Application des circuits intégrés autotestables à la sureté de fonctionnement des systèmes

Serge Noraz

THESE

présentée par

Serge NORAZ

pour obtenir le titre de **DOCTEUR**

de l'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE
(arrêté ministériel du 23 novembre 1988)

(Spécialité : Microélectronique)

==========

APPLICATION DES CIRCUITS INTEGRES AUTOTESTABLES A LA SURETE DE FONCTIONNEMENT DES SYSTEMES

==========

Date de soutenance : 20 Décembre 1989

Composition du Jury : M.	MAZARE	Guy	Président
MM.	BOUTEILLER	Patrick	
	COURTOIS	Bernard	
	DAVID	René	Rapporteur
	GEFFROY	Jean-Claude	Rapporteur

Thèse préparée au sein du Laboratoire TIM3 / INPG.

AVANT - PROPOS

Je tiens à exprimer ma reconnaissance à :

Monsieur Guy MAZARE, Professeur à l'ENSIMAG, pour l'honneur qu'il me fait en présidant le jury de cette thèse.

Monsieur Bernard COURTOIS, Directeur de recherche au CNRS et directeur-adjoint du laboratoire TIM3, pour m'avoir accueilli dans l'équipe d'Architecture des Ordinateurs et pour avoir pris la responsabilité 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, pour avoir accepté d'être rapporteur de ce travail et membre de ce jury.

Monsieur Jean-Claude GEFFROY, Professeur à l'INSA de Toulouse, pour avoir accepté d'être rapporteur de cette thèse et membre du jury.

Mon collègue Michael NICOLAIDIS, Chargé de recherche au CNRS, pour l'intérêt permanent qu'il a porté à mon travail, pour son aide et sa collaboration qui ont fait que mes travaux ont évolué dans les meilleures conditions.

Je tiens à remercier Monsieur Guy LEMARCHAND pour m'avoir accueilli au département Systèmes Electroniques de Sûreté (SES) de MERLIN-GERIN à l'époque où il dirigeait ce département.

Je tiens particulièrement à remercier Monsieur Patrick BOUTEILLER, chef du service Conception Technique et Industrialisation (CTI) de SES, pour son aide, ses remarques pertinentes et le rôle qu'il a tenu dans ce projet. Je le remercie aussi pour avoir accepté de faire partie du jury de cette thèse.

Je ne saurais oublier dans mes remerciements tous les membres de l'équipe de recherche en Architecture des Ordinateurs pour leur collaboration en des moments divers et à différents titres.

Enfin que ce document soit pour tous les membres des différentes équipes du département dans lequel j'ai travaillé, le témoignage de ma reconnaissance pour l'intérêt, les conseils et l'aide qu'ils m'ont accordés.

Cette thèse a été réalisée dans le cadre d'une convention CIFRE N^{0}149/86 avec la société MERLIN-GERIN.

A Frédérique,

à mes parents.

" Il me semble que nous allons aujourd'hui vers une société composée de deux grands types d'intelligence si différents en qualité qu'ils ne seront jamais directement concurrents mais compléméntaires : d'un coté, les êtres humains avec des cerveaux à base de carbone et de l'autre , les robots avec des cerveaux à base de silicium. Plus généralement, on peut dire que nous aurons ainsi une vie-carbone et une vie silicium."

ISAAC ASIMOV

" X comme inconnu "

SOMMAIRE

APPLICATION DES CIRCUITS INTEGRES AUTOTESTABLES A LA SURETE DE FONCTIONNEMENT DES SYSTEMES

INTRODUCTION

INTRODUCTION

L'utilisation de circuits ayant des propriétés d'autotestabilité est très intéressante pour la conception de systèmes où la sûreté de fonctionnement est fondamentale, car ils permettent la détection immédiate (circuits "self-checking" ("autocontrôlables")) ou rapide (circuits "BIST" ("conçus en vue du test")) des erreurs. Les difficultés liées à la conception de ces circuits sont dues surtout au fait que l'on cherche à avoir des éléments conçus avec le minimum d'augmentation de surface possible tout en leur procurant la capacité majeure de détection de pannes qui peuvent apparaître dans le circuit (selon les hypothèses de pannes considérées qui dépendent du niveau de description du circuit).

Bien que ces circuits autotestables soient promis à un bel avenir du fait de l'évolution de la complexité et de la densité des fonctions intégrées sur une puce, ils ne peuvent malheureusement pas se substituer aux circuits "fail-safe" (circuits "sécuritaires") conventionnels. Ces derniers sont généralement utilisés dans des systèmes de contrôle/commande qui se trouvent dans tous les domaines où les normes de sécurité sont draconiennes. Malheureusement, les circuits "fail-safe" conventionnels ne peuvent pas être intégrés, de ce fait ils se prêtent très mal à la réalisation de systèmes complexes qui coûtent très cher en matériel et en surface. D'autre part, la présence d'une panne latente dans ce type de circuits, qui ne comporte généralement aucun moyen de détection d'erreurs, peut se combiner avec une autre panne pour aboutir à une panne composée pour laquelle la propriété "fail safe" du système n'est plus garantie. Cela peut conduire à une situation catastrophique.

L'idée développée dans cette thèse consiste à généraliser le concept des systèmes "fail-safe". Cela pour que nous puissions utiliser des techniques de conception "self-checking" et "BIST" combinées avec des interfaces "totally fail-safe" ("totalement sécuritaire") pour être en mesure d'obtenir des systèmes "fail-safe" à haute densité d'intégration. Comme nous le verrons, la théorie générale donnée par la suite et l'emploi d'interfaces "strongly fail-safe" ("fortement sécuritaires") nous permettront de concevoir non seulement des circuits "fail-safe" combinatoires mais aussi des machines séquentielles sûres.

Le chapitre I expose les différentes approches de conception en vue du test intégré des circuits qui sont principalement utilisées par les concepteurs et les fabricants de VLSI. De là nous abordons la conception de systèmes autotestables en-ligne.

Le chapitre II consiste en un résumé de ce que l'on trouve dans la théorie classique concernant les techniques de conception "self-checking". Les propriétés globales et les hypothèses d'occurrence de pannes dans les systèmes conçus suivant les règles imposées par ces techniques dépendent de l'organisation interne de ces derniers, des codes utilisés pour l'information et , avant tout, des propriétés de chacun des blocs qui composent le système.

Ces propriétés alliées à certaines hypothèses de fonctionnement ont pour but d'assurer que la première sortie erronée du circuit sera reconnue donc détectée. Ce but, que doit accomplir tout système "self-checking", est le "totally self-checking goal" ("but des circuits totalement autocontrôlables").
Enfin, comme les conditions imposées par la théorie précédente ne sont pas toujours pratiques à respecter dans le cas de réalisations concrètes de systèmes, une méthode qui permet d'associer les atouts du test intégré en-ligne et du test intégré hors-ligne est proposée. Cette technique UBIST ("Unified-Built-In-Self-Test" ("conception en vue de l'autotest unifié")) est l'étape ultime du test intégré. Elle permet une certaine souplesse de conception que n'autorisent pas les propriétés données pour les circuits autotestables en-ligne et elle tend à augmenter le taux de couverture des pannes susceptibles d'apparaître à différents stades de la vie d'un circuit intégré.

Le chapitre III se sert de la théorie résumée au chapitre précédent et de quelques définitions relatives aux circuits séquentiels autotestables pour la conception d'automates "self-checking" appelés à être utilisés dans des systèmes de contrôle/commande de processus critiques. L'intérêt de l'étude qui est faite ici porte sur la réalisation en blocs combinatoires internes pour la réalisation de ces machines séquentielles. Un exemple concret de conception UBIST est exposé.

Les circuits digitaux ne délivrent pas de données adéquates pour le pilotage d'éléments électromécaniques qui se trouvent en bout de chaîne de nombreux systèmes de surveillance utilisés dans l'industrie chimique, nucléaire ou des transports. C'est pourquoi le chapitre IV traite de la théorie générale des systèmes "fail-safe". Cette dernière permet d'utiliser des systèmes "self-checking", combinatoires ou séquentiels, avec une interface qui transforme les données binaires des circuits précédents en signaux sûrs capables d'activer des actionneurs électromécaniques. Le grand avantage de cette théorie est qu'elle réunit les circuits "self-checking" et les circuits "fail-safe" au sein d'une classe plus générale définie dans ce chapitre. Cela abouti, avec certaines conditions et avec des hypothèses de fonctionnement spécifiques, à l'intégration à haute densité de systèmes "fail-safe" (VLSI). De la même manière que les circuits "self-checking" réalisent le "totally self-checking goal", les circuits "fail-safe" que nous proposons ont pour but d'accomplir le "totally fail-safe goal" ("but des circuits fortement sécuritaires").

La conclusion permet d'ajouter des considérations, de mettre en valeur l'avantage de l'emploi des circuits "self-checking" ou "fail-safe" dans les systèmes à haute sûreté de fonctionnement. Ceci en vue de l'application et de la poursuite des idées exposées dans cette thèse.

CHAPITRE I

I - LA CONCEPTION EN VUE DU TEST DES CIRCUITS INTEGRES

I - LA CONCEPTION EN VUE DU TEST DES CIRCUIT INTEGRES

1 - LA TESTABILITE

1.1 A quoi sert la testabilité ?

Pour être compétitifs, les produits d'aujourd'hui doivent être de plus en plus performants, de moins en moins chers et fabriqués de plus en plus vite. Cependant, tous ces objectifs ne doivent pas être atteints au détriment de la qualité qui reste le facteur essentiel sur lequel l'utilisateur porte son jugement.
L'électronique intégrée n'échappe pas à cette loi du marché qui est d'autant plus sévère dans ce domaine que la concurrence est nombreuse et l'évolution des technologies est rapide. Dans un tel contexte, ce qui différencie un fabricant de circuits d'un autre, ce n'est pas seulement sa capacité à répondre très vite aux besoins du marché mais encore sa compétence à fournir un produit qui satisfait pleinement les spécifications données par le client en fonction de l'usage final que celui-ci souhaite en faire. Cet usage se traduit en terme d'objectifs (performances, fonctionnalité, fiabilité, sécurité, maintenabilité, disponibilité, ...) et bien qu'ils ne soient pas toujours faciles à définir ou à formuler, ce n'est que par rapport à eux qu'une stratégie de test peut être proposée, choisie, appliquée et améliorée.
C'est donc dans ce sens que la **Testabilité** est un moyen de parvenir à la maîtrise de la qualité, qu'elle permet d'améliorer la productivité et d'apporter une plus grande confiance dans le produit fabriqué.

1.2 Le concept de la testabilité

En fonction des objectifs économiques et techniques du produit à fabriquer, la testabilité peut être considérée comme la prise en compte, à la conception, des moyens qui vont permettre l'exécution et l'optimisation des phases - nécessaires et suffisantes - de test et de diagnostic afin que celles-ci soient financièrement les plus rentables en délai, coût et qualité.
Sa mise en oeuvre concerne aussi bien les concepteurs que les gens du test ou les utilisateurs du produit. Cette concertation entre tous les protagonistes est d'autant plus nécessaire que la testabilité ne doit pas être une préoccupation à un moment donné, mais elle doit être un souci permanent dans le cycle de vie du produit (spécifications, ..., maintenance).

1.3 Quand ?

Chaque étape de la réalisation d'un système peut introduire des défauts. Alors au fur et à mesure de la progression dans le processus de fabrication (composant, carte, système, sur site), le coût de détection et de localisation d'une panne croît en amplitude d'un facteur 10 (fig.1). Le moment le plus opportun pour influencer les coûts du test se situe donc au plus bas niveau de la fabrication, c'est à dire au niveau du composant. Même à ce niveau, compte tenu des augmentations de densité d'intégration et de la complexité croissante des composants VLSI, il est important de disposer de très bons moyens de test.

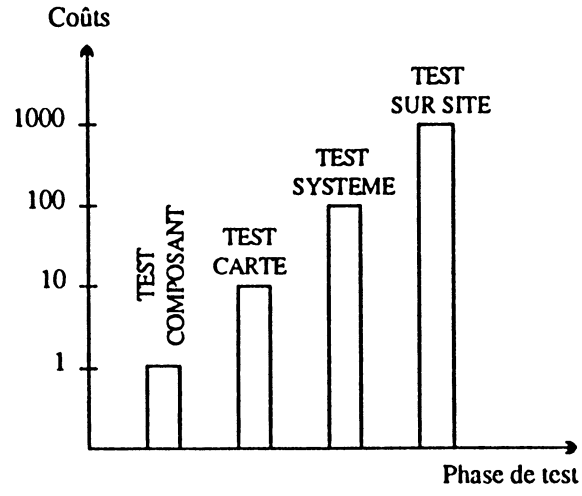


Fig I - 1 - Economie de la testabilité

1.4 Comment ?

Comme le montre la définition, la qualité de détection, le coût et la rapidité sont les données importantes à considérer dans le choix d'une méthode et d'un dispositif de test. La première de ces données est fonction de la connaissance des pannes possibles, quant aux coûts (temps de conception, prix d'un testeur ou surface de test supplémentaire) et à la rapidité, ils sont liés à l'approche de conception choisie. La nature et les quantités sont aussi des critères à ne pas ignorer lors de la prise en compte de la testabilité sur une famille de produits. Nous distinguons :

La testabilité extrinsèque, elle concerne l'environnement de test du produit et à ce titre elle est fonction des quantités. Cet aspect de la testabilité n'est pas abordé dans la présente étude.

La testabilité intrinsèque, elle se rapporte à tout ce qui est implanté sur le produit. C'est sur ce type de testabilité que se porte notre intérêt et plus spécifiquement sur les moyens à mettre en oeuvre au niveau composant puisque c'est à ce stade qu'il est le plus rentable de tester.

La prise en compte de la testabilité d'un composant fait appel aux notions de **contrôlabilité** et d'**observabilité**. L'application de ces deux points ne peut se prévoir que lors de la conception, sinon la testabilité est subie et non optimisée. Enfin la testabilité peut se concevoir à différents niveaux d'intégration :

- **Passive**, il y a un apport de composants simples hors du chemin fonctionnel (points de test),
- **Active**, il y a un apport de circuits actifs dans le chemin fonctionnel,
- **Structurée**, elle concerne les architectures élaborées en vue du test (bus de testabilité, logiques structurées, logiques à scrutation ("scan", BILBO), test intégré).

2 - LES DIVERS STADES DE VERIFICATION DES CIRCUITS INTEGRES

Tout au long du processus de conception - fabrication - utilisation, un circuit intégré subit un certain nombre de tests pour vérifier son bon fonctionnement. Ces différents tests se situent comme suit :
Dès la sortie des premiers prototypes, la conception et la fonctionnalité du circuit sont validées par **le test de validation en fin de conception.**
Ensuite pendant la phase de production les circuits défectueux sont éliminés par **le test de fin de fabrication.**
Une fois les circuits assemblés sur une carte, il faut vérifier que cette dernière est fonctionnellement bonne et ainsi de suite jusqu'au système complet.
Enfin, durant la vie du système (fonctionnement) il faut appliquer périodiquement **un test de maintenance** pour détecter si un circuit est tombé en panne.
Des logiciels et des mécanismes spécifiques peuvent être aussi utilisés pour vérifier en permanence la validité des opérations effectuées, ceci pour ne pas laisser une panne dans le système qui serait signalée longtemps après son apparition et qui pourrait empêcher toute reprise éventuelle.

A ces différentes phases de vérification correspondent trois types de test :

Le test de mise au point : C'est le test de fin de conception ou de validation, il est destiné à révéler et à diagnostiquer les erreurs de conception. Compte tenu du fait que les méthodes de conception mettent en oeuvre des processus plus ou moins automatiques (de la conception faite entièrement "dessinée au micron" à la compilation de silicium "presse bouton" (cf : annexe 1)), deux types de vérifications peuvent être envisagées à ce stade : vérification du résultat obtenu et/ou vérification préalable des outils de conception automatique.

Le test hors-ligne : C'est le test utilisé en fin de fabrication qui est destiné à déterminer quels sont les circuits bons à l'issue de la fabrication. Ce type de test est le plus généralement abordé.

Le test en-ligne : ou test concurrent. C'est le test de maintenance dont le but est de détecter l'apparition d'une panne au cours de l'application exécutée par le circuit. La détection peut-être assurée par des mécanismes ad-hoc (par exemple, chien de garde sur CPU [COU 79]) ou par des systèmes autotestables en-ligne.

3 - CLASSIFICATION DES METHODES DE TEST DES CIRCUITS INTEGRES

Avec l'accroissement de la complexité et de la densité d'intégration des circuits - avant l'an 2000, la puce égalera en nombre de transistors les processeurs des grands calculateurs actuels atteignant le million de transistors! - aujourd'hui il n'est plus question de concevoir des composants sans se préoccuper, dès le stade des spécifications, de la stratégie de test et des moyens à mettre en oeuvre pour obtenir une bonne testabilité. Evidemment, les objectifs et les caractéristiques (taille, coûts, fiabilité, rapidité, ...) d'un dispositif nécessaire au test d'un circuit influent sur son domaine d'utilisation et sur le choix de son implantation.Toutefois, les problèmes de testabilité du matériel dépendent toujours des notions suivantes :

- **la contrôlabilité**, elle mesure la facilité avec laquelle la partie interne d'un circuit peut être excitée à partir de ses entrées primaires.

- **l'observabilité**, elle mesure la facilité avec laquelle la réponse de la partie interne d'un circuit sous test peut être obtenue à partir de ses sorties primaires.

- **la génération de vecteurs de test**.

Les méthodes de conception en vue du test peuvent se regrouper en trois catégories selon :

- Qu'elles facilitent les problèmes d'accessibilité (contrôlabilité et observabilité) et de génération de vecteurs de test sans pour autant les faire disparaître.

- Qu'elles éliminent les problèmes d'accessibilité mais pas ceux qui sont spécifiques à la génération de vecteurs de test (LSSD, "scan path", ...).

- Qu'elles suppriment tout, ou en partie, les problèmes d'accessibilité et de génération de vecteurs de test (test intégré hors-ligne (BIST), test intégré en-ligne).

4 - PRINCIPES DU TEST DES CIRCUITS INTEGRES

4.1 Terminologie

Les notions de panne et d'erreur étant précisées ultérieurement (cf : chapitre 2), les définitions suivantes serviront de référence pour le vocabulaire employé dans la suite de l'étude :

Entrées et sorties primaires : Ce sont respectivement les connexions d'entrée et de sortie du circuit accessibles de l'extérieur.

La détection d'une panne dans un circuit est la mise en évidence de celle-ci, **la localisation** permet de la situer topologiquement. La détection et la localisation de toutes les pannes du circuit sous test constituent **le diagnostic**.

Vecteur de test : un vecteur de test est un affichage de valeurs logiques sur les entrées primaires du circuit.

Séquence de test : Une séquence de test est déterminée par l'ensemble des vecteurs de test nécessaires à la détection d'une panne ou à toutes les pannes d'un circuit.

Longueur d'une séquence : La longueur d'une séquence de test est donnée par le nombre de vecteurs qui la compose.

Méthode de test : La méthode de test caractérise les moyens mis en oeuvre pour la génération des séquences de test et l'observation des réponses du circuit sous test.

4.2 Les différentes approches

Placé dans son environnement fonctionnel, un circuit reçoit normalement des vecteurs d'entrée qu'il transpose suivant sa fonctionnalité en vecteurs de sortie.
Tester ce circuit consiste à lui appliquer (contrôlabilité), à un moment donné et par les moyens les plus adéquats, une séquence de test sur ses entrées, fonctionnelles ou non, et à observer ses sorties (observabilité) pour juger du bon fonctionnement du circuit. Plusieurs approches sont envisageables **[LAP 89]** pour atteindre ce but :

L'approche déterministe et l'approche non-déterministe.

L'approche déterministe : C'est le cas où les vecteurs de test sont prédéterminés en vue de leur utilisation. Soit la séquence de vecteurs de test est déterminée à partir d'une fonction à laquelle est attaché un modèle de panne (par exemple : fonction = multiplexeur et modèle de panne = deux entrées sont simultanément sélectionnées) et le test est qualifié de **test fonctionnel** [THA 78], soit la séquence de vecteurs de test est élaborée à partir de la structure des circuits qui comporte aussi son propre modèle de panne (par exemple : structure en portes logiques et modèle de panne = collage de ligne) et le test est qualifié de **test structurel** [COU 81 a].
Dans ces deux cas, l'analyse des sorties peut se faire soit par comparaison, au "coup par coup", avec des valeurs prédéterminées soit par analyse de signature (analyse par compaction).

L'approche non-déterministe [DAV 79] : C'est le cas où les vecteurs de test qui ne sont pas prédéterminés sont générés lors du test. Cette méthode qui permet de détecter des pannes qui ne sont pas modélisées est connue sous le nom de **test aléatoire** (ou pseudo-aléatoire si la séquence de vecteurs est reproductible). Un paramètre important de ce type de test est lié à la longueur de la séquence de test qui est fonction de la panne la plus difficile à détecter dans le circuit suivant le degré de validation recherché.
L'analyse des sorties est faite en général par comparaison avec les sorties d'un circuit de référence (circuit réputé "bon").

4.3 Les critères de test

Quelle que soit la nature du test appliqué à un circuit intégré, plusieurs critères servent à en préciser le type.

4.3.1 Test en-ligne / test hors-ligne

Le test en-ligne d'un circuit lui permet de détecter ses propres pannes au cours du déroulement de l'application contrairement au test hors-ligne qui s'effectue en dehors du contexte d'exécution de cette application.

4.3.2 Test continu / test discontinu

Le critère de test continu ou discontinu sert toujours à qualifier un test en-ligne. Ce dernier est continu si la simultanéité du test et de l'application est parfaite, tandis qu'il est discontinu lorsqu'il y a alternance d'une phase de test et d'une phase d'application.

4.3.3 Test in-situ / test ex-situ

Le critère de test in-situ ou ex-situ ne peut être employé que pour qualifier un test hors-ligne. Pour un test hors-ligne in-situ le circuit est testé dans son environnement physique de travail et il n'existe pas de moyen d'observer à tout instant les valeurs des signaux. Dans le cas du test hors-ligne ex-situ, le circuit est sorti de son environnement de travail et il est placé dans un environnement spécifique au test. L'observation des valeurs qui transitent sur les lignes est alors possible.

5 - LA CONCEPTION EN VUE DU TEST DES CIRCUITS INTEGRES

L'avènement des circuits à très grande échelle d'intégration (VLSI) a justifié l'utilisation de techniques de conception permettant de nouvelles stratégies de test particulièrement adaptées aux fonctions de plus en plus complexes portées sur une seule puce de silicium. Dans ce qui suit, nous présentons quelques-unes des principales méthodes de réalisation en vue du test (DFT, BIST,...) des circuits intégrés.

5.1 Les circuits conçus avec facilités de test

Les solutions proposées pour ce type de conception sont connues sous le nom de DFT ("Design For Testability" ("conception pour la testabilité")). Elles facilitent l'observabilité et la contrôlabilité mais le problème de la génération des vecteurs de test subsiste. A l'heure actuelle, la technique du "scan design" (conception pour le test par chemin de balayage) est la plus couramment utilisée.
Le principe des architectures "scan" vise à ramener le problème du test d'une logique séquentielle à celui d'une logique combinatoire. Le propre de la méthode est de sérialiser les éléments mémoires du circuit (bascules, flip-flops), et éventuellement d'ajouter certains de ces éléments, pour former un ou plusieurs registres à décalage qui permettent le contrôle et l'observation des registres internes. Parmi les solutions aux variantes multiples nous retiendrons le "scan path" de NEC et le "LSSD" d'IBM que nous allons décrire brièvement.

5.1.1 Méthode "scan path" (chemin de balayage) [FUN 75] , [FUJ 85]

Dans tous les cas (circuit séquentiel ou circuit combinatoire), le circuit est ramené à un réseau combinatoire plus un registre à décalage. La partie combinatoire reçoit des stimulis de test provenant des entrées primaires et du registre "scan".
L'analyse de la réponse se fait de deux façons. Tout d'abord, un système récupère des données aux sorties primaires du circuit. Ensuite, avec le procédé scan la plupart des réponses du réseau combinatoire sont vérifiées pendant la phase de lecture/écriture du registre à décalage.

Ce procédé a été intégré avec succès par d'importants fabricants au niveau de la carte et du circuit depuis une dizaine d'années.

5.1.2 Méthode "LSSD" (chemin de balayage extérieur au circuit) [EIC 77], [FUJ 85]

La méthode "Level Sensitive Scan Design" ("LSSD") est la plus connue en matière de conception pour ce type de test hors-ligne. Elle diffère peu dans son principe de la méthode précédente si ce n'est que par la conception spécifique des éléments de mémorisation.

Les approches présentées jusqu'à présent sont intéressantes puisqu'elles simplifient le test des circuits séquentiels et dans tous les cas (circuits séquentiels ou purement combinatoires) elles éliminent les problèmes d'accéssibilité. La génération des vecteurs de test reste cependant tributaire d'équipements externes coûteux et d'une durée de test qui peut être très longue en raison des décalages séries de la chaîne "scan". De plus, les techniques "scan" ne sont pas très appropriées pour tester la logique dynamique et les structures très répétitives du type mémoires. En effet, les vecteurs ne peuvent pas être appliqués à une vitesse suffisante pour permettre à un circuit dynamique de continuer à fonctionner. Enfin le fait de rajouter, dans certains cas, des éléments pour obtenir des registres à décalage supplémentaires entraîne un doublement ou triplement de la taille du circuit. Pour pallier ces inconvénients, d'autres approches combinant la technique "scan" et l'autotest intégré sont proposées.

5.2 L'autotest intégré

La solution consiste à intégrer dans le circuit tous les mécanismes nécessaires à son propre test.

5.2.1 Le test intégré hors-ligne

Le principe de cette technique, connue sous le nom de BIST ("Built In Self Test"), est d'incorporer dans le circuit la génération de vecteurs de test et l'observation des sorties. Pour ce faire, différents générateurs de vecteurs de test et des analyseurs de signature (observation des réponses par compactage) sont réalisés avec des registres à décalage à rebouclage linéaire ("Linear Feedback Shift Registers" (LFSRs)). Ces techniques simplifient énormément le test des circuits complexes : le temps d'application des séquences est court, les circuits sont contrôlés à vitesse d'utilisation, la phase d'élaboration des vecteurs de test est évitée (génération de vecteurs "pseudo-aléatoire" le plus souvent), l'initialisation et la lecture des registres ne sont pas coûteuses en temps. Cependant, l'augmentation en surface due à l'emploi de registres spécifiques peut ne pas être négligeable.

Dans ce domaine, la méthode de conception BIST la plus représentative est la méthode BILBO ("Built In Logic Bloc Observer") proposée en [KON 79].

Les différentes techniques présentées jusque là concernent le test hors-ligne des circuits. Toutefois elles peuvent être utilisées pour faire un test en-ligne discontinu en activant périodiquement les séquences de test nécessaires. La détection de la panne est alors réalisée avec un retard par rapport à son apparition qui est liée à la périodicité du test. Ces solutions ne sont pas satisfaisantes pour les applications qui exigent une sûreté de fonctionnement importante où la sécurité est le facteur essentiel [LAP 85] (annexe 2). En effet, dans ce cas il est indispensable d'assurer immédiatement la détection des erreurs causées par une panne dans le système afin d'intervenir rapidement et éviter une évolution catastrophique de la situation. Pour répondre à ce besoin, les circuits sont conçus pour être autotestables en-ligne.

5.2.2 Le test intégré en-ligne

Le but du test en-ligne est de détecter l'apparition d'une panne dans un circuit au cours de son fonctionnement et ce pendant le déroulement de l'application. Parmi les diverses techniques possibles (codage logiciel, redondance massive, ...), la prise en compte du test en-ligne dès la conception (circuits "self-testing" ("autotestables") et "self-checking") permet d'obtenir des circuits où la détection des pannes est assurée par le matériel (blocs fonctionnels et contrôleurs). D'autre part le test intégré unifié propose la conception de composants autotestables aussi bien vis à vis du test hors-ligne (utilisation de LFSRs) que du test en-ligne (utilisation de contrôleurs).

6 - CONCLUSION

Les différentes méthodes proposées dans ce chapitre sont autant de choix possibles pour réaliser des circuits intégrés afin de faciliter les tests de validation des prototypes, les tests de fin de fabrication et les tests de maintenance. Cependant la conception de circuits autotestables en-ligne est certainement la plus adaptée pour la réalisation de systèmes exigeant une sûreté importante (automatismes industriels de sécurité, systèmes de transport, ...). Comme cette étude porte sur la contribution des circuits intégrés autotestables à la sûreté de fonctionnement des systèmes, dans le cas où la sécurité est le principal facteur pris en compte, le chapitre suivant présente la technique de conception des circuits "self-checking".

CHAPITRE II

II - INTRODUCTION A LA CONCEPTION DES CIRCUITS "SELF-CHECKING"

II - INTRODUCTION A LA CONCEPTION DES CIRCUITS "SELF-CHECKING"

1 - INTRODUCTION

L'évolution rapide des circuits intégrés, liée aux progrès réalisés dans le domaine de la technologie, empêche désormais les techniques de prévention couramment utilisées jusqu'à aujourd'hui (techniques "scan", ...) de couvrir l'ensemble des pannes possibles dans les circuits intégrés logiques (pannes fugitives par exemple). D'autre part, pour certaines applications de sécurité où la défaillance d'un système peut être à l'origine de situations catastrophiques (mort d'hommes, perte de mission vitale, coûts engendrés importants, .. .) ou critiques (blessure, perte de fonction importante d'une mission, .. .), il est nécessaire de signaler immédiatement les premières sorties erronées afin d'agir en conséquence. De ce fait toutes les techniques classiques ne sont pas vraiment adaptées pour ce genre de mission.
Pour satisfaire un tel besoin, une solution consiste à prendre en compte le test en-ligne (test effectué en même temps que le déroulement d'un programme utilisateur) du circuit dès la conception. Cette technique de réalisation des circuits autotestables (technique "self-checking") a l'avantage de couvrir des modes de défaillances réels qui sont bien étudiés actuellement [FER 88] étant donné que le modèle de collage logique [FRI 71] a dû être abandonné faute de ne pas représenter toutes les pannes possibles dans les circuits intégrés [GAL 80]. De plus cette approche ne nécessite aucun logiciel pour assurer le test en-ligne, la détection des erreurs se fait par le matériel.
La conception d'un circuit intégré autotestable en-ligne ("self-testing et "self-checking") consiste donc en un ajout de matériel pour que le circuit en fonctionnement soit capable de détecter toute occurrence de panne. Cette conception, basée sur des techniques de codage et des propriétés mathématiques des différents blocs qui constituent la fonction a pour but de réaliser le "totally self-checking goal" [CAR 68], [AND 71], [SMI 78], [NIC 84]. Toutefois, le test en-ligne du circuit ne peut être effectif que si certaines conditions sont satisfaites. Malheureusement, dans la plupart des cas pratiques ces conditions sont difficiles, voire impossibles à vérifier. C'est pourquoi une solution consiste à utiliser judicieusement les atouts et les mécanismes propres au test hors-ligne pour les combiner avec les mécanismes utilisés pour le test en-ligne afin d'obtenir une réalisation en vue du test unifié des circuits ("Unified Built In Self Test" (UBIST)) [NIC 86 b], [NIC 88 a]. L'intérêt d'une telle technique est qu'elle autorise une certaine souplesse de conception et un taux de couverture de pannes élevé.

Ce chapitre est essentiellement consacré aux rappels des principales définitions et règles de conception des systèmes intégrés "self-testing" et "self-checking" [NIC 84] pour les hypothèses de pannes analytiques de la classe I donnée dans [COU 81 b].

2 - CLASSES DES HYPOTHESES DE PANNES

Le test d'un circuit se fait en fonction d'hypothèses de pannes. On parle d'hypothèses car cela fait implicitement référence à un niveau de description du circuit et on se limite volontairement aux pannes jugées les plus fréquentes.

Pendant longtemps, le modèle des pannes pris en compte pour le test des circuits intégrés s'est limité à celui du collage au niveau des portes logiques qui constituaient la fonction [FRI 71]. Ce modèle a dû être abandonné puisqu'il ne traduisait pas toutes les pannes réelles dans un circuit intégré [WAD 78]. Par conséquent de nouveaux modèles sont apparus dans lesquels les circuits sont représentés, au niveau électrique, par des réseaux de transistors MOS et où les pannes possibles sont représentées par des collages de lignes à une valeur logique, des collages de MOS passant ou ouvert ("stuck-on", "stuck-open") et des courts-circuits [GAL 80].

En ce qui concerne cette étude c'est l'implémentation réelle des circuits telle qu'elle est décrite par le dessin des masques qui est considérée, c'est à dire les lignes de diffusion, de polysilicium et d'aluminium, les croisements entre une ligne de polysilicium et une ligne de diffusion pour former un MOS, les contacts et les précontacts entre deux lignes de niveaux différents, ... Ce sont donc les défaillances de ces éléments qui nous intéressent, c'est à dire la coupure d'une ligne, un court-circuit entre deux lignes, un MOS défaillant, ... A ce niveau, [COU 81 b] a établi une première classification des mécanismes de pannes dues à des phénomènes physiques de la matière (transport de particules, claquage de diélectrique, ...). Cette étude a abouti à regrouper les pannes possibles en trois classes (fig.1).

CLASSE 0	CLASSE I	CLASSE II
Un défaut simple : * 1 Contact 1 Précontact * 1 MOS s-on ou 1 MOS s-open * 1 alu Coupé * 1 poly Coupé * 1 diff. Coupée : grille flottante →MOS s-open	Classe 0, plus : * 1 Court-circuit entre alu et alu le plus proche géographiquement * Même chose pour diffusion	Classe I, plus : * Court-circuit entre Alus quelconques * Même chose pour diffusion * Défauts multiples

Fig II - 1 - Classes des hypothèses de pannes

Etant donné que les mécanismes de pannes liés à la durée de vie des circuits intégrés sont très lents, la probabilité d'occurrence simultanée de plusieurs pannes est négligeable. Il est donc tout à fait réaliste, pour le test en-ligne de ces circuits, de ne considérer que des hypothèses de pannes simples. C'est à dire que nous nous intéresserons, par la suite, uniquement à la classe I des hypothèses de pannes.

3 - LES CIRCUITS "SELF-CHECKING" : DEFINITIONS COURANTES

La structure générale d'un circuit autotestable en-ligne est donnée en figure 2, elle se compose d'un bloc fonctionnel et d'un contrôleur.

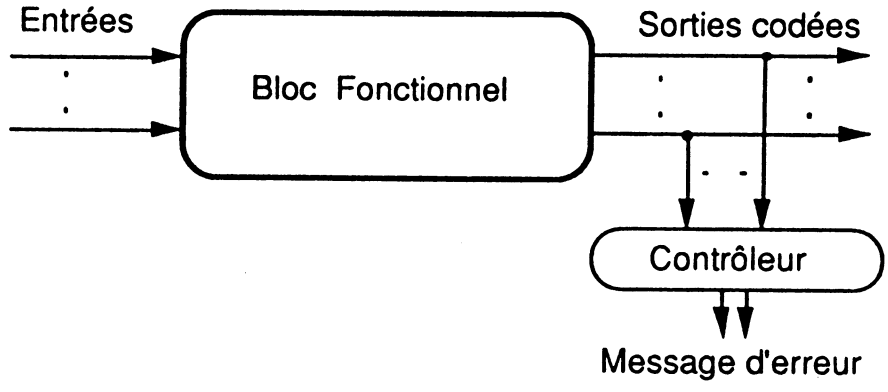


Fig II - 2 - Structure d'un circuit autotestable

Selon sa fonctionnalité, le bloc fonctionnel transforme les valeurs codées sur ses entrées en des valeurs codées sur ses sorties de manière à ce que la première manifestation d'erreur (sur les sorties du bloc), due à une panne au sein du circuit, soit immédiatement signalée par le contrôleur. Pour accomplir correctement ce but, le "totally self-checking goal" ("TSC goal"), les blocs fonctionnels et les contrôleurs d'un circuit "self-checking" doivent vérifier certaines propriétés mathématiques que nous rappelons en adoptant les conventions suivantes [SMI 78] :

Pour un bloc fonctionnel G qui a r entrées primaires et q sorties, les 2^r vecteurs binaires de longueur r forment l'ensemble X des vecteurs d'entrée alors que l'ensemble Y des vecteurs de sortie est constitué par les 2^q vecteurs binaires de longueur q.

En fonctionnement normal (avant l'occurrence d'un panne), le bloc fonctionnel G reçoit sur ses entrées un sous-ensemble A de vecteurs de X correspondant au code d'entrée et il produit un sous ensemble B de vecteurs G(a,ø) de Y appelé code de sortie. De par sa conception, en présence d'une panne f de l'ensemble des pannes F pris comme modèle, ce bloc produit des valeurs G(a,f) en sortie.

Les définitions suivantes sont dues à *Anderson* **[AND 71]** :

DEFINITION D1 : définition d'un circuit "self-testing" (fig.3).

Un circuit G est "self-testing" (ST) pour un ensemble de pannes F si :

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

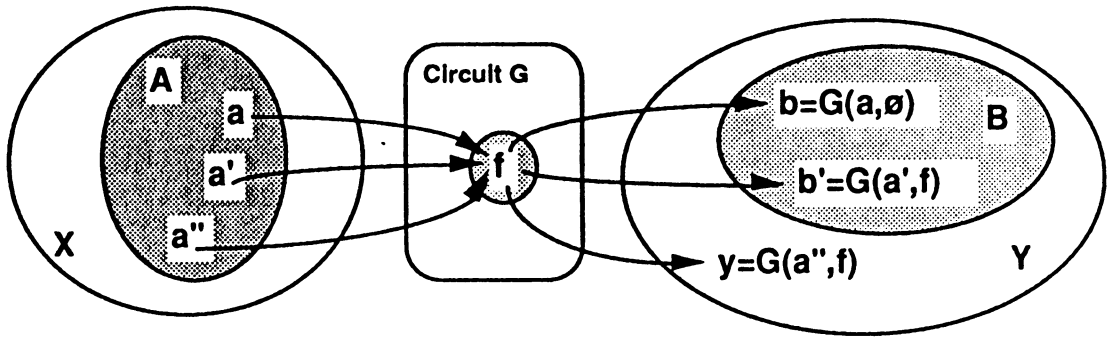


Fig II - 3 - Circuit "self-testing"

- Si $G(a,f) = G(a,\emptyset)$,

la panne n'intervient pas sur le fonctionnement du circuit .

- Si $G(a',f) \neq G(a',\emptyset) \in B$,

la panne influe sur le fonctionnement mais on ne la détecte pas car elle est telle que l'on a un vecteur du code de sortie.

- Si $G(a'',f) \notin B$,

il y a au moins un vecteur du code d'entrée qui détecte la panne.

Autrement dit, pour un circuit "self-testing" qui reçoit tous les vecteurs de A la panne est détectée.

DEFINITION D2 : définition d'un circuit "fault-secure" ("sûr en présence de pannes") (fig.4).

Un circuit G est "fault-secure" (FS) pour un ensemble de pannes F si :

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

Autrement dit, en présence d'une panne soit la sortie du circuit est correcte, soit elle est hors-code.

DEFINITION D3 : définition d'un circuit "totally self-checking".

Un circuit G "totally self-checking" (TSC) est un circuit "self-testing" et "fault-secure".

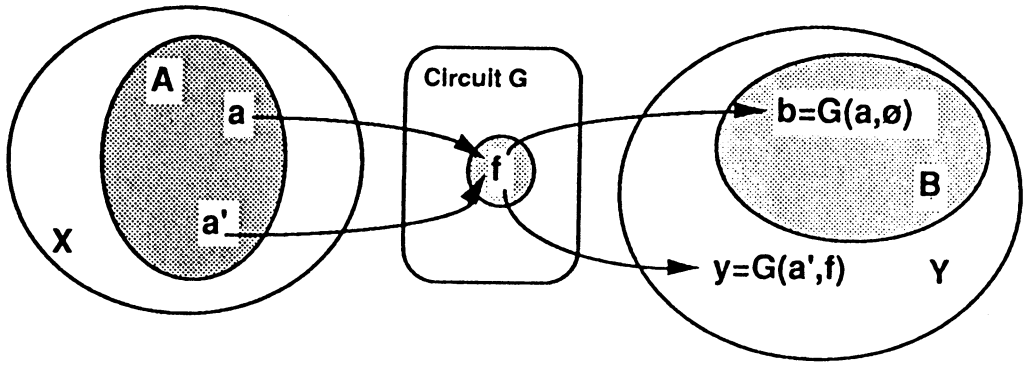


Fig II - 4 - Circuit "fault-secure"

Un circuit G "totally self-checking" accomplit le "totally self-checking goal" si la première sortie erronée due à une panne f de l'ensemble des pannes F est en dehors du code de sortie. Ce but est naturellement atteint si l'hypothèse suivante est respectée :

HYPOTHESE H1 :

Entre l'occurrence de deux pannes quelconques de F, il s'écoule un temps assez long pour que tous les vecteurs du code d'entrée soient appliqués aux entrées du circuit G.

La plus grande classe de circuits fonctionnels permettant d'assurer le "TSC goal" en respectant cette hypothèseest la classe des circuits "strongly-fault-secure" [SMI 78], [DAV 78].

DEFINITION D4 : définition d'un circuit "strongly fault-secure" ("fortement sûr en présence de pannes").

Un circuit G est "strongly fault-secure" (SFS) pour un ensemble de pannes F si pour toute panne $f \in F$: a) soit le circuit est TSC

b) soit le circuit est FS et si une nouvelle panne f de F survient, pour la panne résultante on retombe dans le cas a) ou b).

DEFINITION D5 : définition d'un circuit "code disjoint" ("code-disjoint") (fig.5).

Un circuit G est à "codes disjoints" (CD) si :

$\forall a \in A, G(a,\emptyset) \in B, \forall x \in X\text{-}A, G(x,\emptyset) \notin B.$

DEFINITION D6 : définition d'un contrôleur TSC.

Un circuit G qui est "totally self-checking" et "code-disjoint" est un contrôleur "totally self-checking".

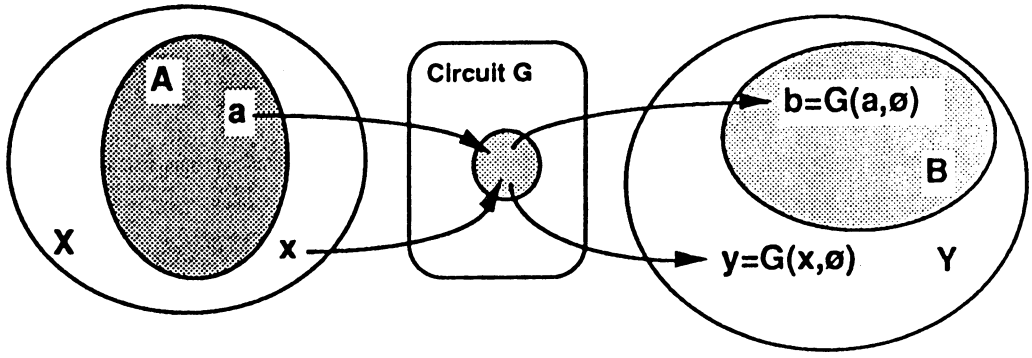


Fig II - 5 - Circuit "code disjoint"

En fait la propriété "totally self-checking" n'est pas forcément nécessaire pour qu'un contrôleur assure sa mission dans la mesure où celui-ci reste "code disjoint" même en présence de pannes internes. A partir de cette remarque *Nicolaidis* [NIC 84] a introduit le concept de contrôleurs "strongly code disjoint" (SCD).

DEFINITION D7 : définition d'un circuit "strongly code disjoint" ("fortement code-disjoint").

Un circuit G est "strongly code disjoint" (SCD) pour un ensemble de pannes F si :

Avant l'occurrence d'une panne f ∈ F, le circuit G est CD.

Après l'occurrence de f on a :

a) soit G est ST

b) soit G transpose les vecteurs hors-code d'entrée en vecteurs hors-code de sortie et si une nouvelle panne f'∈ F survient, on retombe dans le cas a) ou b).

Les contrôleurs "strongly code disjoint" représentent la plus large classe de contrôleurs qui, associés à des circuits fonctionnels "strongly fault-secure" peuvent atteindre le "totally self-checking goal".

La figure 6 montre le fonctionnement d'une cellule de base d'un circuit qui atteint le "totally self-checking goal" avec un bloc fonctionnel SFS et un contrôleur SCD. Dans ce cas l'hypothèse H2, qui concerne l'occurrence des pannes dans un système composé d'un bloc fonctionnel et d'un contrôleur, doit être respectée.

HYPOTHESE H2 :

Après l'occurrence d'une panne dans le contrôleur, 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'une deuxième panne survienne dans le bloc fonctionnel ou dans le contrôleur.

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

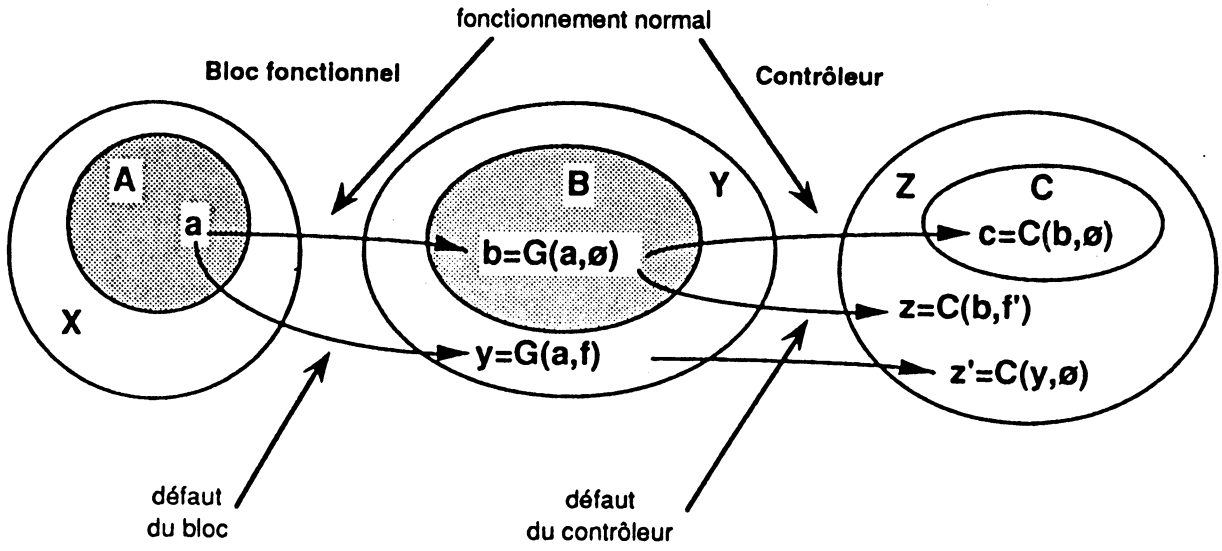


Fig II - 6 - Fonctionnement d'une cellule:
Bloc fonctionnel "SFS" + contrôleur "SCD"

Le comportement de la cellule est le suivant :

- si une panne se produit dans le bloc fonctionnel "SFS",
  ou cette panne (composée) est latente (elle ne provoque aucune erreur) et le circuit reste "FS" jusqu'à l'occurrence d'une autre panne dans le bloc fonctionnel ou dans le contrôleur,
  ou cette panne (composée) donne lieu à une erreur en sortie (vecteur hors-code de sortie) que le contrôleur se charge de la signaler avant l'apparition de la panne suivante en application de H2.

- si une panne se produit dans le contrôleur "SCD",
  ou cette panne (composée) est latente mais le contrôleur reste "CD" donc capable de signaler toute erreur aux sorties du bloc fonctionnel et ce jusqu'à l'apparition d'une autre panne dans le bloc fonctionnel ou dans le contrôleur.
  ou il existe un vecteur d'entrée du contrôleur qui provoque un signal d'erreur avant l'apparition d'une nouvelle panne en application de H2.

4 - CLASSIFICATION DES ERREURS ET CODES DETECTEURS

4.1 Différents types d'erreurs

La transmission des données dans les systèmes informatiques utilise des techniques de codage pour la détection des erreurs. Ces mêmes techniques ont été adaptées et étendues aux circuits intégrés et leurs applications se retrouvent dans le cadre du test intégré. Dans la théorie des circuits "self-checking", les trois types d'erreurs les plus couramment considérés sont :

- Les erreurs simples : où un seul bit du mot est affecté, par exemple 101101 au lieu de 100101.

- Les erreurs unidirectionnelles : où un nombre quelconque de bits est modifié mais dans un seul sens ($1 \rightarrow 0$ ou $0 \rightarrow 1$), par exemple 101101 au lieu de 100100.

- Les erreurs multiples : où un nombre quelconque de bits est modifié dans les deux sens, par exemple 101101 au lieu de 001011.

Dans tous les cas la détection de l'erreur avec un code adapté met en évidence l'existence ou l'absence d'une panne. *Wakerly* [WAK 78] définit un code détecteur d'erreurs comme un sous ensemble S de vecteurs choisis parmi un univers U. Ces vecteurs sont tels que l'ensemble des pannes susceptibles de se produire dans le circuit affectent les vecteurs de S et donnent des vecteurs qui n'appartiennent plus à S mais à (U-S). Les vecteurs de ce dernier ensemble sont des vecteurs hors-code.

Au trois types d'erreurs correspondent différents types de codes. Ces codes peuvent être des code séparables ou non, ils peuvent être ordonnés ou non.

- Les codes séparables : sont des codes pour lesquels les bits d'information et les bits de codage sont distincts et accolés. Les codes de parité, de Berger et les codes à duplication sont les principaux codes séparables utilisés par le test intégré.

- Les codes non-ordonnés : sont des codes où il n'existe pas deux vecteurs du code tels que l'un couvre l'autre (fig.7). Le code m-parmi-n est le code le plus couramment utilisé par les techniques de test intégré.

A = | a1 | - | ai | - | an | couvre B = | b1 | - | bi | - | bn |

si ai = bi chaque fois que bi = "1"

Fig II - 7 - Recouvrement de deux vecteurs

Le code de parité, le code de Berger et les codes à duplication sont en général utilisés pour détecter respectivement les erreurs simples, les erreurs unidirectionnelles et les erreurs multiples (fig.8).

	Erreurs Multiples	Erreurs Unidirectionnelles	Erreurs Simples
Codes à duplication - code dupliqué - code double-rail	*	*	*
Codes non-ordonnés - Code de Berger		*	*
Code de parité			*

Fig II - 8 - Différents types de codes détecteurs d'erreurs

Ces différents codes sont présentés brièvement par la suite :

4.2 Le code à parité

Ce code (ou le code à imparité) est celui qui est le plus couramment utilisé aujourd'hui.C'est un code séparable simple qui permet la détection d'erreurs simples.
Le bloc fonctionnel est conçu de sorte que toute panne ne provoque que des erreurs simples sur ses sorties primaires. Pour cela il faut utiliser des règles de conception très précises au cours de la réalisation.

4.3 Le code de Berger

Lorsque la structure interne du circuit est telle que ce sont des erreurs unidirectionnelles qui sont envisagées sur les sorties primaires du circuit, le code détecteur utilisé est un code non-ordonné comme le code de Berger. Ce dernier est un code séparable obtenu par 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. (fig.9).

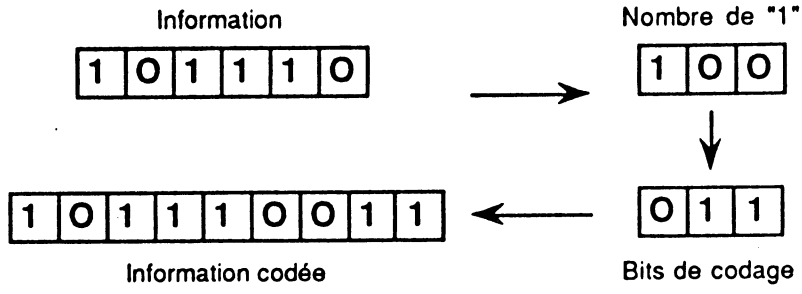


Fig II - 9 - Information codée avec le code de Berger

D'autre part, nous pouvons 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].

Ce code est optimal pour la détection des erreurs unidirectionnelles en ce sens qu'il nécessite le moins de bits de contrôle ($\lceil Log_2 (I+1)$) pour une information de I bits).

D'un point de vue pratique, le principe d'un contrôleur de Berger est donné en figure 10. Le contrôleur est composé de deux éléments :

- un générateur de code qui produit le numéro binaire des uns parmi les I bits d'information. Il est composé à partir d'une 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.

- un contrôleur double-rail qui compare les bits de contrôle de l'information avec les bits délivrés par le générateur de code.

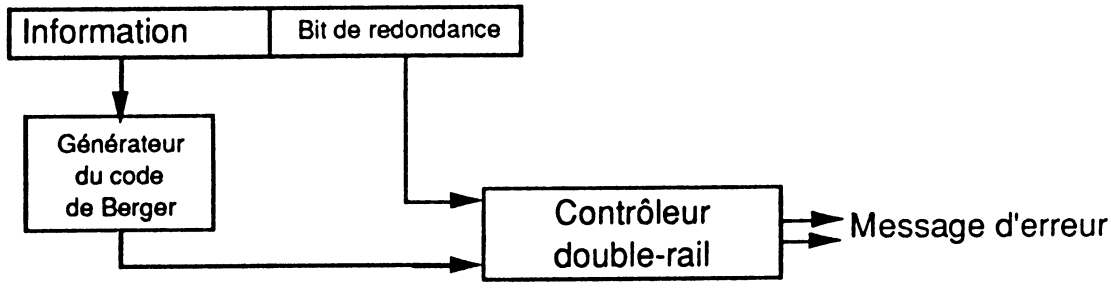


Fig II - 10 - Principe du contrôleur de Berger

4.4 Le code à duplication

Dans le cas d'erreurs multiples, le code le mieux adapté est le code à duplication. Il peut être de deux types:

- le code à duplication direct qui n'est autre que l'information utile et elle même.

- le code à duplication et complémentation (code double-rail) ou le code rajouté est le complément à un de l'information utile.

D'un point de vue pratique la duplication est obtenue en utilisant soit un bloc fonctionnel identique qui génère en parallèle la même information utile (cas de la duplication directe), soit un bloc fonctionnel dual qui délivre en parallèle le complément de l'information. Dans les deux cas le décodage est effectué en comparant les sorties à l'aide d'un contrôleur double-rail (fig.11).

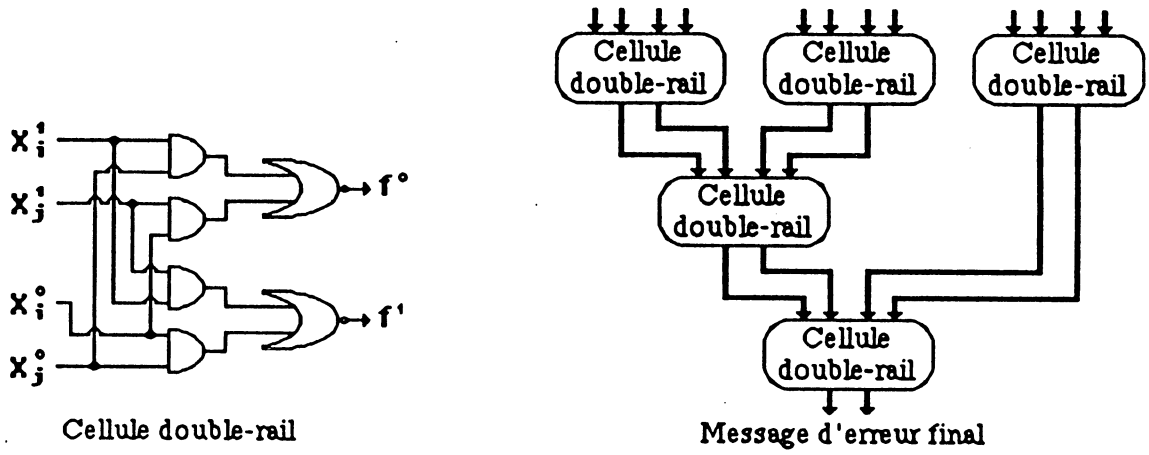


Fig II - 11 - Principe de contrôle d'un code à duplication directe [CAR 68]

Le test de chaque cellule à k paires d'entrée demande l'application des 2^k entrées et, en fait selon [AND 71], un arbre entier avec m paires d'entrées contruites par l'interconnexion de cellules de k paires d'entrées peut être diagnostiqué avec ces 2^k entrées de test adéquatement générées.

Toutes ces techniques de codage sont connues et essentiellement employées dans le test intégré autonome qui nous intéresse. Le principal inconvénient de ces méthodes est la surface supplémentaire de test qu'elles peuvent nécessiter (fig.12). Le code à parité occupe le moins de surface alors que la duplication peut demander plus du double de la surface initialement prévue (bloc initial +bloc dupliqué et contrôleurs associés).

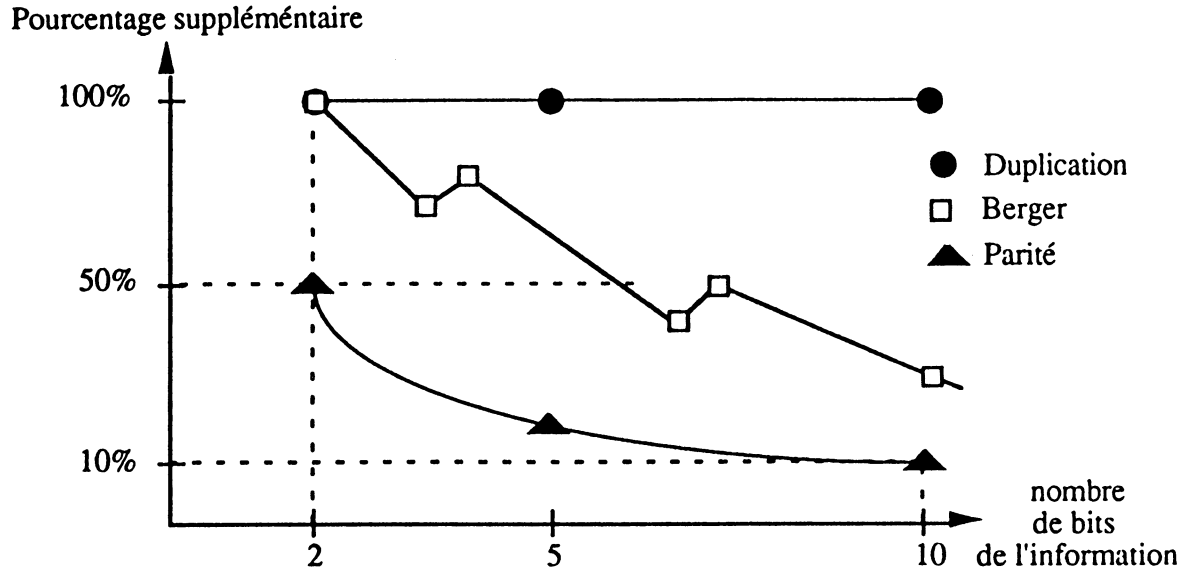


Fig II - 12 - Rapport bits de codage/bits d'information en fonction du code utilisé

Comme nous l'avons vu précédemment, l'analyse des pannes (Classe I) susceptibles de se produire à l'intérieur d'un circuit permet de déduire le type des erreurs provoquées sur ses sorties primaires. De là un type de code et une stratégie de conception sont adoptés. C'est à dire que le circuit est conçu en respectant plusieurs règles qui font que les erreurs internes au circuit se propageront aux sorties primaires de ce dernier en erreurs détectables par un contrôleur. Ces règles confèrent au circuit la propriété "self-checking". Ensuite, d'autres règles assurent que des pannes indétectables ne vont pas invalider les premières règles données, ce qui signifie que le circuit est rendu SFS.

Nous allons donc présenter quelques règles de conception en vue d'obtenir des circuits SFS pour la classe I des hypothèses de pannes.

5 - QUELQUES REGLES DE CONCEPTION DE CIRCUITS SFS POUR LA CLASSE I

Ces règles sont présentées en deux groupes, le premier groupe introduit la propriété FS du circuit avant l'occurrence d'une panne. Le second groupe garantit que le circuit conserve cette propriété en présence de séquences de pannes. Seules les principales règles qui nous intéressent pour les chapitres suivants sont données à titre d'exemple, elles sont tirées d'une liste exhaustive proposée dans [NIC 84].

5.1 Définitions pour les règles de conception

DEFINITION D'UN CHEMIN :

Un chemin est un ensemble ordonné de lignes entre deux points d'un bloc. Si dans l'ensemble <Ii1, ..., Iin> de lignes, la ligne Iij+1 est le successeur de Iij (cf : [NIC 84] pour la relation de succession entre deux lignes) pour j=1 jusqu'à n-1 alors cet ensemble ordonné est un chemin.

Soit deux lignes Ii et Ij où Ij est le successeur de Ii :

DEFINITION D'UN COUPLE INVERSANT :

Un couple de lignes (Ii, Ij) est inversant, si une erreur du type D "niveau bas à la place de niveau haut" (resp. ¬D "niveau haut à la place de niveau bas") sur la ligne Ii ne peut produire qu'une erreur de type ¬D (resp. D) sur la ligne Ij.

DEFINITION D'UN COUPLE NON INVERSANT :

Un couple de lignes (Ii, Ij) est non inversant, si une erreur du type D (resp. ¬D) sur la ligne Ii ne peut produire qu'une erreur de type D (resp. ¬D) sur la ligne Ij.

Par exemple :

- le couple (grille, drain) d'un MOS de signal est inversant.
- le couple (grille, drain/source) d'un MOS interrupteur n'est pas défini vis à vis de la propriété d'inversion.
- tous les autres couples sont non-inversants.

DEFINITION DE LA PARITE D'INVERSION :

La parité d'inversion Ip(P) d'un chemin P, dans lequel la propriété d'inversion est définie pour tous les couples de deux lignes successives, est le nombre binaire Ip(P) $\in$ {1,0} tel que :
Ip(P) = n modulo 2 , où n est le nombre de couples inversants de lignes successives qui se trouvent sur le chemin P.

DEFINITION DU DEGRE DE DIVERGENCE :

Le degré de divergence d'une ligne Ii1 interne d'un bloc fonctionnel est le nombre de sorties primaires Iin du bloc pour lequel il existe des chemins <Ii1, ..., Iin>.

5.2 Cas des erreurs simples [NIC 84], [NIC 85]

Ces règles de conception assurent la propriété SFS d'un circuit dans le cas où un code détecteur d'erreurs simples est utilisé.

Tout d'abord, quelques-unes des règles à respecter pour que le circuit soit "fault-secure" sont :

la REGLE R1 :
Le degré de divergence maximal du bloc fonctionnel, pour l'ensemble de toutes les lignes du bloc est égal à 1.

ou *la REGLE R'1* , qui est équivalente à le règle R1 :
Le degré de divergence maximal du bloc fonctionnel, pour l'ensemble de toutes les entrées primaires du bloc est égale à 1.

et *la REGLE R'2* (lorsque le modèle de pannes est considéré au niveau du transistor) :
Chaque ligne d'alimentation du bloc est utilisée pour alimenter un groupe de portes liées (par des chemins) à une seule sortie primaire de ce même bloc.

ou *la REGLE R"2* (dans les mêmes conditions que la règle précédente) :
Une seule ligne VSS (diffusion ou aluminium) et une seule ligne VDD (diffusion ou aluminium) sont utilisées. l'extrémité de ces lignes sert à alimenter les portes d'un contrôleur.

Par la suite nous présentons quelques règles qui garantissent la propriété SFS du circuit. Ce sont :

la REGLE R3 :
L'élimination des courts-circuits possibles entre deux lignes connectées (via des chemins) avec deux sorties primaires différentes du bloc fonctionnel, se fait soit par séparation topologique de quelques lignes, soit par utilisation de matériaux non court-circuitables.

et *la REGLE R4* (règle qui n'est pas nécessaire si la règle R"2 est vérifiée): :
Pour éliminer les courts-circuits possibles entre deux lignes d'alimentation du même type (VSS ou VDD) qui alimentent des portes dont les sorties sont liées, par l'intermédiaire de chemins, avec deux sorties primaires différentes du bloc, soit il faut déplacer topologiquement quelques lignes, soit il faut utiliser des matériaux non court-circuitables.

5.3 Cas des erreurs unidirectionnelles [NIC 84], [NIC 86 a]

Ces règles de conception assurent la propriété SFS d'un circuit dans le cas où un code détecteur d'erreurs unidirectionnelles est utilisé.

Tout d'abord,pPour la propriété "fault-secure" du circuit, quelques-unes des règles sont :

la REGLE R5 :

Tous les chemins entre une ligne divergente et les sorties primaires du bloc ont la même parité d'inversion.

ou *la REGLE R'5* , qui est équivalente à le règle R5:

Tous les chemins entre une entrée primaire divergente et les sorties primaires du bloc ont la même parité d'inversion.

et *la REGLE R6'* (lorsque le modèle de pannes est considéré au niveau du transistor) :

Chaque ligne d'alimentation d'un bloc est utilisée pour alimenter des groupes de portes dont les sorties sont liées aux sorties primaires du bloc par l'intermédiaire de chemins qui ont la même parité d'inversion.

Pour la propriété "strongly fault-secure", quelques-unes des règles suivantes sont à respecter :

la REGLE R7 :

Il faut éliminer les courts-circuits entre deux lignes connectées aux sorties primaires du bloc par l'intermédiaire de chemins qui ont des parités d'inversion différentes, soit par séparation topologique de certaines lignes, soit par utilisation de matériaux non court-circuitables.

et *la REGLE R8* :

Il faut éliminer les courts-circuits possibles entre deux lignes d'alimentation du même type (VSS ou VDD) telles que : l'une des lignes alimente des portes dont les sorties sont liées avec les sorties primaires du bloc par l'intermédiaire de chemins ayant des parités d'inversion égales à "0" . L'autre ligne alimente des portes dont les sorties sont liées avec les sorties primaires du bloc par l'intermédiaire de chemins ayant des parités d'inversion égales à "1". Ceci est possible si les quelques lignes concernées sont déplacées topologiquement ou si des matériaux non court-circuitables sont utilisés.

5.4 Cas des erreurs multiples [NIC 84]

Les règles de conception suivantes assurent la propriété SFS d'un circuit dans le cas où un code détecteur d'erreurs multiples est utilisé. Pour le code à duplication (resp. le code double rail) les sorties primaires sont partagées en deux groupes et à chaque sortie primaire d'un groupe est associée une sortie dupliquée (resp. double-rail).

Pour que la circuit ait la propriété FS, quelques-unes des règles suivantes sont à respecter :

la REGLE R9 :
Chaque ligne est liée, par l'intermédiaire de chemins, avec des sorties primaires qui appartiennent au même groupe.

et ***la REGLE R'10*** :
Chaque ligne d'alimentation et utilisée pour alimenter des portes dont les sorties sont liées, par l'intermédiaire de chemins, avec des sorties primaires du même groupe.

La propriété SFS est assurée par le respect de règles qui concernent d'une part les courts-circuits entre deux lignes qui sont liées avec des sorties primaires appartenant à deux groupes différents (règle R11). Mais aussi, les courts-circuits entre deux lignes d'alimentation du même type qui servent à alimenter des portes dont les sorties sont liées avec les sorties primaires appartenant à deux groupes différents (règles R12).

5.5 Application

Un exemple d'application des règles de conception à suivre pour la réalisation d'un PLA SFS est présenté en figure 13 dans le cas où des erreurs unidirectionnelles sont envisagées. 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, le circuit peut donc être testé en utilisant le contrôle des entrées primaires et un code de sortie non-ordonné.

5.5.1 Architecture global du PLA SFS

Le PLA (fig.13) est conçu pour qu'à chaque vecteur d'entrée (E1, .., En) corresponde un seul vecteur de sortie (S1, ..., Sn) et un vecteur unique (B1, ..., Bk) qui appartient normalement au code de Berger. D'autre part, à partir de (S1, ..., Sn) une circuiterie supplémentaire génère le code de Berger complémentaire de la sortie et un contrôleur compare ce dernier vecteur (B1*, .., Bk*) au vecteur (B1, ..., Bk). Le contrôleur d'entrée sert à détecter les pannes susceptibles de se produire sur les entrées primaires du PLA (en trait gras sur le dessin) qui provoqueraient des

erreurs multiples aux sorties primaires de ce circuit.

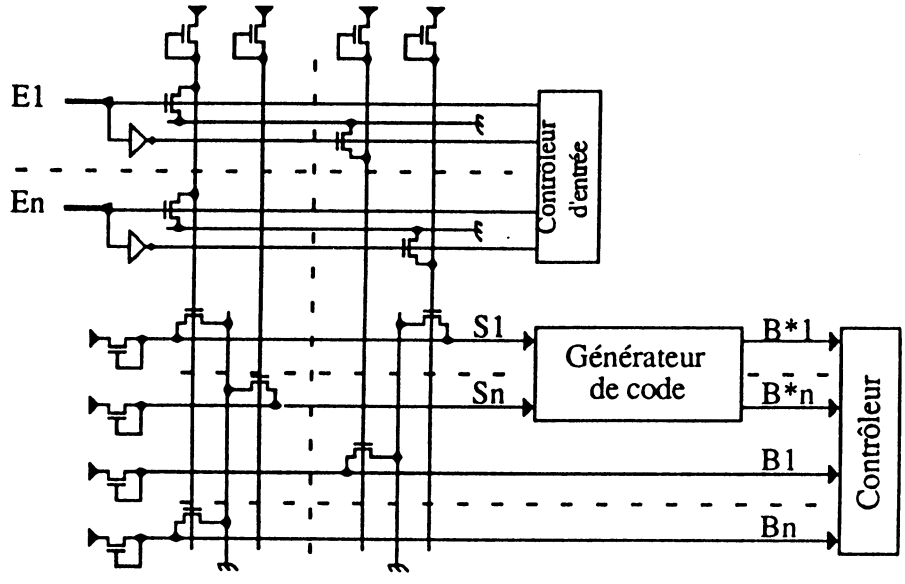


Fig II - 13 - Schéma d'un PLA "strongly fault-secure"

5.5.2 Vérification du respect des règles de conception

Le circuit de la figure 15 est FS pour un ensemble de pannes du type, collage d'une ligne à "1", collage d'une ligne à "0", court-circuit entre deux lignes et la présence ou l'absence anormale d'un transistor [NIC 84]. En effet cette propriété est assurée par le respect de la règle R5 et R6 puisque sur le schéma proposé nous vérifions que :

- Tous les chemins d'une ligne divergente ont la même parité d'inversion, parité qui est paire pour les lignes ne contenant pas d'inverseur et qui est impaire pour les autres (règle R5).

- Chaque ligne d'alimentation (Vss ou Vdd) alimente des groupes de portes dont les sorties sont liées aux sorties primaires du bloc par l'intermédiaire de chemins qui ont la même parité d'inversion (règle R6).

- La propriété SFS est obtenue puisque les règles R7 et R8 sont respectées.

6 - INTERCONNEXION DE BLOCS COMBINATOIRES SFS

Pour qu'un circuit combinatoire composé de plusieurs blocs "SFS" soit lui même "SFS", il faut que cette dernière propriété soit assurée pour les défauts sur les interconnexions. Il est donc

nécessaire, en fonction des pannes possibles dans les circuits et des codes détecteurs d'erreurs utilisés, de prendre des précautions particulières en ce qui concerne les interfaces. De même, les contrôleurs, quand ils sont nécessaires, ne peuvent pas toujours être placés aux hasard sur les lignes des interconnexions.

Par la suite nous considérons l'occurrence de pannes simples dans les circuits dont les sorties sont contrôlées vis à vis d'erreurs simples (code détecteur : code de parité) ou d'erreurs unidirectionnelles (code détecteur : codes non-ordonnés). Le cas des circuits dont les sorties sont contrôlées vis à vis d'erreurs multiples ne sera pas traité puisque leur conception fait appel à la méthode classique de duplication des blocs.

6.1 Cas des blocs fonctionnels SFS et SCD

La méthode propose de réunir deux blocs fonctionnels qui sont chacun "strongly fault-secure" et "strongly code disjoint" pour que la propriété "strongly fault-secure" de l'ensemble puisse être assurée sans utiliser de contrôleurs aux différentes interfaces sauf sur les sorties primaires du système (fig.14).

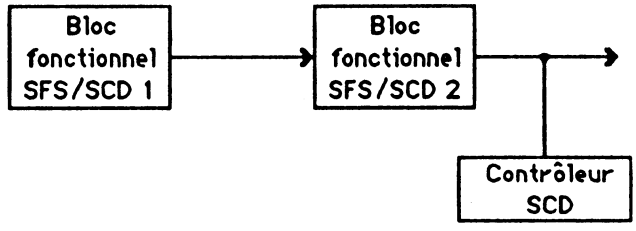


Fig II - 14 - Blocs fonctionnels SFS/SCD et contrôleur SCD

Dans ce cas de figure, dans la mesure où tous les vecterus du code sont appliqués aux entrées du bloc 1 et du bloc 2, les propriétés SFS et SCD de chaque bloc fonctionnel assurent que toute erreur sur les sorties d'un des éléments sera propagée en en un vecteur hors-code sur les sorties primaires du système. Par conséquent, un seul contrôleur "strongly code disjoint" est utilisé pour assurer la détection de pannes susceptibles d'apparaître dans le circuit global.

Toutefois la conception d'une fonction complexe qui soit à la fois "strongly fault-secure" et "strongly code disjoint" peut être difficile à obtenir [NAN 89]

6.2 Cas des blocs fonctionnels SFS

Ici une première méthode consiste à assembler deux blocs fonctionnels "strongly fault-secure" qui sont aussi vérifiés par un seul contrôleur placé sur les sorties primaires du système.

La propriété "strongly fault-secure" de l'ensemble est alors assurée pour les blocs qui respectent les règles suivantes [NIC 84] :

- le premier bloc fonctionnel est testé par un code détectant les erreurs simples et de par sa conception il ne produit que des erreurs simples sur ses sorties primaires. Le deuxième bloc vérifie la règle R1 (R'1) ou la règle R5 (R'5) (§ 5.2 et 5.3) de manière à ce que la propriété concernant la propagation d'erreurs soit satisfaite :

Chaque erreur simple aux entrées d'un bloc qui vérifie la règle R1 (R'1) (fig.15) ou R5 (R'5) est propagée jusqu'aux sorties primaires de ce bloc soit comme une erreur simple, soit comme une erreur unidirectionnelle.

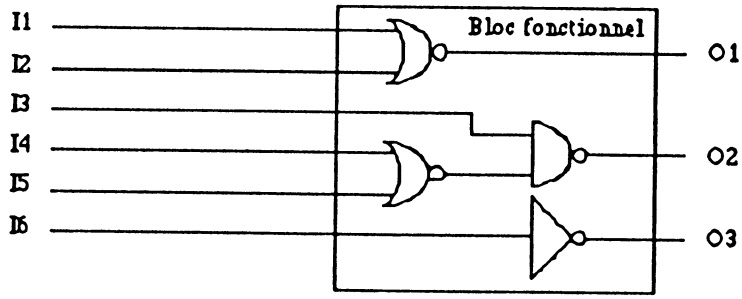


Fig II - 15 - Bloc fonctionnel vérifiant la règle R1

Par conséquent, le circuit global ainsi constitué est testé uniquement par un seul contrôleur "strongly code disjoint" qui vérifie les sorties primaires du dernier bloc (fig.16).

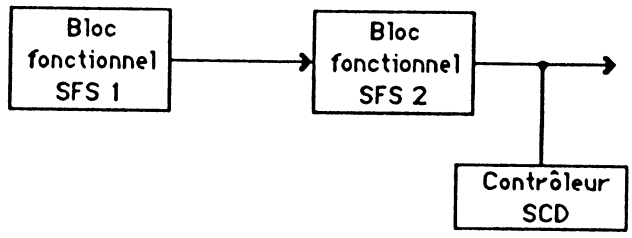


Fig II - 16 - Blocs fonctionnels SFS et contrôleur SCD

- les erreurs présentes aux sorties du premier bloc sont des erreurs unidirectionnelles et ce dernier est testé par un code de sortie non-ordonné. Le code de sortie du premier bloc est utilisé comme code d'entrée du second bloc fonctionnel qui est conçu en vérifiant la règle R'5 et dont le code de sortie détecte les erreurs unidirectionnelles.

Pour ce bloc, la propriété qui concerne la propagation des erreurs est satisfaite :

les erreurs unidirectionnelles présentes aux sorties du premier bloc fonctionnel sont propagées en erreurs unidirectionnelles aux sorties primaires du second bloc.

Ici aussi, le système global est testé par un seul contrôleur "strongly code disjoint" qui vérifie les sorties primaires du dernier bloc uniquement.

Le but étant d'utiliser le moins possible de contrôleurs tout en veillant à assurer la propriété "SFS" globale, *Nanya* [NAN 88] présente la même idée de façon plus formelle en introduisant le concept d'interface "error secure/error propagating" pour des codes non ordonnés qui sont utilisés en sortie de chaque bloc. Toutefois dans cette dernière méthode, la présence d'un vecteur correct en sortie du système global ne signifie pas forcément que les sorties des blocs précédents sont codées correctement.

La seconde méthode que nous présentons considère l'assemblage de deux blocs "strongly fault-secure" (fig.17) pour lequel le second bloc n'est pas capable de détecter la présence d'erreurs simples (resp. d'erreurs unidirectionnelles) sur ses entrées.

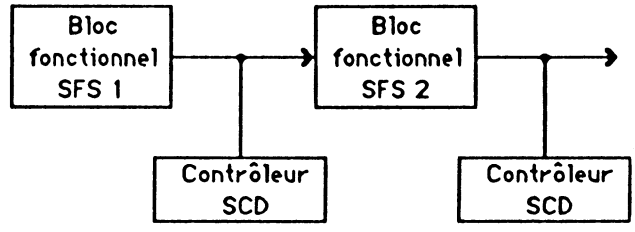


Fig II - 17 - Blocs fonctionnels SFS et contrôleur SCD

Dans ce cas, la propriété "strongly fault-secure" de l'ensemble est assurée si un contrôleur "strongly code disjoint" est placé entre les deux blocs fonctionnels pour tester les interconnexions. Mais ce contrôleur ne peut pas être placé de manière quelconque car les pannes qui apparaissent sur les parties d'interconnexions comprises entre les entrées du contrôleur et les entrées primaires du second bloc ne peuvent ne pas être détectées par ce dernier. Pour palier cet inconvénient, le contrôleur se situe de façon à vérifier les interconnexions jusqu'à certains points de bifurcation (points critiques) [NIC 84] à partir desquels le second bloc est capable de détecter toute présence de panne (fig 18).

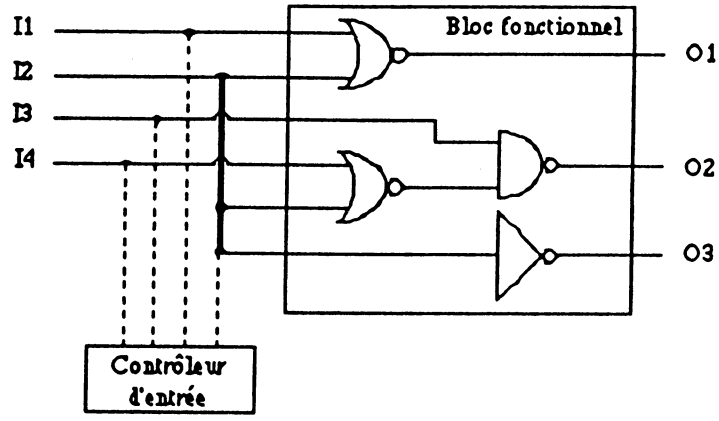


Fig II - 18 - Méthode de contrôle des entrées primaires

7 - LE CAS DES CIRCUITS SEQUENTIELS

La définition des circuits SFS donnée précédemment est valable pour les circuits combinatoires seulement. Pour les circuits séquentiels, il existe des hypothèses de pannes qui leurs sont particulières et des définitions de base données par [OZG 77], [DAV 78], [DIA 79],et [NAN 87]. Une famille de contrôleurs "strongly language disjoint" (SLD) définie en [JAN 85], [JAN88], est associée aux circuits séquentiels "sequentially self-checking" [VIA 80] pour constituer la plus grande classe de circuits séquentiels capables d'atteindre le "totally self-checking goal" sous certaines hypothèses de fonctionnement.
Quelques-unes de ces propriétés relatives aux circuits séquentiels seront précisées dans le chapitre suivant qui traite d'une méthode de conception de machines séquentielles autotestables.

8 - ETAPE ULTIME DU TEST INTEGRE : LE BIST UNIFIE

Pour assurer d'une façon efficace tous les types de tests nécessaires aux circuits intégrés, il faut envisager à la fois le test hors-ligne (techniques BIST) et le test en-ligne (techniques "self-checking"). Comme ces deux types de test ont été étudiés indépendamment, leur intégration commune à nécessité le développement de nouvelles méthodes afin de pouvoir exploiter au maximum les avantages que l'un peut apporter à l'autre.

Les études qui ont été menées dans ce domaine ont apporté des résultats prometteurs en aboutissant à la technique de BIST unifié (UBIST) [NIC 86 b], [NIC 88 a]. Cette dernière représente pour les circuits "self-checking" ce que les schémas classiques du type BIST représentent pour les circuits intégrés ordinaires.

8.1 Préliminaire

Aujourd'hui, bien qu'un niveau très avancé soit atteint en ce qui concerne la théorie des circuits "self-checking", les problèmes suivants persistent :

a) le contrôleur ne peut garantir la propriété SCD ou TSC que si le bloc fonctionnel auquel il est rattaché est capable de générer un ensemble minimal de vecteurs lui permettant d'assurer le test. Très souvent, cette condition n'est pas vérifiée pour les circuits réels. De plus, les contrôleurs ne peuvent être SCD ou TSC que pour des modèles de pannes restreintes (modèle de collage logique).

b) le système étant conçu de manière à ce que les contrôleurs soient SCD et les blocs fonctionnels SFS, pendant le fonctionnement du circuit, chacun d'eux doit recevoir sur ses entrées les vecteurs d'un code donné pour assurer le TSC goal. Ceci implique des restrictions sur le logiciel à utiliser.

c) les contrôleurs et les blocs fonctionnels sont conçus pour couvrir les pannes simples, mais pendant la fabrication, il peut se produire des pannes multiples. Afin de signaler ces dernières, les circuits "self-checking" doivent être testés en fin de fabrication.

C'est donc en grande partie pour résoudre ces différents inconvénients que la technique "Unified-Built-In-Self-Test" (UBIST) spécifique aux "circuits self-checking" a été développée par *Nicolaidis* [NIC 86 b], [NIC 88 a]. Cette technique qui propose de compléter le test en-ligne des circuits par l'activation périodique de séquences de test hors-ligne a comme éléments de base les contrôleurs "self-exercising" SCD [NIC 88 b].

8.2 Les contrôleurs "self-exercising" SCD

Toute la technique UBIST est basée sur le concept des contrôleurs "self-exercising" ("contrôleurs auto-exercés) SCD. Pour ces derniers, la propriété SCD n'est pas assurée au sens donné par [NIC 84] mais en utilisant des mécanismes intégrés de génération de vecteurs de test (fig.19). Pour de tels contrôleurs, les vecteurs de test injectés par un générateur spécifique le sont au cours d'une phase de test insérée pendant la phase de fonctionnement normal. Par ce moyen, certains vecteurs qui appartiennent au code de sortie du bloc fonctionnel contrôlé mais qui ne sont jamais générés par ce bloc (à cause de la structure du bloc ou du programme utilisateur) peuvent être appliqués aux entrées du contrôleur. D'autre part, une partie des vecteurs (ou tous les vecteurs) hors du code d'entrée du contrôleur ont la possibilité d'être injectés sur ses entrées (ce qui est impossible dans le cas des circuits "self-checking" ordinaires). Une circuiterie (ICHC en figure 19) indiquera la nature (élément hors-code ou élément du code) de l'ensemble des vecteurs délivrés par le générateur.

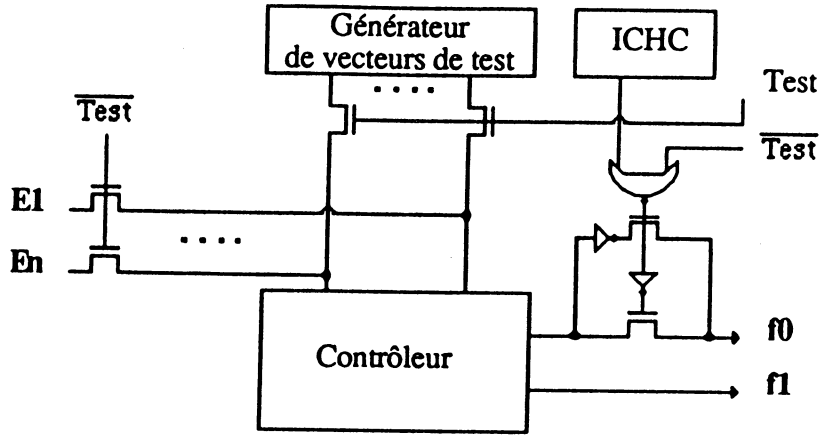


Fig II - 19 - Schéma général d'un contrôleur "self-exercising" [NIC 88 b]

Les contrôleurs "self-exercising" SCD qui ont des structures et des comportements différents des contrôleurs SCD classiques sont définis de la manière suivante :

DEFINITION D8 :

Un contrôleur "self-exercising" est un contrôleur "self-testing" pour un ensemble de pannes F, si pour tout $f \in F$:

- soit un vecteur du code d'entrée est transformé en un vecteur hors-code de sortie pendant le fonctionnement normal du circuit.
- soit pendant la phase "test", il y a aux sorties du contrôleur un vecteur qui n'appartient pas au code de sortie.

Suite à cette définition, le théorème suivant est énoncé et démontré en [NIC 88 b] :

THEOREME 1 :

Si le générateur de vecteurs de test donné en figure 19 génère tous les vecteurs hors-code d'entrée du contrôleur, alors le contrôleur "self-exercising" est SCD :

- pour tous les types de pannes possibles affectant le contrôleur,
- pour toutes les conceptions possibles du contrôleur,
- indépendamment du code de sortie du bloc fonctionnel et du programme utilisateur.

8.3 Schéma de principe d'une conception UBIST

La conception en vue du test unifié (Unified-Built-In-Self-Test) assure la détection de pannes dans le circuit, d'une part pendant la phase de fonctionnement normal (test en-ligne) et d'autre part au cours de la phase de test hors-ligne en autorisant la génération interne de séquences de vecteurs de test qui vérifient le bon fonctionnement des blocs fonctionnels et des contrôleurs.

Les mécanismes qui sont capables de délivrer les séquences de vecteurs de test hors-ligne désirées peuvent être réalisés selon la procédure donnée en [KON 79] (BILBO). Ces éléments ont plusieurs modes de fonctionnement : registre classique d'entrées/sorties, registre à décalage linéaire, registre à décalage à rebouclage linéaire (LFSR utilisé comme générateur de vecteurs de test ou comme analyseur de signature). Ils sont parfaitement adaptés au type de test hors-ligne à effectuer dans le cadre d'une conception UBIST. Par contre pour cette dernière approche, les mécanismes BILBOs subissent quelques modifications de structure pour devenir des registres UBILBOs [NIC 88 a] et permettre le test des contrôleurs spécifiques définis ci-dessus.

Pendant le fonctionnement normal du circuit (fig.20), le test en ligne permet une vérification des blocs fonctionnels SFS et des contrôleurs SCD.

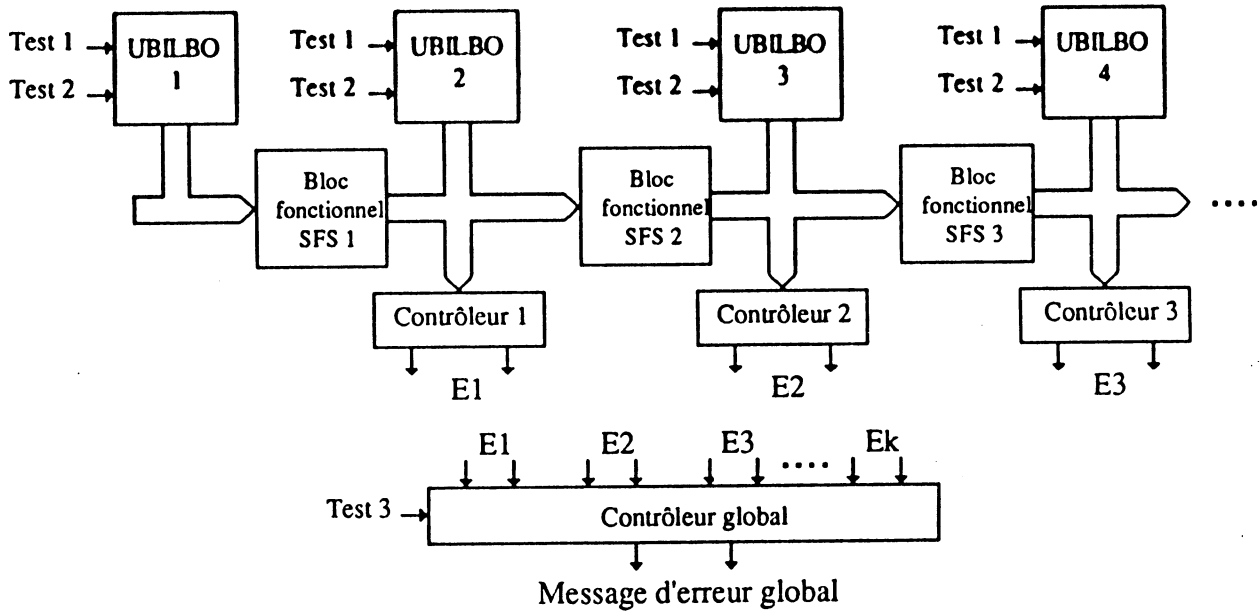


Fig II - 20 - Schéma de principe d'une conception UBIST

En ce qui concerne le test hors-ligne, le bon fonctionnement du contrôleur global qui compacte les signaux d'indication d'erreur (E1k, ..., E1k) est vérifié pendant la phase "Test 3". Durant la phase "Test 1", les UBILBOs impairs (1,3,...) délivrent des vecteurs de test sur les entrées des blocs fonctionnels impairs (1, 3, ...) et sur les entrées des contrôleurs pairs (2,4,...). Les réponses de chacun des blocs sous test (1, 3, ...) sont vérifiées par les contrôleurs impairs correspondants (1, 3, ...) mais elle sont aussi compactées (en vue d'obtenir la signature finale) par les UBILBOs pairs associés (2, 4, ...). La défaillance d'un contrôleur pair (2, 4, ...) pendant cette phase de test est immédiatement signalée par le contrôleur global. A la fin de la période "Test 1" les signatures des différents UBILBOs pairs sont contrôlées.

Le principe est le même pour la phase de test "Test 2" où les blocs fonctionnels SFS pairs et les contrôleurs SCD impairs sont vérifiés.

8.4 Conclusion sur la méthode UBIST

En adoptant une telle méthode de conception, nous pouvons affirmer que,
le test en ligne assure :

- la propriété SFS des blocs fonctionnels pour toutes les pannes simples.
- la propriété SCD des contrôleurs, quel que soit le type de panne considéré.
- l'hypothèse selon laquelle un ensemble suffisant de vecteurs doit être appliqué régulièrement pour la bonne vérification des blocs fonctionnels et des contrôleurs.

le test hors-ligne assure :

- la détection des pannes simples et multiples affectant les blocs fonctionnels.
- la détection de toutes les pannes qui surviennent dans les contrôleurs et les UBILBOs.

Une telle conception qui combine harmonieusement le test en-ligne et le test hors-ligne garantit le bon fonctionnement des contrôleurs et des blocs fonctionnels de manière à ce que le circuit intégré global accomplisse le "totally self-checking goal". Cette technique confère à n'importe quel contrôleur la propriété SCD. Enfin, des dispositifs de test hors-ligne puissants sont disponibles, ce qui permet de garantir une couverture très élevée de pannes et de résoudre les problèmes énoncés aux points a), b) et c) de ce paragraphe.

9 - CONCLUSION

Nous avons pu voir dans ce chapitre les principales techniques utilisées pour la conception de circuits intégrés qui ont la faculté de détecter leurs propres pannes immédiatement après l'apparition des premiers résultats erronés. Cependant les méthodes d'autotest en-ligne ne sont effectives que si certaines conditions sont respectées. Dans le cas de réalisations pratiques de circuits, le respect de ces contraintes devient extrêmement difficile voire impossible à satisfaire. C'est donc pour résoudre ce problème qu'une technique de conception en vue du test unifié (UBIST) a été proposée il y a quelques années par *Nicolaidis*. Elle consiste à intégrer le test en-ligne et le test hors-ligne dans le même circuit (c'est l'étape ultime du test intégré) et elle se fait à des coûts raisonnables en assurant une très haute qualité pour tous les tests nécessaires à la bonne réalisation du produit. Enfin nous pouvons prédire que l'augmentation de la complexité des circuits et l'augmentation du nombre d'applications exigeant une sûreté importante mèneront rapidement à l'industrialisation des deux techniques présentées dans ce chapitre.

CHAPITRE III

III - REALISATION D'AUTOMATES AUTOTESTABLES

III - REALISATION D'AUTOMATES AUTOTESTABLES

1 - INTRODUCTION

Pour satisfaire au mieux les critères de sécurité, de disponibilité, de fiabilité ou de maintenabilité [LAP 85] la conception de systèmes électroniques à haute sûreté de fonctionnement à donné lieu à de multiples solutions. En ce qui concerne les automatismes employés dans des systèmes critiques (systèmes de transport, automatismes industriels de sécurité, ..) les solutions s'avèrent toutes plus ou moins efficaces mais coûteuses puisqu'elles utilisent, pour la plupart, de la redondance matérielle massive et/ou de la redondance logicielle combinées à de la logique à sécurité intrinsèque [LIE 76] dont la réalisation pratique est très lourde.

Aujourd'hui, les progrès réalisés en microélectronique en matière d'intégration, de performances et, comme nous l'avons vu dans les chapitres précédents, dans le domaine du test des circuits digitaux, sont des atouts qui permettent d'envisager l'intégration d'automatismes sûrs (au sens des défaillances catastrophiques [LAP 85]) à moindres coûts.

C'est dans ce contexte que les deux prochains chapitres proposent une solution pour la conception VLSI (Very Large Scale Integration) d'automatismes de sécurité. Après quelques rappels d'ordre général concernant le principe de la sécurité intrinsèque et le savoir faire en matière d'automates finis "self-checking", ce présent chapitre se limitera à la proposition de conception d'un automate autotestable de référence spécifique à un besoin. Dans le chapitre suivant, nous verrons de quelle manière le principe de sécurité intrinsèque est pris en compte et traduit lorsqu'il s'agit de concevoir des automatismes autotestables de sécurité.

2 - QUELQUES RAPPELS

2.1 Le principe de la sécurité intrinsèque

Dès qu'un niveau élevé de sécurité (au sens des défaillances catastrophiques) est imposé à des systèmes complexes (mécaniques, électrotechniques, électroniques, ...), la conception par "intuition", par respect des normes ou des règles traditionnelles ne sont plus des garanties suffisantes. Il faut alors s'appuyer sur des méthodes rigoureuses qui permettent d'identifier tous les "chemins" susceptibles de conduire à un accident et d'en estimer les probabilités. Puis, en fonction d'objectifs de sécurité, il faut déterminer si les risques correspondants sont acceptables ou non, et dans ce cas il devient nécessaire de réduire la probabilité de l'accident (prévention) ou, à défaut, d'en minimiser les conséquences. Parmi toutes les méthodes qui concourent à la maîtrise des risques, certaines de celles qui sont classées dans la catégorie qui vise la prévention des accidents mettent en oeuvre des techniques qui consistent à la limitation des paramètres dangereux [LIE 76].

2.2.1 Les techniques de limitation des paramètres dangereux

Pour de nombreux systèmes, lorsqu'un risque de défaillance non acceptable a été mis en évidence, il n'est pas toujours possible de l'éliminer à des coûts raisonnables. Mais il est parfois possible d'identifier les paramètres dangereux et de limiter leur variation de telle manière que la probabilité d'accident soit considérablement réduite. Pour atteindre ce résultat quatre voies sont possibles :

- La réduction du domaine autorisé,
- La sécurité intrinsèque,
- Le contrôle des limites,
- La surveillance continue des paramètres dangereux.

Mais avant d'en exposer brièvement les principes, il convient de définir les notions de domaine de fonctionnement, domaine autorisé et domaine périphérique.

- Domaine de fonctionnement, domaine autorisé et domaine périphérique :

A un moment donné, un système est caractérisé par un ensemble de paramètres qui peuvent être considérés comme les coordonnées d'un point de fonctionnement. A certains de ces paramètres est associé un risque, c'est à dire qu'il existe un domaine de fonctionnement tel que la probabilité d'accident soit très élevée dès que le point de fonctionnement sort de ce domaine. Pour assurer la sécurité du système, il faut rester dans le domaine de fonctionnement.

Dans ce but, le domaine de fonctionnement est divisé en deux parties, le domaine autorisé et le domaine périphérique.

Le domaine autorisé correspond au domaine dans lequel le point de fonctionnement a une possibilité suffisamment faible , pour être acceptable, de sortir du domaine de fonctionnement.
Le domaine périphérique correspond au domaine dans lequel le point de fonctionnement a une probabilité jugée inacceptable de sortir du domaine de fonctionnement compte tenu de facteurs comme l'influence de l'environnement, l'apparition de pannes, ...

Les frontières du domaine de fonctionnement constituent les limites de fonctionnement, les frontières du domaine autorisé constituent les limites autorisées (fig.1).

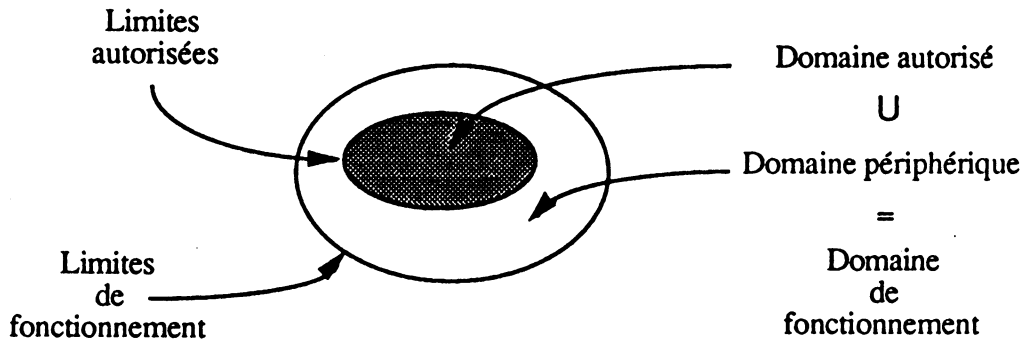


Fig III - 1 - Domaine de fonctionnement d'un système

Dans l'utilisation normale du système, le point de fonctionnement doit rester dans le domaine autorisé, il ne peut entrer dans le domaine périphérique qu'exceptionnellement et il faut alors le faire revenir , en général, dans le domaine autorisé.

La notion de domaine étant précisée, nous présentons maintenant les solutions évoquées pour les techniques de limitation des paramètres dangereux.

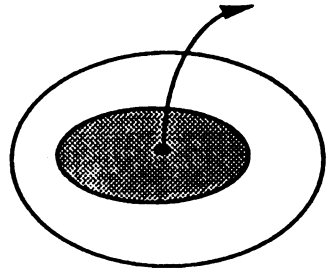
- La réduction du domaine autorisé : La réduction du domaine autorisé permet parfois de notables améliorations en ce qui concerne la sécurité d'un système, qui est jugée insuffisante, en laissant plus de "marge" pour permettre l'éxécution d'une action correctrice avant que le point de fonctionnement franchise les limites de fonctionnement. La limitation légale de la vitesse sur les route constitue un exellent exemple.

- La sécurité intrinsèque : Le but de la sécurité intrinsèque ne consiste pas à éliminer le risque mais de faire en sorte que le pire des cas de fonctionnement du système ne représente pas un gros danger pour le système qui interagit avec lui, le niveau de gravité est considérablement réduit. Par exemple, pour l'utilisation d'un appareil électrique qui fonctionne sur courant alternatif 220V et le risque d'électrocution associé, la prévention du risque consisterait à remplacer cet appareil par un élément ayant la même fonction mais fonctionnant manuellement ou à air comprimé.
Lorsqu'un paramètre a été traité avec le concept de sécurité intrinsèque, le système ne peut plus sortir du domaine autorisé du fait de ce paramètre.

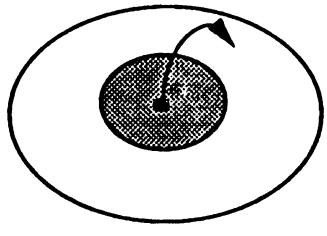
- Le contrôle des limites : Un dispositif de contrôle des limites réduit automatiquement un paramètre dès que celui-ci atteint une intensité qui pourrait porter gravement atteint au niveau de sécurité du système. Ce dispositif fonctionne dès que le point de fonctionnement du système entre dans le domaine périphérique par franchissement de la limite qui fait l'objet du contrôle. Par exemple, l'utilisation de soupapes de sécurité dans des enceintes préssurisées permet, au delà d'une certaine pression, de laisser échapper le fluide pour ramener la pression à une valeur normale.

- La surveillance continue des paramètres dangereux : Lorsqu'un paramètre est traité avec le concept de surveillance continue, associée à un contrôle automatique du danger, ce paramètre est suivi en permanence. Dès que son évolution fait sortir le point de fonctionnement du domaine autorisé, la mise en oeuvre d'un dispositif de contrôle automatique assure le maintien d'un niveau convenable de sécurité. Par exemple, introduction d'un gaz inerte dans une atmosphère où existent des risques d'explosion dès que la concentration de matières inflammables dépasse un certain seuil.

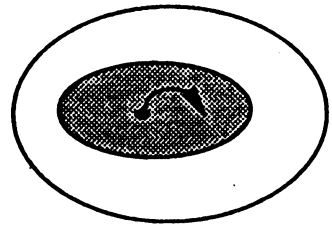
Le trajet possible du point de fonctionnement dans les divers concepts que nous venons d'évoquer fait l'objet de la figure 2. Evidemment, la mise en oeuvre pratique de tous ces concepts est étroitement liée à la nature et à la technologie de chaque système.



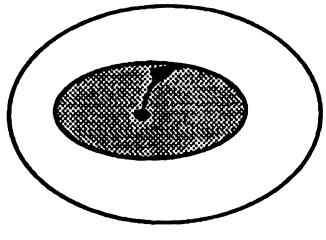
a/ Aucune disposition particulière n'a été prise



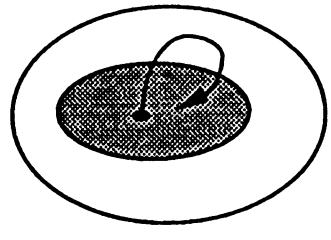
b/ Réduction du domaine autorisé



c/ Concept de sécurité intrinsèque



d/ Contrôle des limites



e/ Surveillance des paramètres dangereux

Fig III - 2 - Trajet du point de fonctionnement
dans les divers concept de limitation des paramètres dangereux

2.2.2 Le principe de la sécurité intrinsèque dans la technologie électronique

Le principe de base mis en oeuvre pour la conception de systèmes électroniques consiste à recenser les composants (discrets ou intégrés) à utiliser et à identifier de façon exhaustive leurs

modes de panne reconnus comme possible. A partir de ce moment, la méthode consiste à réaliser le système de manière à ce que toute les pannes soient polarisées dans un sens restrictif . Par exemple, étudions le cas de la logique dynamique de sécurité positive (où logique à panne orientée) [SAL 75] :

Une logique binaire normale présente deux états logiques possibles "0" et "1" matérialisés le plus souvent par un niveau bas et un niveau haut de tension. Dans de tels systèmes, la défaillance d'un composant se traduit par la génération de valeurs erronées et les deux types d'erreurs possibles sont 0 --> 1 ou 1 --> 0.

Dans une logique dynamique, l'information binaire "0-1" n'a plus pour support les niveaux statiques de tension bas et haut. Pour cette logique :

- l'état logique "1" correspond à une commutation niveau bas - niveau haut qui s'effectue à une fréquence donnée et qui ne peut pas exister en cas de défaillance d'un composant.

- l'état logique "0" correspond à tous les autres états électriques : niveau bas de tension, niveau haut continu de tension, niveau commutant à une fréquence différentes de celle qui est convenue.

Cette logique dynamique est utilisée conformément au principe de la sécurité intrisèque si on fait correspondre :

- à l'état "0", l'action de sécurité. Cet état est un état sûr.
- à l'état "1", l'état normal ou état de veille du système.

L'élément de base d'un système de logique dynamique est la cellule donnée en figure 3.

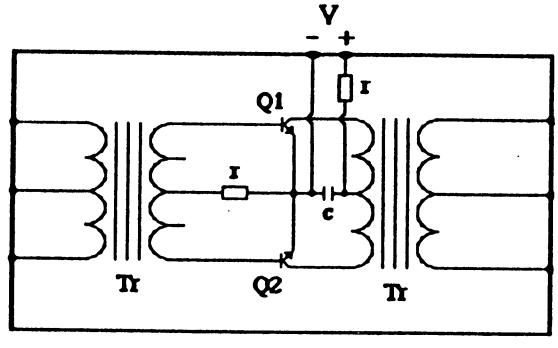


Fig III - 3 - Cellule de base

Cette cellule est apte ou non à transmettre à une autre cellule identique le signal carré de fréquence F appliqué à l'amplificateur selon que celui-ci est alimenté ou non. On vérifiera que toutes les défaillances de composant envisageables dans cet élément de base se traduisent, en sortie, par un état sûr.

2.2 Les automates finis "self-checking"

2.2.1 Les automates à états finis

Les automates à états finis sont des machines séquentielles qui ne peuvent être que dans un nombre fini d'états. Comme la définition exacte d'un tel automate exige une rigueur mathématique qui ne s'impose pas ici, nous nous proposons de faire comprendre simplement la nature du concept à l'aide de quelques définitions courantes **[HAR 65]**.

De façon classique, un automate est une machine séquentielle qui peut être rmodélisée par deux fonctions combinatoires :

- la fonction de transition F : (état courant, conditions) → état suivant.

- la fonction de sortie G : (état courant, (conditions)) → commandes.

Ce qui caractérise un automate déterministe est que pour une condition donnée dans un état donné la commande est toujours la même. Pour un automate non déterministe, cette même condition peut déclencher plusieurs transitions, donc plusieurs commandes.
Si la fonction G donne une commande à partir de chaque état, l'automate est un automate du type de Moore défini en D1. Par contre, si les commandes sont les résultats des transitions plutôt que des états eux-mêmes, l'automate est un automate du type de Mealy défini en D2.

DEFINITION D1 :

Une machine d'états finis affectée par états, ou automate de Moore (fig.4), est définie par un sextuplet $M(Q, X, Z, \delta, \omega, q0)$ où :
Q est un ensemble fini d'états internes,
X est un ensemble fini de vecteurs d'entrée,
Z est un ensemble fini de vecteurs de sortie,
δ est une fonction de transition d'états. $\delta : Q \times X \rightarrow Q$,
ω est une fonction de sortie. $\omega : Q \rightarrow Z$,
$q0 \in Q$ est l'état initial.

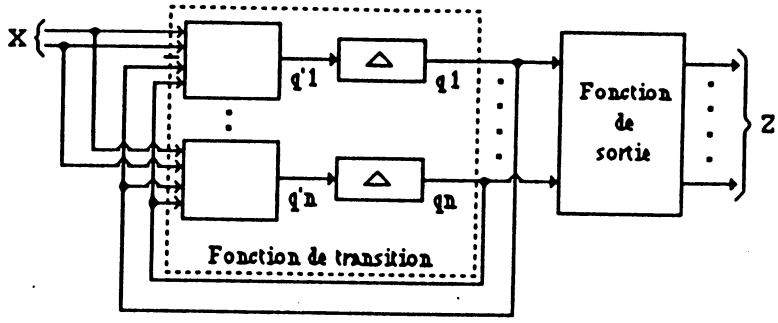


Fig III - 4 - Schéma d'un automate du type de Moore

DEFINITION D2 :

Une machine d'états finis affectée par transitions, ou automate de Mealy (fig.5) est définie par un sextuplet M(Q, X, Z, δ, ω, q0) où :

Q est un ensemble fini d'états internes,

X est un ensemble fini de vecteurs d'entrée,

Z est un ensemble fini de vecteurs de sortie,

δ est une fonction de transition d'états. $\delta : Q \times X \rightarrow Q$,

ω est une fonction de sortie. $\omega : Q \times X \rightarrow Z$,

$q0 \in Q$ est l'état initial.

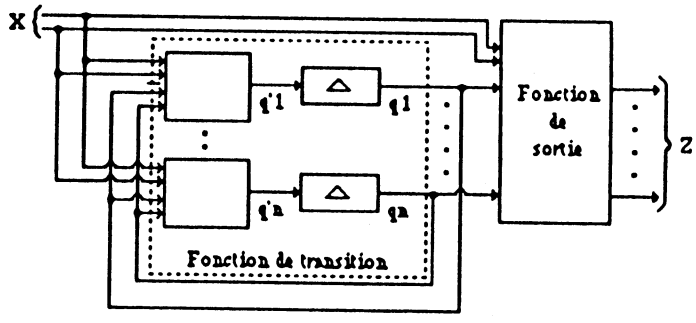


Fig III - 5 - Schéma d'un automate du type de Mealy

Notons que toute tâche qui est accomplie par un type d'automate peut aussi bien être exécutée par l'autre avec, si nécessaire, un nombre d'états différents [HAY 84].

2.2.2 Les machines séquentielles "self-checking"

Comme pour les circuits combinatoires, l'occurrence d'une panne dans un circuit séquentiel peut être détectée dès l'apparition des premières sorties erronées si ce dernier a été conçu suivant certaines caractéristiques. En fait les définitions des circuits séquentiels qui permettent d'accomplir le "totally self checking goal" ("TSC goal") [DAV 78], [DIA 79], [VIA 80] consistent en une généralisation des définitions données dans le cas des circuits combinatoires "self-checking".

D'une façon informelle les circuits séquentiels capables d'atteindre le "TSC goal" ont leurs sorties codées et ils peuvent être caractérisés comme suit [DAV 78], [NAN 87] :

DEFINITION D3 :

Un circuit séquentiel est "fault secure" pour un ensemble de pannes F, si et seulement si pour toute panne f ∈ F le circuit ne produit jamais de sorties erronées appartenant au code de sortie en vigueur.

DEFINITION D4 :

Un circuit séquentiel est "self-testing" pour un ensemble de pannes F, si et seulement si pour toute panne f ∈ F le circuit produit à un moment donné de son fonctionnement une sortie en dehors du code en vigueur.

DEFINITION D5 :

Un circuit séquentiel est "totally self-checking" si et seulement si il est "fault secure" et "self-testing".

DEFINITION D6 :

Un circuit séquentiel est "strongly fault secure" pour un ensemble de pannes F, si et seulement si pour toute panne f1∈ F,

- a / soit le circuit est "totally self-checking",

- b / soit le circuit reste "fault secure" et si une nouvelle panne f2 ∈ F apparaît, l'une des deux propositions précédentes reste vérifiée pour la panne composée (f1, f2). □

De la même façon que pour les circuits combinatoires, les circuits séquentiels "strongly fault secure" accomplissent le "totally self-checking goal" sous certaines hypothèses de fonctionnement spécifiques qui sont :

HYPOTHESE H1' :

Les pannes surviennent une à une.

HYPOTHESE H2' :

Entre l'occurrence de deux pannes dans la machine séquentielle, il s'écoule un laps de temps assez long pour qu'un nombre suffisant de combinaisons en entrée soient appliquées de manière à ce que la première panne, si elle est détectable, soit signalée.

Ces définitions d'un niveau très général permettent une description simple des circuits séquentiels "self-checking". Toutefois, des définitions beaucoup plus formelles et plus générales considèrent les fonctions δ et ω de l'automate et prennent en compte les langages d'entrée et de sortie de la machine [VIA 80].

Nous rappelons brièvement ces définitions :

DEFINITION D7 :

Le langage d'entrée d'une machine M, noté IM, est l'ensemble de toutes les séquences i d'entrée qui peuvent être appliquées à partir de l'état initial q0 de la machine en fonctionnement normal.

DEFINITION D8 :

Le langage de sortie de la machine M, noté SM, est l'ensemble de toutes les séquences de sortie qui peuvent être obtenues à partir de l'état initial q0 de la machine en fonctionnement normal :
$SM = \{s / s = \omega(i, q0) : i \in IM\}$. □

Soit i = i1*i2, où le symbole * exprime la concaténation, les séquences i1 et i2 constituent respectivement le préfixe et le suffixe de i. P(i) et S(i) représentent l'ensemble des préfixes et l'ensemble des suffixes de i. Soit Iq l'ensemble des séquences d'entrée qui peuvent être appliquées à partir de l'état q de la machine en fonctionnement normal : $Iq = \{i2 : i1.i2 \in IM \text{ et } \delta(i1, q0) = q\}$.

$Iq \supset Iq^{\infty}$ ou Iq^{∞} correspond aux séquences d'entrée de longueur non bornée.

Une machine séquentielle est définie par le sextuplet $M = (Q, X, Z, \delta, \omega, q0)$ où $\delta(i,q)$ et $\omega(i,q)$ sont respectivement l'état et la séquence de sortie qui sont obtenus à partir de l'état q de Q lorsqu'une séquence i est appliquée aux entrées de la machine en fonctionnement normal.
D'autre part, si une panne f d'un ensemble F apparaît lorsque la machine M se trouve dans l'état q alors $M^f = (Q^f, X, Z^f, \delta^f, \omega^f, q^f)$ se trouve immédiatement dans l'état q^f ce qui se traduit par : $\delta^f(i, q) = \delta^f(i, \delta^f(\lambda, q)) = \delta^f(i, q^f)$. Dans les mêmes conditions, $\omega^f(i, q)$ est la séquence de sortie obtenue à la place $\omega(i, q)$ si la séquence i est appliquée à la machine.

DEFINITION D9 :

Une machine M est "sequentially self-testing" ("séquentiellement autotestable") pour une panne f d'un ensemble F, un état q de Q et une séquence d'entrée i2 de Iq si et seulement si :
$\forall$ i1 tel que $\delta(i1, q0) = q$ et $i1.i2 \in IM$, $\omega(i1, q0) . \omega^f(i2, q) \notin SM$.

Par la suite, i2m correspond au plus petit suffixe de i2 pour lequel $\omega(i1, q0) . \omega^f(i2m, q) \notin SM$.

DEFINITION D10 :

Une machine M est "sequentially fault secure" ("séquentiellement sûre en présence de pannes) pour une panne f d'un ensemble F, un état q de Q et une séquence d'entrée i2 de Iq si et seulement si :

$\omega^f(i'2, q) = \omega(i'2, q)$ $\forall$ i'2 $\in$ P(i2m) avec i'2 $\neq$ i2m si i2m existe, $\forall$ i'2 $\in$ P(i2) autrement.

DEFINITION D11 :

Une machine M est "sequentially self-checking" ("séquentiellement autocontrôlable") pour un ensemble de pannes F et un mode de fonctionnement normal donné si et seulement si pour toute panne f1 de F, pour tout état q de Q et pour tout i2 de Iq^∞ :

soit a/ la machine est "sequentially self-testing " et "sequentially fault secure" pour (f1, q, i2).

ou b/ la machine est seulement "sequentially fault secure" pour (f1, q, i2) et pour toute occurrence de panne f2 $\in$ F, pour toute séquence i4 de S(i2) soit la condition a/ ou la condition b/ est vérifiée pour : f1 $\cup$ f2 à la place de f1, $\delta^f(i3,q^f)$ tel que i2 = i3.i4 à la place de q et i4 à la place de i2 (fig.6). □

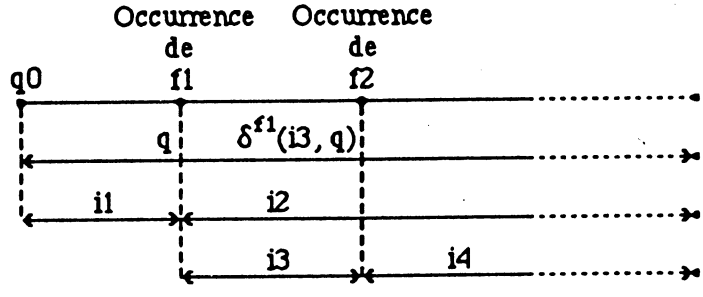


Fig III - 6 - Illustration de la définition D11

Ce qui signifie qu'à chaque fois qu'une panne fi de F apparaît et qu'elle n'est pas détectable, la machine garde la propriété "sequentially fault-secure" pour la panne composée f1 $\cup$ f2 $\cup$ ...$\cup$ fi de l'ensemble F. De plus si la machine devient "self-testing" alors la détection de cette panne composée est assurée avant l'occurrence d'une autre panne.

Plus précisément, pour chaque séquence d'entrée i2 la propriété "fault-secure" doit être vérifiée en présence d'une panne indétectable par cette séquence (ou plus généralement, en présence d'une séquence de pannes indétectables par la séquence i2) et ceci même si il existe d'autres séquences d'entrée qui détectent la panne (ou la séquence de pannes).Cette restriction est très forte et posera de grande difficultés à la construction des systèmes "sequentially self-checking".

Dans le cadre de cette étude, ce problème ne se pose pas étant donné que nous combinons la technique "self-checking" avec la technique BIST. Cette dernière permettant d'envoyer périodiquement aux entrées du circuit séquentiel des séquences d'entrée qui permettent de détecter des pannes. Ainsi les séquences i2 vont contenir des tranches d'une séquence prédéterminée qui permet la détection d'une première panne avant l'occurrence d'une seconde panne. Nous irons même plus loin puisque la technique BIST permet la controlabilité à la fois des entrées primaires et des entrées d'état du circuit séquentiel.. De cette façon, la propriété "self-testing" ne sera pas considérée vis à vis de séquences qui appartiennent à un langage d'entrée et qui sont appliquées seulement sur les entrées primaires de la machine, mais vis à vis d'un ensemble de vecteurs appliqués aux entrées primaires et aux entrées d'états du circuit. Ces mêmes techniques BIST permettront aussi d'assurer à la fois l'observabilité des vecteurs de sorties primaires et des vecteurs de sorties d'états. Ceci ramène la définition de la propriété "self-testing" à son expression la plus simple comme celle donnée dans le cas des circuits combinatoires.
En ce qui concerne la propriété " sequentially fault-secure", dans le cas général il faut assurer que toute sortie erronée de la machine n'appartient pas au langage de sortie du circuit. Ceci semble difficile et néanmoins il n'existe pas de technique connue qui permet d'assurer cette tâche pour un langage de sortie quelconque. De plus la vérification d'un langage de sortie nécessitera l'emploi de contrôleurs "strongly language disjoint" qui peuvent aboutir à des réalisations complexes. Pour cette raison, la détection en-ligne des erreurs sera basée non pas sur la non appartenance de séquences qui appartiennent à un langage de sortie mais sur la non appartenance des vecteurs de sorties primaires et des vecteurs de sorties d'états à un code indicateur d'erreur. Dans le cas où les vecteurs de sorties primaires et les vecteurs de sorties d'états sont contrôlés, l'effet de séquentialité disparaît vis à vis de la propriété "fault secure". Par contre, si seuls les vecteurs de sorties primaires sont vérifiés, il y a toujours un effet de séquentialité vis à vis de cette même propriété car les vecteurs de sorties d'états qui peuvent être erronés sont réinjectés dans la machine sans qu'aucune erreur ne soit signalée.
Ces simplifications importantes nous autorisent à adopter des définitions comme celle proposées dans [DAV 78], [NAN 87].

3 - CONCEPTION DE MACHINES SEQUENTIELLES AUTOTESTABLES

A la lumière des rappels et des précisions précédentes, nous abordons maintenant la réalisation de machines séquentielles qui accomplissent le "totally self-checking goal". Pour chaque proposition de conception que nous ferons, la propriété désirée sera démontrée en assimilant le comportement d'une machine séquentielle dans le temps à celui d'une chaîne de plusieurs sous-systèmes combinatoires.

3.1 Assimilation du comportement d'une machine séquentielle à celui d'un système de blocs combinatoires

La structure que nous allons considérer est celle de la description suivante. La machine séquentielle (fig.7) est constituée par un bloc fonctionnel combinatoire B et une fonction Δ (registre d'états) telle que $y(\Delta(t+1)) = \Delta(x(t)) = x(t)$ (où x et y correspondent respectivement au vecteur d'entrée et au vecteur de sortie de la fonction).

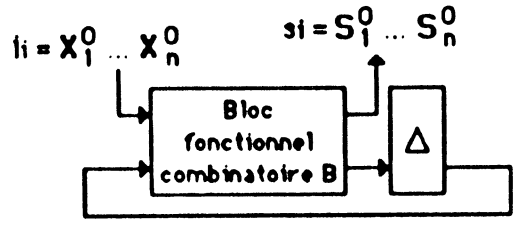


Fig III - 7 - Machine séquentielle

Nous définissons son langage d'entrée I (resp. langage de sortie S) comme l'ensemble des séquences i_i (resp. de séquence s_i) qui sont appliquées (resp. obtenues) en fonctionnement normal à partir de l'état initial Eo de la machine. Chaque séquence i_i (resp. s_i) qui peut se présenter contient une suite de vecteurs X_{i1}, ..., X_{in} (resp. S_{i1}, ..., S_{in}) qui appartiennent tous au code d'entrée (resp. au code de sortie) déterminé. De ce fait, la détection des défauts susceptibles d'apparaître dans l'automate est basée sur le codage vecteur par vecteur des entrées X_i, des états E_i et des sorties S_i de la machine considérée.

La conception "strongly fault secure" vis à vis d'une séquence d'entrée de cet automate peut être abordée en assimilant le comportement de la machine séquentielle considérée, dont le nombre d'états est supposé fini, à celui d'un système de blocs combinatoires (fig.8) comme le propose [BRE 76].

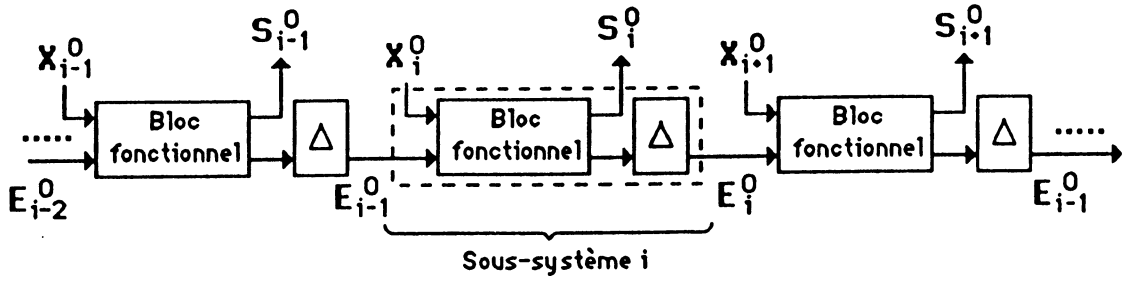


Fig III - 8 - Comportement d'une machine séquentielle assimilé à celui d'un système de blocs combinatoires

Chaque sous-système i (en pointillés sur le dessin) qui est constitué par un bloc fonctionnel Bi et une fonction Δi (où Bi ≡ Bi+1 et $\Delta i \equiv \Delta i + 1$, $\forall i \in \{1, .., n\}$) représente le comportement de la machine pendant la tranche de temps ti. Chaque fonction Δi est donc assimilée à une fonction combinatoire telle que $y = \Delta i\ (x) = x$. En fonctionnement normal, le sous-système i reçoit un vecteur d'entrée X_i^0 et un vecteur d'état E_{i-1}^0 qu'il transpose en un vecteur de sortie S_i^0 et en un vecteur "prochain état " E_i^0.

Pour une telle représentation fictive du système, l'occurrence d'une panne (notée *) dans l'automate pendant la tranche de temps ti correspond donc à la présence de cette panne dans le sous-système i, et dans tous les sous-systèmes suivants i+1, ...n. Les sorties primaires et les sorties d'état, de chaque élément affecté par le défaut, sont respectivement notées S_i^{e1} et E_i^{e1} pour le sous-système i, ..., $S_k^{e1..ek}$ et $E_k^{e1..ek}$ pour le sous-système k (ou $S_k^{e1..ek}$ et $E_k^{e1..ek}$ sont les résultats de la transposition, par le bloc fonctionnel défaillant k, des entrées primaires supposées correctes X_k^0 et des entrées éventuellement erronées $E_{k-1}^{e1..ek-1}$) (fig 9).

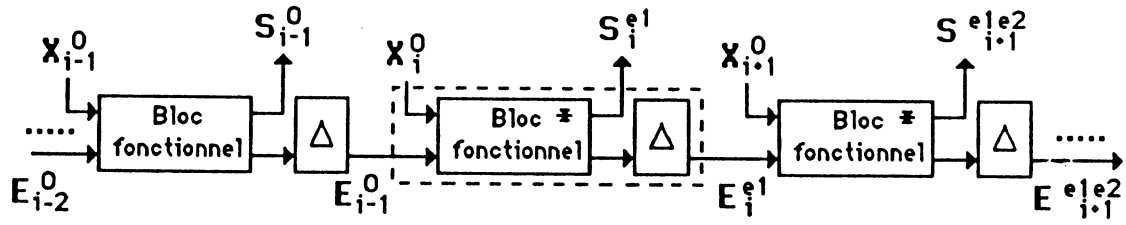


Fig III - 9 - Représentation de l'occurrence d'une panne dans une machine séquentielle dont le comportement est assimilé à celui d'un système de blocs combinatoires

En appliquant les remarques précédentes à propos de la représentation du comportement de la machine séquentielle (fig.8 et fig.9), nous abordons maintenant le problème de la conception d'automates "strongly fault-secure" .

3.2 Conception en blocs de machines séquentielles SFS

Par la suite nous considérons l'occurrence de pannes simples dans le circuit et nous nous limitons à l'étude des caractéristiques des blocs qui constituent un automate "strongly fault-secure" contrôlé vis à vis des erreurs simples (code détecteur : code de parité) et des erreurs unidirectionnelles (les codes détecteurs d'erreurs sont des codes non-ordonnés). Le cas des circuits dont les sorties sont contrôlées vis à vis d'erreurs multiples ne sera pas traité puisque leur conception fait appel à la méthode classique de duplication des blocs.

Les hypothèses de fonctionnement H1' et H2' données dans ce chapitre (§ 2.2.2) doivent toujours être respectées dans les différents cas de propositions que nous étudions.

PROPOSITION P1 :

Pour la conception en blocs d'une machine séquentielle SFS avec la structure présentée à la figure 10, la propriété "strongly fault-secure" est assurée si la machine se compose de :

- Un bloc fonctionnel SFS,
- Un registre d'états SFS/SCD,
- Un contrôleur interne SCD qui vérifie le code des états aux sorties du registre d'états,
- Un contrôleur SCD qui vérifie le code des sorties primaires de la machine. □

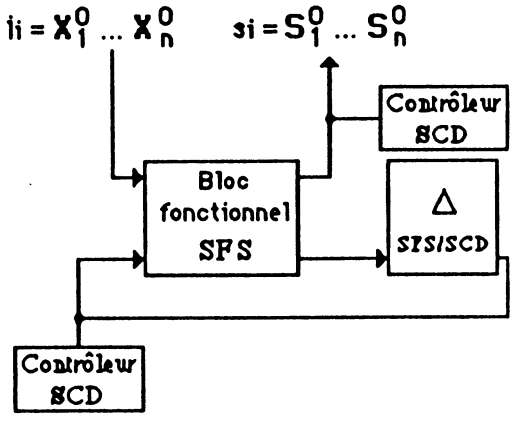


Fig III - 10 - Machine séquentielle SFS selon la proposition P1

Démonstration de la proposition P1

Dans cette proposition, et les suivantes, les entrées primaires du système (X_i^0, i={1, ..., n}) sont correctes par hypothèse (cette hypothèse sera validée par certaines précautions discutées par la suite). Pour simplifier les discussions, nous supposons que chaque fonction Δi (registre d'états) est SFS/SCD vis à vis des codes en vigueur.

Remarque : Le fait que la fonction Δi soit un registre ne pose pas de problème particulier pour une conception SFS et SCD du bloc.

Pour la démonstration de la propriété SFS du circuit séquentiel, conçu selon la proposition P1, nous assimilons le comportement de ce dernier à celui d'un système de blocs combinatoires comme le propose la figure 11. Chaque sous-système SFS (en pointillés sur la figure) est réalisé suivant le principe du paragraphe 6.2 du chapitre 2 en utilisant un bloc fonctionnel SFS et un registre d'états SFS/SCD.

Dans ce cas de figure, l'architecture est telle que les entrées d'état de chaque sous-système sont vérifiées. Un sous-système ne peut donc pas recevoir des entrées d'état erronées qui ne soient pas préalablement détectées par le contrôleur d'état placé en amont de celui-ci.

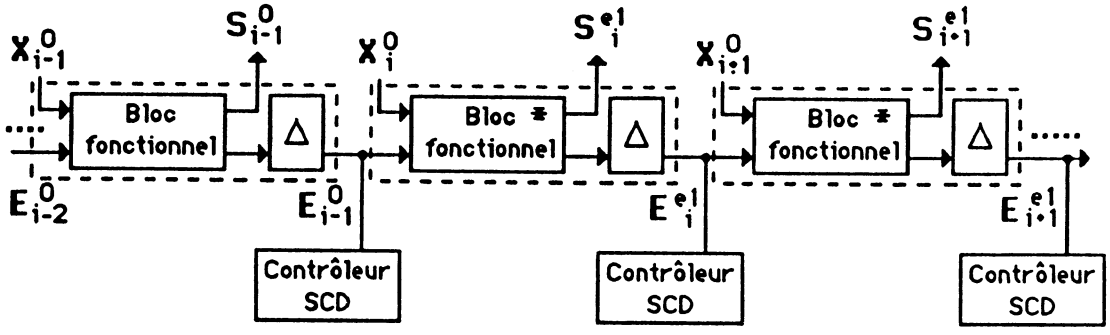


Fig III - 11 - Comportement d'une machine séquentielle, conçue selon la proposition P1, assimilé à celui d'un système de blocs combinatoires

De ce fait et de par la propriété SFS de chaque sous-système, l'occurrence d'une première panne (notée *) détectable dans le système se traduira par des erreurs sur les sorties d'au moins un sous-système k (sorties primaires $S_k^{e1} \neq S_k^0$ et/ou sorties d'état $E_k^{e1} \neq E_k^0$) de la chaîne défaillante. Ces erreurs seront signalées par le contrôleur des sorties primaires et/ou le contrôleur des sorties d'état qui sont associés au sous-système considéré.
Si la panne est indétectable, les sorties primaires et les sorties d'état de tous les sous-systèmes sont correctes et ces derniers restent FS. Le système global est SFS pour l'architecture adoptée.

PROPOSITION P2 :

Pour la conception en blocs d'une machine séquentielle SFS avec la structure présentée à la figure 12, la propriété "strongly fault-secure" est assurée si la machine se compose de :
- Un bloc fonctionnel SFS,
- Un registre d'états SFS/SCD,
- Un contrôleur SCD qui vérifie le code des sorties primaires de la machine,

et si les conditions C1' et C2' données par la suite sont vérifiées. □

Dans cette proposition de conception, contrairement à la proposition précédente, les entrées d'état du bloc fonctionnel ne sont pas vérifiées. C'est pourquoi certaines restrictions, que nous allons préciser par la suite, sont nécessaires pour assurer la propriété désirée.

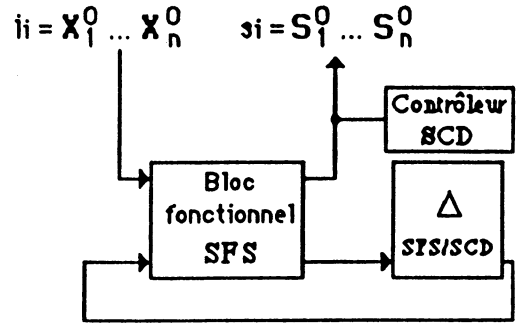


Fig III - 12 - Machine séquentielle SFS selon la proposition P2

Démonstration de la proposition P2

Pour la démonstration de la propriété SFS du circuit séquentiel conçu selon la proposition P2, le comportement de celui-ci peut être assimilé à celui d'un système de blocs combinatoires comme le propose la figure 13. Chaque sous système SFS (en pointillés sur la figure) est réalisé avec un bloc fonctionnel SFS et un registre d'états SFS/SCD selon le principe de la proposition 6.2 du chapitre précédent.

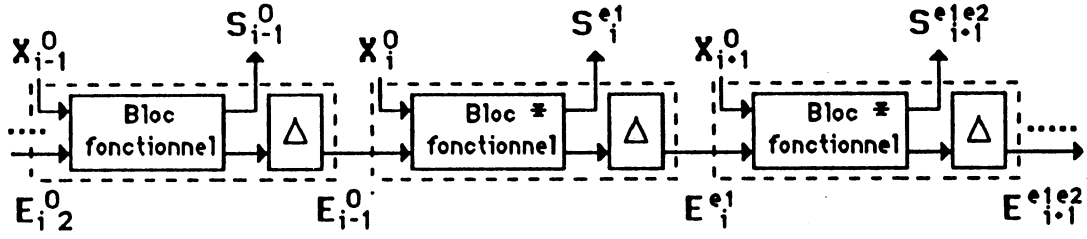


Fig III - 13 - Comportement d'une machine séquentielle, conçue selon la proposition P2, assimilé à celui d'un système de blocs combinatoires

Dans ce cas, comme les entrées d'état de chaque sous-système ne sont pas vérifiées, un sous-système peut être à la fois en panne et recevoir des entrées erronées (une panne identique est présente dans plusieurs blocs à la fois étant donné que le système proposé représente le comportement d'une machine séquentielle). Malgré cela, comme tous les sous-systèmes de la chaîne défaillante sont identiques et qu'ils sont affectés exactement par la même panne, la propriété SFS du système global peut être assurée si les conditions suivantes sont respectées :

CONDITION C1' **:**

Un sous-système qui fonctionne normalement doit transposer toute erreur unidirectionnelle sur ses entrées d'état en erreurs unidirectionnelles de même type sur ses sorties primaires et sur ses sorties d'état.

CONDITION C2' :

Toute occurrence de panne dans un sous-système provoque en sortie des erreurs symétriques (erreurs du type $0 \rightarrow 1$ ou $1 \rightarrow 0$, mais pas les deux à la fois). □

Dans la mesure où toutes les sorties primaires sont vérifiées par des contrôleurs SCD (qui ne sont pas représentés sur la figure 13), si une panne (notée *) apparaît dans l'ensemble des sous-systèmes consécutifs i, i+1, ..., n, alors :

a/ elle est signalée sur les sorties primaires du premier sous-système i défaillant ($S_i^{el} \neq S_i^0$) et la propriété "fault-secure" est assurée.

b/ sinon la panne n'est pas détectable au niveau de ce sous-système et ses sorties d'état sont soit correctes ($E_i^{el} = E_i^0$), soit erronées ($E_i^{el} \neq E_i^0$).

-1- Dans le premier cas ($E_i^{el} = E_i^0$), les entrées d'état du sous-système défaillant qui suit sont celles attendues, donc le sous-système (i+1) peut être considéré comme le premier sous-système de la chaîne défaillante et l'analyse donnée aux points a/ et b/ commence à partir de ce sous-système.

-2- Dans le second cas ($E_i^{el} \neq E_i^0$), pour faciliter la discussion nous considérons le système de la figure 14 où l'occurrence d'une panne dans l'automate ne concernerait que le sous-système i dans la représentation que nous avons adoptée.

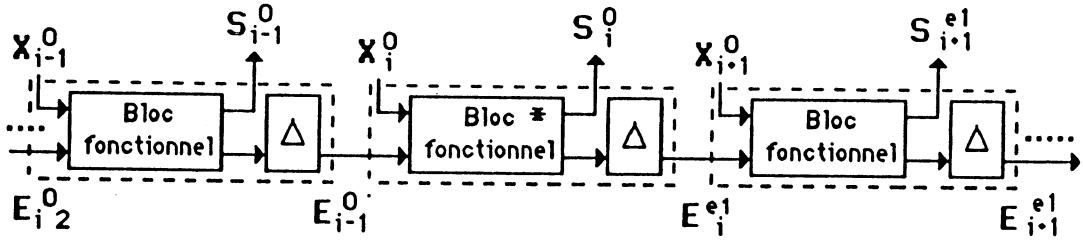


Fig III - 14 - Circuit fictif

Pour ce schéma, l'erreur qui n'apparaît pas sur les sorties primaires (puisque $S_i^{el} = S_i^0$) mais sur les sorties d'état du sous-système i est transposée, d'après la condition C1', en erreurs unidirectionnelles de même type sur les sorties primaires S_{i+1}^{el} et les sorties d'état E_{i+1}^{el} du sous-système suivant. De ce fait, si les erreurs sur E_i sont du type $0 \rightarrow 1$ (resp.$1 \rightarrow 0$), les erreurs sur S_{i+1} et E_{i+1} sont du type $0 \rightarrow 1$ (resp.$1 \rightarrow 0$), donc S_{i+1}^{el} et E_{i+1}^{el} comportent des erreurs du type $0 \rightarrow 1$ (resp.$1 \rightarrow 0$) par rapport à S_{i+1}^0 et E_{i+1}^0.

En réalité (fig.13), le sous-système (i+1) comporte comme son prédécesseur la même panne et cette dernière modifie alors S_{i+1}^{e1} et E_{i+1}^{e1} en S_{i+1}^{e1e2} et E_{i+1}^{e1e2}. Comme, de par la condition C2', une panne provoque toujours des erreurs symétriques sur les sorties du sous-système, c'est à dire qu'elle provoque toujours des erreurs du type $0 \rightarrow 1$ (resp.$1 \rightarrow 0$), S_{i+1}^{e1e2} et E_{i+1}^{e1e2} comportent par rapport à S_{i+1}^{e1} et E_{i+1}^{e1} des erreurs de même type que E_i^{e1} comporte par rapport à E_i^0. Nous en déduisons que S_{i+1}^{e1e2} et E_{i+1}^{e1e2} comportent par rapport à S_{i+1}^0 et E_{i+1}^0 des erreurs du type $0 \rightarrow 1$ (resp.$1 \rightarrow 0$). En conclusion les erreurs sur E_{i+1} sont unidirectionnelles.

Cette démarche peut être appliquée pour n'importe quel nombre de sous-systèmes défaillants successifs, ce qui signifie que les erreurs sur les sorties primaires et les sorties d'état sont toujours unidirectionnelles. Ainsi, la première apparition d'un vecteur erroné sur les sorties primaires d'un sous-système k est immédiatement détectée.

En ce qui concerne la propriété SFS, nous remarquons qu'une panne indétectable au niveau d'un sous-système (c'est à dire si nous observons à la fois les sorties primaires et les sorties d'état) sera indétectable au niveau du sous-système. Si une telle panne apparaît, les sorties primaires et les sorties d'état de chaque sous-système sont toujours correctes et les sous-systèmes restent SFS. Pour que le système global reste SFS, il faudra que les conditions C1' et C2' soient toujours valables. Si la panne est détectable au niveau sous-système mais qu'elle est indétectable au niveau système (toutes les sorties primaires Si, ...,Sn sont correctes), il faut que la propriété SFS de chaque sous-système reste vérifiée (ce qui n'est pas forcément garanti par la propriété SFS au niveau sous-système). Pour résoudre le problème, de telle pannes peuvent être éliminées en observant les sorties d'état pendant une phase de test hors-ligne (analyse de signature).

PROPOSITION P3 :

Une machine séquentielle conçue avec une structure en blocs, comme le montre la figure 15 est "strongly fault-secure" si elle est composée de :

- Un bloc fonctionnel SFS/SCD,
- Un registre d'états SFS/SCD,
- Un contrôleur SCD qui vérifie le code des sorties primaires de la machine,

et si les conditions C1' et C2' données précédemment sont vérifiées. □

Dans ce cas de conception, la différence avec la proposition P2 vient du fait que le bloc fonctionnel est non seulement SFS mais aussi SCD.

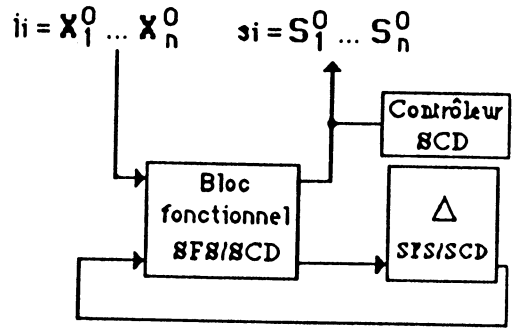


Fig III - 15 - Machine séquentielle SFS selon la proposition P3

Démonstration de la proposition P3

Pour la démonstration de la propriété SFS du circuit séquentiel conçu selon la proposition P3, le comportement de celui-ci peut être assimilé à celui d'un système de blocs combinatoires comme celui qui est décrit par la figure 16. Chaque sous système SFS/SCD (en pointillés sur la figure) est réalisé avec un bloc fonctionnel SFS/SCD et un registre d'états SFS/SCD selon la proposition du paragraphe 6.1 du chapitre 2.

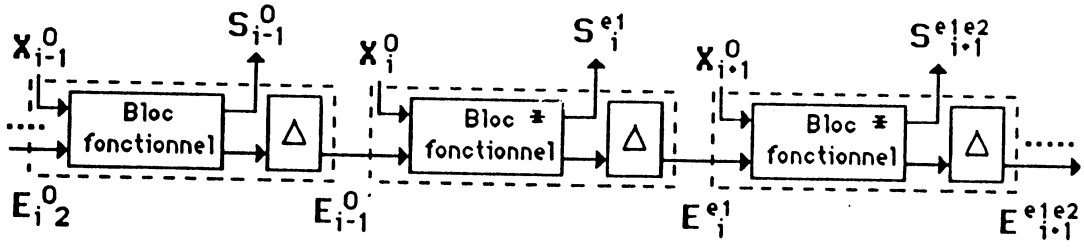


Fig III - 16 - Comportement d'une machine séquentielle, conçue selon la proposition P3, assimilé à celui d'un système de blocs combinatoires

Ici aussi, un sous-système peut être en panne et recevoir des entrées erronées. Comme dans le cas précédent, nous pouvons montrer que la propriété SFS est assurée pour le système global si chaque sous-système qui le compose satisfait aux conditions C1' et C2'. Cependant, comme les contraintes imposées pour la réalisation de tels sous-systèmes (propriété SFS/SCD combinée avec le respect des conditions C1' et C2') risquent d'être trop lourdes pour des applications pratiques, nous proposons une architecture un peu différente qui simplifiera la tâche du concepteur.

Comme nous le verrons dans le prochain paragraphe il est parfois très utile, pour des raisons de conception, de partager le bloc fonctionnel en deux parties : une fonction d'entrée et une partie fonctionnelle. La proposition qui suit permet dans ce cas d'obtenir une machine SFS sans se préoccuper des conditions C'1 et C'2 données précédemment.

PROPOSITION P4 :

Pour la conception en blocs d'une machine séquentielle SFS avec la structure présentée à la figure 17, la propriété "strongly fault-secure" est assurée si la machine se compose de :

- Une fonction d'entrée SFS/SCD
- Un bloc fonctionnel SFS,
- Un registre d'états SFS/SCD,
- Un contrôleur interne SCD qui vérifie le code d'entrée et de sortie de la fonction d'entrée,
- Un contrôleur SCD qui vérifie le code des sorties primaires de la machine. □

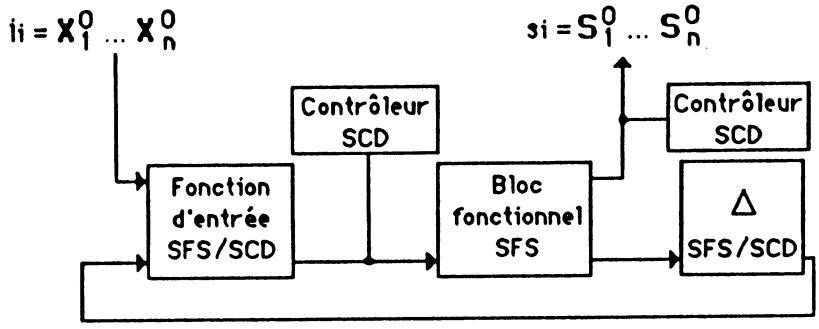


Fig III - 17 - Machine séquentielle SFS selon la proposition P4

Démonstration de la proposition P4

Le circuit séquentiel conçu selon la proposition P4 a un comportement assimilé à celui du système de blocs combinatoires donné par la figure 18. Chaque sous-système fonctionnel combinatoire est constitué d'une fonction d'entrée SFS/SCD et d'un bloc SFS (en pointillés sur la figure) qui est conçu selon le principe du paragraphe 6.1 du chapitre 2.

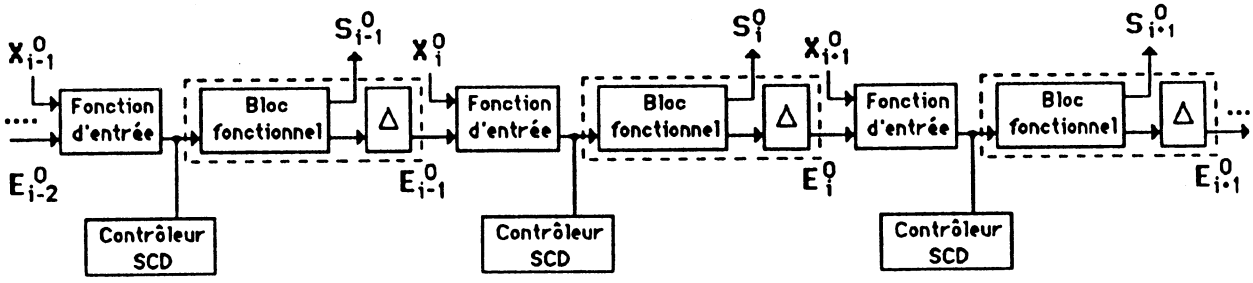


Fig III - 18 - Comportement d'une machine séquentielle, conçue selon la proposition P4, assimilé à celui d'un système de blocs combinatoires

L'architecture du système est telle qu'aucun bloc SFS ne peut recevoir d'entrées erronées qui ne soient pas signalées par le contrôleur SCD placé en amont de celui-ci. De ce fait, si une première panne (notée * en fig.19) apparaît dans les blocs SFS i, i+1, ..., n consécutifs, elle se traduira par

des erreurs sur les sorties primaires S_k^{e1} et/ou sur les sorties d'état E_k^{e1} d'au moins un sous-système k de la chaîne défaillante.des erreurs sur les sorties primaires S_k^{e1} et/ou sur les sorties d'état E_k^{e1} d'au moins un sous-système k de la chaîne défaillante.

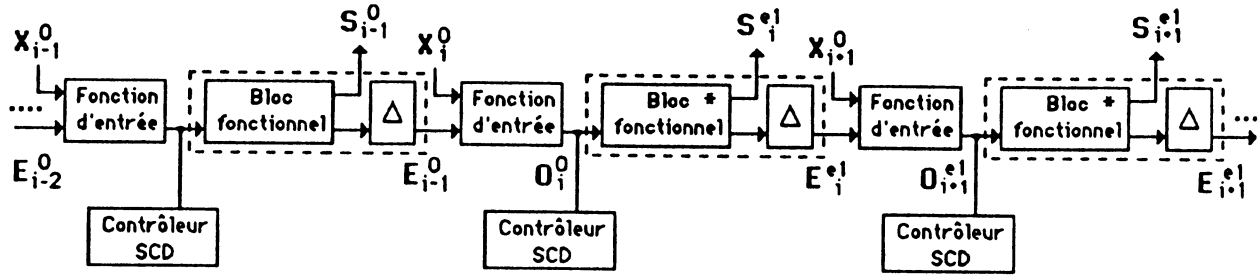


Fig III - 19 - Occurrence d'une panne dans la machine séquentielle, dont le comportement est assimilé à celui d'un système de blocs combinatoires

Si $S_k^{e1} \neq S_k^0$, c'est le contrôleur SCD des sorties primaires du sous-système k (ils ne sont pas représentés sur cette dernière figure) qui signale l'erreur. Si $E_k^{e1} \neq E_k^0$, de par la propriété SFS/SCD des fonctions d'entrée de chaque sous-système k, cela se traduit par un vecteur de sortie $O_{k+1}^{e1} \neq O_{k+1}^0$ qui est mis en évidence par le contrôleur SCD interne du sous-système (k+1). Si la panne est indétectable, les sorties primaires et les sorties d'état de tous les blocs (en pointillés sur le dessin) sont correctes et ces derniers restent FS. Si l'occurrence d'une première panne détectable se produit dans les fonctions d'entrée SFS/SCD des sous-systèmes i, i+1, ..., n, au moins un vecteur de sortie $O_k^{e'1}$ sera erroné et il sera signalé par le contrôleur SCD interne du sous-système k. Sinon toute les sorties O_i^0 sont correctes et chaque fonction d'entrée reste FS.
L'architecture proposée assure donc la propriété "strongly fault-secure" du système global.

Le tableau suivant résume des différentes propositions de conception en blocs d'automates SFS.

Proposition	Prop.P1	Prop.P2	Prop.P3	Prop.P4
Conception en blocs	.Bloc fonct. SFS .Reg. des états SFS/SCD .Contrôle des états .Contrôle des sorties	.Bloc fonct. SFS .Reg. des états SFS/SCD .Conditions C1' et C2' pour la propriété SFS de la machine .Contrôle des sorties	.Bloc fonct. SFS et SCD .Reg. des états SFS/SCD .Conditions C1' et C2' pour la propriété SFS de la machine .Contrôle des sorties	.Fonct. d'entrée SFS .Bloc fonct. SFS .Reg. des états SFS/SCD .Contrôle des entrées du bloc fonct. .Contrôle des sorties

Fig III - 20 - Tableau récapitulatif

3.3 Conception de l'automate à partir de structures régulières

Les différentes architectures proposées au paragraphe précédent sont basées sur la conception de blocs combinatoires qui reçoivent des vecteurs appartenant ou non à un code d'entrée et qui produisent des vecteurs appartenant ou non à des codes déterminés sur les sorties primaires et les sorties d'état. Ces blocs combinatoires ont des propriétés SFS, SFS/SCD, ou SFS avec certaines contraintes supplémentaires. Maintenant nous examinons la conception de ces blocs à partir de PLAs ou de ROMs [MAK 82], [FUC 84], [NIC 84].

Dans [NIC 84] sont présentées des règles et des hypothèses de pannes pour la conception de ces structures régulières SFS dans les cas où des erreurs simples et unidirectionnelles sont considérées en sortie. L'analyse est basée sur des hypothèses de pannes de bas niveau et leur construction est détaillée dans cette référence. Nous nous inspirons ici des conclusions de cette étude.

3.3.1 Conception du bloc fonctionnel combinatoire avec des PLAs

Dans le cas d'un PLA implanté avec un code détectant des erreurs simples (code de parité), ce dernier doit être divisé en trois blocs distincts (entrées primaires, monômes, sorties primaires) où chacun est vérifié directement par un contrôleur de parité. L'insertion de lignes supplémentaires pour éliminer certains courts-circuits possibles qui provoqueraient des erreurs unidirectionnelles en sortie et l'ajout de circuits, pour la génération des bits additionnels utilisés dans le codage de parité des monômes et des sorties, font que le PLA "augmenté" est SFS vis à vis des erreurs simples provoquées par des pannes de la classe I des hypothèses de pannes données au chapitre 2 excepté les coupures de lignes d'alimentation.

Si le code utilisé est un code détectant des erreurs unidirectionnelles, en raison de l'organisation des PLAs le circuit est testé par la méthode de contrôle des entrées primaires et un code de sortie non-ordonné. Les trois groupes de lignes (entrées primaires, monômes, sorties primaires) sont implantés avec des matériaux non court-circuitables et les sorties additionnelles utilisées pour le codage des données sont générées par le PLA lui-même. Ainsi, un PLA conçu selon la procédure décrite rapidement ci-dessus est SFS pour des erreurs unidirectionnelles provoquées par des pannes de la classe I des hypothèses de pannes.

L'obtention d'un PLA SFS/SCD nécessite deux conditions supplémentaires à celles données pour concevoir un PLA SFS pour des codes détectant des erreurs simples ou des erreurs unidirectionnelles :

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 / les fautes ou les séquences de fautes qui seraient détectables uniquement avec l'application de vecteurs hors-code ne peuvent pas se produire (cette condition peut demander des changements de conception du circuit initial).

Si un PLA est conçu selon l'une des deux conditions précédentes et qu'il est programmé pour qu'en recevant des entrées dans le code il produise des sorties dans le code, et qu'en recevant des entrées hors-code il produise des sorties hors-code, et où les fautes ou les séquences de fautes qui ne pourraient être détectées que par des vecteurs hors-code sont éliminées (par des changements topologiques), c'est un PLA SFS/SCD pour des erreurs simples ou unidirectionnelles.

3.3.2 Conception du bloc fonctionnel combinatoire avec des ROMs

Un premier schéma d'une mémoire ROM "strongly fault-secure" est donné en figure 21.

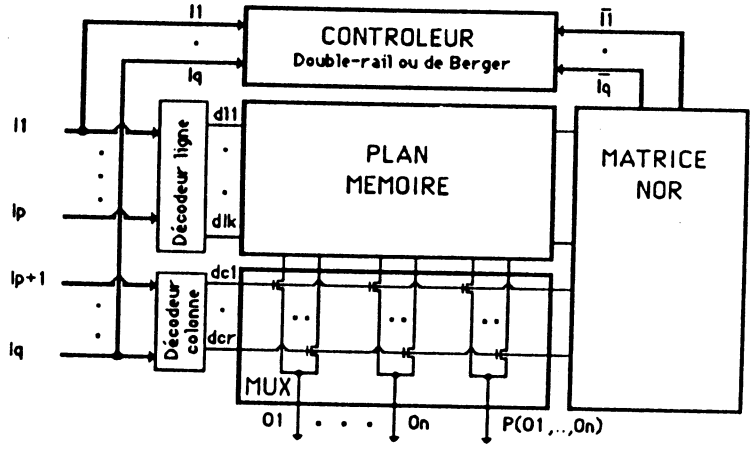


Fig III - 21 - Topologie d'une ROM "strongly fault-secure" [NIC 88 c]

Les entrées primaires (I1, .., Ip) du décodeur ligne et (Ip+1, .., Iq) du décodeur colonne sont codées à l'aide d'un code non-ordonné (code m-parmi-n, code de Berger, code double-rail,...) et sont transformées respectivement en vecteurs (dl1, .., dlk) et (dc1, .., dcr) codés avec le code 1-parmi-k et 1-parmi-r (une seule ligne dli (resp. dci) est activée parmi les k (resp. les r) existantes). A chaque couple de vecteurs obtenu en sortie des décodeurs [(dl1,..,dlk),(dc1,..,dcr)] correspond un mot mémoire qui est codé avec le code de parité.

En bout de ligne, après avoir traversé le plan mémoire ou le multiplexeur, chacun des bits du couple de vecteurs considéré auparavant est destiné à l'entrée d'une matrice NOR qui régénère les bits complémentaires (¬I1, ..,¬Iq).
La cohérence du code du mot mémoire devra être vérifiée par un contrôleur de parité, alors que le contrôleur double-rail (contrôleur de Berger, ...) compare les bits régénérés (¬I1, ..,¬Iq) avec les bits des entrées primaires (I1, .., Iq) et s'assure ainsi du bon fonctionnement des deux décodeurs.
Pour ce cas de figure, l'architecture adoptée et les codes utilisés permettent d'obtenir une ROM "strongly fault-secure" pour un ensemble de pannes simples (ligne coupée, transistor s-on, transistor s-open, ...) susceptibles de se produire dans les décodeurs, le plan mémoire, le multiplexeur et la matrice NOR [NIC 88 c]. Dans cette même référence, l'auteur propose une conception en vue du test unifié qui donne la propriété SFS/SCD au circuit vis à vis du même ensemble de pannes.

Le schéma donné en figure 22 est aussi celui d'une ROM "strongly fault-secure" pour les mêmes pannes considérées ci-dessus mais pour deux ensembles de vecteurs d'entrée codés différemment et pour des vecteurs de sortie codés avec un code non-ordonné.

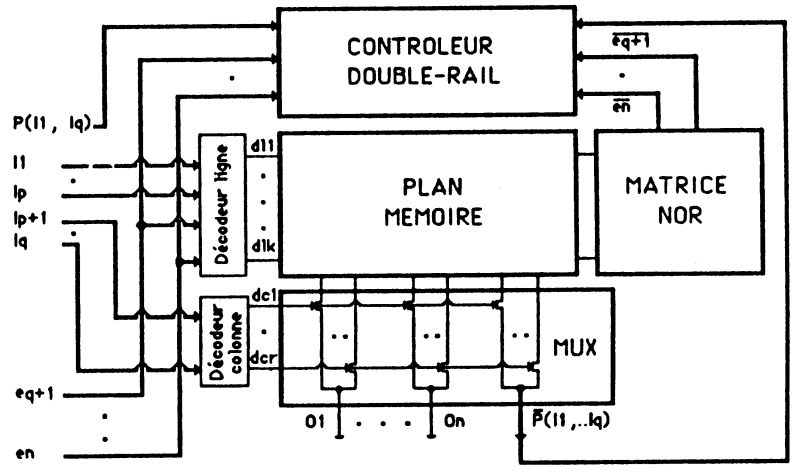


Fig III - 22 - Topologie d'une ROM "strongly fault-secure" [FUC 84]

Pour ce circuit, nous supposons que les p premiers bits (I1, .., Ip) du décodeur ligne et les bits (Ip+1, .., Iq) du décodeur colonne proviennent d'une système A qui délivre des informations codées avec le code de parité P(I1, .., Iq). Alors que les (n-q) derniers bits du décodeur ligne (eq+1, .., en) proviennent d'un système B et sont codés avec un code non-ordonné (code m-parmi-n, code de Berger, ...). Comme dans le schéma précédent, les sorties des deux décodeurs (dl1,.., dlk) et (dc1, ..,dcr) sont codées respectivement avec un code 1-parmi-k et 1-parmi-r en fonction des entrées primaires de la ROM. A chaque couple de vecteurs obtenu correspond un mot mémoire qui est codé avec un code non-ordonné.

En bout de ligne et après avoir traversé le plan mémoire, seuls les bits (dl1, .., dlk) sont destinés aux entrées d'une matrice NOR qui régénère les bits complémentaires (¬e1, ..,¬et) des bits correspondants aux entrées primaires codées avec un code non-ordonné.
La cohérence du code du mot mémoire devra être vérifiée par un contrôleur adéquat (contrôleur double-rail ou contrôleur de Berger, ...). Un contrôleur double-rail compare, d'une part, la parité P(I1, ..,Iq) des entrées primaires codées à l'aide du code de parité avec la parité complémentaire ¬P(I1, ..,Iq) régénérée en sortie de la ROM, d'autre part, les bits (e1, .., et) des entrées primaires codés avec un code non-ordonné avec les bits (¬e1, ..,¬et) régénérés par la matrice NOR.
Ce schéma est à rapprocher de celui proposé en **[FUC 84]**. Alors, la conception adoptée et les codes utilisés permettent d'obtenir une ROM "strongly fault-secure". Cette propriété est vérifiée de la même façon que dans le cas précédent pour un ensemble de pannes simples susceptibles de se produire dans les différents décodeurs, le plan mémoire, le multiplexeur et la matrice NOR.

Remarques : la matrice NOR qui sert à régénérer l'ensemble des bits complémentaires (¬bit i) dans les deux figures précédentes fait partie en réalité du plan mémoire (et du multiplexeur pour le peremier des deux dessin). La séparation en blocs distincts n'est effectuée que pour permettre une meilleure compréhension de la technique employée pour la vérification de certaines entrées ou sorties du circuit.
Le code de parité est employé pour coder les sorties de la mémoire ROM de la figure 21, il n'y a donc aucun inconvénient à ce qu'un code non-ordonné soit utilisé pour coder les mêmes sorties et la propriété "strongly fault-secure" est conservée.

3.3.3 Conception de la logique de mémorisation

La logique de mémorisation est un registre de n cellules où n est le nombre de bits utilisés pour la codification des états. Cet élément n'effectue aucune opération logique, il reçoit l'information du bloc combinatoire et au moment où le signal d'horloge est activé il transfert simplement les valeurs d'entrée sur ses sorties. Si le vecteur d'entrée n'est pas correct le registre propage l'erreur, si le registre comporte un défaut, le vecteur de sortie sera celui attendu ou alors il sera hors-code. Dans **[JAN 85]**, les conditions à respecter sont données pour qu'une telle logique soit SFS/SCD vis à vis des entrées et des sorties pour un ensemble de pannes déterminées.

4 - EXEMPLE : CAHIER DES CHARGES D'UN AUTOMATE SPECIFIQUE

En application de ce qui a été développé dans les paragraphes précédents et sans perdre de vue le but fixé en début de ce chapitre, nous nous sommes donnés comme objectif le réalisation d'une architecture cible d'automates destinés au traitement de fonctions de bas niveau (actionneurs, sémaphores, ...) de systèmes critiques.

Ce circuit a été défini comme un automate autotestable dans lequel il est possible d'intégrer plusieurs centaines d'états (complexité moyenne) et pour lequel la sûreté de fonctionnement est une des conditions principales de son existence.

4.1 Applications envisagées

Les applications envisagées sont toutes les applications qui nécessitent de déterminer les défaillances d'un automate de façon sûre et compatibles avec les caractéristiques techniques suivantes.

4.2 Caractéristiques techniques

L'automate est composé de deux blocs fonctionnels (fig.23), l'organe de séquencement et l'interface de commande (ou fonction de sortie), pour des raisons qui seront données ultérieurement.

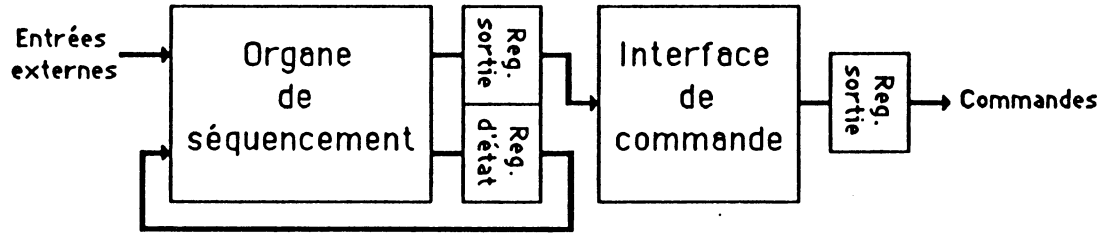


Fig III - 23 - Structure générale de l'automate

Chacun des deux blocs est constitué d'une mémoire ROM dont la surface dépend du nombre total de mots mémoire qu'elle contient et du nombre de bits utilisés pour ces mots :

- la ROM de l'organe de séquencement contient tous les états de l'automate et des informations qui sont fonction de l'état courant de la machine séquentielle et des conditions (entrées externes) appliquées.

- la ROM de l'interface de commande ne sert qu'à transformer les informations délivrées par l'organe de séquencement en commandes qui peuvent être destinées à de multiples systèmes indépendants.

Le nombre d'entrées primaires (conditions) et de sorties primaires (commandes), qui sont toutes données sous forme de vecteurs binaires codés, est limité par le nombre de broches utilisables sur un boîtier.

Suivant la complexité des fonctions à intégrer dans l'automate, le nombre des états peut varier de deux états à un millier d'états.

Remarque : les dernières caractéristiques, entrées/sorties primaires et états de l'automate, devront être données sous forme symbolique par une table des phases ou par un graphe (fig.24). Dans le premier cas, l'évolution du système est décrite par un tableau où l'on note pour chaque combinaison d'état interne et de condition, le nouvel état interne et les sorties correspondantes. Dans le second cas, le comportement du système est décrit sous forme géométrique.

	Entrées où Conditions		
Etats Internes	C1	C2	C3
A	A/S1		B/S6
B		A/S4	C/S5
C	B/S3	C/S2	A/S5

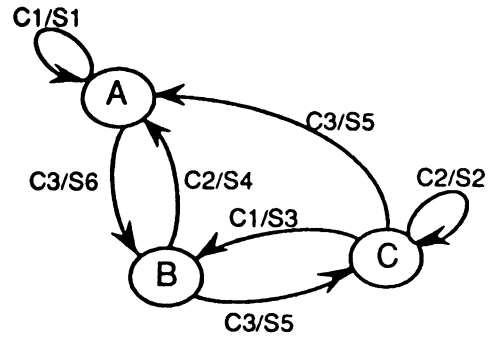


Fig III - 24 - Table des phases et graphe d'un automate

4.3 Justification de la solution retenue

La conception de l'automate de référence à l'aide de ROMs se justifie par le fait que l'architecture de cette structure régulière ne se trouve pas complètement modifiée de la réalisation d'une fonction à une autre (seul l'ajout de quelques lignes peut suffire dans de nombreux cas). De ce fait, deux automates quelconques, qui seront réalisés d'après notre modèle et utilisés pour des missions différentes, auront la même architecture et occuperont une surface pratiquement identique.
Les autres raisons de ce choix sont dues à ce que la méthode de conception d'une ROM autotestable ne pose pas de problème et que son comportement est bien connu puisqu'il a été étudié pour la réalisation de la partie contrôle du 68000 autotestable [NIC 88 c].
Par contre, la division de l'automate en deux parties distinctes est vraiment intéressante lorsque la fonction à intégrer nécessite quelques milliers d'états et si les commandes sont destinées à un nombre élevé de systèmes indépendants placés en aval. Dans ce cas, la surface du circuit est très importante et ses performances sont diminuées. C'est pour ces raisons et pour avoir un modèle le plus ouvert possible que nous avons décidés de concevoir l'automate en deux parties : Le premier élément où l'organe de séquencement est un automate de Mealy (nombre d'états réduit) qui contient tous les états de la machine codés avec un code non-ordonné. Dans ce bloc sont également stockées

les informations destinées à l'interface de commande qui sont codées, avec un type de code identique au précédent, sur un nombre de bits restreint de manière à ce que la largeur de la ROM reste convenable dans les pires cas de conception (très grand nombre d'états, ...). Quant à la hauteur de cette même ROM qui est proportionnelle au nombre d'états ou au nombre d'informations qu'elle contient, elle n'est pas maîtrisable. Le second élément sert d'interface de commande. C'est un circuit combinatoire (fonction de sortie) qui transpose les informations du bloc précédent, délivrées sur un petit nombre de bits, en commandes externes qui peuvent être envoyées à plusieurs systèmes indépendants. Ainsi, nous faisons en sorte que la largeur comme la hauteur de la ROM de ce second bloc reste convenable dans tous les cas de fonctions à intégrer.
Ce modèle d'automate convient parfaitement à nos besoins. Il est de conception simple et permet d'intégrer un grand nombre de fonctions de complexité moyenne sans bouleverser l'architecture de base. De plus, en adoptant le schéma proposé, les coûts de réalisation de n'importe quelle application demandée ne doivent pas être élevés, ce qui ne serait pas le cas si nous nous étions basés sur une architecture de type "partie contrôle d'un microprocesseur" **[NIC 88 c]**.
A partir de la disposition donnée en figure 21 et du modèle architectural adopté (conception des blocs fonctionnels à base de ROMs), nous proposons la conception d'un automate autotestable (en-ligne et hors-ligne) dont les entrées et les sorties primaires sont codées au moyen du code de parité et dont les données internes (état courant, sorties de l'organe de séquencement) sont codées avec un code non-ordonné du type m-parmi-2m.

5 - CONCEPTION D'UN AUTOMATE AUTOTESTABLE

5.1 Schéma global du circuit

Le schéma de l'automate est donné en figure 25.

L'organe de séquencement de la machine comprend :
- la fonction prochain état,
- la fonction "donneur d'ordres", dont les données sont à destination de l'interface de commande,
- et la logique de mémorisation.

Cet organe de séquencement est une machine du modèle de Mealy réalisé à partir d'une ROM.

Quant à l'interface de commande qui est un simple circuit combinatoire réalisée aussi à partir d'une ROM, elle constitue la fonction de sortie de la machine.

Nous supposons que l'horloge est verifiée à part par un contrôleur "totally self-checking" comme le celui proposé en **[USA 75]**.

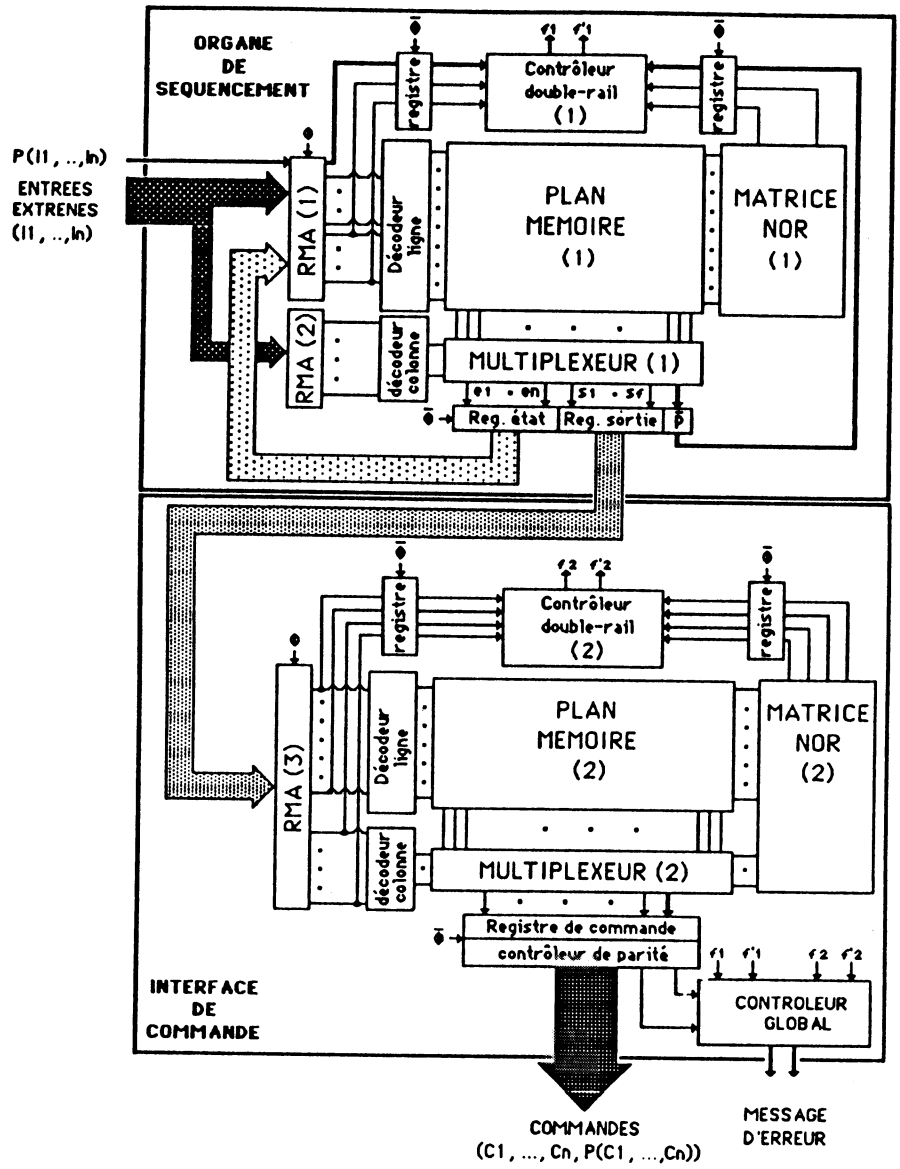


Fig III - 25 - Architecture de l'automate "self-checking"

5.2 Fonctionnement normal

Les entrées externes (I1, ...,In) de l'organe de séquencement sont supposées provenir d'un système amont (automate identique à celui décrit, ensemble de capteurs, ...) qui délivre des informations codées avec le code de parité P(I1, ...,In).

En fonction de ces entrées qui sont supposées correctes et de l'état courant de la machine, l'organe de séquencement fournit trois données distinctes. Le nouvel état courant (e1, ..., en) qui est codé avec un code non-ordonné du type m-parmi-2m, une information ou ordre (s1, ..., sf) destinée à l'interface de commande et qui est codée de la même manière que l'état courant, enfin le bit complémentaire du bit de parité des entrées externes.

L'interface de commande reçoit de l'organe de séquencement l'ordre (s1,...,sf) délivré sur un nombre de bits restreint et le transpose en une commande globalement codée avec le code de parité (C1, ..., Cn, P(C1, ...,Cn)) qui est destinée à un ou de multiples systèmes indépendants placés en aval (ensemble d'automates identiques à celui décrit, interface pour commande d'actionneurs ...). Un contrôleur global reçoit tous les messages des différents contrôleurs du système et délivre une information qui sera exploitée de différentes manières suivant les spécifications de sûreté imposées.

5.2.1 Processus de contrôle en ligne

L'organe de séquencement est comparable au circuit donné par la figure 22. En effet, une partie des entrées externes est destinée au registre des micro-adresses RMA (1) du plan mémoire de la ROM considérée alors que les entrées externes qui restent sont chargées dans le registre des micro-adresses RMA (2) de multiplexeur de cette même ROM. Le registre RMA (1) reçoit aussi les bits d'état courant de l'automate. La propriété "strongly fault-secure" est assuré en vérifiant :

- les entrées externes et l'état courant (fig.26) avec le contrôleur double-rail (1) (fig. 25),

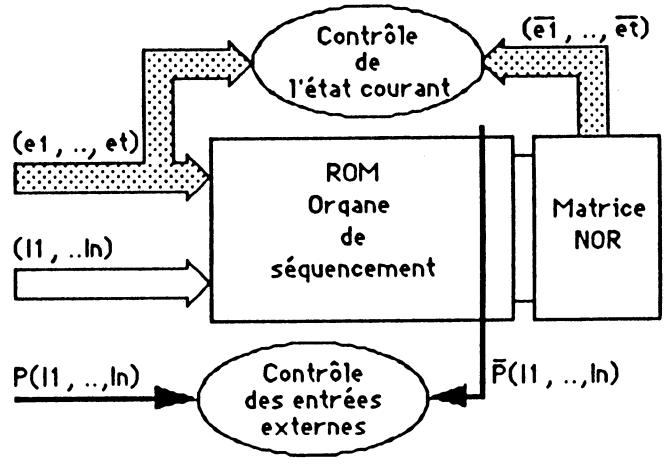


Fig III - 26 - Contrôle des entrées de l'organe de séquencement

Ceci en comparant respectivement les bits P et ¬P et les données (e1, ..., en) et (¬e1, ..., ¬en) où (¬e1, ..., ¬en) est la donnée régénérée par la matrice NOR(1).

- la cohérence du code de "l'ordre" (s1, ..., sf) (fig.27) avec le contrôleur double-rail (2) (fig.25).

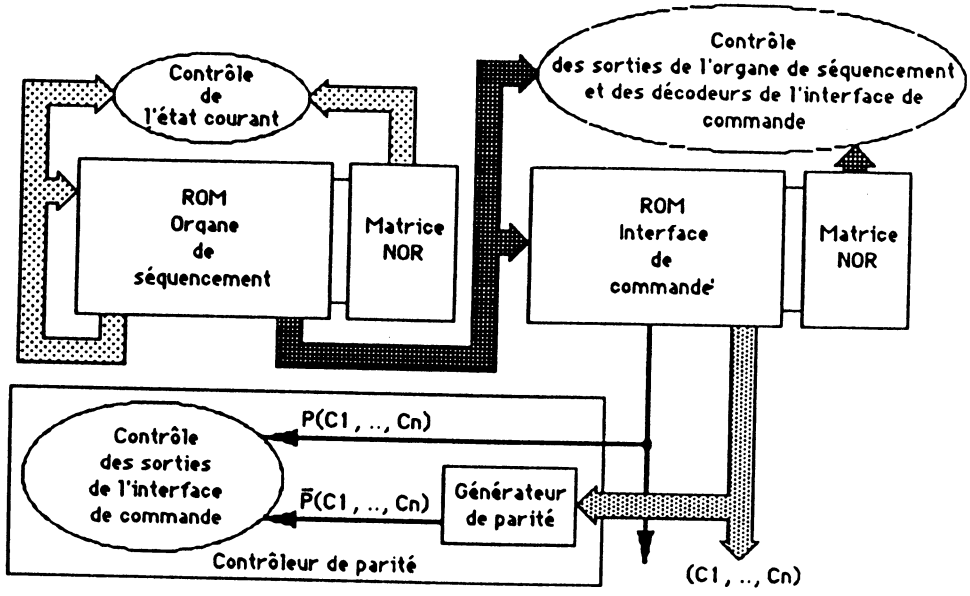


Fig III - 27 - Contrôle des sorties de l'organe de séquencement, des décodeurs et des sorties de l'interface de commande

En ce qui concerne l'interface de commande, son organisation et son comportement sont assimilables a ceux du circuit de la figure 21. La propriété "strongly fault-secure" de ce bloc est assurée par le contrôleur double-rail (2) (vérification des entrées et du bon fonctionnement des décodeurs) et par le contrôleur de parité (contrôle des sorties - fig.27).

Le contrôle des interconnexions entre les deux éléments précédents est assuré par le contrôleur double-rail (2)

Le circuit global obtenu peut être représenté par la figure 28 :

- La fonction d'entrée (1) correspond aux décodeurs et à la matrice NOR de régénération de l'organe de séquencement.

- Le bloc SFS (en pointillés sur le dessin de la figure 28) est constitué par le plan mémoire et le multiplexeur de ce même organe.

Quant à l'interface de commande, elle est aussi partagée en :

- Une fonction d'entrée (2) qui est réalisée par les décodeurs et la matrice NOR de régénération de la seconde ROM utilisée,

et en,

- Un bloc fonctionnel, ou fonction de sortie, correspondant au plan mémoire et au multiplexeur de cette même ROM.

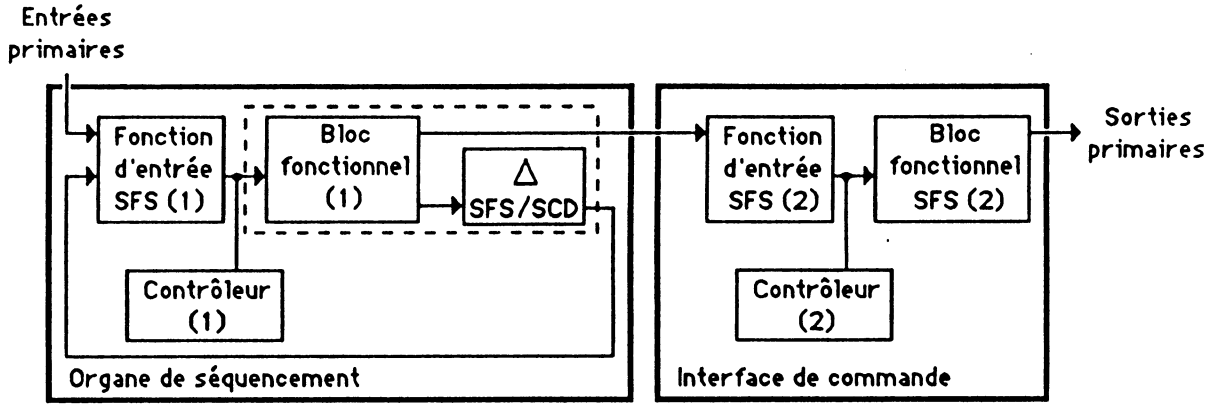


Fig III - 28 - Conception de l'automate en blocs combinatoires

Dans la mesure où le contrôleur (1) est SCD et la fonction d'entrée (1) est SFS et SCD (ceci est possible avec une conception UBIST **[NIC 88 c]** que nous proposerons par la suite), l'organe de séquencement peut être identifié à la machine séquentielle SFS donnée par la proposition 4 du paragraphe 3 (cf : § 3.3). Si la fonction d'entrée (2) de l'interface de commande est aussi SFS et SCD et que le contrôleur (2) est SCD (au moyen d'une conception UBIST), alors le bloc fonctionnel SFS (2), ne peut recevoir que des données correctes qui proviennent du bloc précédent car tout vecteur hors-code est signalé (par le contrôleur (2)).
En conclusion, l'automate qui est conçu selon le schéma proposé ci-dessus est une machine séquentielle "strongly fault-secure" d'après la définition D6 donnée au paragraphe 2 (cf : § 2.2) capable de réaliser le "totally self-checking goal" pour l'ensemble des pannes simples qui peuvent apparaître dans les structures régulières utilisées. Evidemment pour accomplir un tel but, les hypothèses H1' et H2' du paragraphe 2 (cf : § 2.2) doivent être respectées.

Dans la réalité, suivant le besoin et la mission qu'il devra accomplir, l'automate ne recevra pas forcément, en fonctionnement normal, tous les vecteurs utiles pour le test en-ligne complet. C'est pourquoi il est nécessaire d'envisager une conception en vue du test unifié (UBIST) pour assurer la propriété SCD des fonctions d'entrée (1) et (2) et des contrôleurs (1) et (2) de l'automate. Dans ce cas, les séquences de test hors-ligne seront appliquées avec une période suffisamment petite par rapport au MTBF du système.

5.3 Conception en vue du test unifié (UBIST)

Dans ce qui suit nous proposons une architecture et les moyens mis en oeuvre pour obtenir les fonctionnalités désirées. La conception UBIST s'inspire fortement de [NIC 88 c].

5.3.1 Architecture UBIST : justification du choix

Le montage proposé en figure 29 n'est certes pas le plus astucieux ni le plus économique.

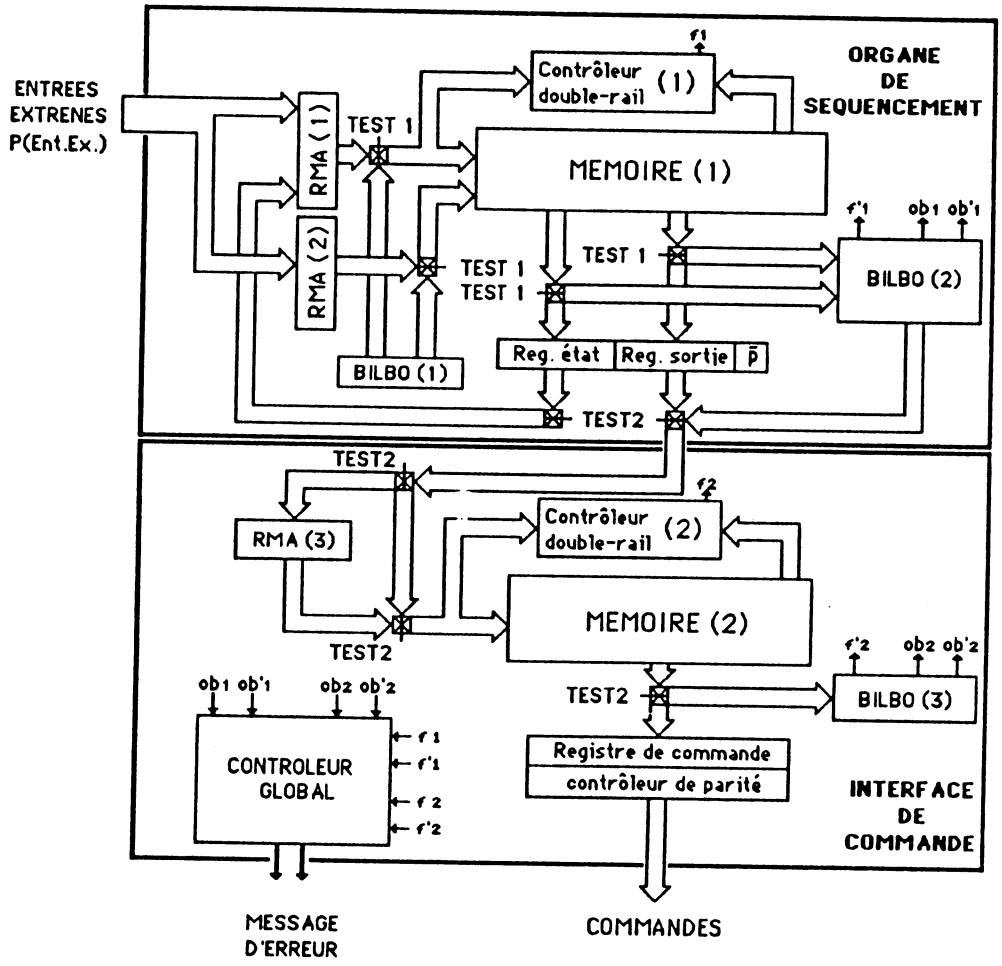


Fig III - 29 - Conception UBIST de l'automate

Cependant, quel que soit l'automate à concevoir (nombre d'états, d'entrées et de sorties, ...) ce schéma est assez général pour servir de référence (architecture "générique"). Si les données présentes en certains points du système avant la phase de test hors-ligne n'avaient pas à être mémorisées, les registres déjà en place pourraient être utilisés pendant cette période de test. Il suffirait de concevoir ces derniers pour qu'ils puissent être reconfigurés et accomplir diverses missions pendant la durée de vie du système.

Toutefois comme nous avons besoin de mémoriser l'ensemble des données présentes dans le circuit, nous avons choisi d'utiliser du matériel supplémentaire (générateur de vecteurs de test, analyseur de signature, ...) qui soit employé uniquement pendant la phase de test hors-ligne et qui soit isolé du reste du système pendant le mode normal de fonctionnement.

5.3.2 Les différentes phases du mode de test hors-ligne

Périodiquement, le fonctionnement normal ("TEST1="0", "TEST2"="0") de l'automate doit être interrompu, plusieurs données sont mémorisées (entrées externes, ordre, état courant et commande) et une phase de test hors-ligne est lancée. Cette dernière se divise en deux périodes distinctes qui peuvent se succéder ou qui sont appliquées à deux instants différents. Chacune d'elles peut aussi, et pour diverses raisons (impossibilité d'interrompre pendant un long moment le mode normal de fonctionnement ...), être décomposées en plusieurs intervalles de temps.
Une période ("TEST1"="1", "TEST2"="0") est consacrée uniquement au contrôle et à l'observation de l'organe de séquencement. L'autre période ("TEST1"="0", "TEST2"="1") concerne seulement le test de l'interface de commande.

5.3.3 Les dispositifs spécifiques au test hors-ligne de l'automate

Comme le montre la figure 29, le test hors-ligne de cet automate nécessite trois registres BILBO (Built In Logic Bloc Observer) qui sont destinés à injecter des séquences de vecteurs aux différents blocs sous test et/ou à compacter les réponses de ces mêmes blocs (générateur de vecteurs de test pour les BILBO (1) et BILBO (2), analyseur de signature pour les BILBO (2) et BILBO (3)). Des dispositifs supplémentaires sont utilisés pour permettre la lecture du contenu des analyseurs de signature et les contrôleurs double-rail sont conçus de manière à ce que cette lecture puisse être prise en compte correctement et en temps voulu par le contrôleur global du système. Les schémas présentés par la suite donnent quelques précisions qui sont détaillées [NIC 88 c]

La séquence de test nécessaire à la mémoire ROM (1) et au contrôleur double-rail (1) est délivrée par le BILBO (1) (fig. 30). Ce dernier contient un générateur de vecteurs de test (fig.31) qui génère les 2^n vecteurs (A1, ..., An), où pour chacun d'eux les k premiers bits (A1, ..., Ak) sont destinés à la partie du décodeur ligne qui reçoit, en mode normal de fonctionnement, des vecteurs codés avec le code non-ordonné m-parmi-2m, alors que les (n-k) bits suivants (Ak+1, ..., An) sont codés avec le code de parité et sont destinés à l'autre partie de décodeur ligne et au décodeur colonne. Le fait que le vecteur (A1, ..., Ak) appartient ou non au code m-parmi-2m est indiqué par un compteur S(1) (vecteur hors-code : S(1) = "0"; vecteur du code S(1) = "1") de manière à ce qu'il n'y ait pas de message d'erreur délivré par le contrôleur global alors que le système testé ne comporte pas de panne.

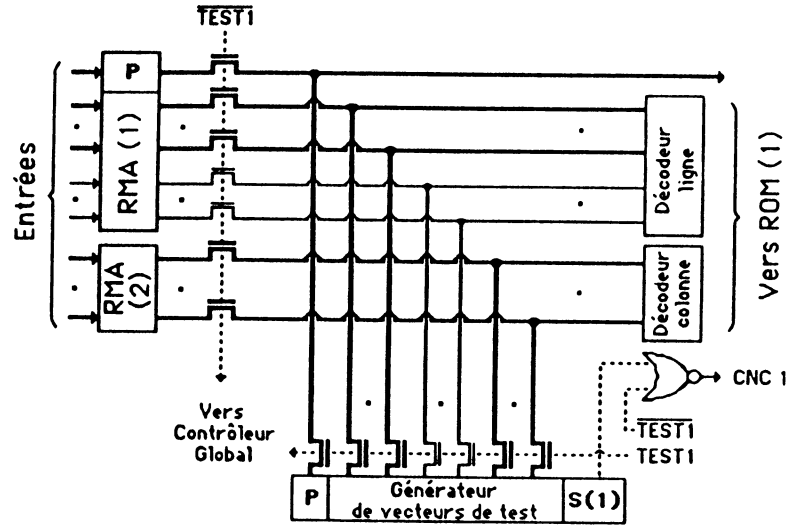


Fig III - 30 - Contrôlabilité de l'organe de séquencement

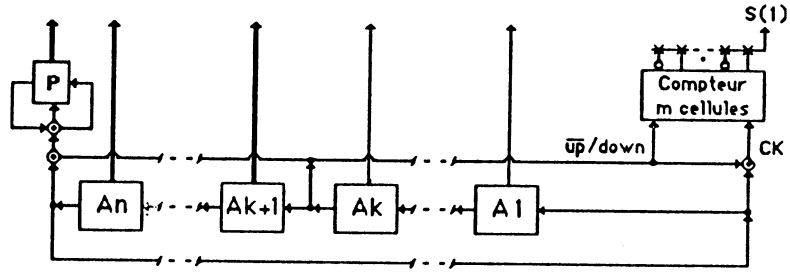


Fig III - 31 - Générateur de vecteurs de test spécifique (BILBO (1))

D'autre part, il faut ajouter un dispositif supplémentaire qui indique si les vecteurs de test hors-ligne qui sont injectés et qui appartiennent au code m-parmi-2m sont, ou non ("don't care conditions"), décodés par les décodeurs aux entrées de la ROM de l'organe de séquencement. Une méthode de conception de ce dispositif est proposé en [NIC 88 c] . Le message que ce dernier dispositif délivre est regroupé avec celui qui indique si le vecteur appartient, ou non, au code donné pour aboutir à un message final (f'1) destiné au contrôleur global .

La séquence de vecteurs de test injectée pendant cette même période ("TEST1" = "1") de test hors-ligne sert aussi à vérifier le bon fonctionnement du contrôleur double-rail (1). La séquence sera donc élaborée pour mettre en évidence tout les défauts non détectable qui peuvent se produire dans l'organe de séquencement pendant la phase de fonctionnement normal.

Ces défauts seront signalés soit pendant la lecture de l'analyseur de signature (BILBO (2) fig.32) à la fin de la compaction des réponses des blocs sous test, soit immédiatement par le contrôleur double-rail (1) pendant le déroulement de la séquence de test hors-ligne considérée.

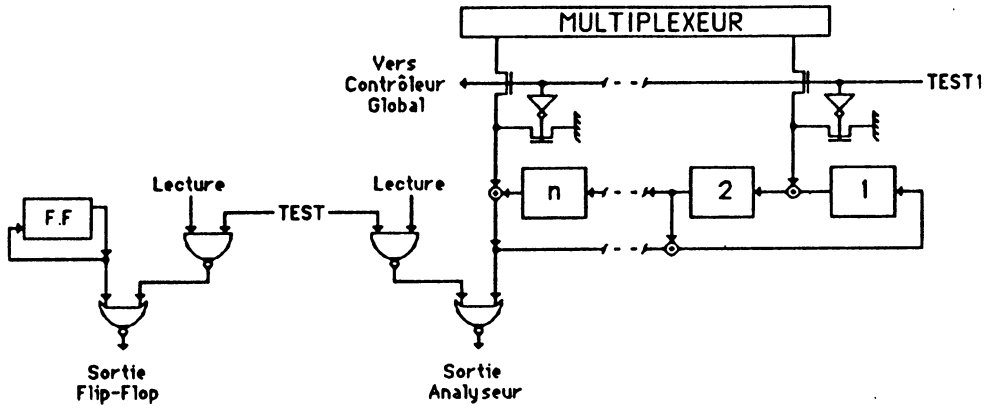


Fig III - 32 - Analyseur de signature (BILBO (2))

Pour le contrôle de l'interface de commande ("TEST 2"="1"), le registre BILBO (2) qui est utilisé comme analyseur de signature pendant la période de test hors-ligne de l'organe de séquencement est transformé en générateur de vecteurs de test (fig.33).

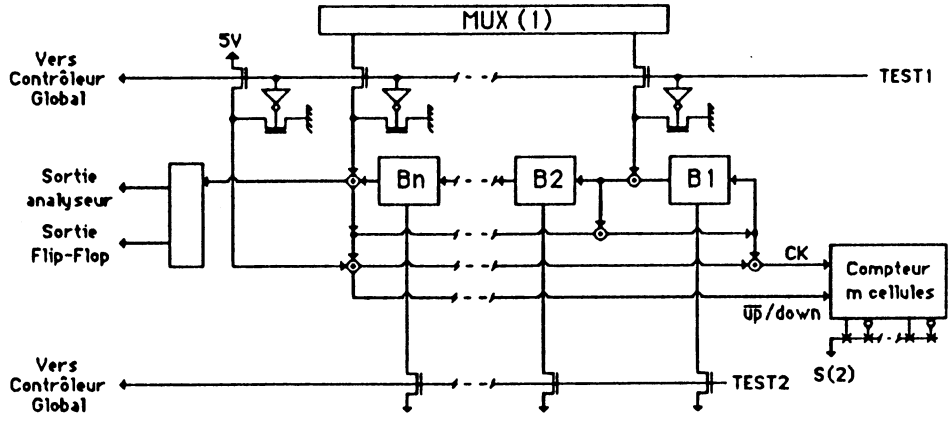


Fig III - 33 - BILBO multi-fonctions (BILBO (2))

Comme dans la cas précédent, ce sont 2^n vecteurs (B1, .., Bn) qui sont utilisés pour le contrôle de la ROM (2). Ici aussi, un dispositif est nécessaire pour signaler si le vecteur injecté appartient ou non au code m-parmi-2m, et s'il peut ou non être décodé par le décodeur de la ROM (2) sous test. Ceci de manière à toujours avoir un message cohérent (f2, f'2) délivré au contrôleur global.

Les diverses réponses de la partie sous test sont compactées dans l'analyseur de signature BILBO (3) prévu à cet effet (fig.34) et toute panne permanente dans l'interface de commande non détectée pendant la phase de test en-ligne doit être signalée soit directement par le contrôleur double-rail (2) soit, un peu plus tard, pendant la lecture du contenu de l'analyseur.

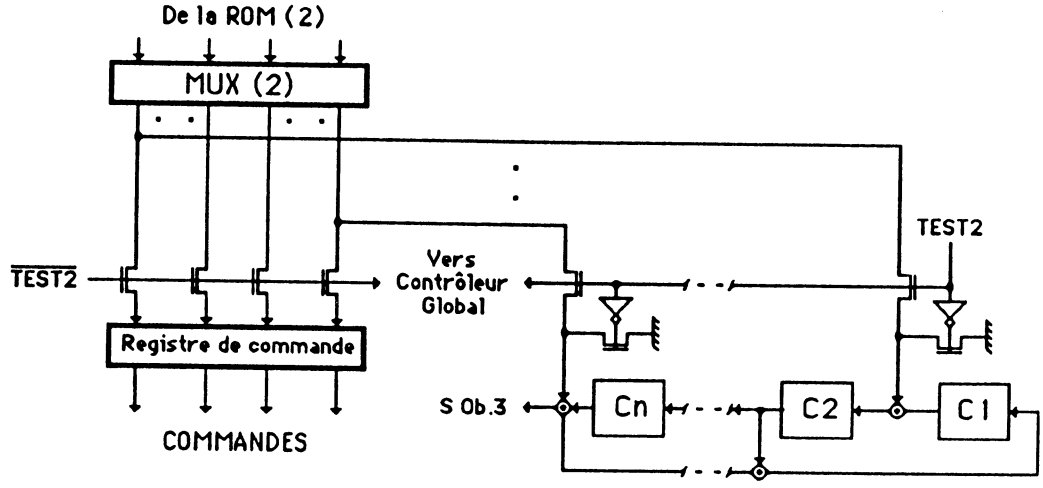


Fig III - 34 - Observabilité de l'interface de commande

6 - CONCLUSION

La raison d'être de ce chapitre a été de proposer une architecture cible pour des automates destinés au contrôle/commande de fonctions de bas niveau de systèmes critiques. La solution développée consiste à réaliser une machine séquentielle de référence à l'aide de mémoires ROMs (pour ne pas bouleverser profondément la conception de l'ensemble entre deux applications de même type) en utilisant les atouts qu'offrent les techniques de conception "self-checking" et BIST pour assurer la sûreté de fonctionnement de l'ensemble.

De par ses spécifications, ce type de circuit pourrait parfaitement jouer le rôle de fonction réflexe dans un système de "détection incendie" ou diverses missions de ce type. Mais alors une autre question se pose quant à l'exploitation du message d'erreur délivré par le contrôleur global de l'automate. De plus, telle qu'elle est présentée, la machine séquentielle dont les sorties sont des vecteurs binaires codés n'est pas adaptée à la commande de systèmes comme les actionneurs qui sont généralement les éléments employés en bout de chaîne des ensembles qui accomplissent les missions qui nous intéressent.

Pour satisfaire pleinement les conditions qui font que cet automate pourrait effectivement être utilisé dans des systèmes de surveillance classiques (systèmes à haute sûreté de fonctionnement) il faut le modifier de manière à ce que ses sorties soient compatibles avec les entrées de systèmes électromécaniques sans pour autant que le système réalisé perde ces caractéristiques relatives à la sécurité. La solution à envisager fera l'objet du chapitre suivant.

CHAPITRE IV

IV - LES SYSTEMES "FAIL-SAFE"

IV - LES SYSTEMES "FAIL-SAFE"

1 - INTRODUCTION

Les nombreux systèmes critiques qui sont developpés pour les transports ferroviaires, aériens, pour l'industrie chimique ou nucléaire sont tous réalisés suivant le même schéma de principe. Ce dernier se décompose en trois parties, une logique de traitement des données, un étage de sortie et des fonctions sécuritaires de commande qui utilisent la technique de codage en fréquence (alarmes, actionneurs, sémaphores, ...).
Les gros problèmes liés à la sécurité (au sens de la non-occurrence de défaillances catastrophiques) se trouvent généralement localisés au niveau de l'étage de sortie. Différentes solutions sont proposées pour résoudre au mieux ces problèmes, mais toutes sont réalisées à partir de la définition des systèmes "fail safe" conventionnels donné par [MIN 67] et [TAK 7I] . Malheureusement, tels qu'ils sont définis ces systèmes ne peuvent pas être intégrés et de ce fait ils posent d'énormes difficultés de conception lorsque l'étage de sortie à réaliser est complexe. Dans ce chapitre, nous proposons une méthode originale qui permet de palier ces inconvénients. Cette méthode propose d'employer de systèmes autotestables ("self-testing" et "self checking") associés à une interface à panne non dangereuse ("fail safe") pour intégrer non seulement la logique de traitement mais aussi l'étage de sortie et obtenir un système global "fail-safe".

Pour ce faire, nous commençons par reprendre la définition des circuits "fail-safe" proposée en [MIN 67] de manière à la rendre homogène aux énoncés donnés pour les propriétés des circuits "self-checking". Puis la théorie généralisée des circuits "fail-safe" est développée. Les relations entre ces circuits et les circuits "self-checking" [DIA 74], [DIA 79] sont rediscutées et d'importantes propriétés concernant les systèmes "fail-safe" sont alors établies. Ensuite, nous introduisons le concept des circuits "strongly fail-safe" capables d'atteindre le "totally fail-safe goal". Finalement, nous proposons la conception d'une interface "fail-safe" qui transforme les sorties des circuits "self-checking" en signaux qui autorisent le pilotage d'éléments qui utilisent la technique de codage en fréquence. Ainsi l'ensemble, circuit "self-checking" et interface, susceptible d'être implanté en VLSI est "strongly fail-safe" et il peut réaliser la logique de traitement et l'étage de sortie d'un système destiné à la commande d'un processus critique complexe.

2 - DEFINITION DE BASE

Les systèmes "fail-safe" [WAT 66], [MIN 67], [TAK 71] ont été conçus pour que toute sortie erronée, qui est le résultat d'une panne interne, ne donne pas lieu à un événement catastrophique. La définition formelle des ces systèmes a été proposée par *Mine et Koga* [MIN 67] pour des fonctions à une sortie primaire. Puis, *Takaota et al* [TAK 71] ont introduit le concept des circuits

"N-fail-safe", dont la sortie primaire est dupliquée, pour distinguer une valeur "sûre" correcte d'une valeur "sûre" qui est le résultat d'une panne dans le système.

Comme les notations utilisées dans les définitions précédentes sont différentes de celles considérées dans la théorie des circuits "self-checking", pour des raisons d'homogénéité nous pouvons énoncer la propriété des circuits "fail-safe" comme suit :

DEFINITION Dø :

Un circuit G, à une seule sortie primaire, est "0 fail-safe" (resp. "1 fail-safe") pour un ensemble de vecteurs d'entrée X et pour un ensemble de pannes F si :

$\forall x \in X, \forall f \in F : G(x, f) = G(x, \emptyset)$ ou $G(x, f) =$ "0" (resp. "1").

$G(x,\emptyset)$ correspond à la fonction correcte et $G(x, f)$ à la fonction défaillante.

Cette définition est similaire à celle donnée en **[MIN 67]** excepté le fait qu'aucune restriction n'est prévue en ce qui concerne l'ensemble des pannes F.

3 - GENERALISATION DU CONCEPT DES CIRCUITS "FAIL-SAFE"

En considérant la définition des systèmes "fail-safe" **[MIN 67]** qui ne comportent qu'une seule sortie primaire avec deux états possibles, **[TOH 71]** et **[TAK 72]** réalisent des systèmes "fail-safe" aux sorties multiples composés de plusieurs systèmes "fail-safe" à sortie unique. Toutefois, la notion d'état sûr en sortie d'un circuit peut être considérée vis à vis d'un groupe de sorties sur lesquelles plusieurs vecteurs sont possibles. Dans ce cas, l'ensemble des vecteurs de sortie noté Y est divisé en deux sous-ensembles Os et On. L'ensemble Os des vecteurs de sortie d'état sûr est composé des vecteurs qui ne peuvent pas entraîner de situation dangereuse même si leur apparition est involontaire (effet d'une panne à l'intérieur du circuit). L'ensemble On des vecteurs de sortie d'état non sûr, regroupe tous les autres vecteurs de Y. Ces derniers peuvent donner lieu soit à une situation catastrophique s'ils sont générés par erreur à la place de vecteurs déterminés, soit ils sont sans conséquence s'ils sont délivrés, dans les mêmes conditions, à la place d'autres vecteurs. Nous avons $On = Y - Os$ et $On \cap Os = \emptyset$.

Bien qu'elle soit importante, cette répartition de l'ensemble des vecteurs de sortie n'est pas suffisante pour caractériser parfaitement la situation engendrée par un système défaillant. En effet, s'il est en panne ce dernier peut générer :

- 1 - un vecteur de sortie d'état sûr à la place d'un autre vecteur de sortie d'état sûr,
- 2 - un vecteur de sortie d'état sûr à la place d'un vecteur de sortie d'état non sûr,
- 3 - un vecteur de sortie d'état non sûr à la place d'un vecteur de sortie d'état sûr,
- 4 - un vecteur de sortie d'état non sûr à la place d'un autre vecteur de sortie d'état non sûr.

Les deux premières situations sont non dangereuses, mais pour être en mesure de classer les situations obtenues dans les deux derniers cas, il est nécessaire de connaître la nature du changement entre le vecteur erroné et le vecteur normalement attendu.

Cette notion est précisée en définissant d'une part Os_{sxn} , l'ensemble des couples sûrs des vecteurs de sortie d'état sûr et des vecteurs de sortie d'état non sûr tel que :

Os_{sxn} = {[a,b]/[a,b] $\in$ Os x On : si b est généré par erreur à la place de a alors la situation résultante n'est pas dangereuse},

et d'autre part Os_{nxn} , l'ensemble des couples sûrs des vecteurs de sortie d'état non sûr tel que :

Os_{nxn} = {[a,b]/[a,b] $\in$ On x On : si b est généré par erreur à la place de a alors la situation résultante n'est pas dangereuse} (avec [a,a] $\in$ On x On $\subset$ Os_{nxn}).

Dans ces deux définitions, pour le couple [a,b] le premier vecteur (a) est celui qui est normalement attendu, le second vecteur (b) est celui qui est obtenu.

Maintenant l'ensemble des couples sûrs de vecteurs de sortie d'un circuit G est défini comme l'ensemble Os_{yxy} tel que :

$Os_{yxy} = Os_{nxn} \cup Os_{sxn} \cup ((On \cup Os) x\ Os)$ (puisque qu'un élément de Os est toujours un vecteur d'état sûr indépendamment du vecteur à la place duquel il est généré).

Alors la définition la plus générale des systèmes "fail-safe" est établie :

DEFINITION G :

Un système G est "fail-safe" de type G pour un ensemble de pannes F, un ensemble de vecteurs d'entrée X et un ensemble de couples sûrs de vecteurs de sortie Os_{yxy} si :

$\forall\ x \in X\ ,\ \forall\ f \in F$: $G(x, f) = G(x, \emptyset)$ ou $[G(x, \emptyset), G(x, f)] \in Os_{yxy}$.

Etant donné que l'existence d'un ensemble Os_{sxn} non vide impose de lourdes contraintes à la conception d'un système "fail-safe" [NIC 89], nous nous intéressons par la suite à une définition moins générale mais plus utilisable pratiquement où $Os_{sxn} = \emptyset$.

DEFINITION D1 :

Un système G est "fail-safe" de type D1 pour un ensemble de pannes F, un ensemble de vecteurs d'entrée X, un ensemble Os de vecteurs de sortie d'état sûr et un ensemble Os_{nxn} de couples sûrs de vecteurs de sortie d'état non sûr si :

$\forall a \in X, \forall f \in F$: $G(x, f) = G(x,\emptyset)$,

ou $G(x, f) \in Os$,

ou $[G(x, \emptyset), G(x, f)] \in Os_{nxn}$. □

Cette définition peut encore être simplifiée compte tenu que pour la plupart des applications pratiques le système doit être conçu de telle sorte qu'un vecteur de sortie d'état non sûr n'est jamais généré à la place d'un autre vecteur de sortie d'état non sûr. Alors $Os_{nxn} = \emptyset$ et la définition D1 se réduit à la forme suivante :

DEFINITION D2 :

Un système G est "fail-safe" de type D2 pour un ensemble de pannes F, un ensemble de vecteurs d'entrée X et un ensemble Os de vecteurs de sortie d'état sûr si :

$\forall a \in X, \forall f \in F$: $G(x, f) = G(x, \emptyset)$,

ou $G(x, f) \in Os$.

A ce stade notons qu'un circuit "fault-secure" qui est conçu de façon à ce que l'occurrence de vecteurs de sortie erronés ne provoque pas de situation dangereuse est considéré comme un circuit "fail-safe" de type D2. En effet, soit Y l'ensemble des vecteurs de sortie d'un circuit G "fault-secure" et soit $B \subset Y$ l'ensemble des vecteurs de sortie de G qui appartiennent à un code donné. Cet ensemble B peut être partagé en un sous-ensemble Bs de vecteurs de sortie codés et sûrs et en un sous-ensemble Bn de vecteurs de sortie codés et non sûrs. Si le concepteur réalise le système "fault-secure" de manière à ce que tout vecteur qui appartienne à l'ensemble Y-B ne provoque pas d'action dangereuse, l'ensemble des vecteurs de sortie sûrs $Os = Bs \cup (Y- B)$ et l'ensemble des vecteurs non sûrs On = Bn. Comme G est "fault-secure", on a [NIC 84] :

Pour un ensemble de pannes F, pour un ensemble de vecteurs d'entrée X et un ensemble de vecteurs de sortie Y dont l'ensemble B est l'ensemble des vecteurs codés suivant un code donné

$\forall a \in X, \forall f \in F$: $G(x, f) = G(x, \emptyset)$,

ou $G(x, f) \in (Y-B)$ où $(Y-B) \subseteq Os$.

Alors que les circuits "fault-secure" génèrent des vecteurs de sortie codés globalement, la notion d'état sûr et d'état non sûr est définie pour chacune des sorties binaires des systèmes "fail-safe" considérés par [MIN 67], [TAK 71], [TOK 71]. Il est aussi évident que les systèmes "fail-safe" définis dans [MIN 67] vérifient la définition D2. Pour finir, les machines séquentielles "fail-safe" qui sont décrites par [TOH 71] produisent des sorties erronées qui sont forcément en dehors du code de sortie déterminé et de ce fait elles sont classées comme des machines "fault-secure".
En conclusion, nous pouvons prétendre que les circuits "fault-secure" définis en [AND 71] et les circuits "fail-safe" donnés en [MIN 67], [TAK 71] et [TOK 71] forment deux classes disjointes qui sont toutes les deux incluses dans la classe plus générale des circuits "fail-safe " obtenue d'après la définition D2 (fig.1).

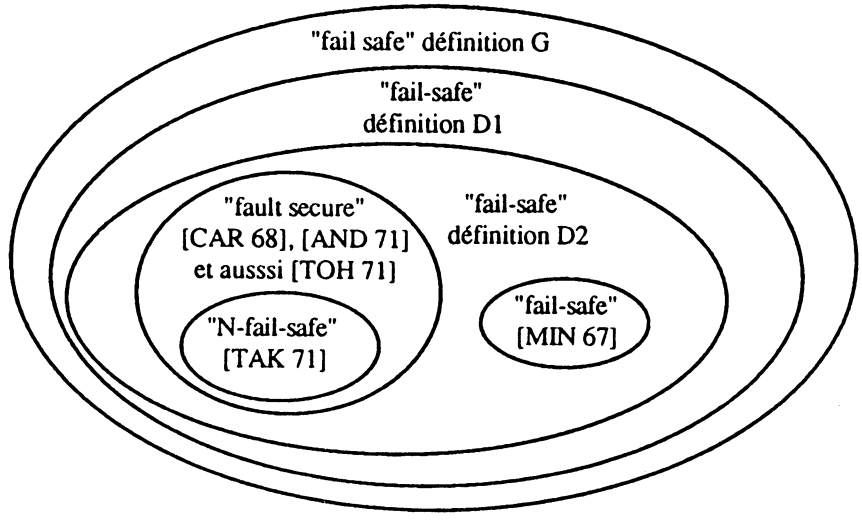


Fig IV - 1 - Relation entre les circuits "fail-safe" et les circuits "fault-secure" [NIC 89]

Une troisième et dernière définition peut être énoncée en ce qui concerne les circuits "fail-safe" pour faciliter la prise en compte de certaines situations dès la conception.
Considérons un circuit qui est tel qu'en présence d'une panne, il peut générer un vecteur de sortie d'état non sûr à la place d'un autre vecteur de sortie d'état non sûr. Pour ce circuit, les sorties sont partagées en plusieurs groupes O^1, ..., O^n où pour chaque groupe de sorties O^i, la notion d'état sûr et d'état non sûr est considérée indépendamment des autres groupes de sorties O^j. De plus, à l'intérieur de chaque groupe les situations décrites par les point 3 et 4, données en début de paragraphe, ne peuvent pas avoir lieu. Si Y^i est l'ensemble de tous les vecteurs de sortie et si O^i_s est l'ensemble des vecteurs de sortie d'état sûr du groupe de sorties O^i, alors si le vecteur de sortie du système global est considéré comme un vecteur de sortie d'état sûr lorsque tous les vecteurs de sortie de tous les groupes sont des vecteurs de sortie d'état sûr, la définition D3 est la suivante :

DEFINITION D3

Soit un système G dont les sorties sont partagées en différents sous-groupes O^1, ..., O^n tels que l'état des sorties d'un sous-groupe est considéré comme sûr ou non sûr indépendamment de l'état des sorties des autres sous-groupes. G est un système "fail-safe" de type D3 si :

$\forall a \in X, \forall f \in F, \forall i \in \{1, 2, .., n\}$: $G^i(x, f) = G^i(x, \emptyset)$,

ou $G^i(x, f) \in O^i_s$.

où : $G^i(x, \emptyset)$ est le vecteur de sortie obtenu normalement pour le sous-groupe de sorties O^i et,
$G^i(x, f)$ est le vecteur erroné présent en sortie de ce même sous-groupe après l'occurrence d'une panne. □

Il est évident que la classe des systèmes donnée par cette dernière définition est une sous-classe des systèmes obtenus à partir de la définition D1 pour laquelle :

$Os = O^1s \times ... \times O^ns$,
$On = (Y^1 \times ... \times Y^n) - Os$ et,

$O^s_{nxn} = \{[a,b]/[a,b] = [(a^1, .., a^n), (b^1, .., b^n)] \in On \times On : b^i \neq a^i \Rightarrow b^i \in O^i_s, \forall i \in \{1,..,n\}\}$.

4 - INTERCONNEXION DE SYSTEMES "FAIL-SAFE"

Lorsque plusieurs sous-systèmes "fail-safe" sont utilisés pour concevoir un système plus complexe, un sous-système A, par exemple, reçoit sur ses entrées les sorties d'un sous-système B (fig.2.a) ou de plusieurs sous-systèmes B1, ..., Bk (fig.2.b). Pour que le système global soit "fail-safe", l'ensemble des vecteurs d'entrée et des vecteurs de sortie des différents sous-systèmes interconnectés devront satisfaire à l'une des conditions données par la suite.

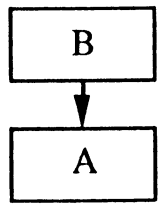


Fig 2.a

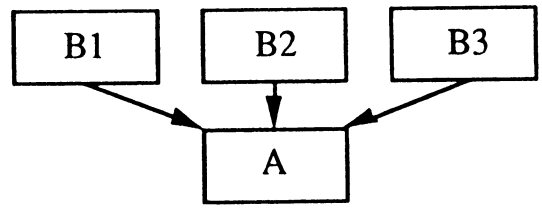


Fig 2.b

Fig IV - 2 - Interconnexion de sous-systèmes "fail-safe"

Tout d'abord considérons l'ensemble des vecteurs d'entrée pour lesquels un système "fail-safe" qui fonctionne normalement délivre des vecteurs de sortie qui appartiennent à Os et notons Is cet ensemble de vecteurs d'entrée d'état sûr du système. Tous les autres vecteurs d'entrée, ceux pour

lesquels le système génère, dans les mêmes conditions de fonctionnement, des vecteurs de sortie d'état non sûr, appartiennent par définition à In l'ensemble des vecteurs d'entrée d'état non sûr du système.

En se reportant à la figure 2.a, nous supposons que $O^b{}_s$ et $O^b{}_n$ sont respectivement l'ensemble des vecteurs de sortie d'état sûr et l'ensemble des vecteurs de sortie d'état non sûr du sous-système B. De même, $I^a{}_s$ et $I^a{}_n$ sont les ensembles des vecteurs d'entrée d'état sûr et des vecteurs d'entrée d'état non sûr du sous-système A. Pour concevoir un système globalement "fail-safe" lorsque deux sous-systèmes A et B vérifient la définition D2, la condition à respecter est la suivante :

CONDITION C2 :

Les deux sous-systèmes A et B vérifient la définition D2, le système global est "fail-safe" de type D2 si la relation $O^b{}_s \subseteq I^a{}_s$ est respectée.

Examinons le cas du sous-système B (fig.2.a) qui vérifie la définition D3 ou, ce qui est équivalent, le cas des sous-systèmes B1, ..., Bk (fig.2.b) qui répondent chacun à la définition D2. Dans un premier temps nous admettons que le sous-système A vérifie la définition D2 et que le sous-système B est connecté à ce dernier par l'intermédiaire de plusieurs groupes de lignes de sorties O^{b1}, ..., O^{bn} . Pour chacun des sous-groupes de lignes O^{bi} les différents vecteurs possibles seront des vecteurs de sortie d'état sûr ou non sûr indépendamment de l'état des autres vecteurs présents sur les autres sous-groupes de lignes O^{bj}. Soit I^{a1}, ..., I^{an} les groupes de lignes d'entrée correspondantes du sous-système A. Nous définissons $\{I^{a1}{}_s,..., I^{an}{}_s\}$ comme l'ensemble des sous-ensembles des vecteurs d'entrée d'état sûr qui sont tels que chaque fois qu'un vecteur erroné e^{ai} de l'ensemble $I^{ai}{}_s$ est appliqué sur le groupe de lignes d'entrée I^{ai}, et que les autres vecteurs appliqués sur les autres groupes de lignes d'entrée I^{aj} sont corrects, alors le sous-système A qui fonctionne normalement délivre soit le vecteur de sortie attendu soit un vecteur de sortie d'état sûr. La condition à respecter pour que les deux sous-systèmes A et B constituent un système globalement "fail-safe" est la suivante :

CONDITION C3 :

Le système B vérifie la définition D3, le système A vérifie la définition D2, le système global réalisé à l'aide des sous-systèmes A et B précédents est "fail-safe"de type D2 s'il existe un ensemble $\{I^{a1}{}_s,..., I^{an}{}_s\}$ pour lequel la relation $O^{bi}{}_s \subseteq I^{ai}{}_s$, $\forall\, i \in \{1, ... ,n\}$, est satisfaite. □

Si le système A vérifie la définition D3, ce dernier peut être partagé en plusieurs sous-systèmes Ai qui répondent tous à la définition D2 et alors la condition C3 est appliquée à chacun des sous-systèmes Ai considéré. Le système global sera "fail-safe" de type D3.

Maintenant nous établissons la condition C1 dans le cas où les sous-systèmes A et B répondent à la définition D1 :

Soit O^s_{nxn} l'ensemble des couples sûrs des vecteurs de sortie d'état non sûr d'un système G qui vérifie la définition D1. Notons I^s_{nxn} l'ensemble des couples sûrs de vecteurs d'entrée d'état non sûr [a,b] qui sont transposés, par ce même système G, en couples de vecteurs qui appartient forcément à O^s_{nxn}. Si O^{sb}_{nxn} est l'ensemble O^s_{nxn} correspondant au sous-système B et si I^{sa}_{nxn} est l'ensemble I^s_{nxn} correspondant au sous-système A, dans le cas le plus général où les deux sous-systèmes (fig.2.a) appartiennent à la classe des circuits donnée par la définition D1, la condition C1 peut être énoncée.

CONDITION C1 :

Si les sous-systèmes A et B vérifient chacun la définition D1, la condition à respecter pour que le système global soit "fail-safe" de type D1 est la suivante :

a / $O^b_s \subseteq I^a_s$ et,

b / $\forall$ [a,b] $\in O^{sb}_{nxn}$, soit b $\in I^a_s$ soit [a,b] $\in I^{sa}_{nxn}$. □

Pour un système donné par la définition D3, nous pouvons remarquer que tous les couples [a,b] ([a,b] =[(a^1, ..., a^n), (b^1, ..., b^n)]) pour lesquels $b^i \in I^i_s$ chaque fois que $b^i \neq a^i$ sont des couples sûrs. Alors :

$I^s_{nxn} = \{[a,b] / [a,b] = [(a^1, .., a^n), (b^1, .., b^n)] \in In \times In : b^i \neq a^i \Rightarrow b^i \in I^i_s, \forall i \in \{1, .. ,n\}\}$.

De là, le lemme 1 (annexe 3)

LEMME 1 :

La condition C3 et un cas particulier de la condition C1. □

La condition C1 est aussi valable pour le schéma de la figure 2.b, puisque les sous-systèmes B1, ..., Bk qui vérifient tous la définition D1 constituent un sous-système B qui répond lui aussi à la définition D1 (annexe 4).

Enfin, dans [NIC 89] une condition C plus générale est proposée pour la conception de systèmes "fail-safe" de type G à partir des sous-systèmes A et B qui vérifient chacun la définition G. Comme cette condition et cette définition sortent du cadre de notre étude nous ne la détaillerons pas.

Après l'énoncé de toutes ces conditions, nous pouvons établir le premier théorème de la théorie générale des systèmes "fail-safe" :

THEOREME 1 :

Si un système est composé de plusieurs sous-systèmes S1, ..., Sn qui sont chacun "fail-safe" d'après la définition D1, D2 ou D3 et, respectivement, si la condition C1, C2 ou C3 est satisfaite par les différents sous-systèmes interconnectés entre eux, alors le système global est "fail-safe" de type D1, D2 ou D3 pour un ensemble de pannes $F = F1 \cup ... \cup Fn$ où Fi, $i \in \{1, ..., n\}$ qui est l'ensemble des pannes vis à vis duquel Si est "fail-safe". □

Lorsqu'une panne apparaît dans un sous-système Si qui répond à la définition D1, D2 ou D3, les vecteurs de sortie générés par ce sous-système défaillant sont propagés, de par la condition C1, C2 ou C3, sur les sorties primaires du système en vecteurs qui vérifient les conditions imposées par la définition D1, D2 ou D3.

5 - LE "TOTALLY FAIL-SAFE GOAL" ET LES CIRCUITS "STRONGLY FAIL-SAFE"

Si un système G est "fail-safe" pour un ensemble de pannes F, l'occurrence d'une première panne $f1 \in F$ ne provoque pas de vecteur de sortie erroné d'état non sûr, alors la sécurité est assurée. Cependant, les vecteurs d'état sûr présents sur les sorties peuvent être ceux qui appartiennent à l'ensemble des vecteurs qui sont obtenus en fonctionnement normal. Dans ce cas la première panne n'est pas signalée puisque aucun moyen n'existe pour cela (les systèmes "fail-safe" ne possèdent pas de moyens pour mettre en évidence des erreurs). Si le système est utilisé pendant un temps assez long après l'occurrence de cette première panne, l'apparition d'une seconde panne $f2 \in F$ est probable et la panne composée (f1, f2) présente dans le système peut ne plus appartenir à F et la propriété "fail-safe" n'est plus garantie. Il est donc absolument nécessaire d'utiliser des techniques qui permettent de détecter assez rapidement la présence d'une première panne pour empêcher que le système défaillant fonctionne encore longtemps après l'occurrence de cette panne. La technique de détection qui est proposée pour les systèmes "fail-safe" s'inspire de la propriété "self-testing" énoncée dans le cas des circuits "self-checking".
Pour les systèmes "fail-safe" qui ne possèdent pas de mécanismes pour la détection des erreurs (i.e. techniques de codage, voteurs, ...), des modes spéciaux de fonctionnement sont utilisés (test hors-ligne, ...) pour la mise en évidence des pannes. Par contre, pour les systèmes qui possèdent leurs propres mécanismes de détection le mode normal de fonctionnement est suffisant. La présente définition est assez générale pour prendre en compte tous les cas possibles de situation.

DEFINITION D4 :

Un circuit est "self-testing" pour un ensemble de pannes F, si pour chaque panne f de F, il y a au moins un mode de fonctionnement du circuit durant lequel la panne f est détectée.

DEFINITION D5 :

Un circuit est "totally fail-safe" ("totalement sécuritaire") s'il est "fail-safe" et "self-testing".

Si pendant un certain temps un circuit "fail-safe" défaillant fournit plusieurs vecteurs de sortie erronés, cela ne présente aucun inconvénient dans la mesure où ce sont des vecteurs d'état sûr. Ce but à atteindre par les circuits "fail-safe" est le "totally fail-safe goal" par analogie au "totally self-checking goal" des circuits "self-checking". Les circuits "totally fail-safe" accomplissent le "totally fail-safe goal" si l'hypothèse H1" est respectée :

HYPOTHESE H1" :

a/ les pannes apparaissent une à une dans le circuit,

b/ entre l'occurrence de deux pannes consécutives, il s'écoule un laps de temps assez long pour que le circuit soit suffisamment contrôlé avec les moyens mis en oeuvre pendant les différents modes de fonctionnement possibles.

De la même manière que la définition des circuits "strongly fault-secure", nous définissons les circuits "strongly fail-safe" qui forment la classe la plus générale des circuits "fail-safe" capables d'atteindre le "totally fail-safe goal" si l'hypothèse H1" est respectée :

DEFINITION D6 :

Un circuit G est "strongly fail-safe" pour un ensemble de pannes F, si pour chaque panne $f1 \in F$:

a/ G est "totally fail-safe" ou,

b/ G est "fail-safe" et si une nouvelle panne f2 de F apparaît, pour la panne composée (f1, f2]) la clause a/ ou la clause b/ est vérifiée.

Toutefois, il est possible de concevoir des systèmes "strongly fail-safe" qui ne possèdent aucun moyen de détection d'erreurs. Le théorème suivant donne la condition nécessaire et suffisante de l'existence de tels systèmes .

THEOREME 2 :

Un système G qui ne possède pas ses propres moyens de détection d'erreur est "strongly fail-safe" pour un ensemble de pannes F si et seulement si il est "fail-safe" pour l'ensemble des pannes F*, où F* est constitué par des pannes multiples qui sont formées par la combinaison des pannes simples de F.

Preuve :

Si le système G est "strongly fail-safe" pour l'ensemble des pannes F, chaque fois qu'une nouvelle panne f de F apparaît, la clause b/ de la définition D6 est vérifiée (dans ce cas la propriété

"self-testing est impossible puisque le système ne possède aucun mécanisme de détection d'erreur et la clause a/ n'est jamais considérée). Le système G reste donc "fail-safe" pour la panne multiple obtenue. Comme ceci est vrai pour toute combinaison de pannes de f, alors G est "fail-safe" pour l'ensemble F*.
Si le système G est "fail-safe" pour un ensemble de pannes F*, après toute occurrence d'une nouvelle panne dans le circuit la clause b/ de la définition D6 reste valable. G est donc "strongly fail-safe" pour l'ensemble des pannes F.

COROLLAIRE 1 :

Si un système G est "fail-safe" pour un ensemble de pannes F* alors il est "strongly fail-safe" pour ce même ensemble F*.

En effet si le système G est "fail-safe" pour l'ensemble F*, il est forcément "fail-safe pour (F*)* puisque (F*)* = F*. Alors d'après le théorème 2, le système est "strongly fail-safe" pour l'ensemble F*.

Concevoir un système complexe "strongly fail-safe" qui n'a pas de mécanisme de détection d'erreurs est une tâche qui peut se révéler très difficile puisqu'il faut que ce système soit "fail-safe" vis à vis d'un ensemble de pannes multiples.
Le problème est simplifié si nous adoptons une solution qui consiste à découper le système global en sous-systèmes. Alors nous donnons deux théorèmes qui reportent la difficulté au niveau de la conception de chaque sous-système "fail-safe". De ce fait, la réalisation pratique de systèmes complexes "fail-safe" est facilitée.

Dans un premier temps, nous supposons que chaque sous-système est "fail-safe" de type D2 ou de type D3

THEOREME 3 :

Soit G un système composé :

- de n sous-systèmes S1, ..., Sn "fail-safe" - de type D2 ou de type D3 - pour les ensembles de pannes respectifs F1, ..., Fn,

et :

- tel que les n sous-systèmes interconnectés respectent les conditions C2 ou C3 suivant la définition à laquelle ils répondent,

alors G est "fail-safe" de type D2 ou de type D3 pour l'ensemble des pannes F = F1 x ... x Fn. □

Dans ce théorème, l'ensemble des pannes possibles Fi comprend aussi la panne fi = ø. Alors une panne f ∈ F est composée de n pannes (f1, .., fn) ∈ F = F1 x ... x Fn , ce qui revient à dire que le système est "fail-safe" pour les pannes qui affectent simultanément n'importe quel ensemble des sous-systèmes qui composent G.

Preuve :

La preuve de ce théorème est donnée en annexe 5.

COROLLAIRE 2 :

Si dans le précédent théorème, chaque sous-système Si est "fail-safe" pour un ensemble de pannes multiples Fi*, alors le système global G est "fail-safe" pour l'ensemble des pannes multiples F* = F1* x ... x Fn* et par conséquent (corollaire 1) il est "strongly fail-safe" pour ce même ensemble F* .

En effet, d'après le théorème 3 le système G est "fail-safe" pour l'ensemble des pannes qui affectent simultanément n'importe quel groupe de sous-systèmes Si. Comme pour chacun d'eux la panne considérée peut être une panne multiple fi*, le système global est donc "fail-safe" pour toute panne multiple (f1*, ..., fn*) ∈ F* = F1* x ... x Fn*.

Dans le cas où tous les sous-systèmes Si qui composent G répondent à la définition D1, le théorème 4 peut être énoncé si la propriété de transitivité suivante est respectée.

PROPRIETE DE TRANSITIVITE :

Un ensemble de couples sûrs de vecteurs de sortie d'état non sûr O^s_{nxn} vérifie la propriété de transitivité si :

$\forall$ a, b, c : [a,b] ∈ O^s_{nxn} $\wedge$ [b,c] ∈ O^s_{nxn} $\Rightarrow$ [a,c] ∈ O^s_{nxn}. □

Cette propriété de transitivité est toujours vérifiée par les systèmes "fail-safe" définis par D2 ou D3 mais elle n'est pas forcément vraie pour ceux qui répondent à la définition D1 [NIC 89].

THEOREME 4 :

Soit G un système composé :

- de n sous-systèmes S1, ..., Sn "fail-safe" - de type D1- pour les ensembles de pannes respectifs F1, ..., Fn,

et :

- tel que les n sous-systèmes interconnectés respectent la condition C1,

- tel que tous les ensembles O^{si}_{nxn} correspondant aux sous-systèmes Si vérifient la propriété de transitivité,

alors G est "fail-safe" de type D1 pour l'ensemble des pannes F = F1 x ... x Fn. □

Preuve :

La preuve de ce théorème est donnée en annexe 4.

Le corollaire 2 peut aussi être énoncé dans le cas des systèmes considérés par le théorème ci-dessus.

Les circuits "fail-safe" décrits dans la littérature sont généralement conçus en utilisant des composants dupliqués (techniques de redondance) [MIN 67] ou des composants dont le MTBF est très élevé pour les pannes considérées comme "dangereuses" (technique d'évitement de pannes) [FUT 88]. En ce qui concerne la redondance, la propriété "fail-safe" ne peut pas être assurée pour les pannes multiples, alors que la technique d'évitement des pannes semble être une solution satisfaisante pour l'application de cette théorie, elle n'est cependant pas envisageable [NIC 89] dans la mesure où nous voulons intégrer des systèmes VLSI "strongly fail-safe".
Par la suite, nous proposons diverses conceptions de systèmes "fail-safe" intégrés en utilisant des techniques de codage (circuits "self-checking") et des techniques BIST. Dans ce cas de conception, la propriété "fail-safe" des systèmes vis à vis d'un ensemble de pannes multiples n'est pas nécessaire pour assurer la propriété "strongly fail-safe".

6 - TRANSFORMATION DES CIRCUITS "SELF-CHECKING" EN CIRCUITS "FAIL-SAFE"

Bien que les circuits "fault-secure" forment un sous-ensemble des circuits "fail-safe" (fig.1), pour de nombreuses applications ces derniers ne peuvent pas être remplacés efficacement par des circuits dont les données en sortie sont codées globalement (i.e. code de parité, code de Berger, ..). En effet, certains processus critiques demandent à ce que chaque signal de sortie d'une fonction soit "fail-safe" individuellement. Un circuit "fault-secure" ne peut pas répondre à ce besoin, à moins qu'une solution utilisant des techniques de codage en fréquence soit envisagée. Cette technique consiste à faire correspondre à un état "non sûr" une fréquence (état "1"), ou un intervalle de fréquences, et à considérer tous les autres états électriques (état "0") comme des "états sûrs". Une telle solution à l'avantage de convenir parfaitement à la commande des éléments électromécaniques (actionneurs) qui se trouvent généralement placés en bout de chaîne des fonctions critiques. Elle est déjà à l'origine de la conception de plusieurs systèmes "fail-safe" existants [SAL 75], [FUT 88].

6.1 La conception d'interfaces "fail-safe"

Ce que nous cherchons donc à réaliser, c'est un mécanisme qui transforme les sorties d'un système "self-checking" de telle manière que les sorties du système global (circuit "self-checking" et mécanisme) respectent toujours les conditions imposées par la définition Dø.

6.1.1 Principe de réalisation

Pour ce faire, nous proposons en figure 3 un premier schéma d'une interface "fail-safe". Le système de traitement des données est conçu avec des blocs fonctionnels "strongly fault-secure" et des contrôleurs "strongly code disjoint". Il délivre n sorties fonctionnelles (I1, ..., In) codées sur k bits (C1, ..., Ck) (code de Berger par exemple) et des signaux d'indication d'erreur issus des différents contrôleurs qui vérifient les blocs fonctionnels. Le message d'erreur final (f1, f2) est codé en double-rail à la sortie du contrôleur global. Ce dernier et les divers contrôleurs utilisés dans le système, y compris celui destiné à la vérification des sorties fonctionnelles (contrôleur de Berger), sont supposés constituer un système "strongly code disjoint".

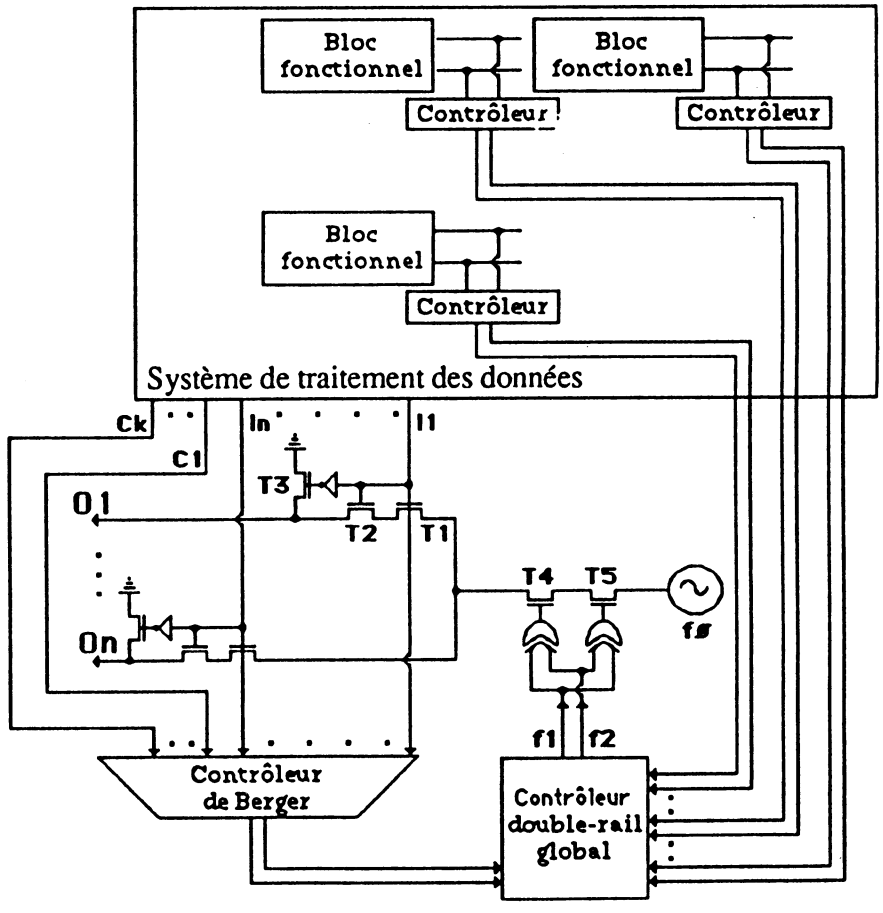


Fig IV - 3 - Schéma de principe d'une interface "fail-safe"

En considérant l'ensemble des signaux I1, I2, ..., In, f1, f2, comme le vecteur de sortie global, les vecteurs hors du code sont ceux pour lesquels (f1,f2) = (0,0) ou (f1,f2) = (1,1). Compte-tenu de la propriété SFS de chaque bloc fonctionnel et de la propriété SCD de l'ensemble réalisé par les différents contrôleurs, chaque vecteur erroné sera donc un vecteur hors-code. Le système qui délivre les vecteurs (I1, I2, ..., In, f1, f2) est un système "fault-secure (et même "strongly fault-secure").

Un générateur fournit une fréquence Fø correspondant à l'état "non sûr" (état "1"). L'interface transforme les signaux binaires I1, ..., In ("1"= 5v, "0"= 0v) en signaux O1, ..., On codés en fréquence.

Pour assurer la propriété "fail-safe" de l'interface quelques éléments sont dupliqués (transistors T1 et T2, portes XOR, transistors T4 et T5) et des précautions particulières sont prises en ce qui concerne chaque branche de l'interface pour éviter les problèmes d'interférences [NIC 89]. Par construction, l'interface est "fail-safe" vis à vis de toute panne simple du type transistor passant ("stuck-on"), transistor ouvert ("stuck-open"), et court-circuit entre deux lignes adjacentes ("bridging fault").

La généralisation du concept proposé dans les paragraphes précédents nous a permis de réaliser un système global "fail-safe" composé de deux sous-systèmes inhomogènes, l'un "self-checking et l'autre "fail-safe". En effet, l'interface est "fail-safe" par construction. Quant au circuit de traitement des données ("strongly fault-secure"), il est lui aussi "fail-safe" d'après la définition D2 puisque les vecteurs de sortie erronés sont forcément des vecteurs hors-code (i.e. (f1,f2) = (0,0) ou (1,1)) et sont considérés à ce titre comme des vecteurs d'état sûr. Comme l'apparition d'un vecteur (I1, .., In, f1, f2) erroné en sortie du système de traitement des données isole la fréquence Fø de l'interface, toutes les sorties O1, ..., On sont placées dans un état "sur". Alors la condition C2 est satisfaite et d'après le théorème 3 nous pouvons affirmer que le système global est "fail-safe" pour l'ensemble des pannes simples qui affectent simultanément le circuit de traitement des données et l'interface.

Remarque : Si par hypothèse seule l'occurrence d'une panne simple est possible dans le système global, alors la conception de l'interface est simplifiée en supprimant une porte XOR et le transistor correspondant.

6.1.2 Cas des systèmes dupliqués

Le cas des systèmes dupliqués est un cas particulier des systèmes "self-checking". L'interface à prendre en compte pour ce type de conception est proposée en figure 4. Pour ce schéma, nous pouvons facilement vérifier que le système global est "fail-safe" pour toute panne simple qui peut

se produire dans l'interface, ou pour toute panne simple qui affecte soit le sous-système A soit le sous-système A' (mais pas les deux à la fois).

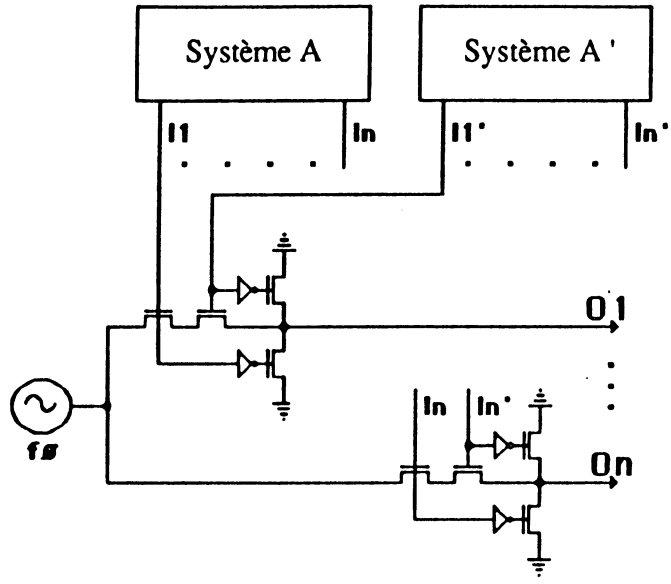


Fig IV - 4 - Interface "fail-safe" pour des systèmes dupliqués

6.1.3 Cas des systèmes tripliqués

Un autre cas intéressant est présenté en figure 5. Les trois systèmes A1, A2, A3 sont identiques et ils effectuent les mêmes opérations en même temps.

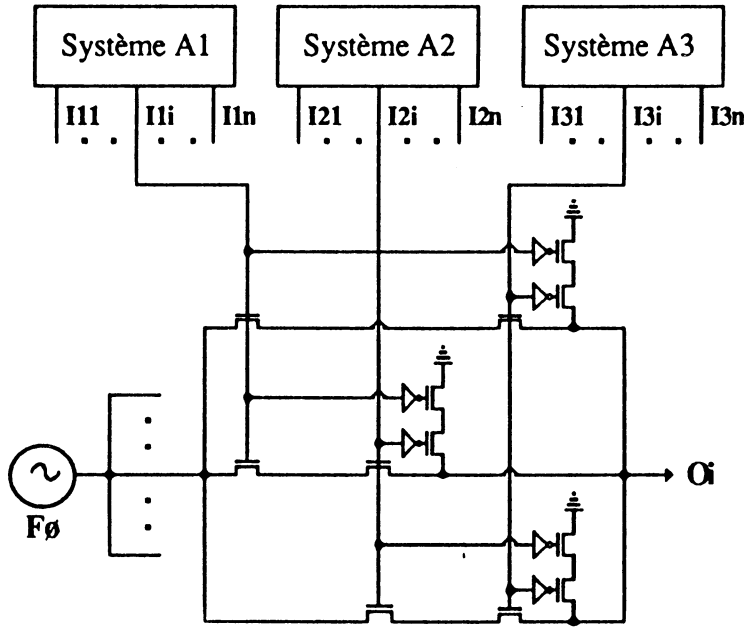


Fig IV - 5 - Interface "fail-safe" pour des systèmes tripliqués

L'interface traite la redondance et autorise la connexion entre le générateur de fréquence et la sortie Oi si au moins deux des trois signaux I1i, I2i, I3i, ont la même valeur binaire "1". Comme le cas précédent, le système est "fail-safe" vis à vis d'une panne simple dans l'interface ou vis à vis de toute panne simple qui peut apparaître dans l'un des trois systèmes (mais pas deux à la fois). Une telle architecture à l'avantage d'être tolérante aux pannes, c'est à dire que le système est conçu de telle manière que la présence d'une donnée erronée ou d'une panne simple dans l'interface permet, quand même, d'obtenir le résultat prévu lors des spécifications.

6.2 La conception d'interfaces "strongly fail-safe"

La partie du système de la figure 4 qui génère le vecteur (I1, ..., In, f1, f2) est "strongly fault-secure" et par conséquent elle est considérée comme "strongly fail-safe". Si nous voulons que le système global soit "strongly fail-safe", l'interface doit aussi avoir cette dernière propriété. Toutefois nous remarquons que les différentes interfaces proposées au paragraphe précédent ne restent pas "fail-safe" en présence de certaines pannes multiples (i.e. pour l'interface de la figure 3, les pannes multiples du type transistor T1 et transistors T2 passants, collages à "1" des sorties des deux portes XOR combinés avec avec un collage à "1" d'une entrée Ii, ..., sont des pannes pour lesquelles l'interface perd la propriété "fail-safe"). Par conséquent, pour assurer la propriété "strongly fail-safe" de l'interface il faut prévoir des mécanismes pour mettre en évidence une première panne avant l'occurrence d'une seconde .
Ceci peut être obtenu en utilisant des techniques BIST. Un générateur de vecteurs de test (TVG) délivre la séquence utile pour détecter les pannes simples dans l'interface. Un analyseur de signature, qui est initialisé de façon à ce que la signature finale normalement obtenue soit 010101..., prend en compte l'ensemble des réponses du bloc sous test. A la lecture du contenu de l'analyseur, à la fin de la phase de test hors-ligne, un mécanisme approprié (flip-flop) génère la séquence complémentaire 101010... pour que le message d'erreur obtenu (g1,g2) soit codé en double-rail ([NIC 88 a]).

Le schéma de la figure 6 donne le principe utilisé pour le test hors-ligne d'une branche de l'interface proposée en figure 4. Le générateur de vecteurs de test à cinq sorties (A1, ..., A5) est utilisé pour le test de l'ensemble des branches du système. D'autre part, l'analyseur de signature reçoit les réponses des n branches de l'interface sous test. Cet analyseur est donc réalisé avec au moins n+2 cellules de base. La phase de test hors-ligne de l'interface est activée avec une période plus faible que le MTTF du circuit pour être certain de détecter la présence d'une première panne avant qu'une seconde apparaisse (pour satisfaire l'hypothèse H1').

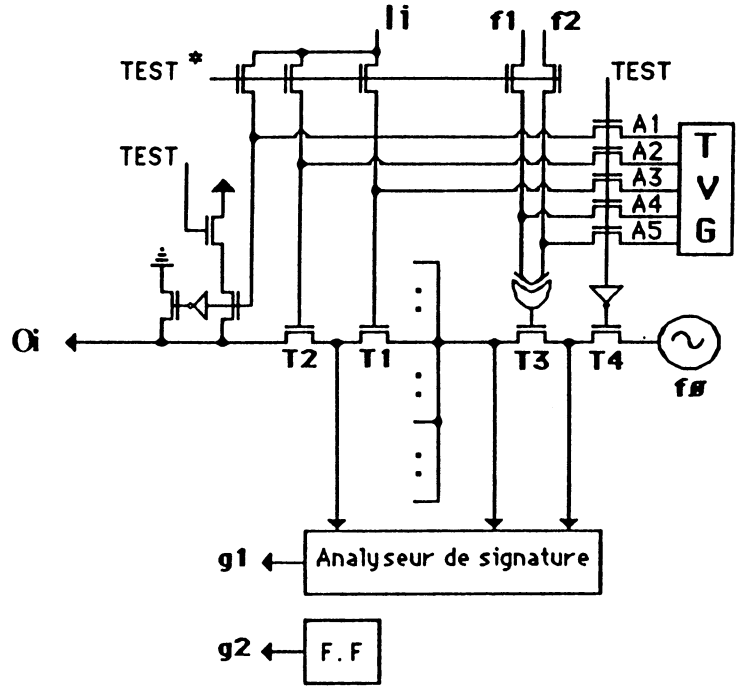


Fig IV - 6 - Interface "strongly fail-safe" (utilisation de techniques BIST)

Dans ce cas de figure les contraintes suivantes sont respectées :

- a / L'activation du mode de test hors-ligne (TEST = "1", TEST* = "0") de l'interface ne perturbe pas le fonctionnement normal du système global. Comme les systèmes électromécaniques (du type actionneurs) sont sensibles à des fréquences de l'ordre de la ms, cette contrainte est aisément satisfaite puisque la phase de test hors-ligne ne dure que quelques µs.

- b / L'interface proposée en figure 6 où des techniques BIST sont utilisées reste toujours "fail-safe", car toutes les pannes simples pour lesquelles cette interface pourrait perdre la propriété "fail-safe" sont détectées (i.e. transistor T4 s-on, ...). D'autres pannes (i.e. transistor T4 s-open ...) n'altèrent en rien la propriété désirée et du point de vue de la sécurité elles n'ont pas à être obligatoirement signalées.

La séquence de test qui accomplit le but fixé par le point b est donnée en figure 7.a. Elle met en évidence toutes les pannes simples de l'interface dont la combinaison pourrait entraîner une situation dangereuse. Le système global est alors "strongly fail-safe".

A1	A2	A3	A4	A5
1	1	1	1	0
0	1	1	1	0
1	1	1	0	0
0	1	1	0	1
1	1	1	0	1
0	1	1	1	1
1	1	0	1	1
0	0	0	0	0

Fig 7.a - Séquence de test de l'interface de la fig.6

Ii*	Ii	Si	Oi	NC
1	1	1	1	1
0	0	1	0	1
1	1	1	1	1
0	1	0	0	0
0	0	0	0	1
1	1	1	1	1
1	0	1	0	0

Fig 7.b - Séquence de test de l'interface de la fig.8

Fig IV - 7 - Séquences de vecteurs de test hors-ligne pour différentes interfaces

La figure 8 montre la conception d'un système global "strongly fail-safe" réalisé avec deux systèmes de traitement des données, A et A*, qui sont des systèmes dupliqués complémentaires.

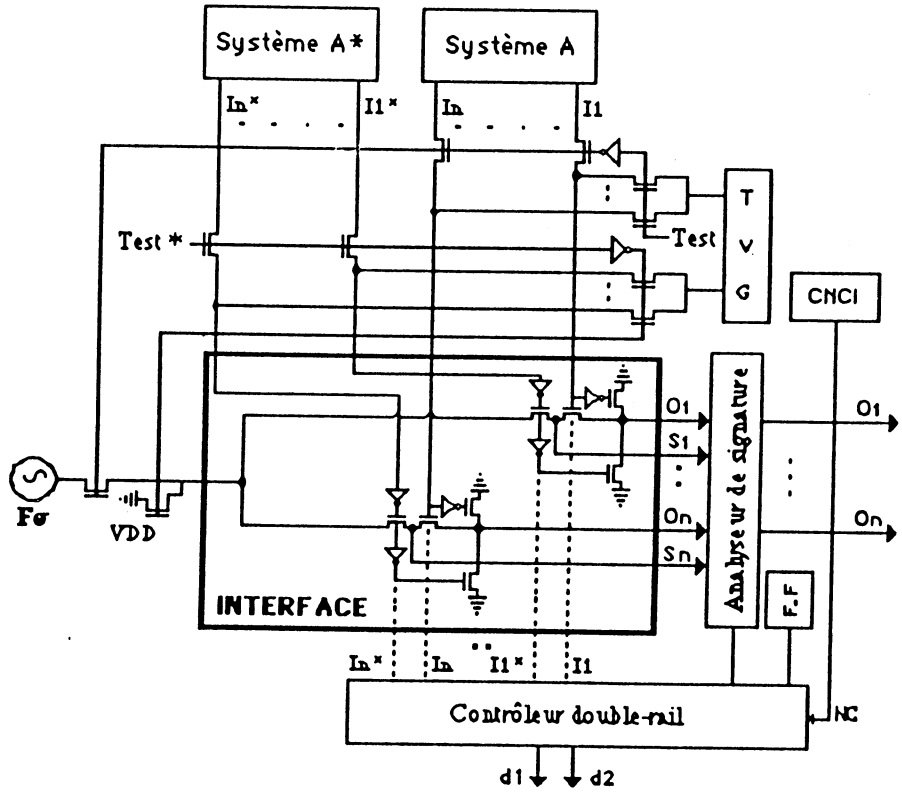


Fig IV - 8 - Interface "strongly fail-safe" pour des systèmes dupliqués

Pour cette architecture et pendant la phase de test hors-ligne, comme le générateur (TVG) ne fournit pas que des vecteurs qui appartiennent au code en usage (code double-rail) une circuiterie (CNCI) indique (signal NC) quelle est la nature de la donnée (vecteur du code ou vecteur hors-code) qui est injectée sur les entrées de l'interface (voir [NIC 88 a] pour cette technique). La figure 7.b donne la séquence de vecteurs appliqués sur Ii, Ii* et NC et la valeur binaire des réponses normalement attendues sur Oi et Si.
La vérification de la signature est effectuée à l'aide du contrôleur double-rail du système en procédant de la même façon que dans le cas de la figure 6. Ici aussi, les trois contraintes a/ b/ et c/ sont satisfaites.

Remarque : un montage similaire pourrait être proposé si nous voulions obtenir un système "strongly fail-safe" constitué avec les éléments tripliqués de la figure 5.

Les propositions précédentes ont été données pour que la propriété "strongly fail-safe" des systèmes soit toujours respectée. Nous n'avons cependant jamais indiqué comment le système était placé dans un état sûr après qu'une première panne ait été détectée (message d'erreur (d1,d2) de la figure 8 avec (d1,d2) = (1,1) ou (0,0)). Le mécanisme qui accomplit cette tâche et qui est intégré dans le circuit est celui proposé en [NIC 89] où les auteurs utilisent des indicateurs d'erreurs "self-checking" définis en [GAI 85]. Avec de tels moyens, dès la mise en évidence d'une première panne dans le système global, quelle que soit la phase de fonctionnement, les sorties O1,..., On sont toutes placées dans un état sûr de façon irréversible.

7 - LE CAS DES CIRCUITS SEQUENTIELS

Comme pour les circuits combinatoires, pour la réalisation de machines séquentielles "fail-safe" [TAK 72], [CHU 78], basée sur la définition donnée en [MIN 67] et [TAK 71], la propriété est considérée vis à vis de chaque sortie. *Nicolaidis* [NIC 89] montre que la généralisation du concept "fail-safe" proposée dans ce chapitre peut être modifiée pour satisfaire les contraintes imposées dans le cas des circuits séquentiels. De ce fait, la notion de sortie d'état sûr ou non sûr peut aussi être estimée vis à vis d'un ensemble de sorties sur lesquelles plusieurs vecteurs sont possibles.
Toutefois cette théorie des circuits séquentiels "fail-safe" n'est pas forcément utile pour obtenir des machines séquentielles sûres. En effet, avec les interfaces proposées au paragraphe précédent et les définitions que nous avons données, une fois les problèmes de synchronisation résolus, il est tout à fait possible de placer en amont du système de codage en fréquence un automate "self-checking" comme celui du chapitre III pour former un ensemble "strongly fail-safe".

8 - CONCLUSION

Tels qu'ils étaient définis, les circuits "fail-safe" conventionnels ne peuvaient malheureusement pas être intégrés et de ce fait ils se prêtaient difficilement à la réalisation de systèmes complexes. D'autre part, la majorité de ces circuits ne possèdaient pas de moyens propres qui leur permettaient de détecter la présence d'une première panne (panne latente) et alors la propriété "fail-safe" pouvait être perdue après l'occurrence d'une seconde panne pour aboutir à une situation catastrophique.
Une partie de la théorie générale des systèmes "fail-safe" donnée dans ce chapitre propose de résoudre ces problèmes . En effet, les nouvelles définitions et les conditions de conception que nous avons établies permettent d'utiliser des techniques "self-checking" et "BIST" combinées avec l'emploi d'une interface "fail-safe" (*) pour obtenir des systèmes "fail-safe" à haute densité d'intégration (circuits "fail-safe" VLSI). De ce fait, il est maintenant possible de concevoir des systèmes combinatoires, ou séquentiels, "fail-safe" qui soient complexes. Ceci avec une diminution sensible des coûts de réalisation des produits et une augmentation de la sûreté de fonctionnement : en terme de sécurité puisque l'occurrence de plusieurs pannes est possible sans que le système perde sa propriété et en terme de disponibilité puisque les techniques "self-checking" et "BIST" sont des moyens de détection, et parfois de localisation, rapides des pannes.

(*) *Brevet MERLIN-GERIN n°89 02636.*
"Système de codage en fréquence à haute sûreté de fonctionnement".
M.NICOLAIDIS (IMAG/TIM3) - S.NORAZ (MERLIN-GERIN).

CONCLUSION

CONCLUSION

Le travail présenté dans ces pages montre dans quelle mesure les circuits intégrés autotestables peuvent être envisagés comme une solution intéresssante pour la conception de systèmes complexes à haute sûreté de fonctionnement.

Dans la mesure où chaque fonction comporte des mécanismes de test en-ligne ("self-checking") et/ou hors-ligne ("BIST"), ces derniers permettent d'accroître sensiblement la disponibilité et la maintenabilité (détection immédiate de la panne, possibilité de localisation rapide de cette panne, donc intervention dans un bref délai) du système qui est implanté en VLSI. Mais c'est certainement pour le critère de sécurité (au sens des défaillances catastrophiques) que l'emploi de fonctions autotestables est le plus intéressant puisque une donnée erronée en sortie du système est immédiatement signalée alors l'exploitation adéquate du message d'erreur permettra d'éviter une situation catastrophique. Toutes ces considérations nous ont amenés à proposer une solution pour la conception d'automatismes industriels de sécurité à l'aide de composants autotestables.

Les quelques techniques de conception en vue du test des circuits intégrés qui ont été exposées nous ont permis de préciser les moyens à utiliser pour la conception de systèmes dont la sûreté de fonctionnement est la principale raison d'être. De là, nous nous sommes attardés sur les techniques de conception "self-checking" et "Unified-Built-In-Self-Test" qui sont certainement les mieux adaptées à notre problème. Cependant, pour pouvoir exploiter au mieux l'emploi de circuits intégrés "self-checking" pour la réalisation de systèmes de sécurité ou de nombreux problèmes se posent au niveau de l'étage de sortie, il a fallu élargir le concept des systèmes "fail-safe". Des relations avec les circuits "self-checking" ont été établies, de nouvelles définitions ont été proposées ainsi que des conditions de conception. Les circuits "strongly fail-safe" ont été définis. Le résultat de ce travail a aboutit à la possibilité d'obtenir des systèmes "fail-safe" à très large échelle d'intégration capable d'accomplir des missions critiques. L'application concrète de ces recherches à donné lieu à l'étude de faisabilité d'un automate intégré "strongly fail-safe" composé d'une machine séquentielle "self-checking" et d'une interface" fail-safe" qui assure, dans le pire cas de fonctionnement, la "polarisation" des sorties du système global (circuit "self-checking" et interface) de telle manière qu'aucun événement catastrophique ne peut être envisagé. Ce genre d'automates est susceptible d'être employé dans des applications du type "fonctions réflexes intelligentes".

Evidemment, la maîtrise complète de la sûreté de fonctionnement ne peut être envisagée que si le système est considéré d'un point de vue matériel et logiciel.

Par la suite, il faudrait s'attarder sur l'architecture d'interface "fail-safe" tolérante aux pannes. Aucune étude poussée n'a été entreprise, le sujet a simplement été évoqué dans cette thèse bien qu'il soit très important en sûreté de fonctionnement.
Enfin, les circuits intégrés "strongly fail-safe" semblent être promis à un bel avenir puisqu'ils sont déjà à la base de la conception d'un micro-automate programmable de sécurité (MAPS) destiné au traitement de fonctions sécuritaires de bas niveau dans un réseau ferroviaire (passages à niveau, aiguillages, ..).

REFERENCES

REFERENCES

[AND 71] **ANDERSON D.A.**
"Design of self-checking digital networks using coding techniques".
Urbana, CSL Univ. of Illinois, September 1971 (report 527).

[BER 61] **BERGER J.M**
"A note on error detection codes for asymmetric channels".
Information and Control, New York, March 1961.

[BRE 76] **BREUER M.A., FRIEDMAN A.D.**
"Diagnosis and reliable design of digital systems".
Computer Science Press. 1976.

[CAR 68] **CARTER W.C., SCHEIDER P.R.**
"Design of dynamically checked computers".
IFIP Congress, Edinburgh 1968, Inf.Proc. 68, Amsterdam, North Holland, 1969.

[CHU 78] **CHUANG H., DAS S.**
"Desqign of fail-safe sequential machines using separable codes".
IEEE Trans. Comp., Vol C-27, March 1978.

[COU 79] **COURTOIS B.**
"Some result about the efficiency of simple mechanisms for the detection of microcomputer malfunctions".
9th Int. Symp. on Fault-Tolerant Comp., Madison, June 1979.

[COU 81 a] **COURTOIS B.**
"Test et LSI".
Thèse d'état, USM Grenoble/ INP Grenoble, Juin 1981.

[COU 81 b] **COURTOIS B.**
"Failure mechanisms, fault hypotheses and analytical testing of LSI-NMOS (HMOS) circuits".
VLSI 81, Univ of Edinburgh, August 1981.

[DAV 78] **DAVID R. THEVENOD-FOSSE P.**
"Design of totally self-checking asynchronous modular circuits".
Journal of Design Automation and Fault-Tolerant Computing, Vol 2, October 1978.

[DAV 79] **DAVID R. THEVENOD-FOSSE P.**
"Panorama des méthodes de test non déterministes des circuits logiques".
RAIRO, Vol 13, No.1, 1979.

[DIA 74] **DIAZ M.**
"Design of totally self-checking and fail-safe sequential machine".
14th Int. Symp. on Fault-Tolerant Comp., Urbana, June 1974.

[DIA 79] **DIAZ M., AZEMA P. and AYACHA J.M.**
"Unified design of self-checking and fail-safe combinational and sequential machines".
IEEE Trans. Comp., Vol C-28, March 1979.

[EIC 77] **EICHELBERGER E.B., WILLIAMS T.W.**
"A logic design structure for LSI testability".
14th Design Automation Conference, June 1977.

[FER 88] **FERGUSON F. J., SHEN J.P.**
"Extraction and simulation of realistic C-MOS faults using inductive fault analysis".
ITC 1988.

[FRI 71] **FRIEDMAN A.D.**
"Fault detection in digital circuits".
Prentice Hall Inc. 1971.

[FUN 75] **FUNATSU S., WAKATSUKI N., ARIMA T.**
"Test generation in Japan".
12th Design Automation Conference, June 1975.

[FUC 84] **FUCHS K.W., ABRAHAM J.A.**
"Error detection in highly structured logic arrays".
14 th Int. Symp. on Fault-Tolerant Comp., Kissemmee, June 1984.

[FUJ 85] **FUJIWARA H.**
"Logic testing and design for testability".
MIT Press series in computer systems. Cambridge 1985

[FUT 88] **FUTSUHARA K., SUGIMOTO N., MUKAIDONO M.**
"Fail-safe logic elements having upper and lower thresholds and their application to safety control".
18th Int. Symp. on Fault-Tolerant Comp., Tokyo, June1988.

[GAI 85] **GAITANIS N.**
"A totally self-checking error indicator".
IEEE Trans. Comp., Vol C-34., August 1985.

[GAJ 86] **GAJSKI M.**
"Silicon compilation - Tutorial".
Proc. ICCAD 86, Santa Clara, November 1986.

[GAL 80] **GALAY J.A., CROUSET Y. and VERGNIAULT M.**
"Physical versus logical fault models MOS-LSI circuits : impact of their testability".
IEEE Trans. Comp., Vol C-29, June 1980.

[HAY 84] **HAYES B.**
"Les automates finis".
Récréations informatiques. Pour la Science. Mars 1984.

[HAR 65] **HARRISON M.A.**
"Introduction to switching and automata theory".
New York : McGraw-Hill, 1965.

[JAN 85] **JANSCH-SHREIBER I.E.**
"Conception de contrôleurs autotestables pour des hypothèses de pannes analytiques".
Thèse de docteur-ingénieur, INP de Grenoble, Janvier 1985.

[JAN 88] **JANSCH-SHREIBER I.E., COURTOIS B.**
"Strongly language checkers".
IEEE Transactions on Computers, june 1988.

[KON 79] **KONEMANN B., MUCHA J. and ZWIEHOFF G.**
"Built-In Logic Bloc Observation Techniques".
Proc IEEE 1979 Int'l. Test conference, Cherry-Hill, New Jersey, October 1979.

[LAP 85] **LAPRIE J.C.**
"Sûreté de fonctionnement des systèmes informatiqueset tolérance aux fautes : concepts de base".
Technique et science informatiques.Vol 4, No.5, 1985.

[LAP 89] **LAPRIE J.C., COURTOIS B., GAUDEL M.C., POWELL D.**
"Sûreté de fonctionnement des systèmes informetiques (matériels et logiciels)".
AFCET - Dunod informatique - Mars 1989.

[LIE 76] **LIEVENS C.**
"Sécurité des systèmes".
Sup'Aéro - CEPADUES-Editions. 1976.

[MAK 82] **MAK G.P., ABRAHAM J.A. and DAVIDSON E.S.**
"The design of PLAs with concurrent error detection".
12 th Int. Symp. On Fault-Tolerant Comp., Santa Monica, June 1982.

[MIN 67] **MINE H. and KOGA Y.**
"Basic properties and a construction method for fail-safe logical systems".
IEEE Trans. Elec. Comp., Vol. EC-16, June 1967.

[NAN 87] **NANYA T., KAWAMURA T.**
"A note on strongly fault secure sequential circuits".
IEEE Trans. Comp., Vol C-36, September 1987.

[NAN 88] **NANYA T., KAWAMURA T.**
"Error secure/propagating concept and its application to the design of strongly fault secure processors".
IEEE Trans. Comp., Vol C-37, January 1988.

[NAN 89] **NANYA T., UCHIDA M.**
"A strongly fault secure and strongly code disjoint realization of combinational circuits".
19th Int. Symp. on Fault-Tolerant Comp., Chicago, June 1989.

[NIC 84] **NICOLAIDIS M**
"Conception de circuits intégrés autotestables pour des hypothèses de pannes analytiques".
Thèse de docteur-ingénieur, INP Grenoble, Janvier 1984.

[NIC 85] NICOLAIDIS M., COURTOIS B.
"Layout rules for design of self-checking circuits".
VLSI Conference, Tokyo, August 1985.

[NIC 86 a] NICOLAIDIS M., COURTOIS B.
"Design of self-checking circuits using unidirectional error detection codes".
16th Int. Symp. on Fault-Tolerant Comp., Vienna, July 1986.

[NIC 86 b] NICOLAIDIS M., COURTOIS B.
"An unified BIST approach using specifique strongly code disjoint checkers design".
IMAG report 599, March 1986.

[NIC 88 a] NICOLAIDIS M.
"A Unified Built In Self Test Scheme : UBIST".
18th Int. Symp. on Fault-Tolerant Comp., Tokyo, June 1988.
IEEE Trans. Comp. on CAD/ICAD.

[NIC 88 b] NICOLAIDIS M., COURTOIS B.
"Strongly code disjoint checkers".
IEEE Trans. Comp., June 1988.

[NIC 88 c] NICOLAIDIS M.
"UBIST implementation for a microprocessor sequencing scheme".
IMAG Report RR-690I. January 1988.

[NIC 89] NICOLAIDIS M., NORAZ S. and COURTOIS B.
"A generalized theory of fail-safe systems".
19th Int. Symp. on Fault-Tolerant Comp., Chicago, June 1989.

[NOR 89] NORAZ S., NICOLAIDIS M. and COURTOIS B.
"VLSI implementation for control of critical systems".
IFIP/IFAC. SAFECOMP 89. Vienna. December 1989.

[OZG 77] OZGUNER F.
"Design of totally self-checking asynchronous and synchronous sequential machines".
7th Int. Symp. on Fault-Tolerant Comp., Los-Angeles, June 1977.

[SAL 75] SALOMON G., THEROND J.C.
"Systèmes de logique dynamique pour les circuits de protection des réacteurs nucléaires".
Journées d'information des industries nucléaires. "NUCLEX 75".

[SMI 78] SMITH J.E., METZE G.
"Strongly fault secure logic networks".
IEEE Trans. Comp., Vol C-27, June 1978.

[TAK 71] TAKAOKA T., MINE H.
"N-fail-safe logical systems".
IEEE Trans. Comp., vol C-20, May 1971.

[TAK 72] **TAKAOKA T., IBARAKI T.**
"N-fail-safe sequential machines".
IEEE Trans. Comp., vol C-21, November 1972.

[THA 78] **THATTE S.M, ABRAHAM J.A.**
"A methodology for functional level testing of microprocessors".
8th Int. Symp. on Fault-Tolerant Comp., Toulouse, June 1978.

[TOH 71] **THOMA Y., OHYAMA Y. and SAKAI R.**
"Realization of fail-safe sequential machines by using a k-out-of-n code".
IEEE Trans. Comp. , Vol C-20, November 1971.

[TOK 71] **TOKURA N., KASAMI T. and HASHIMOTO A.**
"Fail-safe logic nets".
IEEE Trans. Comp., Vol C-20, March 1971.

[USA 75] **USAS M.A.**
"A totally self-checking checker for the detection of errors in periodic signals".
IEEE Trans. Comp., Vol C-24, May 1975.

[VIA 80] **VIAUD J., DAVID R.**
"Sequentially self-checking circuits".
10 th Int. Symp. on Fault-Tolerant Comp., Kyoto, October 1980.

[WAD 78] **WADSACK R.L.**
"Fault modeling and logic simulation of CMOS and MOS integrated circuits".
The Bell system technical journal, May/June 1978.

[WAK 78] **WAKERLY J.**
"Error detecting codes, self-checking circuits and applications".
Elsevier, North-Holland Inc 1978.

ANNEXES

ANNEXES

Annexe 1 - *les compilateurs de silicium*

1 - INTRODUCTION

Avec les techniques de conception classiques, la fabrication des circuits VLSI (Very Large Scale Integration) peut demander des dizaines d'hommes/année et autant pour les vérifier. Par exemple, comme les limites théoriques de l'intégration sur silicium sont prévues à dix millions de transistors par puce, cela demanderait actuellement un travail de vérification d'environ six milles hommes / année.
Alors pour pouvoir exploiter au mieux les potentialités de l'évolution technologique, il est nécessaire de définir de nouvelles méthodes de conception et de vérification adaptées à la complexité des circuits à produire et qui prennent en compte les contraintes économiques propres à l'industrie des semi-conducteurs.
La compilation de silicium [GAJ 86] est certainement une réponse à ce besoin.

2 - GENERALITES

Un compilateur de silicium est un logiciel capable de générer les masques nécessaires à la réalisation physique d'un circuit à partir de sa description en langage de haut niveau. C'est une traduction des spécifications.
Malheureusement, la majorité de ces outils ne sont capables d'effectuer qu'une partie limitée de la traduction attendue. C'est pourquoi, nous sommes conduits à distinguer les trois définitions suivantes :

2.1 Les définitions

2.1.1 Le compilateur de layout : c'est une station pour la description procédurale de cellules.
2.1.2 Le compilateur d'architecture : il se charge de placer les blocs générés par des compilateurs spécialisés et d'établir la communication entre eux.
2.1.3 Le compilateur de comportement : c'est un outil qui conçoit des circuits, en utilisant une architecture cible, suivant la fonction à réaliser.

2.2 Classification des outils

Ces trois types d'outils peuvent être classés en fonction du langage de description de départ qu'ils acceptent.

Classe 1 : Le langage d'entrée permet un découpage fonctionnel. Ainsi l'architecture interne des circuits réalisés est définie par le concepteur et la tâche du compilateur se réduit à la génération du dessin des masques associée aux différents blocs fonctionnels.

Cette classe comprend :

- la classe 1.a, où l'utilisateur écrit lui même ses générateurs (compilateur 2.1.1),

- la classe 1.b, où ce sont des modules générateurs standards pour le layout qui sont utilisés (Compilateur 2.1.2).

Classe 2 : le langage d'entrée permet un découpage algorithmique. Dans ce cas l'architecture interne des circuits réalisés est déterminée par le compilateur lui même qui génère par la suite le dessin des masques (compilateur 2.1.3).

2.3 les outils de conception

Dans le processus de conception, il y a trois phases majeures :

- La spécification du circuit
  entrées / sorties
  fonctions
- La spécification de l'architecture
  découpage en blocs simples
  complexes
- La génération du layout

Des outils sont associés à chacune de ces étapes pour :

- La création : description
- L'édition : modification
- La vérification : simulation, test

Un compilateur de silicium va partir de l'une de ces étapes pour arriver aux dessins des masques.

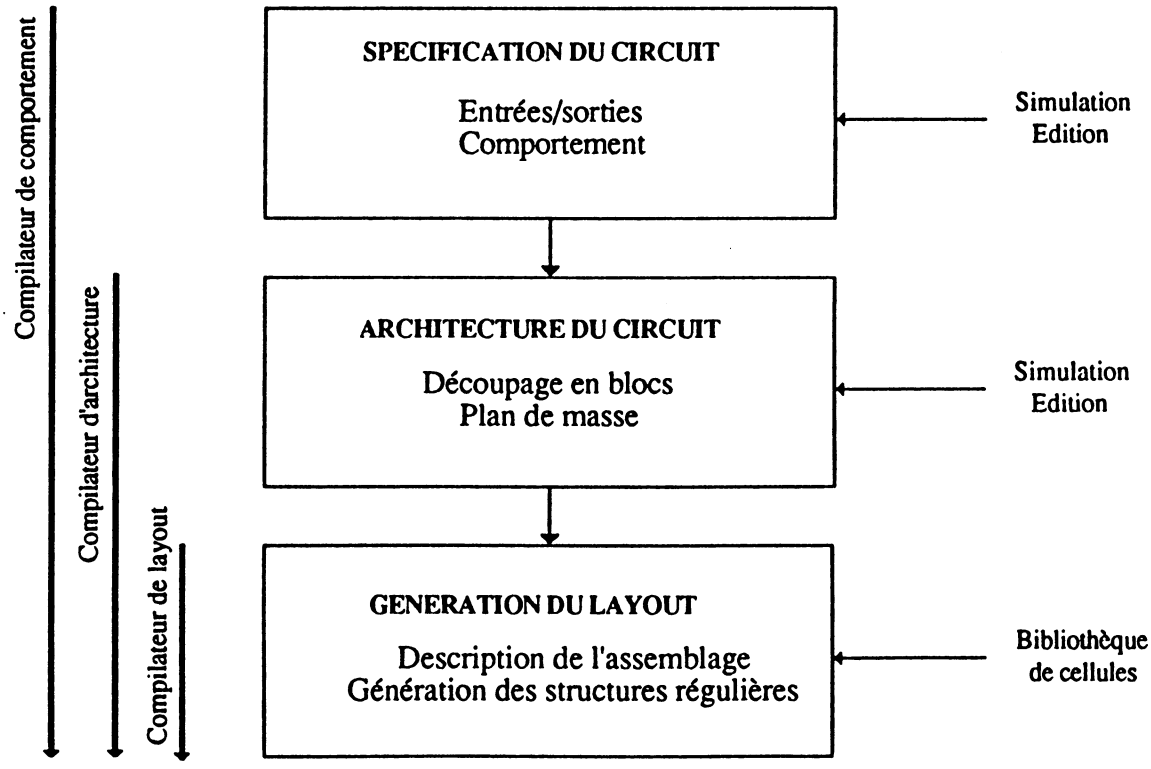


Fig - 1 - Processus de fabrication

3 - PRINCIPES DE FONCTIONNEMENT

Les trois types de compilateurs donnés ci-dessus apparaissent à l'utilisateur comme de simples générateurs de fonctions. Toutefois, chacun d'eux se différencie par sa complexité interne, la manière dont il peut être utilisé et la méthode qu'il propose pour concevoir les circuits.

3.1 Le fonctionnement du compilateur de layout

Cet outil consiste en un environnement de "bibliothèques" et d'outils qui servent à la manipulation de symboles, à l'assemblage de motifs et aux calculs en fonction du paramètrage. Ils autorisent aussi la simulation et l'analyse des circuits et la conversion d'une forme de représentation à une autre (en particulier celle de l'implantation des masques). Enfin ils permettent la vérification des règles de construction et des règles d'implantation ainsi que le calcul prévisionnel des performances du circuit intégré.
Les éléments constitutifs d'un circuit, ou d'une cellule, se trouvent dans la "bibliothèque" de ce compilateur de silicium qui, à l'inverse d'une bibliothèque de cellules standard, renferme des "générateurs".

Exemple : A partir du menu de bibliothèque, le concepteur choisit un compilateur de cellules. Le logiciel lui demande des valeurs de paramètres tels que la fonction à réaliser, le nombre de bits à traiter, la disposition souhaitée pour les entrées/sorties ainsi que le nombre, le type et la disposition relative des cellules. A l'aide des paramètres passés par l'utilisateur, le compilateur génère la représentation sur silicium.

Ces compilateurs de cellules sont capables de créer de nombreuses versions différentes pour une seule version générique. Par conséquent il y a un nombre restreint de cellules pour réaliser toutes les fonctions requises par les concepteurs.
Ce type de compilateur à l'avantage de pouvoir réaliser tous les circuits, mais il faut vérifier que le composant élaboré fonctionne bien. Malheureusement, cet outil nécessite que l'utilisateur soit un expert en circuits intégrés et en programmation. De plus, le temps de conception est long.

SYCOMORE (Thomson Semiconducteurs) contient un compilateur de layout.

3.2 Le fonctionnement du compilateur d'architecture

C'est un outil qui travaille à l'aide de compilateurs spécialisés, qui acceptent des langages de description (Desc POP, Desc RAM, Desc PLA....), et d'une bibliothèque de cellules simples.

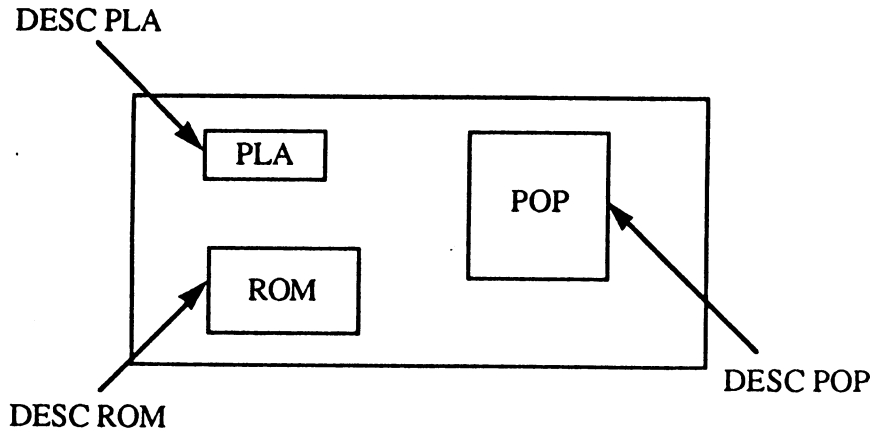


Fig - 2 - Plan de masse

Le compilateur place les blocs compilés (Plan de masse), qui peuvent avoir une structure complexe (bloc de blocs), simple (élément de base) ou régulière (PLAs, ROMs, RAMs..) avec ou sans l'aide de l'utilisateur. Il établit les connexions entre les composants (Routeur), et il simule le fonctionnement du circuit élaboré (Simulateur).

Ce qui fait l'intérêt de ce type de compilateur, c'est que son emploi ne nécessite pas de grandes notions d'architecture (un ingénieur système est parfaitement qualifié pour utiliser ce compilateur), de plus, les temps de conception d'un circuit sont très courts. Malheureusement avec cet outil, il est impossible de réaliser un circuit faisant appel à un bloc pour lequel il n'existe pas de compilateurs spécialisés. D'autre part, comme les blocs sont générés indépendamment puis assemblés par la suite, ceci engendre des restrictions sur la communication entre les blocs.

GENESIL (Silicon Compiler) est considéré comme un compilateur d'architecture.

3.3 Le fonctionnement du compilateur de comportement

C'est un compilateur de circuit de type microprocesseur, filtre ... .
A partir d'un langage de description l'outil élabore l'architecture du circuit et fixe les paramètres des générateurs (les fonctions des deux compilateurs précédents se retrouvent dans ce compilateur). Malheureusement, son utilisation est limitée à un domaine d'applications (celui pour lequel il a été conçu).Toutefois, le temps de conception très court et sa convivialité sont à l'avantage de cet outil.

SYCO (INPG) est un compilateur de comportement.

4 - PRESENTATION DE QUELQUES COMPILATEURS

4.1 Les compilateurs de layout

Bull a adapté certaines parties d'un outil universitaire (PAOLA (INPG)) de manière à obtenir un "module générateur" utilisé pour l'optimisation topologique de PLAs. Le schéma global de la chaîne de génération est donné en figure 3. Cet outil peut ainsi traiter des composants à trois cents entrées, trois cents sorties et mille monômes (portes NOR).
PAOLA est un exemple de compilateur spécialisé utilisable par des compilateurs du type 2.1.2 et 2.1.3.

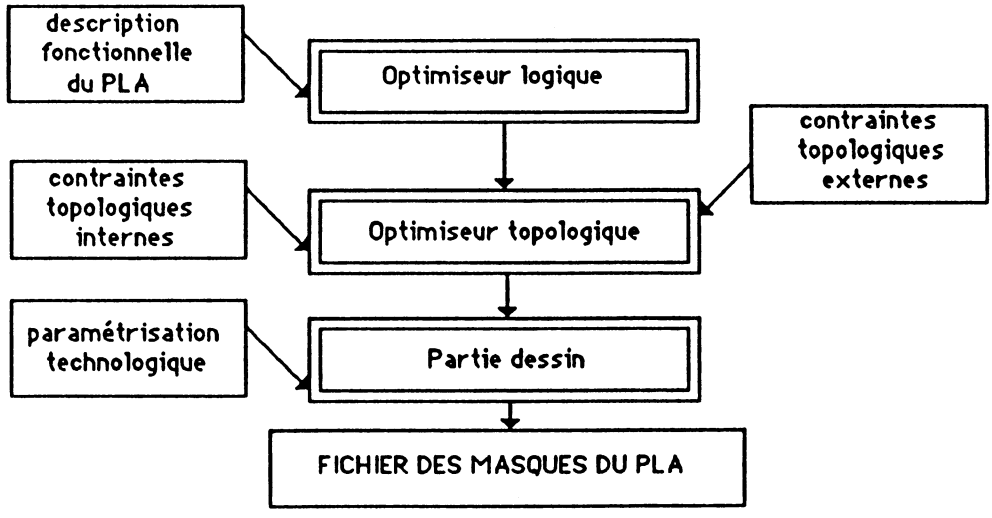


Fig - 3 - Schéma global

Le "Generator Development Tools" (GDT) (utilisé chez ESD) de"Silicon Design Labs" (SDL) se présente comme un compilateur de compilateur. GDT représente une approche nouvelle pour améliorer la productivité de la conception de circuits par la création de compilateurs de circuits complètement paramétrisés et réutilisables.C'est le premier "silicon compiler compiler" complètement intégré, indépendant du processus de fabrication et adaptable à toute technologie.

4.2 Les compilateurs d'architecture

GENESIL de chez Silicon Compiler est un compilateur d'architecture dont l'originalité est de travailler avec un minimum d'informations provenant du concepteur. Partant d'un schéma global du circuit à réaliser, le système ne se préoccupe que des détails relatifs au niveau sur lequel il travaille (fig.4) et propose plusieurs solutions à chaque fois (conception explorative).

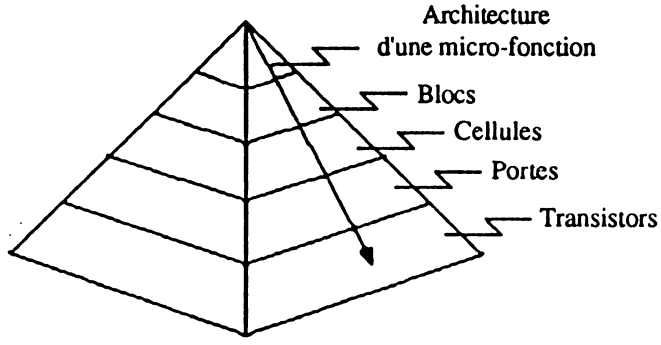


Fig - 4 - Les différents niveaux de décomposition

Le compilateur de silicium utilisé par European Silicon Structure (ES2) est un compilateur d'architecture. Il s'utilise comme un Macintosh (fenêtres, menus, souris) et il peut être adapté à n'importe quel fondeur de silicium. Inversement, un client peut concevoir ses circuits avec ses propres outils, et ceux-ci seront adaptés pour être fabriqués par ES2.

4.3 Les compilateurs de comportement

SYCO (INPG) est un compilateur de silicium spécialisé pour la génération de circuits de type microprocesseur . L'élaboration des composants est basée sur une architecture cible (architecture de référence). De ce fait, il n'est capable de compiler qu'un certain type de circuits.

5 - CONCLUSION

Le mathématicien J.Von Neumann a imaginé des "automates autoreproducteurs" s'adaptant à leur environnement sans nécessiter d'intervention humaine : une fois le programme lancé, la machine détecterait à l'aide de capteurs spécialisés, les matériaux de base servant à sa construction que des robots commandés par elle iraient chercher. Puis elle fabriquerait, à partir de ces éléments les différents composants et les assemblerait pour réaliser une nouvelle machine.
Ce scénario de science-fiction n'est pas encore devenu réalité. Mais, malgré les efforts qu'il reste à faire pour rendre les compilateurs de silicium plus performants, une étape est déjà franchie. L'ordinateur est capable de concevoir des circuits électroniques et de réaliser des masques qui serviront à fabriquer les composants de machines "filles" plus performantes que leur "mère".

Annexe 2 - *A propos de la sûreté de fonctionnement des systèmes*

La sûreté de fonctionnement est un domaine dont le champ d'application est potentiellement immense. Etudiée à l'origine pour épargner des vies humaines, il est d'ores et déjà établi que la prise en compte de ce concept permet d'éviter bien des dépenses dans les secteurs liés à l'informatique, l'électronique et l'automatique.

1 - DEFINITION

La sûreté de fonctionnement d'un système est la propriété qui permet aux utilisateurs de placer une confiance justifiée dans le service qu'il délivre.

Le service délivré par un système est son comportement tel qu'il est perçu par son (ou ses) utilisateur(s) physique ou humain qui interagit avec le système considéré.

La sûreté de fonctionnement est une notion générique qui englobe principalement :

- La fiabilité,
- La sécurité,
- La disponibilité,
- la maintenabilité.

D'autres qualificatifs peuvent être ajoutés suivants les cas (durabilité, ...) mais ils restent d'usage limité.

2 - LES PRINCIPALES MESURES DE LA SURETE DE FONCTIONNEMENT

Les mesures de la sûreté de fonctionnement d'un système permettent d'apprécier la qualité du service délivré.

2.1 Grandeurs ne tenant pas compte des réparations

♦ **Fiabilité**

Notée R(t), elle mesure la probabilité que le système ait fonctionné jusqu'à l'instant t.
C'est l'évaluation du comportement d'un système jusqu'à la défaillance (c'est à dire jusqu'à ce que le service délivré dévie du service spécifié) dans des conditions d'environnement spécifiées.
La variable associée est le MTTF (Mean Time To Failure) ou MTBF (Mean Time Between Failure) par abus de langage. C'est le temps moyen jusqu'à la défaillance du système.

♦ **Sécurité**

Notée S(t), elle mesure la probabilité que le système n'ait pas eu, jusqu'à l'instant t, de défaillance catastrophique. Ceci suppose que les défaillances dites dangereuses puissent être identifiées par rapport à la mission du système.

2.2 Grandeurs intégrant les réparations

♦ Maintenabilité

Notée M(t), elle mesure la probabilité que le système soit restauré à l'instant t sachant qu'il est tombé en panne à t = 0.
La variable associée est le MTTR (Mean Time To Repair) . C'est le temps moyen de la durée des opérations nécessaires pour remettre le système en service.

♦ Disponibilité

Notée A(t), elle mesure la probabilité qu'à l'instant t le système accomplisse sa mission.
L'indisponibilité d'un système est donnée par :
1-A = (MTTR / (MTTR + MTTF)).

Annexe 3 - *Démonstration du lemme 1*

LEMME 1 :
La condition C3 et un cas particulier de la condition C1.

Pour un système "fail-safe" qui est conçu pour respecter les propriétés de la définition D3 et qui vérifie la condition C3 nous avons:

$O^b{}_S = O^{b1}{}_S \times O^{b2}{}_S \times ... \times O^{bn}{}_S$. (1)

$O^{sb}{}_{nxn} = \{[c,d] / [c,d] = [(c^1, c^2, ..,c^n),(d^1, d^2, .., d^n)] \in O^b{}_n \times O^b{}_n :$

$d^i \neq c^i \Rightarrow d^i \in O^{bi}{}_S, \forall i \in \{1, 2, ..., n\} \}$. (2)

$I^{sa}{}_{nxn} = \{[c,d] / [c,d] = [(c^1, c^2, ..,c^n),(d^1, d^2, .., d^n)] \in I^a{}_n \times I^a{}_n :$

$d^i \neq c^i \Rightarrow d^i \in I^{ai}{}_S, \forall i \in \{1, 2, ..., n\} \}$. (3)

$O^{bi}{}_S \subseteq I^{ai}{}_S, \forall i \in \{1, 2, ..., n\}$. (4)

(1), (4) $\Rightarrow O^b{}_S = O^{b1}{}_S \times ... \times O^{bn}{}_S \subseteq I^{a1}{}_S \times ... \times I^{an}{}_S \subseteq I^a{}_S$. Alors la clause a) de la condition C1 est vérifiée.

D'autre part, si $[c, d] = [(c^1, c^2, ..., c^n), (d^1, d^2, ..., d^n)] \in I^a{}_S \times I^a{}_n$ alors :

$\forall i \in \{1, 2, ..., n\}$, $c^i \in I^{ai}{}_S \wedge \exists d^i \in \{d^1, d^2, .., d^n\} : d^i \neq c^i \wedge d^i \in I^{ai}{}_n$, et puisque nous avons $I^{ai}{}_n \cap I^{ai}{}_S = \{\emptyset\} \wedge O^{bi}{}_S \subseteq I^{ai}{}_S$ nous trouvons que $d^i \neq c^i \wedge d^i \notin O^{bi}{}_S$. Ceci combiné avec la relation (2) implique que si $[c,d] \in I^a{}_S \times I^a{}_n$ alors$[c,d] \notin O^{sb}{}_{nxn}$.

Par conséquent, $\forall [c,d] = [(c^1, c^2, ..,c^n),(d^1, d^2, .., d^n)] \in O^{sb}{}_{nxn}$:

soit $d \in I^a{}_S$, (5)

ou $[c,d] \in I^a{}_n \times I^a{}_n$. (6)

De (2), nous avons $d^i \neq c^i \Rightarrow d^i \in O^{bi}{}_S, \forall i \in \{1, 2, ..., n\}$ et puisque $O^{bi}{}_S \subseteq I^{ai}{}_S$ (de 4) alors la relation suivante est donnée $d^i \neq c^i \Rightarrow d^i \in I^{ai}{}_S$, ce qui signifie que le couple [c,d] de (6) appartient à $I^{sa}{}_{nxn}$.

Donc (5) et (6) donnent : $\forall [c,d] \in O^{sb}{}_{nxn}$ soit $d \in I^a{}_S$ ou $[c,d] \in I^{sa}{}_{nxn}$,
la clause b) de la condition C1 est vérifiée.

(C.Q.F.D.)

Annexe 4 - *Démonstration du théorème 4*

THEOREME 4 :

Soit G un système composé :
- de n sous-systèmes S1, ..., Sn "fail-safe" - de type D1- pour les ensembles de pannes respectifs F1, ..., Fn,

et :
- tel que les n sous-systèmes interconnectés respectent la condition C1,
- tel que tous les ensembles O^{si}_{nxn} correspondant aux sous-systèmes Si vérifient la propriété de transitivité,

alors G est "fail-safe" de type D1 pour l'ensemble des pannes F = F1 x ... x Fn.

Soit un sous-système A qui reçoit sur ses entrées les données des sous-systèmes B1, ..., Bk , où chacun d'eux est "fail-safe" de type D1. Alors nous pouvons considérer que les entrées du sous-système A sont les données fournies par un seul sous-système B "fail-safe" de type D1. Plus précisément,

$Y^b = Y^{b1} \times ... \times Y^{bk}$

$O^b{}_s = O^{b1}{}_s \times O^{b2}{}_s \times ... \times O^{bk}{}_s$, $O^b{}_n = Y^b - O^b{}_s$ et

$O^{sb}{}_{nxn} = \{[a,b] / [a,b] = [(a^1, ...,a^n), (b^1, ...,b^n)] \in O^b{}_n \times O^b{}_n : bi \in O^{bi}{}_s \vee [a^i, b^i] \in O^{sbi}{}_{nxn}, \forall\, i \in \{1,...,n\}\}$

Il est donc aisé de montrer que le sous-système B vérifie la définition D1 pour l'ensemble des pannes F qui affectent simultanément toute partie des sous-systèmes B1, ..., Bk.
Par conséquent, pour la suite nous supposerons que le sous-système A reçoit sur ses entrées les données d'un seul sous-système B de type D1 et que la condition C1 est respectée en ce qui concerne les vecteurs de sortie de B et les vecteurs d'entrée de A.

Du théorème 1 nous savons qu'un système composé de deux sous-systèmes A et B "fail-safe" reste "fail-safe" même si le sous-système A ou le sous-système B est défaillant. Par contre si nous considérons le cas où les deux sous-systèmes A et B sont en panne, le raisonnement est le suivant : le vecteur de sortie erroné du sous-système B qui comporte une panne fb est noté yf = B(x,fb) tandis que le vecteur de sortie correct est noté yø = B(x,ø). De la définition D1 nous pouvons affirmer que :

$yf = y\text{ø}$ ou, (1)

$yf \in O^b{}_s$ ou, (2)

$[y\text{ø}, yf] \in O^{sb}{}_{nxn}$. (3)

Comme les sorties de B sont appliquées sur les entrées de A qui comporte une panne fa :

- dans le cas (1) le sous-système A qui comporte une panne fa reçoit du sous-système B des vecteurs d'entrée corrects. Alors, d'après le théorème 1 le sous-système A délivre des vecteurs de sortie qui respectent les restrictions imposées par la définition D1.

- dans le cas (2), $yf \in O^b{}_s$ et, de par la condition C1, $O^b{}_s \subseteq I^a{}_s$, alors $yf \in I^a{}_s$ soit :

$A(yf,\text{ø}) \in O^a{}_s$. (4)

puique le sous système A vérifie la définition D1:

- $A(yf,fa) = A(yf,\emptyset)$ et de (4) $\Rightarrow A(yf,fa) \in O^a{}_S$, ou

- $A(yf,fa) \in O^a{}_S$, ou

- $[A(yf,\emptyset), A(yf,fa)] \in O^{sa}{}_{nxn}$, mais d'après (4) ce cas n'est pas considéré car si un couple $[a,b] \in O^{sa}{}_{nxn}$ alors $a \in O^a{}_n \Lambda\ b \in O^a{}_n$.

Donc dans le cas (2) les vecteurs de sortie du sous-système A appartiennent à $O^a{}_S$ et la définition D1 est vérifiée.

- pour le cas (3), $[y\emptyset,yf] \in O^{sb}{}_{nxn}$ et d'après la condition C1 :

soit $yf \in I^a{}_S$, ce qui signifie que $A(yf,\emptyset) \in O^a{}_S$, ou (5)

$[y\emptyset,yf] \in I^{sa}{}_{nxn}$, c'est à dire $[A(y\emptyset, \emptyset), A(yf,\emptyset)] \in O^{sa}{}_{nxn}$. (6)

puisque A vérifie la définition D1, nous avons un des trois cas suivants :

- $A(yf,fa) = A(yf,\emptyset)$, combiné avec (6) cela donne $[A(y\emptyset,\emptyset), A(yf,fa)] \in O^{sa}{}_{nxn}$ et la définition D1 est vérifiée.
- $A(yf, fa) \in O^a{}_S$ et la définition D1 est vérifiée.
- $[A(yf,\emptyset), A(yf, fa)] \in O^{sa}{}_{nxn}$, combiné avec (6) et la propriété de transitivité, cela donne $[A(y\emptyset,\emptyset), A(yf,fa)] \in O^{sa}{}_{nxn}$. Alors la définition D1 est vérifiée.

En remplaçant $y\emptyset$ et yb dans les trois situations précédentes nous obtenons :

- $A(B(x,fb),fa) = A(B(x, \emptyset), \emptyset)$.
- $A(B(x,fb),fa) \in O^a{}_S$.
- $[A(B(x, \emptyset), \emptyset), A(B(x,fb),fa)] \in O^{sa}{}_{nxn}$.

Ce qui signifie que le système composé des deux sous-systèmes A et B est "fail-safe" (de type D1) vis à vis des pannes $fa \in Fa$ et $fb \in Fb$ qui affectent simultanément A et B.
Si le sous-système B est lui-même composé de plusieurs sous-systèmes B1, ..., Bk, alors l'ensemble Fb comprend les pannes qui peuvent apparaître simultanément dans n'importe quel groupe des sous-systèmes B1, ..., Bk qui composent B, donc le système composé de A, B1, ..., Bk est "fail-safe" pour l'ensemble des pannes qui peuvent apparaître simultanément dans tous (ou en partie) ces sous-sytèmes.
Ce raisonnement reste valable quel que soit le nombre de couches de sous-systèmes qui composent le système global .

(C.Q.F.D.)

Annexe 5 - *Démonstration du théorème 3*

THEOREME 3 :

Soit G un système composé :

- de n sous-systèmes S1, ..., Sn "fail-safe" - de type D2 ou de type D3 - pour les ensembles de pannes respectifs F1, ..., Fn,

et :

- tel que les n sous-systèmes interconnectés respectent les conditions C2 ou C3 suivant la définition à laquelle ils répondent,

alors G est "fail-safe" de type D2 ou de typpe D3 pour l'ensemble des pannes F = F1 x ... x Fn.

Le théorème 3 peut être démontré en remarquant qu'il s'agit d'un cas particulier du thèorème 4.

Puisque la définition D2 est un cas particulier de la définition D1 avec $O^s_{nxn} = \{\emptyset\}$, il est évident que la condition C2 est un cas particulier de la condition C1 et la propriété de transistivité est vérifiée pour tous les systèmes qui répondent à la définition D2. D'autre part, la définition D3 est aussi un cas particulier de la définition D1 et d'àprès le lemme 1 la condition C3 est un cas particulier de la condition C1. Alors, les systèmes qui sont décrits par la définition D3 vérifient aussi la propriété de transitivité. En effet :

nous avons $O^s_{nxn} = \{[a,b] / [a,b] = [(a^1, ..., a^n), (b^1, ..., b^n)] \in O_n \times O_n : b^i \neq a^i \Rightarrow b^i \in O^i_s , \forall i \in \{1,2, ..., n\}\}$.

Si $[a, b] \in O^s_{nxn} \wedge [b,c] \in O^s_{nxn}$ alors :

- $b^i \neq a^i \Rightarrow b^i \in O^i_s \ \forall i \in \{1,2, ..., n\}$. (1)
- $c^j \neq b^j \Rightarrow c^j \in O^j_s \ \forall_j \in \{1,2, ..., n\}$. (2)
- $[a,c] \in O_n \times O_n$. (3)

d'autre par, $\forall k \in \{1,2, ..., n\}$, si $c^k \neq a^k$ alors :

- soit $c^k \neq b^k$, ou (4)
- $c^k = b^k \wedge b^k \neq a^k$ (5)

(4) et (2) $\Rightarrow c^k \in O^k_s$.

(5)et (1) $\Rightarrow c^k = b^k \wedge b^k \in O^k_s \Rightarrow c^k \in O^k_s$.

soit $\forall k \in \{1,2, ..., n\}$ si $c^k \neq a^k \Rightarrow c^k \in O^k_s$ (6)

(3) et (6) $\Rightarrow [a,c] \in O^s_{nxn}$, la propriété de transitivité est donc vérifiée. En conséquence, le théorème 3 est bien un cas particulier du théorème 4.

(C.Q.F.D.)

Annexe 6 - *Etude qualitative de la sûreté de fonctionnement d'une interface "fail-safe"*

1 - INTRODUCTION

L'étude concerne essentiellement la composante "sécurité" de la sûreté de fonctionnement d'un système composé d'une logique de commande "self-checking" et d'une interface "fail-safe".
Ce que nous voulons mettre en évidence, c'est le rôle joué par une interface "strongly fail-safe" du type de celles présentées au troisième chapitre.

L'objectif de cette étude est de modéliser le système global, donné en figure 1, à l'aide **d'un arbre de défaillance** et d'effectuer une analyse qualitative à partir **des coupes minimales** de cet arbre de défaillance.

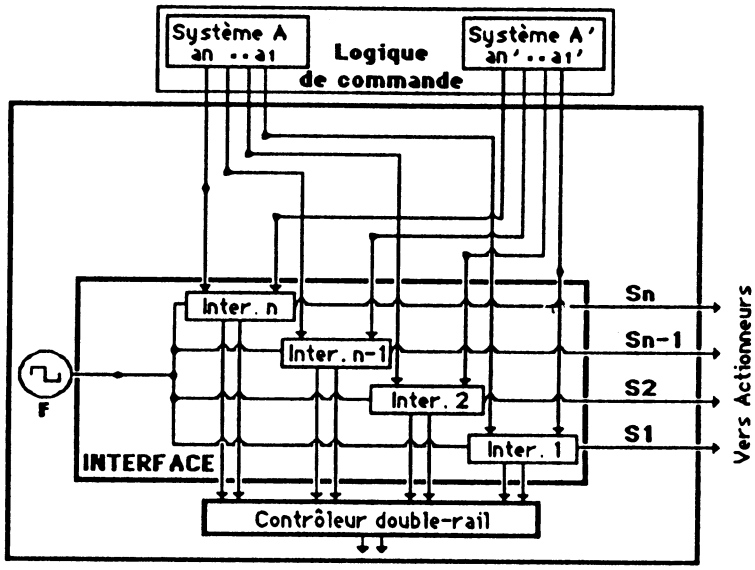


Fig - 1 - Système global étudié

L'événement redouté est le suivant :

- Sortie Si dans un état non sûr, c'est-à-dire commutant à une fréquence F,

sachant qu'en hypothèse, nous considérons les ordres qui arrivent à l'entrée de l'interface comme définissant la sortie dans un état sûr (tension continue).

2 - ANALYSE FONCTIONNELLE ET AMDE

A partir de l'analyse fonctionnelle du système qui permet de définir exhaustivement :

- les fonctions assurées par le système,
- les équipements participant à la réalisation de ces fonctions,

nous identifions, pour chaque fonctionnalité, tous les modes de défaillance, nous recençons toutes les causes possibles correspondantes et nous recherchons pour chacune d'elles les effets sur la mission du système ("Analyse des modes de défaillances, de leurs effets (et de leur criticité)" (AMDE(C)).

2.1 Analyse fonctionnelle

Le système étudié se compose d'une logique de commande qui est réalisée par deux systèmes dupliqués et d'une interface qui transpose des ordres codés en binaire (ai et ai') en signaux codés en fréquence. Les diagrammes suivants décrivent l'architecture du système dans les deux positions du profil de vie :

- fonctionnement normal (fig.2) :

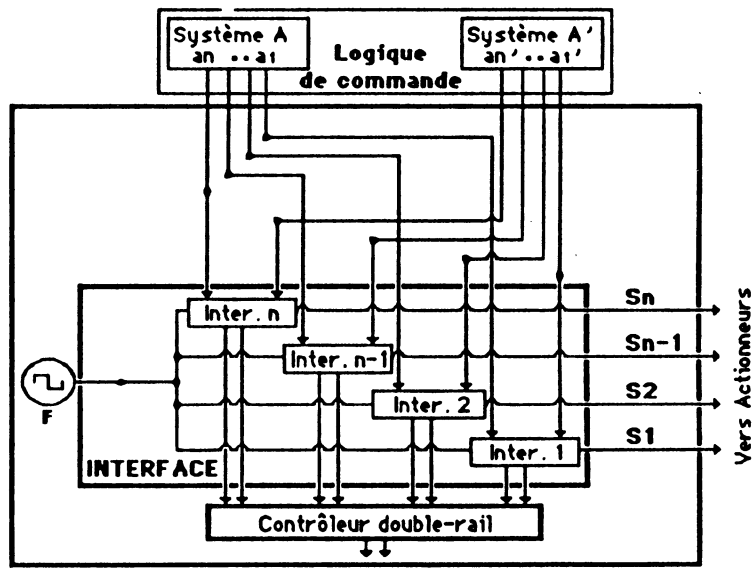


Fig - 2 - Phase normal de fonctionnement

ai = ai'= "1" : Si $\rightarrow$ F

ou ai = ai' = "0" : Si $\rightarrow$ "0",

le message du contrôleur est correctement codé en double-rail.

- procédure de test (fig 2) :

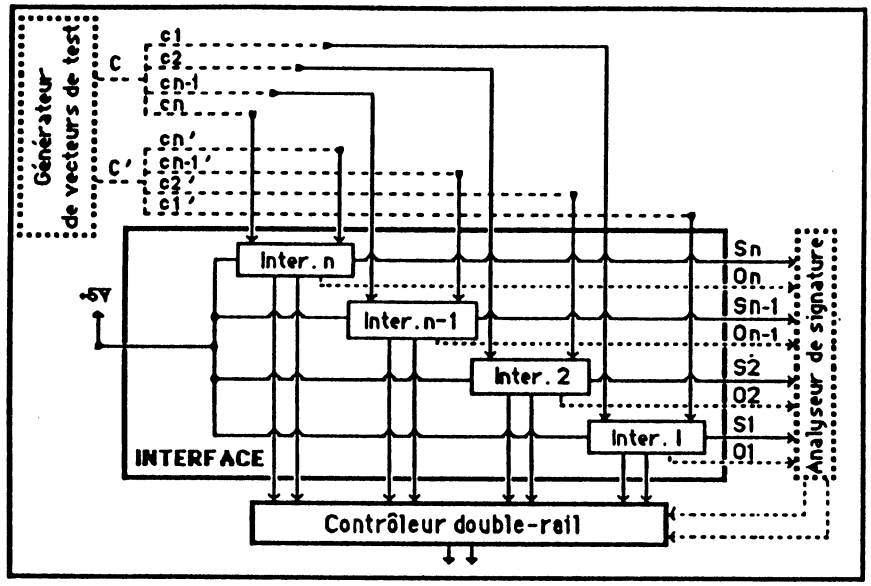
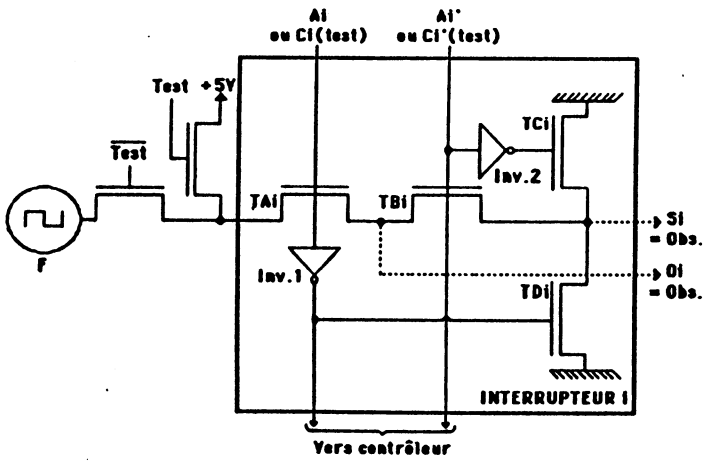


Fig - 3 - Procédure de test

La logique de commande est déconnectée de l'interface, le générateur de vecteurs de test injecte une séquence déterminée pour tester les interrupteurs de sécurité. Le bon déroulement de la séquence est vérifiée en permanence par le contrôleur double-rail. A la fin de la période de test hors-ligne, ce même contrôleur et l'analyseur de signature vérifient les réponses des blocs sous test.

Il est aussi nécessaire d'effectuer l'analyse fonctionnelle sur le sous-système "interrupteur" (fig.4), ce dernier représentant le sous-système le plus pertinent vis-à-vis de la sécurité du système global.



Ci	Ci'	Oi	Si	Panne détectée
1	0	1	0	Initialisation
0	0	1	0	TBi S-On
0	1	0	0	TBi S-Open
0	0	0	0	TAi S-On
1	0	1	0	TAi S-Open
1	1	1	1	TCi ou TDi S-On
1	0	1	0	TCi S-Open
1	1	1	1	
0	1	0	0	TCi S-Open
0	0	0	0	

Panne latente dangereuse détectée

Fig - 4 - Interrupteur de sécurité et séquence de test

2.2 Analyse des modes de défaillances et de leurs effets

Une première analyse rapide a été réalisée sur les sous-ensembles fonctionnels suivants :

- contrôleur double-rail,
- générateur de test,
- interrupteur,
- analyseur de signature,
- générateur de fréquence.

(Les résultats ne sont pas donnés dans cette annexe)

Les tableaux des pages 134 à 136 sont le résultat d'une deuxième analyse, de niveau plus fin, qui a été menée sur les composants qui constituent l'interrupteur .

3 - ETUDE DE SECURITE

A ce stade de l'étude, nous recherchons les combinaisons de pannes significatives (arbre de défaillance) sur un interrupteur compte tenu de l'évènement indésirable (racine de l'arbre) défini au paragraphe 1.

L'arbre de défaillance d'un interrupteur du système étudié est donné dans les figures 5 et 6.

A partir de cet arbre, nous nous intéressons à l'ensemble des coupes minimales d'événements (fig 7), et plus précisément à la longueur (ou ordre) de ces coupes. Néanmoins, cette notion qualitative est à pondérer par le taux de défaillance des événements de base constituant ces coupes.

4 - CONCLUSION

Pour l'ensemble des coupes minimales données précédemment, nous observons que l'ordre obtenu est au minimum un ordre 3, c'est-à-dire qu'il faut l'occurence de 3 événements de base pour que l'événement sommet se réalise. Ceci prouve que qualitativement, l'interface a un haut niveau de sécurité vis-à-vis de l'occurence de l'événement redouté.

Au niveau quantitatif, nous pouvons faire un calcul rapide et simplifié en prenant les hypothèses suivantes :

- intervalle entre tests : 1 mn.
- taux de défaillance des événements (hypothèses pessimistes) :

Pour un interrupteur i :

TAIF : $10^{-5}\ h^{-1}$
TBIF : $10^{-5}\ h^{-1}$
M5V : $10^{-5}\ h^{-1}$

$P(Ev^t\ Sommet) = P(TAIF) * P(TBIF) * P(M5V)$

avec :

$P(TAIF) = 1 - e^{-\lambda t} = 1.66 * 10^{-7}$

$P(TBIF) = 1.66 * 10^{-7}$

$P(M5V) = 1.66 * 10^{-7}$

D'où $P(Ev^t\ Sommet) = 4.57 * 10^{-21}$

En prenant comme hypothèse que le résultat des 2 autres coupes minimales est identique, nous obtenons une probabilité pour l'événement indésiré :

$P(Evt\ Redouté) \approx 1.3 * 10^{-20}$

NB : ce résultat est indépendant du temps de mission si les taux de défaillance des composants sont supposés constants.

Pour l'interface composée de n interrupteurs :

Inf [P(def. due au test), P(def sur un interrupteur i)] = P1
P (def. due au test ou d'une defaillance sur un des interrupteur) = P2

$P1 < P(Ev^t\ Redouté) < P2$

$4.5\ 10^{-21} < P(Ev^t\ Redouté) < 4.5\ 10^{-21} + n * (2*4.5\ 10^{-21})$

$4.5\ 10^{-21} < P(Ev^t\ Redouté) < (2n+1)* 4.5\ 10^{-21}$

Cette quantification rapide démontre que l'interface de sécurité étudiée répond parfaitement aux objectifs du critère "sécurité".

ETUDE :

SYSTEME : Interface

SOUS-S : Interrupteur

A M D E C

ANALYSE INDUCTIVE DES RISQUES

ELEMENT OU EQUIPEMENT COMPOSANT	FONCTIONS ETATS	MODES DE DEFAILLANCE	CAUSES POSSIBLES INTERNES EXTERNES	EFFETS SUR L'ELEMENT SUIVANT SUR LE SYSTEME	MOYENS DE DETECTION	FREQUENCE GRAVITE CRITICITE	ACTIONS QUALITE	OBSERVATIONS
TAi	assure le passage ou non de F → fct interrupteur	Pas de fonction collage à "1"		etat sûr ou etat commutant, voulu sur Si	Test H.L			Panne dangereuse à détecter impérativement
		Pas de fonction collage à "0"		etat sûr sur Si	Test H.L			
TBi	idem	//		//	//			//

ETUDE :

SYSTEME : Interface

SOUS-S : Interrupteur

AMDEC

ANALYSE INDUCTIVE DES RISQUES

ELEMENT OU EQUIPEMENT. CONSTITUE	FONCTIONS ETATS	MODES DE DEFAILLANCE	CAUSES POSSIBLES INTERNES EXTERNES	EFFETS SUR L'ENSEMBLE DU SYSTEME	MOYENS DE DETECTION	FREQUENCE GRAVITE CRITICITE	ACTIONS QUALITE	OBSERVATIONS
INV 1	inversion de l'ordre ai	Pas de fonction collage à "1"	court-circuit	TDi tirs passent état sûr sur Si (niveau bas de tension)	Tstr en ligne (entr double rail) ou Tstr hors-ligne			TDi dimensionné pour imposer le "0" sur Si dans le cas où ai = ai' = "1"
		Pas de fonction collage à "0"	"	TDi tirs arrêt le signal sur Si et celui désiré	Tstr en ligne ou Tstr hors-ligne			
TDi		Pas de fonction collage à "1"	"	TDi tirs passent	Tstr hors-ligne			
		Pas de fonction collage à "0"	"	TDi tirs arrêt	Tstr hors-ligne			panne dangereuse à détecter impérativement.

A M D E C

ETUDE :

SYSTEME : Interface

SOUS-S : Interrupteur

ANALYSE INDUCTIVE DES RISQUES

ELEMENT DU EQUIPEMENT. COMPOSANTS	FONCTIONS ETATS	MODES DE DEFAILLANCE	CAUSES POSSIBLES INTERNES EXTERNES	EFFETS SUR L'ENSEMBLE DU SYSTEME	MOYENS DE DETECTION	FREQUENCE GRAVITE CRITICITE	ACTIONS QUALITE	OBSERVATIONS
INV 2	inversion de l'ordre ai'	Pas de fonction collage à "1"	court-circuit	TCi fonctionnel dès sur Si (niveau bas de Tension)	Test hors-ligne			
		Pas de fonction collage à "0"	"	TCi fonctionnel si signal sur Si est celui désiré	Test hors-ligne			
TCi		idem	idem	idem	idem			la panne TCi avant doit être impérativement détectée.

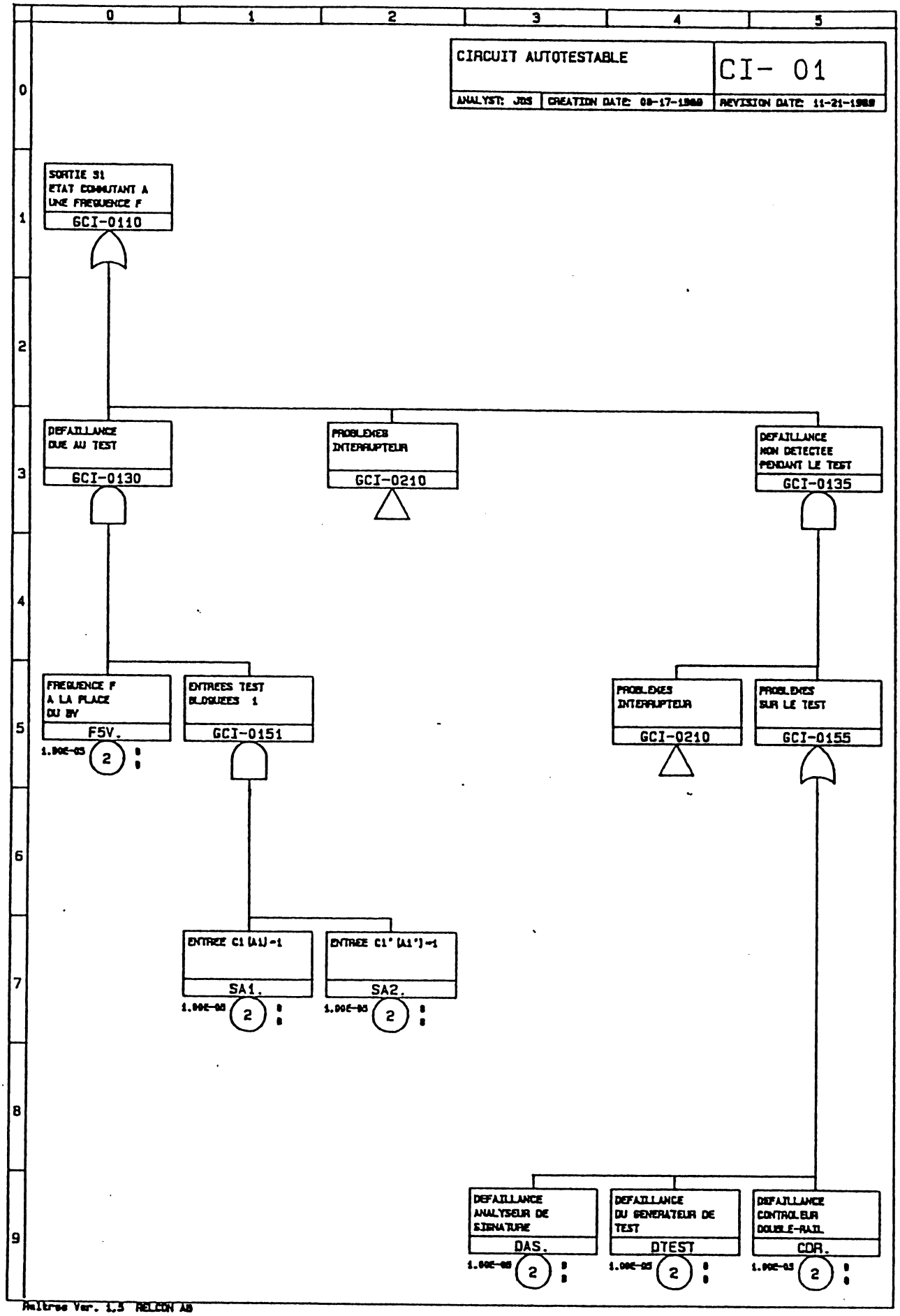


Fig - 5 - Arbre de défaillance d'un interrupteur i

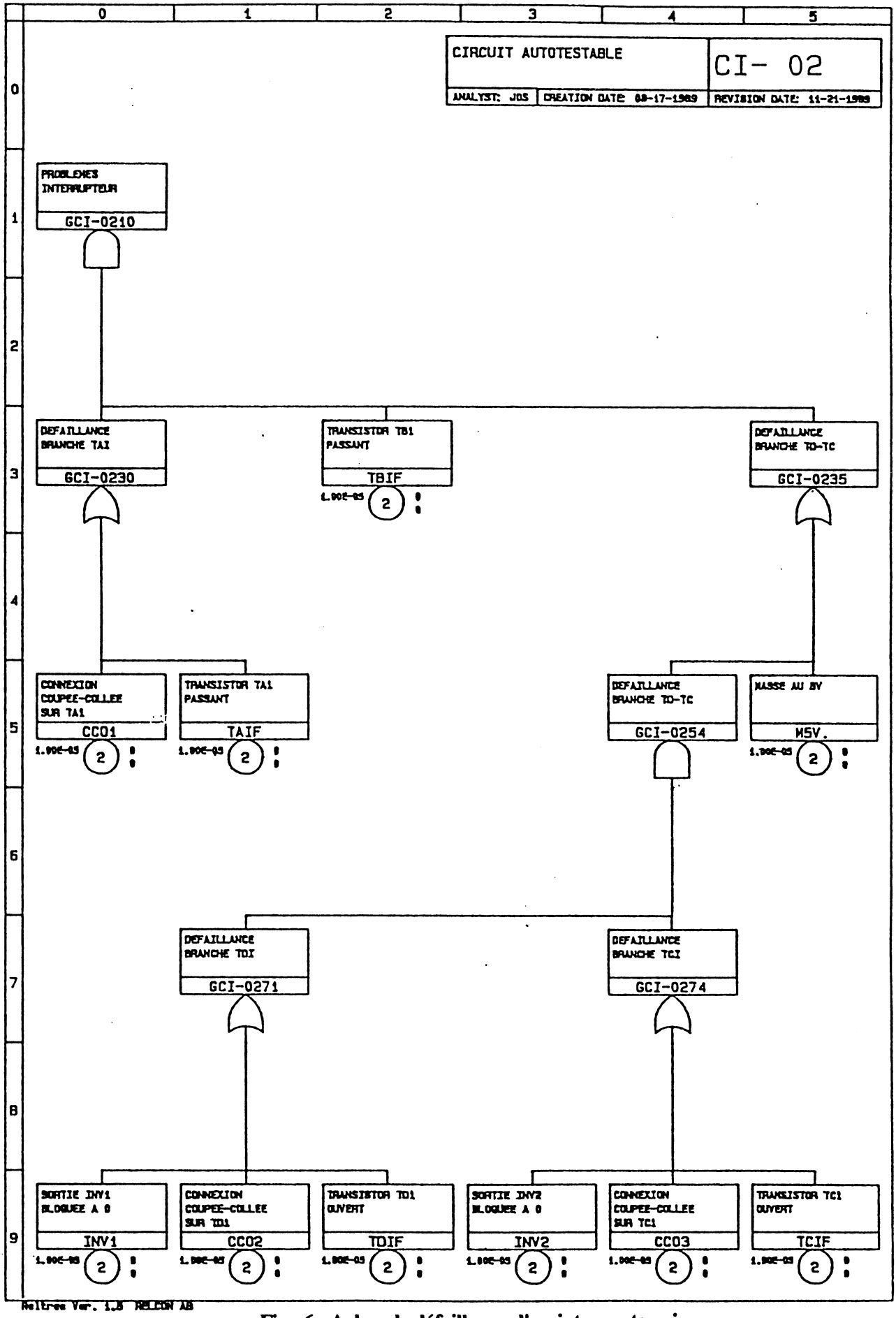


Fig - 6 - Arbre de défaillance d'un interrupteur i

***** CUTLIST.CSL 11-21-1989 *****

Fault tree: CI-TEST Serial no.= 6

CIRCUIT AUTOTESTABLE

Top event: GCI-0110

*** CUTOFF BY ORDER USED ***
*** NO NUMERICAL RESULTS CALCULATED ***

Cutoff value used = 5
No cut sets removed by cutoff

Number of Boolean Indicated Cut Sets = 81

Number of MCS evaluated = 21
Number of MCS listed = 21

MINIMAL CUT SETS SORTED BY CUT SET ORDER

1.	M5V.	TBIF	CCO1	
2.	F5V.	SA1.	SA2.	
3.	TAIF	TBIF	M5V.	
4.	CCO2	TBIF	TAIF	CCO3
5.	CCO2	TBIF	TAIF	TCIF
6.	INV2	TAIF	TBIF	CCO2
7.	TDIF	TBIF	TAIF	CCO3
8.	TDIF	TBIF	TAIF	TCIF
9.	INV2	TAIF	TBIF	TDIF
10.	CCO3	TAIF	TBIF	INV1
11.	TCIF	TAIF	TBIF	INV1
12.	INV1	TBIF	TAIF	INV2
13.	CCO1	TBIF	INV1	INV2
14.	CCO1	TBIF	CCO2	CCO3
15.	CCO1	TBIF	CCO2	TCIF
16.	INV2	CCO2	TBIF	CCO1
17.	CCO1	TBIF	TDIF	CCO3
18.	CCO1	TBIF	TDIF	TCIF
19.	INV2	TDIF	TBIF	CCO1
20.	CCO3	INV1	TBIF	CCO1
21.	TCIF	INV1	TBIF	CCO1

Fig - 7 - Coupes minimales

DATA BASE: CI-TEST ANALYST: JOS CDATE: 08-17-1989 RDATE: 11-21-1989

CIRCUIT AUTOTESTABLE

CODE	TYPE	FAIL.RATE	MTTR	TEST INT.	TTFT				
SA1.....	2	1.00E-05		0	0	/ENTREE Ci(Ai)=1			/
SA2.....	2	1.00E-05		0	0	/ENTREE Ci'(Ai')=1			/
F5V.....	2	1.00E-05		0	0	/FREQUENCE F	A LA PLACE	DU 5V	/
TBIF.....	2	1.00E-05		0	0	/TRANSISTOR TBi	PASSANT .		/
DAS.....	2	1.00E-05		0	0	/DEFAILLANCE	ANALYSEUR DE	SIGNATURE	/
DTES.....	2	1.00E-05		0	0	/DEFAILLANCE	DU GENERATEUR DE	TEST	/
CDR.....	2	1.00E-05		0	0	/DEFAILLANCE	CONTROLEUR	DOUBLE-RAIL	/
INV1.....	2	1.00E-05		0	0	/SORTIE INV1	BLOQUEE A 0		/
INV2.....	2	1.00E-05		0	0	/SORTIE INV2	BLOQUEE A 0		/
CCO1.....	2	1.00E-05		0	0	/CONNEXION	COUPEE-COLLEE	SUR TAi	/
CCO2.....	2	1.00E-05		0	0	/CONNEXION	COUPEE-COLLEE	SUR TDi	/
CCO3.....	2	1.00E-05		0	0	/CONNEXION	COUPEE-COLLEE	SUR TCi	/
TAIF.....	2	1.00E-05		0	0	/TRANSISTOR TAi	PASSANT		/
TDIF.....	2	1.00E-05		0	0	/TRANSISTOR TDi	OUVERT		/
TCIF.....	2	1.00E-05		0	0	/TRANSISTOR TCi	OUVERT		/
M5V.....	2	1.00E-05		0	0	/MASSE AU 5V			/

Fig - 8 - Taux de défaillance

INPG département des études doctorales

AUTORISATION de SOUTENANCE

VU les dispositions de l'Arrêté du 23 novembre 1988 relatif aux Etudes doctorales

VU les rapports de présentation de

- Monsieur DAVID René
- Monsieur GEFFROY Jean-Claude

Monsieur NORAZ Serge

est autorisé(e) à présenter une thèse en soutenance en vue de l'obtention du diplôme de DOCTEUR de L'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE, spécialité "Microélectronique"

Fait à Grenoble, le 23 Novembre 1989

Pour le Président de l'I.N.P.G.
et par délégation,
le Vice-Président
P. VENNEREAU