLAIDIS M., JANSCH I., COURTOIS B.
"Strongly code disjoint checkers". The 14th International Symposium on Fault-Tolerant Computing, Kissimmee, Juin 20-22, 1984. Digest of papers. New-York, IEEE, 1984 . p.16-21.

[ÖZG 77] ÖZGUNER F.
"Design of totally self-checking asynchronous and synchronous sequential machines". The 7th Annual International Conference on Fault-Tolerant Computing, Los Angeles, Juin 28-30,1977. Digest of papers. New York, IEEE, 1977. p.124-9.

[PIE 83] PIESTRAK S.
"Design methods of totally self-checking checkers for m-out-of-n codes". The 13th Annual International Conference on Fault-Tolerant Computing, Milan, Juin 28-30,1983. Digest of papers. New York, IEEE, 1983. p.162-7.

[RED 74a] REDDY S.M., WILSON J.R.
"Easily testable cellular realizations for the (exactly P)-out-of-n and (p-or-more)-out-of-n logical functions". IEEE Transactions on Computers. New York, IEEE, 23(1) : 98-100, Janvier 1974.

[RED 74b] REDDY S.M.
"A note on self-checking checkers". IEEE Transactions on Computers. New York, IEEE, 23(10) : 1100-2, Octobre 1974.

[SMI 78] SMITH J.E., METZE G.
"Strongly fault secure logic networks". IEEE Transactions on Computers. New York, IEEE, 27(6) : 491-9, Juin 1978.

[TAK 73] TAKAOKA T., IBARAKI T.
"Fail-safe realization of sequential machines". Information and Control. New York, Academic Press, 22(1) : 31-55, Février 1973.

[TOH 71] TOHMA Y., OHYAMA Y., SAKAI R.
"Realization of fail-safe sequential machines by using a k-out-of-n code". IEEE Transaction on Computers. New York, 20(11) : 1270-5, Novembre 1971.

[VIA 79] VIAUD J.
"Circuits logiques séquentiellement autotestestables." Grenoble, INPG / Université de Grenoble, 1979. (Thèse).

[VIA 80] VIAUD J., DAVID R.
"Sequentially self-checking circuit". The 10th International Symposium on Fault-Tolerant Computing, Kyoto, Octobre 1-3, 1980. Digest of papers. New York, IEEE, 1980. p.263-8.

# AUTORISATION DE SOUTENANCE

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

VU les rapports de présentation de

Monsieur Bernard COURTOIS

Monsieur Michel DIAZ,

Mlle Ingrid Eleonora SCHREIBER JANSCH

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

Fait à GRENOBLE, le 4 Janvier 1985.

Le Président de l'I.N.P.G.
D. BLOCH
Président
de l'Institut National Polytechnique
de Grenoble

P.O. le Vice-Président,

## RESUME

Dans cette étude nous nous intéressons aux contrôleurs utilisés dans des systèmes autotestables, pour le test des sorties, combinatoires ou séquentielles, du bloc fonctionnel.

Deux classes de contrôleurs sont abordées : les "Strongly Code Disjoint" (SCD) qui vérifient une propriété combinatoire, et les "Strongly Language Disjoint" (SLD), où la propriété vérifiée est séquentielle.

Pour la première, nous examinons la conception des contrôleurs NMOS à partir de l'assemblage des cellules, des règles de conception pour celles-ci, et des hypothèses de pannes pouvant survenir dans les systèmes aussi bien que dans quelques structures spécifiques de contrôleurs.

Les contrôleurs "Strongly Language Disjoint" définis ici composent la plus large classe qui, associée à des circuits "sequentially self-checking", permet au système d'atteindre le "TSC goal" sous certaines hypothèses de pannes. Ils conservent la propriété "language-disjoint" même en présence de fautes. Des propositions pour la conception de ces contrôleurs sont également données -nous vérifions la possibilité de les construire à partir de blocs combinatoires.

Toutes les considérations pratiques sont basées sur des hypothèses de pannes analytiques.



## MOTS-CLES

circuits autotestables - circuits autocontrôlables - contrôleurs - codes - conception de contrôleurs - contrôleurs "Strongly Code Disjoint" - contrôleurs "Strongly Language Disjoint" - test (en ligne/hors ligne) - conception VLSI.