Test en ligne des systèmes digitaux linéaires

A. Abdelhay

INSTITUT **N**ATIONAL **P**OLYTECHNIQUE DE **G**RENOBLE

N° **attribué par la bibliothèque**
/ / / / / / / / / /

THESE

Pour obtenir le grade de

DOCTEUR DE L'INPG

***Spécialité* : MICRO-ELECTRONIQUE**

Préparée au Laboratoire TIMA
Dans le cadre de l'Ecole Doctorale ''ELECTRONIQUE, ELECTROTECHNIQUE, AUTOMATIQUE, TELECOMMUNICATIONS, SIGNAL''

Présentée et soutenue publiquement

Par

Ahmad ABDELHAY

Le 20 avril 2001

Titre :

Test en Ligne des Systèmes Digitaux Linéaires

Directeurs de thèse :

Michael **NICOLAIDIS** (TIMA-CNRS)
Emmanuel **SIMEU** (TIMA-UJF)

Jury :

Mr.	**R. DAVID**	Directeur de recherche EMERITE, CNRS	Président
Mr.	**A. DANDACHE**	Professeur, Université de METZ	Rapporteur
Mr.	**B. ROUZEYRE**	Professeur, Université de MONTPELLIER II	Rapporteur
Mr.	**M. NICOLAIDIS**	Directeur de recherche, CNRS	Directeur de thèse
Mr.	**E. SIMEU**	Maître de conférences, UJF	Co-encadrant
Mme	**M. L. FLOTTES**	Chargée de recherche, CNRS	Examinateur

Remerciements

Cette thèse a été préparée au sein de l'équipe de Sûreté de fonctionnement des Systèmes Intégrés (**RIS**) au laboratoire de Technique de l'Informatique et de la Microélectronique pour l'Architecture d'ordinateurs (**TIMA**).

Je voudrai tout d'abord exprimer ma gratitude à mon responsable de thèse Monsieur Emmanuel SIMEU, Maître de conférences à l'université Joseph Fourier, qui m'a encadré et m'a soutenu tout au long de ma thèse. Ses remarques et ses conseils m'ont été d'une aide précieuse.

Un grand merci à Monsieur Michael NICOLAIDIS mon directeur de thèse, Directeur de recherches CNRS, pour l'accueil qu'il m'a offert dans son équipe, pour son encouragement continu et son soutien.

Je suis très reconnaissant à Monsieur Abbas DANDACHE, Professeur à l'université de Metz, ainsi qu'à Monsieur Bruno ROUZEYRE, Professeur à l'université de Montpellier II, qui m'ont fait l'honneur d'accepter d'être rapporteurs et membre du jury de cette thèse. Merci pour la lecture soigneuse du manuscrit et les commentaires très pertinents et très enrichissants.

Je tiens à remercier Monsieur René DAVID, Directeur de recherches Emerite CNRS, qui m'a fait l'honneur d'accepter de présider le jury de cette thèse. Merci pour la lecture attentive du manuscrit et l'encouragement.

Je tiens à remercier Madame Marie-Lise FLOTTES, Chargée de recherche CNRS, pour avoir accepté d'être examinateur et membre du jury. Merci pour la lecture attentive du manuscrit et les commentaires pertinents et très enrichissants

Je remercie infiniment ma famille, mes amis ainsi que tous ceux qui ont contribué à ce travail sans attendre mes remerciements.

Résumé

Les systèmes digitaux linéaires représentent une classe importante de circuits utilisés dans plusieurs applications critiques militaires, nucléaires, spatiales …etc. C'est pourquoi le problème du test en ligne des systèmes digitaux linéaires est très important car une erreur de données, pendant la période de fonctionnement normal, peut entraîner de graves conséquences. Les recherches développées pour le test des systèmes digitaux sont orientées vers la détection de fautes des structures spécialisées tandis que le problème de détection de fautes en ligne des systèmes digitaux linéaires n'est pas visé et adressé directement.

L'objet de ce travail est d'étudier une nouvelle approche de conception et d'intégration des détecteurs de défauts en ligne pour les systèmes digitaux linéaires. Afin d'atteindre l'objectif, une nouvelle méthodologie qui permet de concevoir des circuits de test concurrent (détecteurs de défauts ou de fautes en ligne) pour les systèmes digitaux linéaires a été développée. La méthode proposée de détection de fautes est basée sur l'exploitation de la redondance analytique décrivant les relations entre l'historique des signaux d'entrées et de sorties du système sous test. Les algorithmes développés garantissent aussi une détection robuste de fautes, c'est-à-dire une sensibilité maximale pour les fautes et minimale pour le bruit (le bruit généré à l'intérieur des systèmes n'est pas considéré comme fautes et doit être toléré). Le circuit de test concurrent d'un système digital linéaire est relié aux entrées et aux sorties du système pour calculer la relation de la redondance. Il sert comme indicateur d'erreur (fautes) car sa sortie est zéro en l'absence de fautes (les mesures disponibles des signaux d'entrées/ sorties du système correspondent au comportement nominal) et différente de zéro en présence de fautes. Les circuits robustes et optimaux requis pour l'implémentation des testeurs (circuits de test concurrent) en ligne restent très raisonnables en taille. A l'inverse des autres techniques de test traitant des cas (architectures) spécifiques des systèmes et posant, souvent, des conditions sur les conceptions à tester, la méthode proposée est générale et applicable à tout système digital pourvu qu'il soit linéaire.

Les paramètres ainsi que la description VHDL du circuit robuste et optimal de test, pour chaque système digital linéaire, peuvent être générés automatiquement par un outil dédié développé au cours de la thèse. La génération se fait soit à partir de la description VHDL du système à tester soit à partir des paramètres du système fournis directement à l'outil.

TABLE DES MATIERES

Introduction générale

Les prédictions sur les tendances des technologies des circuits intégrés pour la prochaine décennie prévoient une réduction de la géométrie des transistors qui devrait atteindre le niveau de 0.07-mic et 7 niveaux de métallisation pour l'année 2010. Cette tendance permettra l'intégration de systèmes complexes comprenant jusqu'à un milliard de transistors. La maîtrise de la complexité des nouvelles générations de systèmes pose déjà un problème aujourd'hui, et a donné naissance au concept de «system-on-chip» utilisant des modules enterrés profitant des fonctions préexistantes au sein d'une compagnie ou provenant éventuellement de plusieurs compagnies. Le problème majeur concernant les nouvelles générations de systèmes est la testabilité. En général, un système est regardé comme une collection des modules de hardware coopérants, et son test est un test dans lequel tous les modules sont compris. Le problème de test donc est un problème d'identification des modules (circuits) qui ne satisfont pas à leurs propres spécifications fonctionnelles désignées par le concepteur. Un circuit ne peut accomplir sa fonction en raison de plusieurs facteurs, par exemple fautes permanentes (fautes physiques causées par les processus de fabrication, vieillissement, . . . etc.) et fautes temporaires (fautes causées par l'environnement). Un test de conception a pour objectif de :

1- Avant la fabrication, s'assurer que le comportement du circuit satisfait aux spécifications fonctionnelles désignées par le concepteur.

2- Après la fabrication, détecter le circuit en panne, soit pendant la période de repos (test hors ligne) soit pendant le cycle de fonctionnement normal (test en ligne).

Aujourd'hui, tester les circuits après fabrication est le problème le plus important auquel tant le concepteur que l'ingénieur de test doivent faire face. Il est considéré potentiellement comme l'obstacle majeur que doit affronter l'exploitation complète des avantages des nouvelles technologies du VLSI. La testabilité des circuits complexes pose un problème important depuis déjà un certain temps. Il a amené à l'adoption des techniques de test hors ligne telles que AD-HOC, full-scan, boundry-scan et Built-in-self-test (BIST).

Certaines applications qui utilisent des systèmes électroniques ne peuvent tolérer l'interruption de leur fonctionnement même pas quelques secondes par année. Ces applications nécessitent donc de tester les systèmes au cours de leur fonctionnement normal.

Dans ces conditions, il devient pratiquement impossible de tester les systèmes en utilisant les techniques de test hors ligne. Il devient donc de plus en plus évident que les systèmes du futur devront inclure des moyens d'auto-test, donnant un avantage évident aux techniques de test en ligne telles que les techniques de redondance du matériel, de redondance de l'information (techniques de codage) et de redondance du temps.

Le test en ligne signifie que le circuit (système) est testé pendant le fonctionnement normal qui peut comprendre une phase de repos («idle» en anglais) et une phase de fonctionnement. Ce test peut être en effet un test concurrent ou non-concurrent. Le test non-concurrent signifie que le circuit est testé pendant la phase de repos tandis que le test concurrent signifie que le circuit est testé pendant la phase de fonctionnement.

Les systèmes digitaux linéaires qui sont utilisés dans plusieurs domaines critiques d'application représentent une classe importante de circuits. Par conséquent, le problème de détection de fautes en ligne dans ce type de systèmes est très important car une erreur de données pendant la période de fonctionnement normal peut produire de graves effets. Les recherches effectuées dans le domaine de test en ligne des systèmes digitaux [ChAb93, ChCh99, JoAb86, JoAb88, HuAb84, NaAb90, ReBa90, SeRa95] sont orientées vers la détection de fautes des structures spécialisées tandis que le problème de test en ligne des systèmes digitaux linéaires n'est pas visé et adressé directement.

Le but de notre étude est donc d'étudier une nouvelle approche aboutissant à des conceptions destinées à détecter en ligne des fautes des systèmes digitaux linéaires. La méthode proposée de détection de fautes est basée sur la technique de la redondance analytique décrivant les relations entre l'historique des signaux d'entrée et de sortie du système à tester. Cette méthode garantissant une détection concurrente de fautes est générale et applicable à tout système digital pourvu qu'il soit linéaire.

Avant d'aborder la méthode proposée, nous parlerons dans le premier chapitre des structures de systèmes digitaux linéaires, de leurs représentations et de leurs implémentations.

Le deuxième chapitre expose les techniques actuelles de test hors ligne et de test en ligne des circuits digitaux.

Dans le troisième chapitre nous développons les principes de la méthode proposée de test en ligne des systèmes digitaux linéaires, nous exposons le concept de la redondance analytique et expliquons son utilisation pour la conception de détecteurs de défauts en ligne.

La conception robuste et optimale des détecteurs de défauts est traitée dans le quatrième chapitre, des exemples d'illustration et des résultats expérimentaux sont donnés.

Le cinquième chapitre est consacré à la génération automatique des codes **VHDL** des circuits (détecteurs) robustes et optimaux de test.

Le sixième chapitre conclut le travail et donne des perspectives.

La figure suivante résume les techniques utilisées de test des systèmes digitaux et indique le domaine dans lequel on travaille.

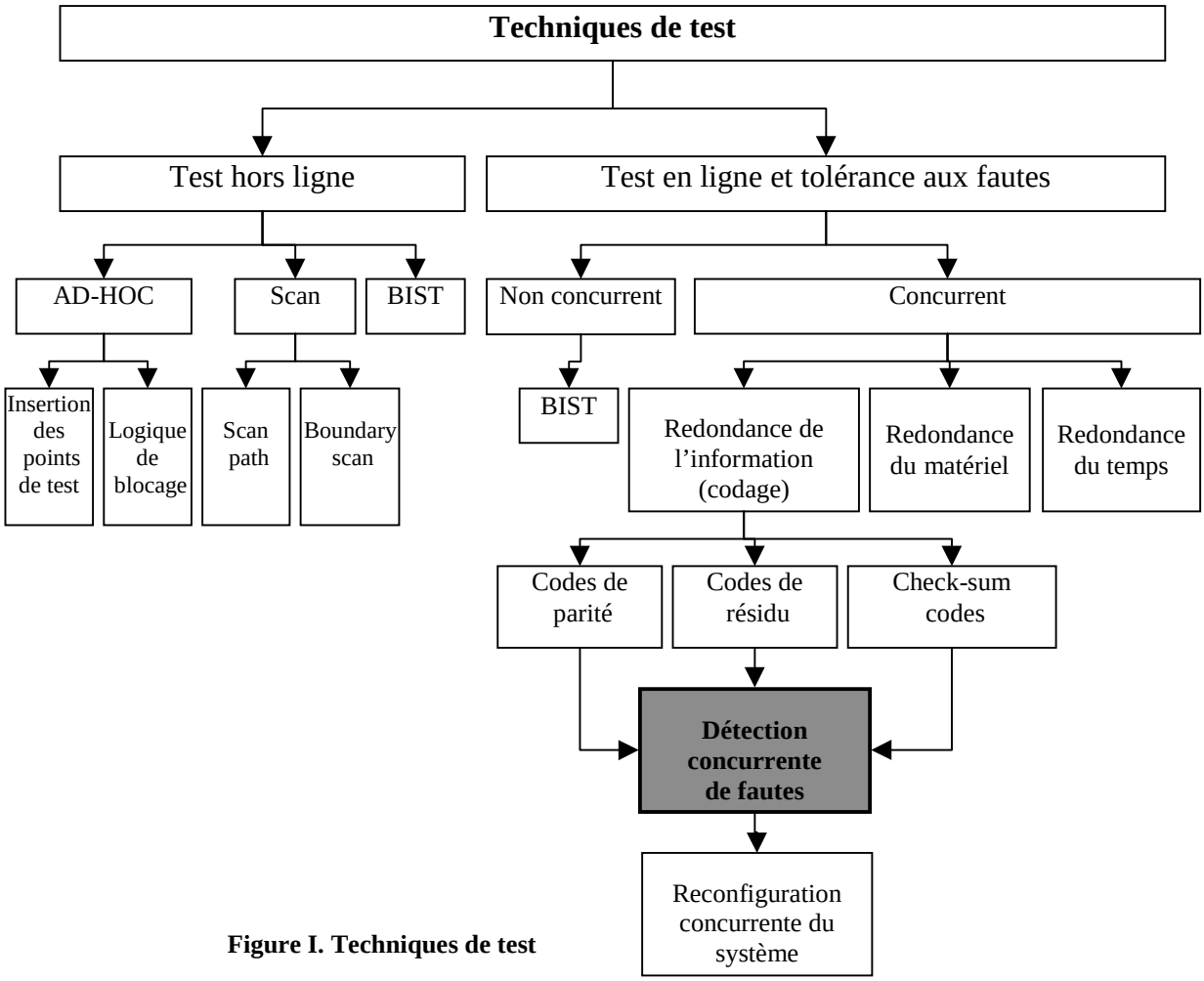


Figure I. Techniques de test

CHAPITRE1 : Systèmes digitaux linéaires

1.1 Introduction

L'évolution des technologies **VLSI** a permis de réaliser des systèmes complexes et dédiés à des applications importantes. Les systèmes digitaux linéaires, étant comptés parmi les avantages des nouvelles technologies, représentent une classe très importante des circuits utilisés dans plusieurs applications critiques militaires, nucléaires, spatiales et de télécommunication. Dans ce chapitre nous présentons les systèmes digitaux linéaires, nous donnons tout d'abord leurs représentations principales, puis nous parlons de leurs implémentations matérielle et logicielle.

1.2 Représentations des systèmes digitaux linéaires

Par définition, un système digital linéaire est un système (circuit) digital accomplissant une transformation linéaire de ses entrées. Les signaux traités par un système digital linéaire sont des signaux numériques et les différents types d'opérations qu'il est possible d'effectuer sont la multiplication, l'addition et la mémorisation de données. Généralement, ce type de systèmes peut être présenté selon quatre méthodes différentes. Celles-ci comprennent :

1- La représentation par le graphe général de l'écoulement de données [ChAb93, ChCh99].
2- La représentation d'état [DiRo90, GiCl90, Lfer72].
3- La représentation par des fonctions de transfert [DiRo90, FoGe87, OpSc75, Smit99].
4- Le graphe d'état général [ChAb93, OpSc75].

1.2.1 Graphe général de l'écoulement de données (structure générale)

Généralement, un système digital linéaire est un assemblage d'additionneurs, de multiplieurs et de mémoires interconnectés pour réaliser une fonction complexe. La figure1-1, représente la structure générale d'un système digital linéaire [ChAb93, ChCh99].

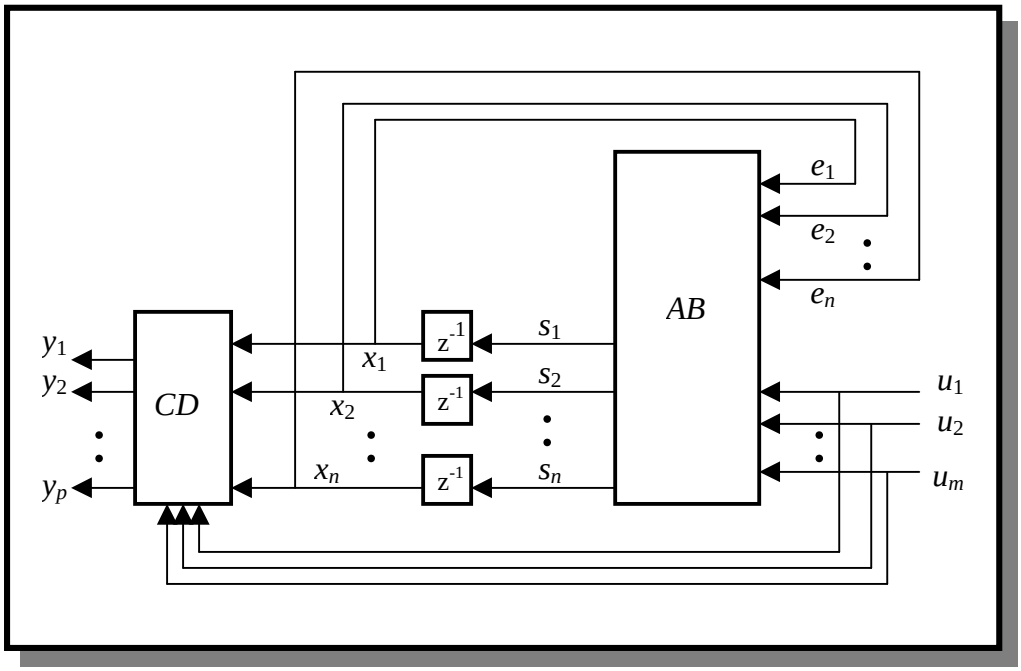


Figure 1-1. Structure générale d'un système digital linéaire

z^{-1} représente un retard (mémoire) pour synchroniser le système. *AB* est un circuit séquentiel synchronisé traitant des ensembles de données d'entrées appelées mots des données. *AB* calcule une transformation linéaire des valeurs de ses $n+m$ entrées et génère n valeurs de sortie. Les valeurs des mots de données fournies aux entrées $e_1, e_2,..., e_n,$ à un temps quelconque t, sont données par le vecteur :

$$[x_1(t), x_2(t),..., x_n(t)]^T \tag{1-1}$$

Les valeurs des mots de données obtenues aux sorties $s_1, s_2,\ldots, s_n,$ sont fournies au temps $t+1$ aux entrées correspondantes de *AB*. Les valeurs de ces mots de données sont données par le vecteur :

$$[x_1(t+1), x_2(t+1),..., x_n(t+1)]^T \tag{1-2}$$

A un temps quelconque t, *CD* calcule une transformation linéaire des valeurs de ses $n+m$ entrées et génère p valeurs de sorties. Les valeurs des mots de données fournies aux entrées de *CD* sont données par les vecteurs :

$$[x_1(t), x_2(t),..., x_n(t)]^T$$

$$[u_1(t), u_2(t),..., u_m(t)]^T \tag{1-3}$$

Les valeurs des mots de données correspondantes obtenues aux sorties sont données par le vecteur

$$[y_1(t), y_2(t),..., y_p(t)]^T \tag{1-4}$$

u_i désigne la $i^{ème}$ entrée externe, $1 \le i \le m$, y_i désigne la $i^{ème}$ sortie externe, $1 \le i \le p$.

Le vecteur

$$x(t) = [x_1(t), x_2(t),..., x_n(t)]^T \tag{1-5}$$

est défini comme un vecteur d'état tandis que

$$x(t+1) = [x_1(t+1), x_2(t+1),..., x_n(t+1)]^T \tag{1-6}$$

est le vecteur d'état suivant.

x_1, x_2,..., x_n : représentent les variables d'état associées avec les retards (z^{-1}) par conséquent elles sont associées avec les vecteurs d'état $x(t+q)$, q = 0, 1, 2, … etc.

$x_i(t+q)$: représente la valeur de variable d'état x_i au temps $t+q$, et $1 \le i \le n$.

1.2.2 Représentation d'état

Le comportement des systèmes digitaux linéaires peut être exprimé par des modèles mathématiques. Ceux-ci correspondent à des équations mathématiques décrivant les relations directes et indirectes entre les entrées et les sorties du système [DiRo90, GiCl90, Lfer72].

Equations d'état et de sortie

De la représentation générale des systèmes digitaux linéaires, le vecteur d'état $x(t)$ à un temps quelconque t est relié au vecteur d'état suivant $x(t+1)$ au temps $t+1$ par l'équation d'état (1-7) qui traduit la dynamique du système :

$$x(t+1) = A.x(t) + B.u(t) \tag{1-7}$$

$u(t)$: est le vecteur d'entrée de l'ordre m.

$x(t)$: est le vecteur d'état de l'ordre n.

$t+1$: définit l'instant suivant.

Les matrices A, B sont des matrices de dimensions appropriées.

Le vecteur de sortie à un temps quelconque t est relié au vecteur d'état $x(t)$ par l'équation de sortie :

$$y(t) = C.x(t) + D.u(t) \tag{1-8}$$

$y(t)$: est le vecteur de sortie de l'ordre p.

Les matrices C, D sont des matrices de dimensions appropriées.

Les équations 1-7 et 1-8 sont les équations qui représentent le modèle mathématique temporel (dynamique) d'un système digital linéaire.

1.2.3 Fonction de transfert

Les systèmes digitaux linéaires peuvent aussi être exprimés par leurs fonctions de transfert [DiRo90, FoGe87, OpSc75, Smit99]. Pour les systèmes mono-variable (mono-entrée 'u' et mono-sortie 'y'), La forme générale de la fonction de transfert est donnée par :

$$H(z) = \frac{y(z)}{u(z)} = \frac{d + \sum_{i=1}^{n} b_{n-i}.z^{-i}}{1 + \sum_{i=1}^{n} a_{n-i}.z^{-i}} \tag{1-9}$$

ou 'n' représente l'ordre du système et a, d, b sont des constantes.

Cette fonction de transfert correspond à l'équation aux différences suivante

$$\begin{aligned} y(t) = d.u(t) + b_{n-1}.u(t-1) + b_{n-2}.u(t-2) + \cdots + b_0.u(t-n) \\ - a_{n-1}.y(t-1) - a_{n-2}.y(t-2) - \cdots - a_0.y(t-n) \end{aligned} \tag{1-10}$$

Pour les systèmes multi-variables, chaque système peut être exprimé par une matrice de transfert dans lequel chaque élément dans cette matrice est une fonction de transfert.

1.2.4 Passage de la représentation d'état à la fonction de transfert

Ce passage est fait en effectuant la transformée en z des équations d'état (1-7) et de sorties (1-8). La transformée en z de ces équations est donnée par:

$$z.x(z) = A.x(z) + B.u(z) \tag{1-11}$$

$$y(z) = C.x(z) + D.u(z) \tag{1-12}$$

De l'équation 1-11 on trouve [FoGe87] :

$$x(z) = [z.I - A]^{-1}.B.u(z) \qquad (1\text{-}13)$$

où I est la matrice d'identité. Remplaçant l'équation 1-13 en l'équation 1-12 on trouve :

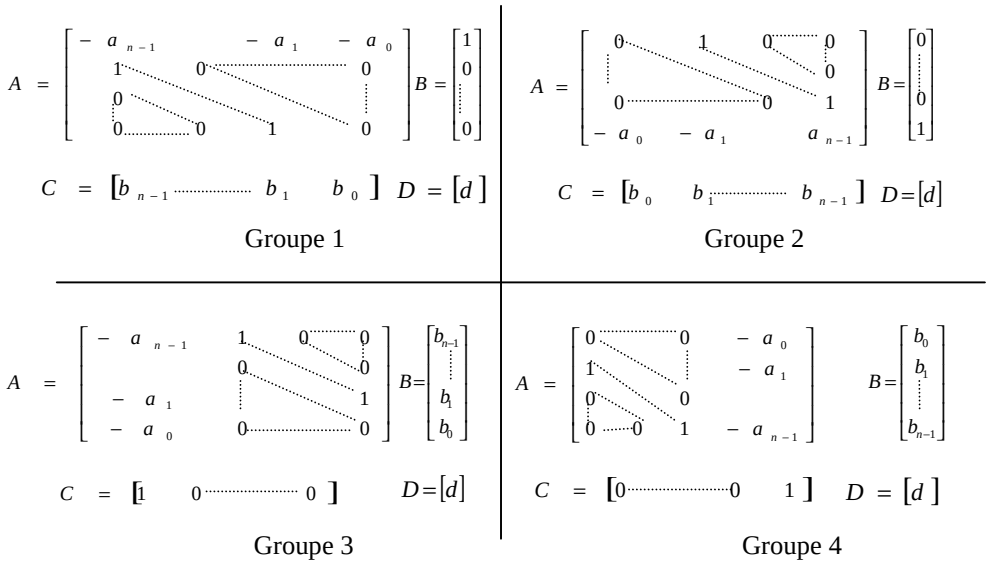
$$y(z) = \{C.[z.I - A]^{-1}.B + D\}.u(z) \qquad (1\text{-}14)$$

$$H(z) = y(z) / u(z) = C.[z.I - A]^{-1}.B + D \qquad (1\text{-}15)$$

L'équation 1-15 représente la fonction de transfert extraite de la représentation d'état.

1.2.5 Passage de la fonction de transfert à la représentation d'état

Le passage de la fonction de transfert ou de la matrice de transfert à la représentation d'état se fait selon les règles de changement de représentation [FoGe87]. Ce passage est nécessaire pour pouvoir appliquer notre méthode de test si le système à tester est exprimé par une fonction de transfert telle que dans l'équation 1-9. Pour les systèmes mono-variable, ces règles sont illustrées par les groupes des matrices suivants.



Les groupes des matrices montrent la méthode de placement de coefficients (les constantes *a*, *b* et *d*) de la fonction de transfert dans les matrice *A*, *B*, *C* et *D* des équations

d'état et de sortie. Chaque groupe peut être utilisé pour le passage à la représentation d'état. Pour les systèmes multi-variables, le passage de la matrice de transfert à la représentation d'état est faisable, mais il est un peu compliqué que dans le cas de systèmes mono-variable. Pour plus d'informations sur le changement de représentation voir la référence [FoGe87].

1.2.6 Graphe d'état

Les systèmes digitaux linéaires peuvent encore être représentés par des graphes d'état [ChAb93, OpSc75]. Ceux-ci décrivent les relations entre les différentes variables comprises dans le système : entrées, sorties et variables d'état. Un graphe d'état correspond graphiquement à des nœuds interconnectés par des flèches, figure1-2 suivante.

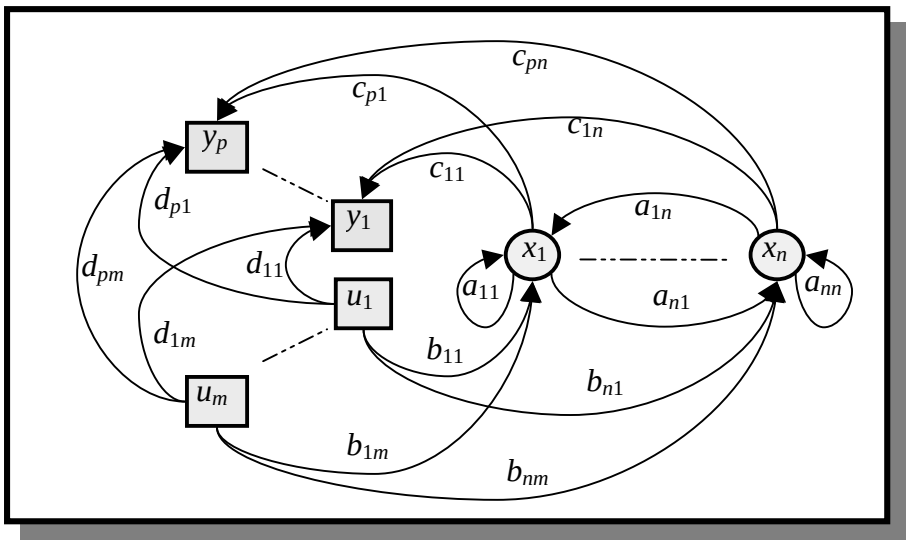


Figure 1-2. Graphe d 'état général d'un système digital linéaire

Note *: Il ne peut y avoir de flèche entre un nœud carré et lui-même*.

Un nœud carré dans la figure représente une entrée ou une sortie externe, par contre un nœud rond représente une variable d'état « x_i » $1 \le i \le n$. Les flèches représentent les chemins de propagation de données entre les nœuds (une flèche peut correspondre à un chemin ou à plusieurs chemins). La constante associée avec une flèche représente le gain de(s) chemin(s) représenté(s) par cette flèche. Par exemple la constante a_{11} représente le gain de(s) chemin(s) représenté(s) par la flèche reliant le nœud x_1 avec lui-même tandis que a_{1n} représente le gain de(s) chemin(s) représenté(s) par la flèche partant de x_n et reliant x_1 avec x_n.

1.2.7 Passage du graphe d'état à la représentation d'état

Les matrices *A*, *B*, *C*, *D* d'équations d'état peuvent être déduites facilement du graphe d'état. Les règles d'extraction, voir équation (1-16) ci-après, sont :

1- Rassembler dans la matrice *A* toutes les constantes associées avec les flèches reliant seulement les nœuds de variables d'état entre eux.

2- Rassembler dans la matrice *B* les constantes associées avec les flèches reliant les nœuds d'entrées avec les nœuds de variables d'état.

3- Rassembler dans la matrice *C* les constantes associées avec les flèches reliant les nœuds de variables d'état avec les nœuds de sorties.

4- Rassembler dans la matrice *D* les constantes associées avec les flèches reliant les nœuds d'entrées avec les nœuds de sorties.

Pour le graphe de la figure1-2, les matrices *A*, *B*, *C* et *D* sont :

$$A=\begin{bmatrix} a_{11} & a_{12} & .. & a_{1n} \\ a_{21} & a_{22} & .. & a_{2n} \\ : & : & : & : \\ a_{n1} & a_{n2} & .. & a_{nn} \end{bmatrix}, B=\begin{bmatrix} b_{11} & b_{12} & .. & b_{1m} \\ b_{21} & b_{22} & .. & b_{2m} \\ : & : & : & : \\ b_{n1} & b_{n2} & .. & b_{nm} \end{bmatrix}, C=\begin{bmatrix} c_{11} & c_{12} & .. & c_{1n} \\ c_{21} & c_{22} & .. & c_{2n} \\ : & : & : & : \\ c_{p1} & c_{p2} & .. & c_{pn} \end{bmatrix}, D=\begin{bmatrix} d_{11} & d_{12} & .. & d_{1m} \\ d_{21} & d_{22} & .. & d_{2m} \\ : & : & : & : \\ d_{p1} & d_{p2} & .. & d_{pm} \end{bmatrix} \quad (1\text{-}16)$$

Les équations d'état, sous une forme matricielle, peuvent être réécrites :

$$\begin{bmatrix} x_1(t+1) \\ x_2(t+1) \\ : \\ x_n(t+1) \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & .. & a_{1n} \\ a_{21} & a_{22} & .. & a_{2n} \\ : & : & : & : \\ a_{n1} & a_{n2} & .. & a_{nn} \end{bmatrix} . \begin{bmatrix} x_1(t) \\ x_2(t) \\ : \\ x_n(t) \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} & .. & b_{1m} \\ b_{21} & b_{22} & .. & b_{2m} \\ : & : & : & : \\ b_{n1} & b_{n2} & .. & b_{nm} \end{bmatrix} . \begin{bmatrix} u_1(t) \\ u_2(t) \\ : \\ u_m(t) \end{bmatrix} \quad (1\text{-}17)$$

$$\begin{bmatrix} y_1(t) \\ y_2(t) \\ : \\ y_p(t) \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & .. & c_{1n} \\ c_{21} & c_{22} & .. & c_{2n} \\ : & : & : & : \\ c_{p1} & c_{p2} & .. & c_{pn} \end{bmatrix} . \begin{bmatrix} x_1(t) \\ x_2(t) \\ : \\ x_n(t) \end{bmatrix} + \begin{bmatrix} d_{11} & d_{12} & .. & d_{1m} \\ d_{21} & d_{22} & .. & d_{2m} \\ : & : & : & : \\ d_{p1} & d_{p2} & .. & d_{pm} \end{bmatrix} . \begin{bmatrix} u_1(t) \\ u_2(t) \\ : \\ u_m(t) \end{bmatrix} \quad (1\text{-}18)$$

Exemple

Soit le graphe d'état de la figure1-3 suivante. Il y a deux variables d'état (n = 2), une entrée (m = 1) et une sortie (p = 1). Pour cela, la matrice *A* est des dimensions 2×2, la matrice

B est des dimensions 2×1, la matrice *C* est des dimensions 1×2 et la matrice *D* est des dimension 1×1.

Selon les règles d'extraction on trouve :

$a_{11} = 0, \quad a_{12} = -1/2, \quad a_{21} = -1/3, \quad a_{22} = 2/3, \quad b_{11} = 1/2, \quad b_{21} = -1/2.$

$c_{11} = 0, \quad c_{12} = 1, \quad d_{11} = 0.$

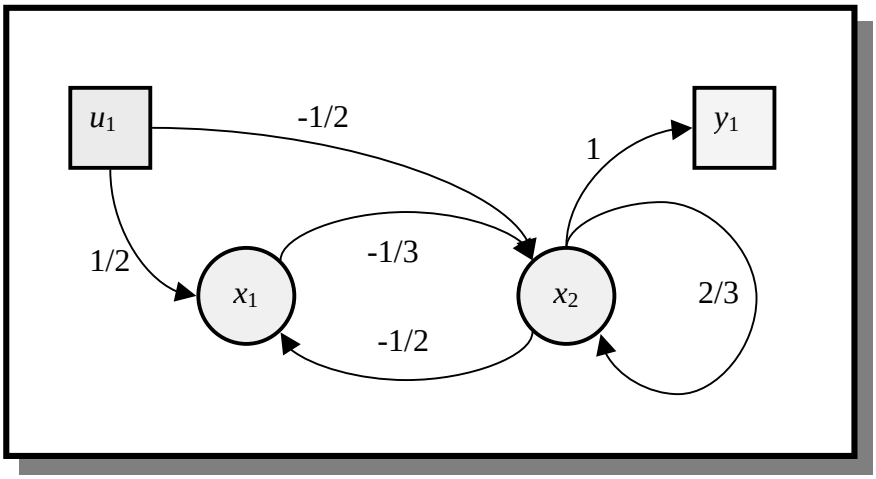


Figure 1-3.

Les matrices *A*, *B*, *C*, *D*, sont $A = \begin{bmatrix} 0 & -1/2 \\ -1/3 & 2/3 \end{bmatrix}$. $B = \begin{bmatrix} 1/2 \\ -1/2 \end{bmatrix}$. $C = \begin{bmatrix} 0 & 1 \end{bmatrix}$. $D = \begin{bmatrix} 0 \end{bmatrix}$.

1.3 Implémentation des systèmes digitaux linéaires

Les nouvelles technologies du VLSI ont permis d'intégrer des millions de transistors dans une seule puce de silicium. Ceci a permis de réaliser des systèmes intégrés complexes et peu coûteux. Cette réduction du coût avec cette possibilité d'intégration, a donné naissance à des nouvelles générations de systèmes spécifiques et dédiés à des domaines d'applications très importants, par exemple les processeurs de traitement digital linéaire du signal [BaVi96, CCCE84, KONa86, Kung85, Madi95, MaWi98, RoBr97, Wade94]. Ces nouveaux processeurs, appelés DSP linéaires et qui traitent les différents algorithmes de traitement digital linéaire du signal, peuvent avoir deux formes différentes d'implémentation [BaVi96, Madi95, Wade94]. Cela désigne les DSP linéaires programmables et les DSP linéaires dédiés.

1.3.1 Les DSP linéaires programmables

Les DSP linéaires programmables sont en général des machines séquentielles multitâches. Un DSP programmable peut traiter plusieurs algorithmes écrits sous forme d'un programme d'ordinateur stocké dans la mémoire du DSP. La figure1-4 représente une architecture simple d'un DSP programmable [Madi95].

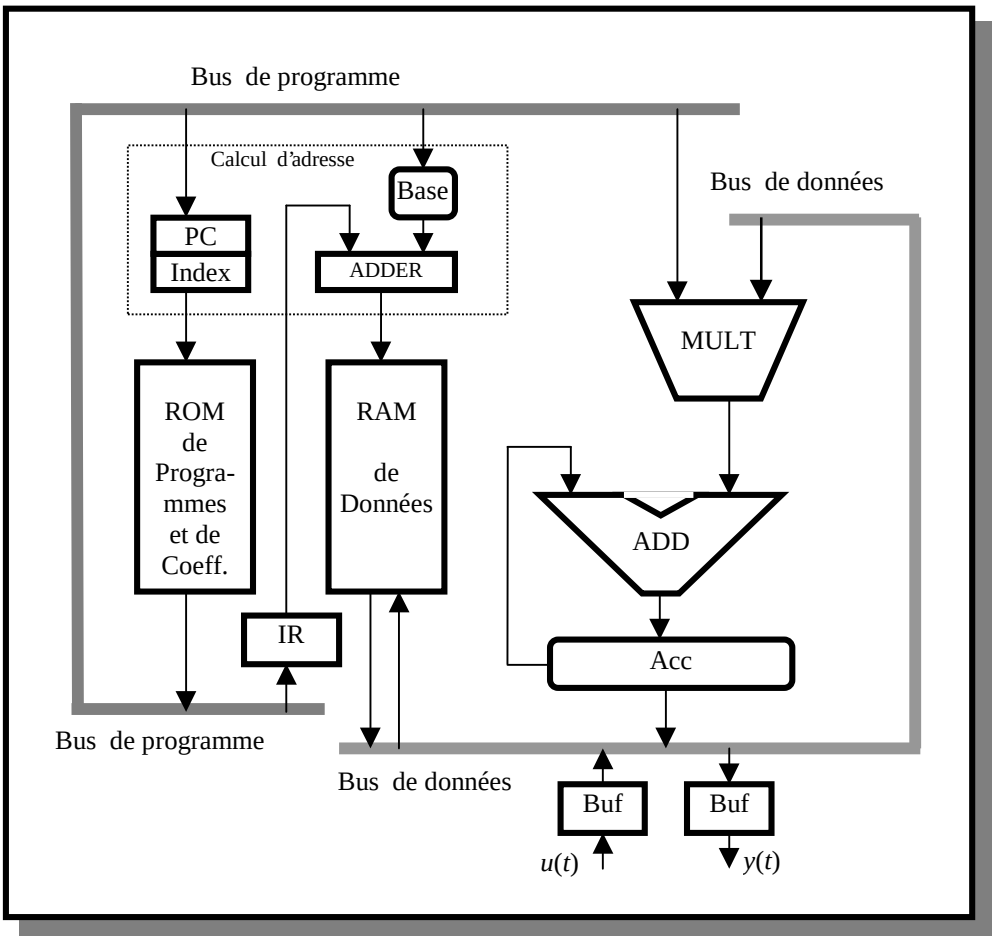


Figure 1-4. DSP programmable

Les programmes contenant les algorithmes de traitement ainsi que les coefficients sont stockés dans une mémoire ROM accessible à lecture seulement et rien ne peut être écrit dans cette mémoire. Les données à traiter $u(t)$ (entrée du DSP) peuvent être écrites dans la mémoire RAM. Ces données peuvent être ensuite lues de la RAM pour traitement. Le compteur de programme (PC) et le registre d'indexation (Index) sont utilisés pour calculer les adresses de l'instruction suivante et des coefficients. L'instruction est décodée par le registre d'instruction (IR) pour calculer l'adresse de données en utilisant l'additionneur (ADDER) et le registre

d'indexation (Base). Le multiplieur (MULT), l'additionneur (ADD) et le registre (Acc) sont utilisés pour effectuer la tâche de traitement. Les données de sortie $y(t)$ sont disponibles sur le tampon de sortie (Buf). Les instructions et toutes les autres opérations (adressage, multiplication, addition mémorisation … etc.) sont effectuées d'une manière séquentielle.

1.3.2 Les DSP linéaires dédiés

Certaines applications n'ont pas besoin de toutes les fonctionnalités des DSP programmables. Donc, pour simplifier, améliorer les performances et réduire le coût, une autre forme d'implémentation peut être utilisée. Celle-ci distingue l'implémentation dédiée à une tâche (algorithme) spécifique. Cette implémentation correspond à deux approches différentes appelées implémentation dédiée séquentielle (DSP dédies séquentiels) et implémentation dédiée parallèle (DSP dédiés parallèle) [Madi95, Wade94].

1.3.2.1 Les DSP dédiés séquentiels

Ce type d'implémentation ressemble à peu près à l'implémentation des DSP programmables (traitement séquentiel des données), mais ici, le DSP est adapté et dédié à un algorithme spécifique. La figure1-5 représente un DSP linéaire dédié pour réaliser la fonction d'un filtre numérique linéaire non récursif («finite impulsion response (FIR) » en anglais) [Wade94]. Le système calcule la sortie $y(t)$ donnée par la relation de convolution suivante :

$$y(t) = \sum_{i=0}^{n} h_i u(t-i) \qquad (1\text{-}19)$$

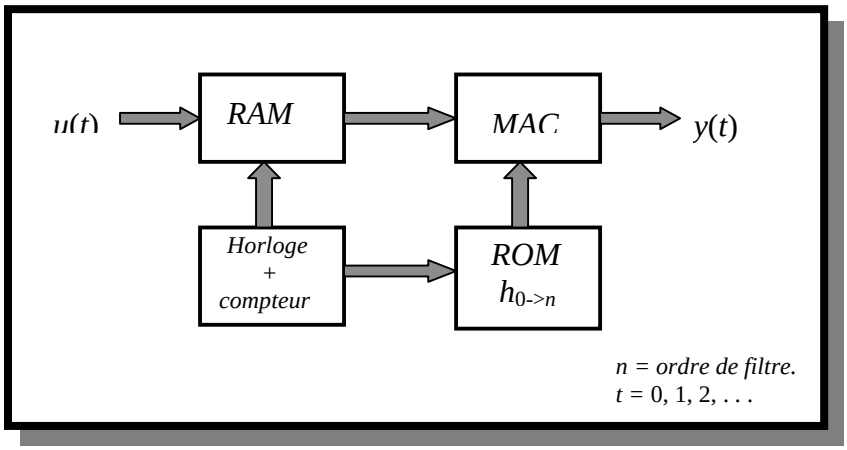


Figure 1-5. DSP linéaire séquentiel pour filtre FIR

La *RAM* est une mémoire vive d'une taille de (***n***+1) mots. Elle reçoit les mots des données d'entrée. La *ROM* est une mémoire morte stockant (***n***+1) constantes représentant les coefficients $\boldsymbol{h}_i$ du filtre, $0 \leq i \leq \boldsymbol{n}$. La boîte '*Horloge + compteur*' est une boîte de commande pour l'adressage et la synchronisation. Le block *MAC* est un multiplieur/accumulateur effectuant les opérations de multiplication et d'accumulation de données dans son registre. A chaque coup d'horloge un mot de données, $u(t)$, entre dans la *RAM*, un autre mot, $u(t-n-1)$, la quitte et un mot, $y(t)$, est obtenu à la sortie du *MAC*. Le compteur effectue la fonction d'adressage pour pouvoir accéder aux contenus des mémoires. Chaque opération d'adressage est accompagnée d'une opération composée (multiplication/accumulation) du *MAC*. Les opérations sont effectuées d'une manière séquentielle. Donc, le calcul de la relation de convolution (équation 1-19) a besoin de (***n***+1) opérations composées successives du *MAC*. Les opérations dans leur totalité doivent être effectuées pendant un seul cycle d'horloge.

1.3.2.2 Les DSP dédiés parallèles

Les DSP programmables et les DSP dédiés séquentiels traitent les informations d'une manière séquentielle, c'est-à-dire que les opérations sont effectuées successivement l'une après l'autre. Donc, pour effectuer la tâche de traitement, les implémentations séquentielles peuvent demander un cycle d'horloge (période d'échantillonnage) relativement large. Ceci peut imposer des contraintes sur la fréquence de fonctionnement du DSP.

De toute façon les DSP séquentiels (programmables et dédiés) sont limités en fréquence et ne peuvent pas répondre aux besoins d'application demandant une gamme de fréquence élevée. Pour résoudre ce problème, des implémentations parallèles (DSP dédiés parallèles) peuvent être utilisées. Avec le traitement parallèle, des opérations peuvent être effectuées en même temps et le temps nécessaire pour effectuer, parallèlement, (n+1) opérations est équivalent à celui nécessaire pour effectuer une seule opération. La figure1-6 suivante représente une architecture parallèle d'un DSP linéaire qui a pour tâche la réalisation de la fonction exprimée par la relation de convolution dans l'équation 1-19.

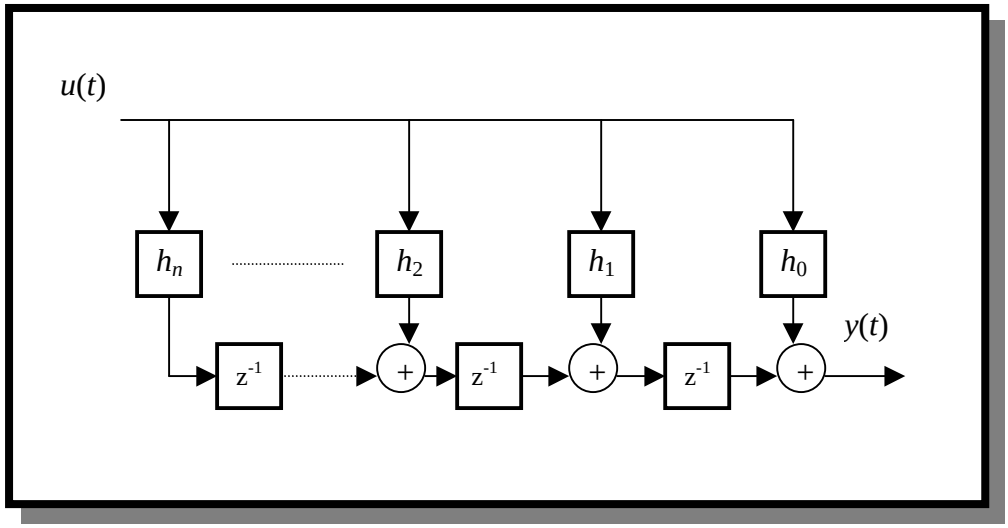


Figure 1-6. Filtre FIR digital linéaire

Pour cette architecture, il est clair que toutes les opérations de multiplication sont effectuées en même temps. Ensuite la réalisation parallèle de toutes les opérations d'addition aura lieu ; et à la fin les opérations de mémorisation, toutes, seront réalisées en même temps. Donc, le temps nécessaire pour effectuer toutes les opérations de multiplication, d'addition et de mémorisation est équivalent au temps nécessaire pour réaliser une seule opération composée d'une multiplication, d'une addition et d'une mémorisation.

1.4 Conclusion

Les systèmes digitaux linéaires ou les DSP linéaires traitent des signaux déjà convertis dans une forme numérique. Ils ont été inventés pour améliorer la qualité et réduire le coût de transmission/réception des signaux de télécommunication. La première génération de DSP a vu le jour dans les années 80s grâce aux progrès technologiques spectaculaires dans le domaine du VLSI. Différentes formes d'implémentation, adaptées aux besoins des applications différentes, ont été réalisées. Ces implémentations désignent les DSP programmables et les DSP dédiés. Les DSP dédiés sont des systèmes de traitement séquentiels ou parallèles. Les DSP de traitement parallèle sont favorisés pour des applications de fréquences élevées.

CHAPITRE 2 : Test des systèmes digitaux (état de l'art)

2.1 Introduction

Le test de circuits intégrés est mis en œuvre pour déceler la présence de dysfonctionnement dans le circuit sous test. La testabilité des circuits complexes pose un problème important depuis un certain temps déjà. Il a amené à l'adoption des techniques de test hors ligne telles que les techniques AD-HOC, full-scan, boundry-scan et built-in-self-test.

Certains domaines d'application ne peuvent tolérer l'interruption de leur fonctionnement même pas quelques secondes par année et demandent de montrer l'état des systèmes au cours de leur fonctionnement normal. Dans ces conditions, il devient pratiquement impossible de tester les systèmes en utilisant les techniques de test hors ligne. Il devient donc de plus en plus évident que les systèmes du futur devraient inclure des moyens d'auto-test donnant un avantage évident aux techniques de test en ligne comme par exemple les techniques de redondance du matériel et de redondance de l'information.

Avant d'aborder les notions de test hors ligne et de test en ligne, pour la détection de fautes dans les circuits digitaux, il nous paraît important de parler tout d'abord des fautes, de l'origine de fautes et des modèles de fautes. Puis, nous discutons quelques techniques de test hors ligne en détaillant leurs avantages et leurs limitations. Nous parlons ensuite de techniques de test en ligne et de leurs avantages par rapport aux techniques de test hors ligne.

Des techniques nouvelles de test en ligne des systèmes digitaux complexes sont détaillées. Nous concluons le chapitre en exposant les avantages, les limitations et les problèmes actuels de test des systèmes digitaux.

2.2 Fautes et modèles de fautes

Le problème de test est un problème d'identification des modules qui ne satisfont pas à leurs propres spécifications fonctionnelles. Le fonctionnement erroné d'un circuit est principalement dû à des fautes physiques et à des fautes dues à l'environnement. En général, les fautes peuvent se classer en deux groupes :

1- Fautes permanentes.
2- Fautes temporaires.

2.2.1 Fautes permanentes

Les fautes permanentes sont en général des fautes physiques principalement dues aux procédés de fabrication de circuits intégrés ou au vieillissement de composants [AbFu86, Fant84, EiBe81, Goor91, MaAv80, Mang84, RuSa89]. Elles sont de deux types :

1-Fautes statiques.
2-Fautes dynamiques (fautes de délai).

Définition : *un circuit logique est le siège d'une faute statique s'il présente un dysfonctionnement à basses et à hautes fréquences.*

Définition : *un circuit logique est le siège d'une faute dynamique s'il présente un dysfonctionnement à hautes fréquences alors que le fonctionnement logique est correct à des fréquences plus basses.*

Donc, une faute statique entraîne un dysfonctionnement logique du circuit à basse fréquence et elle est caractérisée, en général, par un défaut localisé tel que : une connexion rompue, un nœud de circuit isolé ou des nœuds court-circuités. Par contre, une faute dynamique (également appelée faute de délai) entraîne un dysfonctionnement logique qui se caractérise à haute fréquence par un ralentissement de la transmission des signaux dans les différents blocs du système [More98].

Origine physique d'une faute permanente

Avec l'avènement des technologies submicroniques, l'impact de certains paramètres sur les performances des circuits s'est intensifié. En effet, la fréquence de fonctionnement des circuits augmente régulièrement avec l'évolution de ces technologies d'intégration. A basse fréquence, certains défauts liés au processus de fabrication ne perturbent pas le fonctionnement du circuit, alors que, dans le cas de fréquences élevées, ces mêmes défauts peuvent influencer le comportement temporel des sorties. De même, les outils d'optimisation utilisés lors de la conception ont tendance à augmenter la densité des circuits, ce qui donne lieu à une augmentation du nombre de portes logiques sur les mêmes chemins critiques. Ainsi, même les défauts provoquant des retards de petites tailles peuvent rendre un circuit défectueux.

Les défauts physiques qui sont à l'origine de fautes permanentes sont répartis en deux catégories : les défauts liés au procédé de fabrication et les défauts qui peuvent apparaître pendant la phase de fonctionnement normal (défauts dus au vieillissement des composants).

Défauts de fabrication

Les défauts de fabrication résultent des imperfections dans les procédés de fabrication. Ces imperfections peuvent se traduire par des défauts localisés tels que les connexions rompues (circuit ouvert) [AbSh85], les connexions détériorées (circuit ouvert incomplet) [TBGH83], des nœuds de circuit isolés (grille flottante) [BaAb83], des nœuds court-circuités (court-circuit) ou par une dérive générale d'un certain nombre de paramètres (rapport W/L, tension de seuil) [Koep86].

Les défauts à l'origine de ces nœuds flottants ou de ces ruptures de lignes sont très variés et dépendent fortement des procédés de fabrication propres à chaque technologie. A titre d'exemple, voici comment un défaut local peut apparaître et entraîner des fautes permanentes.

La réalisation d'un circuit comporte différentes opérations technologiques (dopage, diffusion, métallisation, … etc.) qui correspondent aux modifications que doit subir la plaquette de silicium sur laquelle sont réalisés les circuits intégrés. Une première phase de dopage permet de produire les effets électriques qui forment la base des circuits électroniques (transistors, résistances, condensateurs). Les différentes fonctions logiques vont être diffusées

sur la tranche de silicium. Avant de passer à la phase de métallisation qui consiste à implanter des bandes de métal entre les différents points du circuit à connecter, il faut procéder à l'ouverture des contacts. La plaquette est enduite de résine qui, après séchage, est exposée à un rayonnement ultraviolet au travers d'un masque de métallisation. Les contacts sont ouverts par révélation. La résine étant positive, seule la partie exposée aux ultraviolets disparaît. Or, si un grain de poussière vient à se déposer sur la résine avant l'étape de rayonnement, ce grain de poussière peut masquer la résine qui demeure, par conséquent, insensible à l'attaque chimique destinée à ouvrir les contacts nécessaires à la création des connexions de métal. Après l'étape de métallisation, la présence de cette poussière se caractérise par une tache sur la plaquette de silicium. Ce type de défaut entraîne quelquefois l'occurrence de fautes statiques et une modification de la résistance de l'interconnexion qui se traduit par un retard. Les effets d'une telle défaillance sont présentés sur la figure2-1 [More98]. Le schéma représente trois lignes d'un même métal dans un circuit.

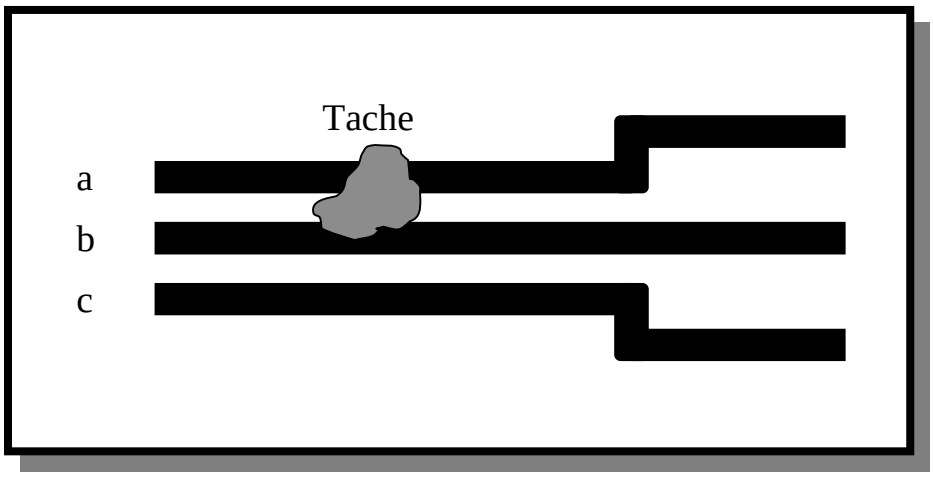


Figure 2-1: Exemple d'un défaut physique provoquant sur a : une faute statique, sur b : une panne temporelle, sur c : aucune incidence

Les lignes **a** et **b** sont affectées par le défaut. La connexion sur la ligne **a** est rompue. Donc, la ligne a est le siège d'une faute statique. La connexion sur la ligne **b**, quant à elle, est détériorée puisque l'interconnexion est rétrécie en un endroit. Bien que la connexion soit toujours présente, le rétrécissement provoque une augmentation de la résistance. Par conséquent, le chargement d'une capacité à travers cette ligne ne sera pas interrompu mais uniquement ralenti. Ceci peut donner lieu à une faute dynamique ou faute de retard. Le temps nécessaire pour charger une capacité est donné par la loi :

$$t = -R.C.ln(Vc/V) \qquad (2\text{-}1)$$

où *C, R* sont la capacité et la résistance de chargement, respectivement, *Vc* est la tension momentanée aux bords de capacité et *V* est la tension à atteindre. La figure2-2 illustre l'influence d'une connexion détériorée. La ligne **c**, quant à elle, n'est pas touchée par la tache. Elle n'est donc pas affectée par le défaut.

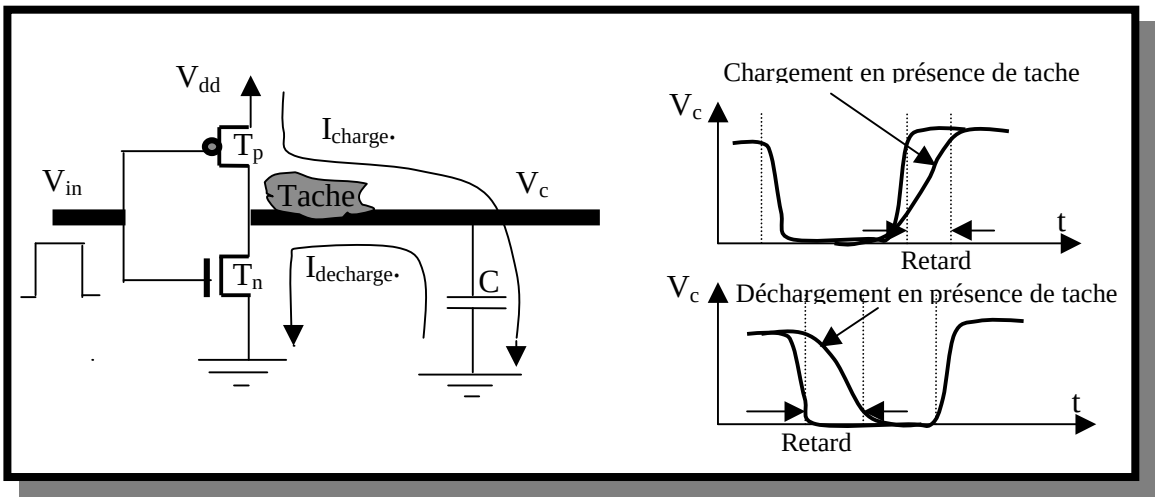


Figure 2-2: Influence d'une connexion détériorée

Indépendamment de l'affectation locale, certains défauts de fabrication peuvent également avoir un effet global, c'est-à-dire, que toutes ou un grand nombre de portes du circuit sont affectées par une modification de leur délai de propagation. Dans le cas d'une altération non uniforme des délais de propagation dans le circuit, ces variations peuvent être provoquées par des conditions de fabrication non homogènes pendant la réalisation du circuit.

Par exemple, au cours de la phase de diffusion, la plaquette de silicium est placée dans un four où sont introduits les gaz dopants de manière à imprégner les zones non masquées. Si la chaleur n'est pas répartie uniformément, la portion du circuit soumise à une température supérieure subit une accélération du processus de diffusion, ce qui conduit à des concentrations de dopage inégales. Ceci provoque une altération des délais de propagation des portes de la zone affectée. Ce type de défaut engendre une répartition globale de fautes de retard.

Outre les cas cités précédemment, de nombreux exemples de défauts de fabrication sont disponibles dans la littérature [AbFu86, AbSh85, BaAb83, CIRo87, Mang84, Syrz87].

Défauts de vieillissement des composants

Les circuits électroniques ayant été préalablement conçus, fabriqués et testés sont destinés à fonctionner dans certains domaines d'application. Un circuit peut ne pas remplir sa fonction correctement, ou tomber en panne, sous l'influence de certains mécanismes tels que injection des porteurs de charge dans la zone d'oxyde de la grille du transistor MOS [EiBe81, Fant84], défaillance de l'oxyde [Fant84], métallisation défaillante [Fant84, Sze83], … etc.

Injection des porteurs chauds de charge

Les transistors MOS, qui sont largement utilisés dans les circuits VLSI, ont deux caractéristiques très importantes : la tension de seuil et la transconductance. Le changement d'un seul paramètre (caractéristique) peut affecter la performance du circuit.

Si des porteurs chauds (électrons) sont injectés dans la zone d'oxyde de grille, alors la tension de seuil sera modifiée et la transconductance sera dégradée (les électrons injectés produisent un courant de fuite). Par conséquent, des fautes peuvent être introduites dans le circuit.

Pour que les électrons puissent pénétrer dans la zone d'oxyde, ils doivent avoir une énergie élevée (ils deviennent des porteurs chauds). Les électrons chauds sont générés en général par la soumission à un champ électrique élevé dû au dépassement de la tension permise de l'alimentation appliquée entre la source et le drain du transistor MOS.

Défaillance de l'oxyde

L'injection d'électrons chauds dans la zone d'oxyde de grille peut créer un autre phénomène que le courant de fuite. Ceci correspond au changement de caractéristiques de l'oxyde. Si les électrons injectés ont suffisamment d'énergie, alors ils peuvent créer des paires d'électron-trou par le phénomène d'ionisation. Les électrons créés par ce mécanisme sont rapidement recueillis (ramassés) par la grille (cas de NMOS) en laissant derrière eux une charge positive (trous bougeant lentement), laquelle induit un courant élevé conduisant à augmenter la formation de paires d'électron-trou. Par cet effet, le volume de charge positive

augmente. L'augmentation en charge entraîne une nouvelle augmentation en courant entraînant à son tour une nouvelle augmentation en charge, et la boucle se répète ainsi sans arrêt. Par conséquent l'oxyde peut perdre ses caractéristiques d'isolation et se comporter comme un conducteur. Ainsi, un court circuit peut être établi et le circuit sera le siège de fautes.

Défaillance de la métallisation

Le phénomène d'électro-migration des atomes dans les transistors MOS représente un grand risque, car il peut changer les caractéristiques de métallisation. Ce phénomène apparaît quand la densité du courant dépasse certaines limites. Les atomes de métal sont, en général, soumis à deux forces différentes : la force électrostatique due à la chute de tension tout au long du conducteur et l'interaction entre atomes et électrons qui sont influencés par un champ électrique. Pour les métaux utilisés dans les circuits intégrés, la dernière force tend à dominer. Quand les électrons entrent en collision avec les atomes de métal, ils bougent vers la fin positive du conducteur tandis que les lacunes (ions positifs formés par cet effet) bougent dans l'autre sens (migration vers la deuxième fin du conducteur). Si la charge positive (ions cumulés sur la deuxième fin) augmente en volume, alors la densité de courant augmente. Par cet effet, de nouveaux atomes métalliques (ions) auront immigrés. L'augmentation en charge entraîne ainsi une augmentation en densité de courant entraînant à son tour une nouvelle augmentation en charge et la boucle se répète ainsi. Par conséquent, le métal peut perdre ses caractéristiques et se comporter comme un isolant. Pour cela, un circuit ouvert sera établi.

2.2.2 Fautes temporaires

Généralement, les fautes temporaires sont des fautes physiques principalement dues à l'environnement du travail. Elles sont beaucoup plus difficiles à détecter que les fautes permanentes, car elles arrivent aléatoirement. Les fautes temporaires peuvent être aussi classées en deux groupes :

1-Les fautes transitoires.

2-Les fautes intermittentes.

Les fautes transitoires tendent à être aléatoires. Une faute transitoire peut être causée par des particules ionisées frappant le semi-conducteur ou par l'interférence électromagnétique. Les fautes intermittentes se répètent aléatoirement et apparaissent, habituellement, dans la même partie d'un circuit. Les fautes intermittentes peuvent se manifester dans certaines conditions, par exemple dans le cas d'un composant surchargé. Les différentes sources de fautes temporaires comprennent : le bruit électrique [SRHA84], l'interférence électromagnétique [FoRi79, RuSa89] et les particules ionisées frappant le silicium [Goor91, May79, MaWo79, RuSa89, SRHA84,].

Bruit électrique

Le bruit électrique, sous la forme d'un bruit thermique et bruit de vacillement de signal dans les circuits électroniques est un problème déjà connu. C'est ainsi qu'il peut être une source importante de fautes temporaires. Ce type de bruit peut apparaître si les dimensions du transistor pendant la phase de conception sont réduites sans beaucoup tenir compte des autres paramètres (dissipation de l'énergie, tension de l'alimentation, …etc.). Si un composant est surchargé pendant une période quelconque, alors il peut être le siège d'un bruit dépassant la marge permise et le circuit peut être le siège d'une faute intermittente.

Interférence électromagnétique

L'interférence électromagnétique a deux sources principales : de l'extérieur du circuit de la radiation électromagnétique et de l'intérieur de la fluctuation de l'alimentation et du couplage capacitif entre deux lignes du signal adjacentes.

Généralement, les circuits VLSI sont protégés contre la plus grande part des radiations externes par des moyens convenables. Par contre, il est très difficile de protéger ces circuits contre les radiations de fréquence et d'énergie élevées (radiation micro-onde), car ces sources peuvent induire des pulsations dans les fils métalliques du circuit imprimé et cela, à son tour, peut réduire les marges de bruit dans le circuit. Par conséquent une faute intermittente sera introduite.

L'origine de fautes temporaires qui sont générées intérieurement est : la fluctuation de l'alimentation et le couplage capacitif entre deux conducteurs. La fluctuation de l'alimentation ainsi que les mécanismes de commutation (passer d'une logique à une autre) peuvent générer des courants transitoires considérables pouvant entraîner des fautes dans le

circuit. Le couplage capacitif entre des conducteurs peut changer la tension d'un conducteur. Ce changement peut induire un effet dans le conducteur adjacent et entraîner une faute intermittente dans le circuit.

Particules ionisées

Les radiations Alpha et Bêta représentent des sources majeures de radiations induisant des fautes dans les circuits VLSI. Si Alpha ou Bêta particules frappent un semi-conducteur, alors des paires d'électron-trou peuvent être générées tout au long du chemin de propagation de la radiation. Les paires d'électron-trou générées par cette méthode d'ionisation créent deux courants. Si les paires sont générées dans la zone de déplétion (région du champ électrique élevé) d'un transistor NMOS, alors les électrons seront ramassés rapidement dans la région du canal tandis que les trous vont rejoindre le substrat. Le courant résultant de cet effet (appelé «drift current » en anglais ou courant des porteurs majoritaires) se produit pendant une courte période de temps. Pour se faire, les porteurs de charge générés sont ramassés avant qu'ils ne se réunissent. Si la fin du chemin de radiation est dans la région de déplétion, alors le mécanisme de drift current sera le problème majeur.

Le deuxième courant se produit quand les paires d'électron-trou sont générées hors de la région de déplétion. Dans ce cas, un courant de diffusion dû au gradient de concentration de porteurs de charge sera produit. Les porteurs de charge se diffusent lentement jusqu'à ce qu'ils arrivent à la région de déplétion où le courant de glissement se produit. Par cet effet, (mouvement ralenti des porteurs), le courant de diffusion se produit pendant une période plus longue que celle du drift current. C'est ainsi que les composants du courant de diffusion peuvent être source de problèmes majeurs car ils sont capables de se déplacer loin et de produire des erreurs corrélées.

Un autre élément important à considérer est l'angle sur lequel les particules frappent les circuits intégrés. Si l'angle d'incidence est petit, alors les particules peuvent affecter un ou plusieurs nœuds, ce qui fait que plusieurs erreurs corrélées peuvent être présentées.

A titre d'exemple, si une particule (Alpha) frappe une mémoire NMOS qui contient la valeur logique '1' et si la charge produite par cet effet est plus grande que celle nécessaire pour maintenir cette valeur, alors la valeur logique sera changée à '0'.

2.2.3 Modèles de Fautes

Les modèles de fautes sont les moyens de description d'effets de fautes pouvant produire des erreurs à la sortie du circuit sous test. Afin de comprendre le comportement de fautes et générer des tests pour les détecter ou concevoir des circuits de détection, les fautes sont exprimées au moyen des modèles appelés modèles de fautes. Développer un modèle décrivant bien l'effet de faute est un problème très compliqué. De toute façon, il existe actuellement trois niveaux pour modéliser les fautes [ABFr90, AbFu86, Goor91, RuSa89].

1-Modèles de fautes au niveau du transistor (Modèles structurels).
2-Modèles de fautes au niveau de la porte logique (Modèles structurels).
3-Modèles de fautes au niveau de la fonction (Modèles fonctionnels).

Les modèles au niveau du transistor [AbFu86, RuSa89] sont utilisés pour représenter directement des fautes au niveau du transistor. Six types de fautes peuvent être modélisés ; nœuds connectés directement à la masse ou à l'alimentation du transistor, transistor toujours passant ou fermé, court-circuit ou circuit ouvert entre deux nœuds.

Le modèle de faute le plus connu au niveau de la porte logique est le modèle de collage [AbFu86, Goor91, RuSa89, WiPa83]. Ce modèle suppose que les entrées et les sorties de portes peuvent être collées en permanence à 0 ou à 1.

Les modèles de fautes au niveau du transistor ou au niveau de la porte logique peuvent donner une représentation précise ou très raisonnable de fautes pouvant se manifester dans les circuits. Les modèles au niveau du transistor ou au niveau de la porte logique sont convenables pour traiter des circuits de petites tailles.

Pour les circuits de grandes tailles, l'utilisation de modèles structurels est très compliquée car elle nécessite la connaissance de la structure du circuit. La solution consiste donc à générer des tests fonctionnels. Dans ce cas, toutes les fautes modélisées au niveau du transistor ou au niveau de la porte logique seront représentées dans le niveau le plus haut (niveau fonctionnel) [ABFr90, AbFu86, Goor91, WiPa83]. Les modèles fonctionnels sont utilisés pour générer des tests basés sur la description fonctionnelle du circuit sous test.

2.3 Techniques de test hors ligne

Les techniques de test hors ligne sont développées en vue de faciliter la génération et l'application de vecteurs de test pour détecter des fautes de circuit sous test. Compte tenu de la complexité des circuits, et pour garder les coûts de test acceptables, il est devenu populaire et nécessaire d'ajouter des moyens de test sur les designs. Ces techniques comprennent : les méthodes AD-HOC, Les approches structurées ou techniques de scan et les techniques de test intégré BIST («built-in-self-test » en anglais).

2.3.1 Les méthodes AD-HOC

Les techniques AD-HOC ajoutent du matériel supplémentaire sur la conception, soit pour améliorer la contrôlabilité et l'observabilité des signaux de nœuds internes du circuit à tester soit pour morceler le circuit en fonctions simples [ABFr90, GrNa80, RuSa89]. Ces méthodes ont été développées pour tester des circuits particuliers et essayer de résoudre des problèmes de test des circuits complexes. Les méthodes AD-HOC nécessitent un très bon niveau de compétence de la part du concepteur; par exemple savoir où placer les points optimaux de test. Les méthodes sont variées et dépendent des structures des circuits. Voici à titre d'exemple deux méthodes [RuSa89].

Insertion des points de test

Les points de test sont connectés au circuit à tester pour faciliter l'accessibilité aux nœuds internes du circuit et par conséquent pouvoir contrôler ou observer les valeurs des signaux internes. Si l'observabilité seulement d'un signal est nécessaire, alors le point de test peut être connecté directement à la ligne du signal, figure 2-3. Si l'ajout de ce nouveau point peut affecter les performances du composant chargé de la nouvelle connexion, alors il faut améliorer les performances de ce composant étant charge de la nouvelle connexion.

Les points de test sont utilisés, en général, pour pouvoir observer ou contrôler les signaux dans le circuit à tester. Pour pouvoir contrôler un signal sur une ligne, il est nécessaire d'insérer dans le circuit, et avant le nœud de connexion du point de test, figure 2-3, un

composant à trois états («tri-state-driver» en anglais) [RuSa89]. Pour contrôler le signal, le composant à trois états doit être dans l'état d'une impédance élevée.

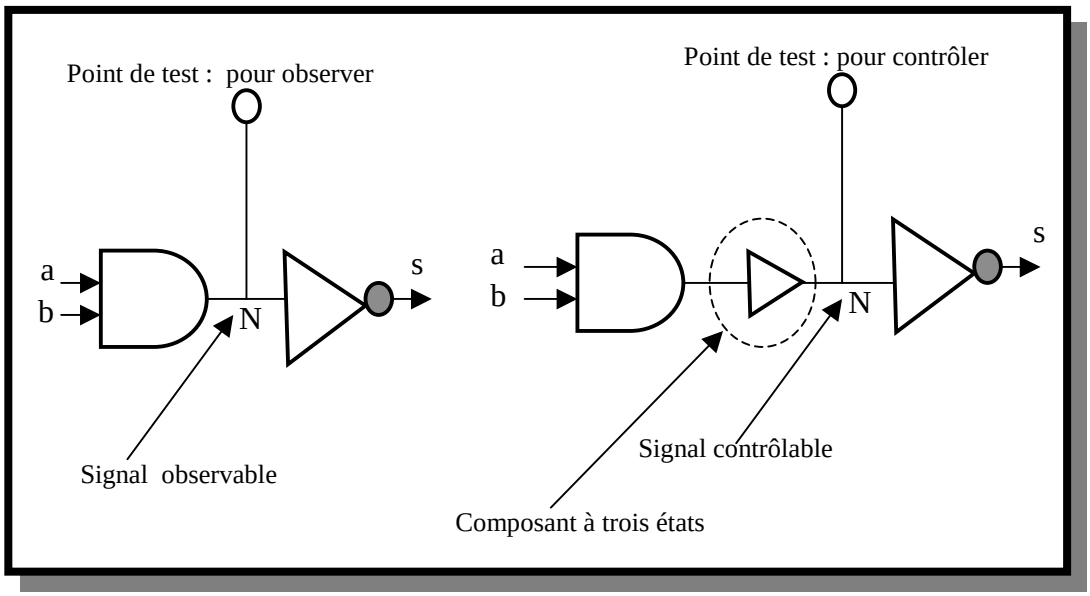


Figure 2-3 : Insertion des points de test

<u>Logique de blocage</u>

Le principe de cette technique est d'ajouter de nouvelles portes logiques sur le design pour contrôler le flux de données tout au long de certains chemins de propagation des signaux dans le circuit. Par conséquent, pour tester un circuit, la méthode morcelle le système en simples fonctions ou modules faciles à tester.

Les portes de blocage sont des portes simples à deux entrées ; L'une des entrées est l'entrée de signal de données normales (sortie secondaire sur le schéma) tandis que l'autre est l'entrée de signal de blocage (signal de contrôle) commandée par une entrée externe de test.

Le signal de blocage est gardé inactif pendant la phase de fonctionnement normal, c'est-à-dire qu'il a une valeur logique égale à '1' pour les portes AND et NAND, et une valeur égale à '0' pour les portes OR et NOR.

La figure 2-4 suivante illustre le principe de cette technique.

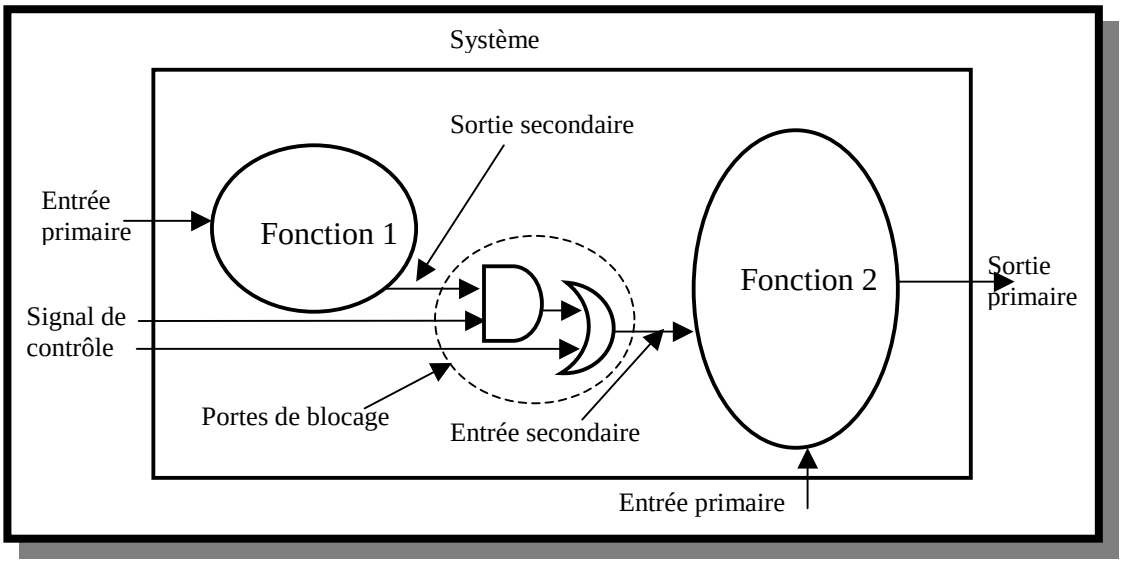


Figure 2-4 : Principe de la logique de blocage

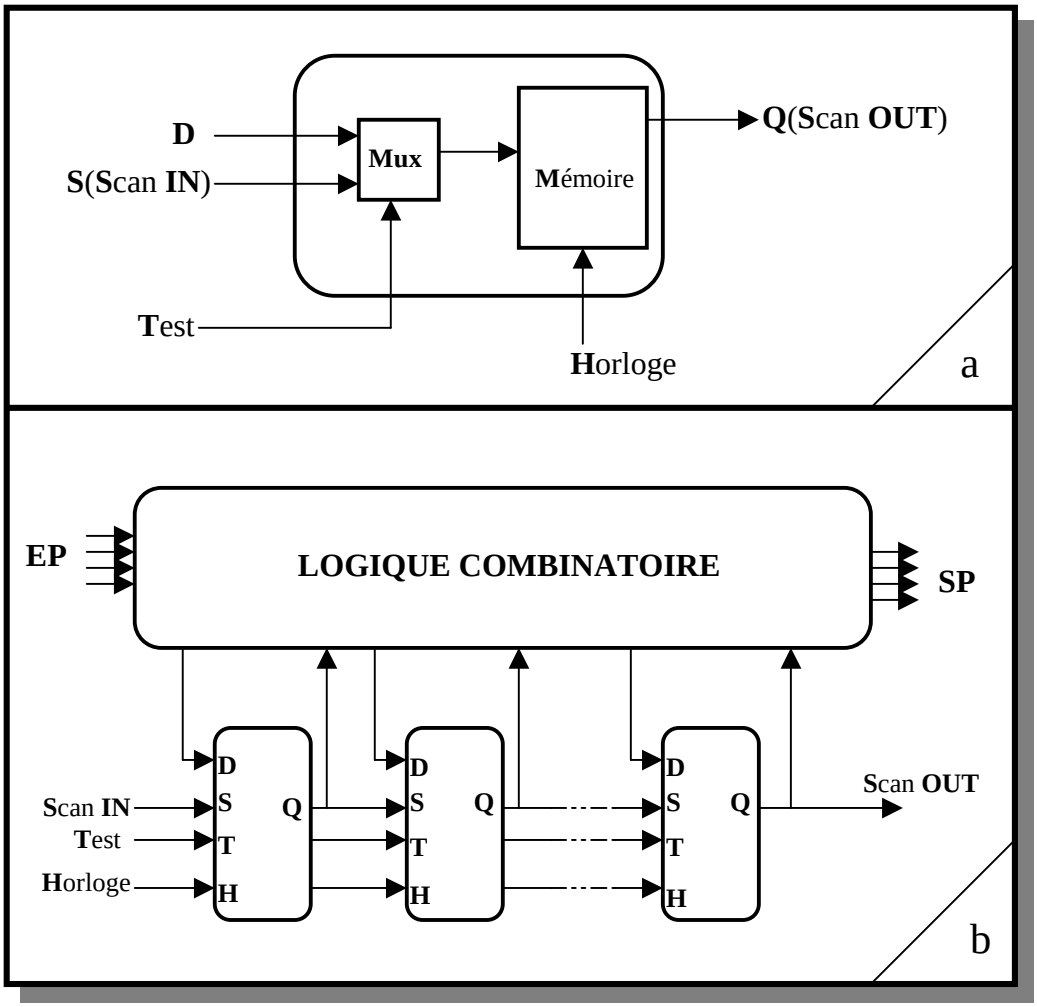
2.3.2 Les approches structurées

Les approches structurées sont développées pour faciliter le test des circuits séquentiels complexes. En effet, la réponse de sortie d'un circuit séquentiel dépend des entrées et de l'état interne du circuit. Donc le problème majeur du test des circuits séquentiels est de déterminer l'état interne. Pour cela, les approches structurées sont orientées pour améliorer la contrôlabilité et l'observabilité de l'état interne du circuit. Pouvoir contrôler et observer l'état interne peut ramener les problèmes de test du circuit séquentiel aux problèmes de test d'un circuit combinatoire. Le point de départ des approches structurées est d'utiliser les mémoires du circuit pour contrôler et observer les signaux dans le circuit. Parmi les premières approches, il existe la technique de scan-path [ABFr90, BMcS87, RuSa89].

Scan-path

L'objectif de la technique de scan-path est de transformer un circuit séquentiel, pour le but du test, en un autre circuit combinatoire. La transformation est assurée si toutes les mémoires internes peuvent se connecter, entre elles, pour former un registre de décalage, en série, appelé registre scan ou scan-path. Par conséquent, l'état interne du circuit peut être contrôlé et observé par ce registre. Ainsi, tester un circuit séquentiel contenant un registre

scan revient à tester un circuit équivalent combinatoire. Les circuits conçus avec scan-path ont, en général, deux modes d'opération : mode de fonctionnement normal et mode de test qui contrôle les cellules du registre scan pour accéder à l'état interne. La figures 2-5a et la figure 2-5b représentent respectivement une cellule scan avec ses connexions et une architecture scan-path.



Figures 2-5. a : cellule scan, b : L'architecture générale de scan-path

Si le signal de mode **T**est est égal à '1', alors la cellule accepte les données de l'entrée **S**can **IN** (Données du Test) sinon elle les accepte de l'entrée **D** (Données du Circuit).

Les démarches nécessaires pour tester un circuit avec scan-path sont :

1- Mettre **T**est = 1.

2- Décaler un vecteur de test dans les cellules pour contrôler l'état interne.

3- Appliquer un vecteur de test aux entrées primaires **EP**.

4- Mettre **T**est = 0, et permettre au circuit de se stabiliser et montrer ses sorties primaires **SP**.

5- Activer le signal d'horloge, **H**orloge, pour capter l'état suivant du circuit par les éléments de mémorisation (**M**émoire) dans les cellules.

6- Mettre **T**est = 1, et décaler les valeurs captées vers la sortie **S**can **OUT** pour être observées et analysées, en décalant un autre vecteur de test dans les cellules.

Outre de pouvoir contrôler et observer l'état interne, le registre scan découpe le circuit en simples modules [RuSa89], figure 2-6, tout en permettant à l'efficacité de génération des vecteurs de test d'être très améliorée.

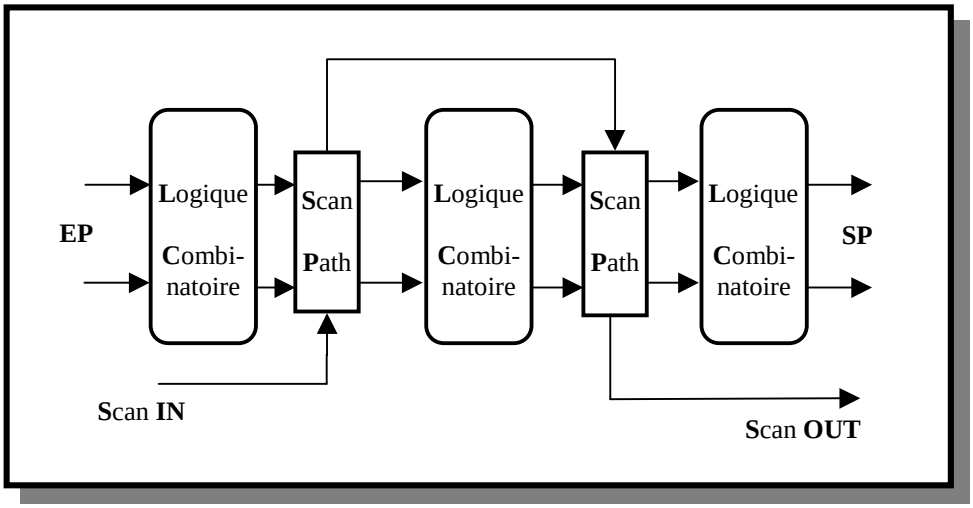


Figure 2- 6 : Decoupage en modules

Avant que le registre scan soit utilisé pour appliquer les vecteurs de test et capter leurs réponses, le registre lui-même doit être testé. Le test du registre scan peut être fait en deux opérations [RuSa89]. La première consiste à décaler dans le registre une séquence soit de '1's seulement (111... 1) soit de '0's seulement (000…0). La deuxième opération comprend le décalage de vecteurs de '1's et de '0's (par exemple '00110'). Par les deux opérations, chaque cellule est examinée pour tous les états possibles. Des exemples d'utilisation de la technique de scan pour tester des circuits digitaux se trouvent dans [DBHa92, Grob95, McCl84, McCl85, POBB88, TMcCL96].

2.3.3 Test intégré hors ligne (BIST hors ligne)

Compte tenu des fréquences de fonctionnement élevées des circuits intégrés actuels, les équipements automatiques de test doivent être extrêmement performants (fréquences élevées, mémoires importantes). Ils sont donc très coûteux. Par ailleurs, même avec les équipements de test les plus perfectionnés, il peut être impossible de tester certains blocs combinatoires d'un circuit à leur fréquence nominale dans les conditions d'utilisation normale. Pour pallier à ces problèmes, l'utilisation de technique de test intégré BIST peut s'avérer très intéressante. En effet, cette technique de test permet de réduire de manière significative le coût du test en déportant un certain nombre de fonctionnalités du testeur sur le circuit à tester.

Le test intégré est une technique de conception dans laquelle certaines parties du circuit sont utilisées pour le tester. L'utilisation d'une telle technique entraîne une modification de la structure du circuit de manière à le rendre totalement ou partiellement testable par lui-même.

Les méthodes de test intégré peuvent être classées en deux catégories : le test intégré en ligne et le test intégré hors ligne. Dans le test intégré en ligne, le test s'effectue pendant le fonctionnement normal du circuit afin de donner une image instantanée de la validité de son comportement. Ainsi, le test en ligne permet de tester le circuit de manière totalement autonome, ce qui s'avère particulièrement intéressant dans le cadre de la maintenance, ou du test, de systèmes embarqués (circuits pour l'aéronautique). Le test intégré hors ligne, quant à lui, signifie que le circuit est testé dans un mode de test particulier au cours duquel les fonctions normales du circuit sont désactivées. Cette approche est utilisée pour des applications moins critiques que celles du test en ligne. Dans ce cas, le comportement du circuit est vérifié régulièrement en passant du mode de fonctionnement normal au mode de test.

D'une manière générale, le test intégré est une technique permettant de réduire de façon significative le coût du test en déportant un certain nombre de fonctionnalités du testeur vers le circuit à tester. Pour cela, un (ou plusieurs) générateur(s) de vecteurs de test et un (ou plusieurs) analyseur(s) de signature sont inclus dans le circuit lors de sa conception [ABFr90, AlKi96, BMcS87, ChPa96, Coun93, DuVi95, GAYe95, GAYe96, GoOr97, More98,

POBB88, RuSa89, YeBe97]. Le schéma de principe du test intégré est présenté dans la figure2-7.

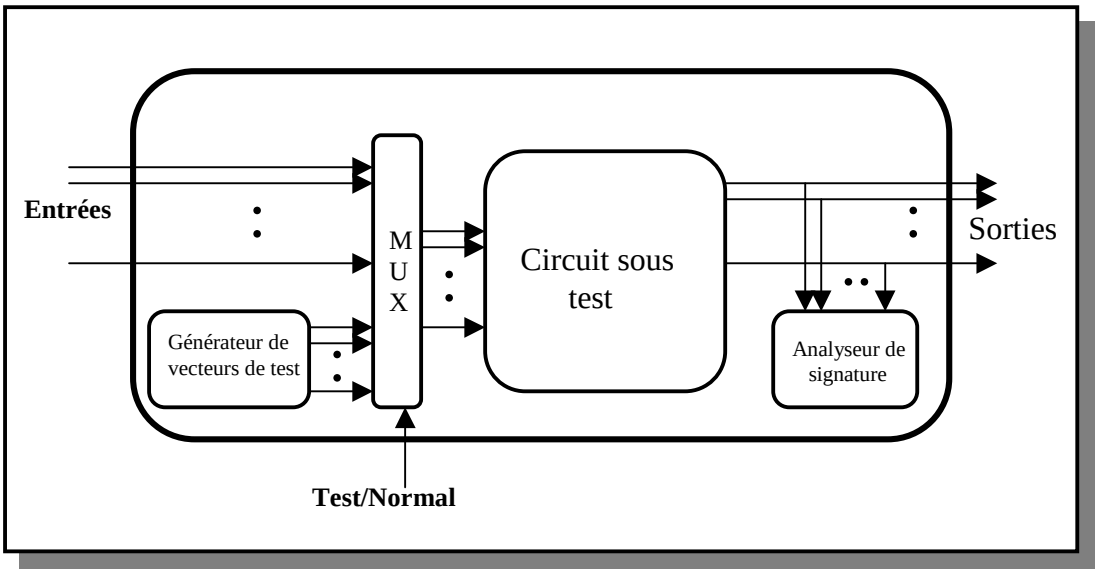


Figure 2-7. Schéma de principe du test intégré (BIST)

En mode test, les vecteurs produits par le générateur sont appliqués au circuit sous test au travers d'un multiplexeur. Afin de déterminer si le circuit est défectueux ou non, la réponse du circuit est comparée à celle du circuit sain en utilisant un analyseur de signature.

De nombreux générateurs de vecteurs de test et analyseurs de signature existent et sont disponibles dans la littérature [ABFr90, Beta95, BMcS87, ChGu95, Davi86, Davi98, DCBh98, DuVi95, DuYe97, Laty95, RuSa89, TMcCl96].

L'utilisation de techniques de test intégré présente de nombreux avantages par rapport aux techniques de test classiques. Ce type de test permet notamment de réduire le coût du test (compte tenu de la non-utilisation de testeurs complexes et onéreux) et d'améliorer la contrôlabilité et l'observabilité du circuit. De même, puisque le test peut être effectué à la vitesse nominale de fonctionnement du circuit, la vérification du bon fonctionnement logique du circuit à haute fréquence peut être assurée. De ce fait, le test intégré permet de détecter, outre les fautes de collage, l'occurrence d'un éventuel dysfonctionnement temporel.

2.4 Techniques de test en lignes

Certaines applications qui utilisent des systèmes électroniques ne peuvent tolérer l'interruption de leur fonctionnement. Ce type d'applications nécessite de montrer l'état de systèmes (sains ou en panne) pendant leur fonctionnement normal. Dans ces conditions, il devient pratiquement impossible de tester les systèmes en utilisant les techniques de test hors ligne. Il apparaît de plus en plus évident que les systèmes du futur doivent inclure des moyens d'auto-test privilégiant évidemment les techniques de test en ligne telles que les techniques de redondance du matériel, de redondance du temps et les techniques de codage.

2.4.1 Techniques de redondance du matériel

La technique de redondance du matériel compte parmi les techniques primitives utilisées pour la détection de fautes en ligne des systèmes électroniques. Pour cette technique, des copies du système à tester sont utilisées [Meum56, RuSa89, SiMc73]. La conception la plus simple qui peut illustrer la technique est le **TMR** («Triple Modular Redundancy» en anglais) [Neum56, RuSa89]. Dans cette conception, figure 2-8, le système **S** est triplé et les sorties **x, y** et **z** de copies sont connectées à un élément de vote de majorité **M**.

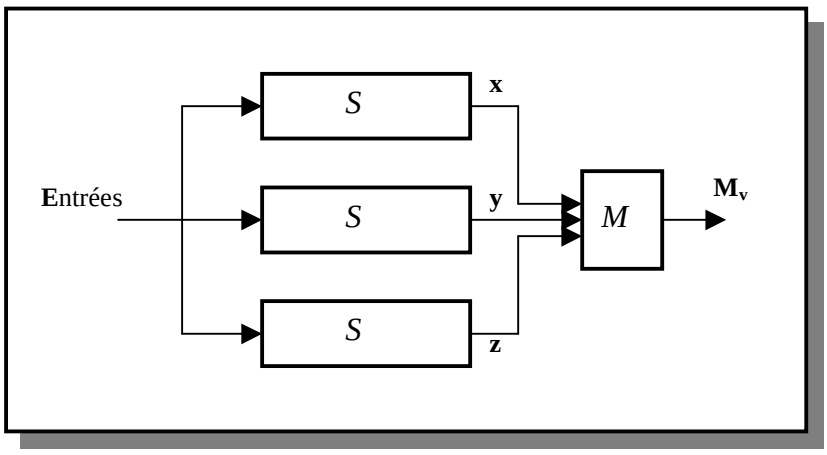


Figure 2-8: TMR

L'élément de vote génère la fonction de majorité $\mathbf{M_v = x.y + x.z + y.z}$. Si une seule copie est fautive, alors les deux autres seront dominantes et la faute sera masquée. Si une

autre copie peut être fautive, alors le résultat peut être erroné. Le concept de **TMR** est la forme de base de la technique de masquage de fautes. De toute façon, il est possible d'utiliser **n** copies identiques. Avec **n** copies, le schéma peut tolérer **n**/2 copies fautives.

La redondance du matériel où les copies ont des points fixes de connexion avec l'élément de vote est appelée redondance statique. Selon cette condition, les copies fautives ne peuvent être remplacées. Par contre, il y a une autre méthode pour inclure des copies dans le schéma. Cette méthode correspond à la redondance dynamique [SiMc73]. Dans le cas de la redondance dynamique, les copies fautives, lorsqu'elles tombent en panne, sont remplacées automatiquement par de nouvelles copies. Les copies de remplacement, selon les critères de l'application, peuvent être gardées avec ou sans alimentation.

2.4.2 Technique de redondance du temps

La technique de redondance du temps est une technique basée sur la répétition d'opérations dans le circuit sous test. Les avantages de cette technique par rapport aux autres sont : le matériel supplémentaire nécessaire pour l'implémentation est négligeable et la dégradation de performances du circuit sera banalisée. Pour cette technique, chaque opération est répétée plusieurs fois avant que le circuit ne délivre le résultat final. Les sorties, à chaque fois, sont stockées pour la comparaison [ChMa85, LaPa83, RuSa89]. Par exemple, un additionneur peut être testé en répétant l'opération de l'addition trois fois [LaPa83]. Tout d'abord, l'opération d'addition est effectuée sur les données A, B selon la relation de sortie S = A + B et la valeur de sortie obtenue est stockée. La deuxième opération sera effectuée selon la relation S = B + A et la nouvelle valeur de sortie est comparée avec la valeur stockée. chaque différence est marquée et stockée. La troisième opération d'addition sera effectuée selon la première relation S = A + B et l'opération de comparaison se répète en remarquant s'il y a un désaccord entre la valeur calculée et la valeur stockée. S'il n'y a pas de désaccord, alors le résultat obtenu à la première étape peut être utilisé, il n'y a pas de faute. S'il y a des désaccords, alors il est possible de déterminer les bits erronés et de les corriger.

2.4.3 Technique de redondance de l'information ou techniques de codage

Les techniques de codage représentent les moyens les plus importants utilisés aujourd'hui pour tester en ligne les systèmes électroniques. En effet, les techniques de codage augmentent l'information nécessaire pour représenter les données. Donc elles font référence à la technique de redondance de l'information. En général, les différents types de codes utilisés effectuent la même idée de base : celle de transformer le mot de données par des opérations logiques sur les bits de données en un autre mot codé et augmenté en taille. Parmi les codes les plus connus, on compte les codes de parité [Hamm50, PrRe72, RuSa89] et les codes de résidu [MoRa72, RuSa89]. Chaque mot codé comporte les bits de mot de données original (bits d'information) et les bits de vérification («check bits » en anglais) résultant de l'opération de codage. Les bits de vérification sont construits pour examiner les bits d'information. La figure 2-9 illustre le principe de cette technique de codage.

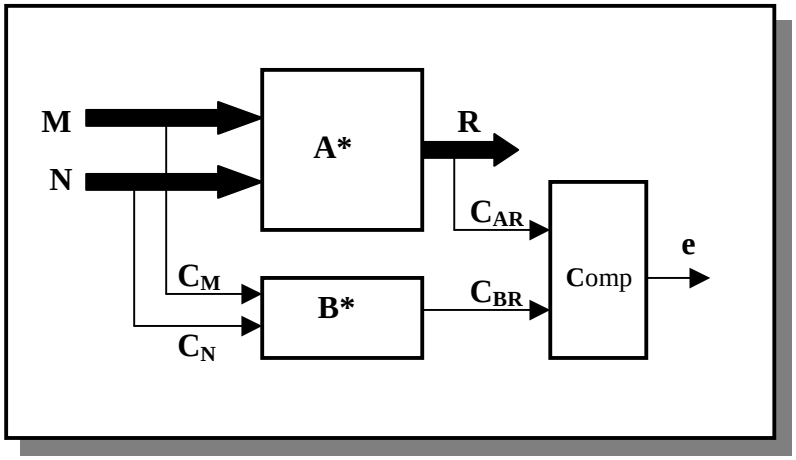


Figure 2-9: Principe de la technique de codage

Le bloc **A*** représente un système effectuant la fonction «*****» sur les données d'entrées. La sortie du bloc est donnée par **R = M*N**. Les mots de données **M** et **N** sont codés, et les bits de vérification $\mathbf{C_M}$ et $\mathbf{C_N}$ seront fournis au bloc supplémentaire **B*** qui effectue la même fonction «*****» sur $\mathbf{C_M}$ et $\mathbf{C_N}$. La sortie **R** est codée et les bits de vérification $\mathbf{C_{AR}}$ qui en résultent sont compares avec $\mathbf{C_{BR}}$ par le bloc **C**omp. Si **M**, **N**, **R** sont codés par le même code, alors $\mathbf{C_{AR}}$ et $\mathbf{C_{BR}}$ doivent être égaux s'il n'y a pas de faute et **e** = 0.

2.4.3.1 Codes de parité

Le code de parité le plus simple comprend l'ajout d'un seul bit de vérification sur le mot de données. Le nouveau bit, appelé bit de vérification de parité **p**, examine le nombre de 1s et de 0s dans le mot de données. Si le nombre de 1s est impair, alors la valeur logique du bit de vérification de parité sera **p**= 1 sinon **p**= 0. Par exemple, si les bits d'information dans le mot de données sont **MI** = 00110, alors **p**= 0. Il est clair que le bit de vérification changera sa valeur à **p**= 1 pour n'importe quelle erreur simple de bit du mot de données **MI**. Donc l'erreur sera détectable. Le code de parité simple ne peut détecter que les erreurs simples, mais il peut être développé pour la détection des erreurs multiples [Hamm50]. Le code de parité de l'ordre **n** (**n** bits de vérification sont ajoutés) peut détecter **n** erreurs produites en même temps dans le mot de données. Les bits de vérification, en général, sont formés par des opérations logiques **XOR**s entre les bits d'information.

2.4.3.2 Codes de résidu

Les codes de résidu représentent un groupe efficace de codes compatibles d'examiner des opérations arithmétiques ainsi que des opérations logiques. De plus, ils sont capables de détecter des erreurs multiples[MoRa72, RuSa89]. Les bits de vérification pour les codes de résidu peuvent être formés par la division de bits d'information par une base donnée, **b**. Le reste de la division donne les bits de vérification. Par exemple, considérons un nombre **I**, lequel est égal à $\mathbf{I} = \mathbf{k.b} + \mathbf{r}$, **k** est une constante et **r** est le reste de la division ($0 \leq \mathbf{r} < \mathbf{b}$), donc **r = I** *modulo* **b**. Le nombre de bits de vérification, pour un code, est donné par $[\log_2(\mathbf{b})]$. Par exemple si **b** = 7, alors le nombre de bits de vérification nécessaire est égal à 3. Si **I** = 011111 alors **r =** 011 (puisque cela est égal à 31 modulo 7 = 3). Les codes de résidu les plus importants sont les codes **LCR** («**LCR** = **L**ow **C**ost **R**esidue » en anglais). Pour ces codes, la base de vérification est $\mathbf{b} = 2^{\boldsymbol{a}} - 1$, ($\boldsymbol{a} \geq 2$). Ainsi, l'exemple précédant de **b** = 7 est un code **LCR**. L'avantage principal de ces codes est la facilité de construction du résidu. Au lieu d'effectuer une opération compliquée de division sur les bits d'information, il est tout simplement nécessaire de produire le résidu par un arbre d'additionneurs de modulos ($2^{\boldsymbol{a}} - 1$). L'arbre d'additionneurs, figure 2-10, peut être formé ou construit avec des additionneurs de ***a***-bit avec la retenue de sortie bouclée sur la retenue d'entrée.

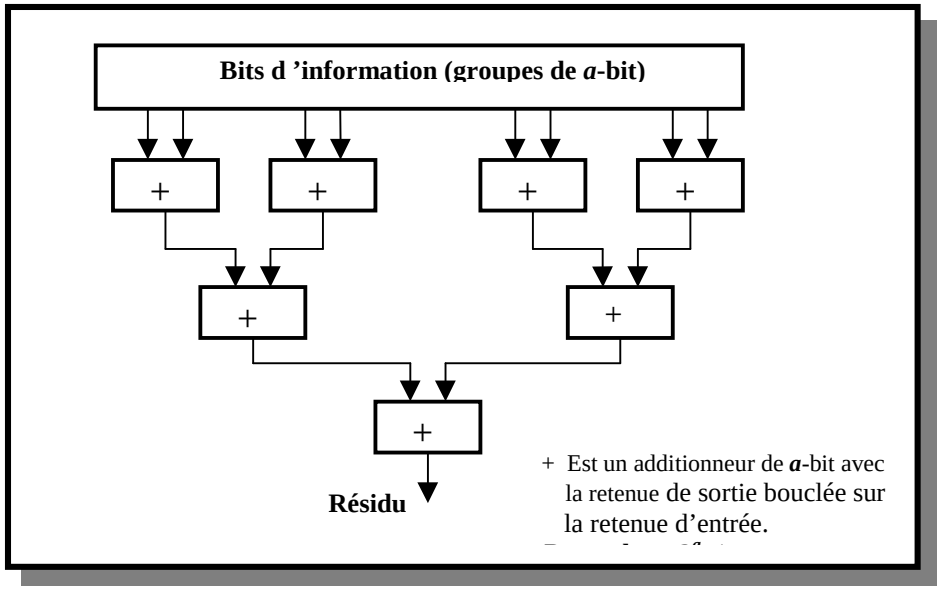


Figure 2-10. Générateur du résidu (bits de vérification)

2.4.4 Test en ligne des systèmes digitaux complexes

Les nouvelles technologies du VLSI ont permis d'intégrer des millions de transistors dans une seule puce de silicium. Ceci a permis de réaliser des systèmes intégrés complexes comprenant des unités fonctionnelles qui coopèrent entre elles pour réaliser une fonction complexe. Citons à titre d'exemple les systèmes multiprocesseurs de calcul et les DSPs.

Compte tenu de la complexité des nouvelles générations de systèmes, l'utilisation des techniques traditionnelles de test en ligne comme celles qui sont déjà exposées peut être coûteuse et très compliquée. Cela a amené l'adoption des nouvelles techniques de test basées sur l'utilisation des codes arithmétiques comme par exemple les « check sum codes » [ChAb93, ChCh99, HuAb84, JoAb86, JoAb88, NaAb90, ReBa90].

L'utilisation de techniques de «check sum codes» pour tester des systèmes digitaux complexes peut être illustrée par les exemples suivants.

2.4.4.1 Test des structures multiprocesseurs

Certains domaines d'application qui ont recours aux notions de calcul matriciel pour résoudre leurs problèmes requièrent une performance élevée des systèmes de calcul. Grâce aux nouvelles technologies VLSI, il est possible de pallier ce problème par la réalisation de systèmes multiprocesseurs dans lesquels la tâche de calcul est distribuée sur les processeurs.

Certaines applications critiques exigent en plus une sûreté de fonctionnement élevée. Des niveaux de tolérance aux fautes doivent alors être introduits. La figure 2-11 illustre l'utilisation de «check sum codes» pour que le résultat (*C*) de la multiplication de deux matrices (*A* et *B*) effectuée dans un réseau de processeurs interconnectés soit valide, même si un processeur est défectueux [HuAb84].

Le principe de la méthode est le suivant :

La matrice *A*, de dimensions $n \times m$, est codée par un vecteur de codage $v^T = [1\ 1\ 1\ \ldots 1]$. Cela donne une nouvelle matrice appelée la matrice de vérification de sommes de colonnes de *A* telle que :

$$A_c = \left[\begin{array}{c} A \\ \hline v^T A \end{array} \right].$$

Les lignes de la matrice *A* occupent les *n* premières lignes et le vecteur de sommes de colonnes de *A* ($v^T A$) occupe la $(n+1)^{\text{ème}}$ ligne. La matrice A_c est de dimensions $(n+1) \times m$ et les éléments de vecteur de sommes de colonnes de *A* sont générés tels que :

$$a_{n+1,j} = \sum_{i=1}^{n} a_{i,j} \quad \text{pour} \quad 1 \le j \le m.$$

La matrice *B* de dimensions $m \times n$ est codée par le vecteur v^T pour avoir une nouvelle matrice appelée la matrice de vérification de sommes de lignes de *B* telle que :

$$B_r = [B \mid Bv].$$

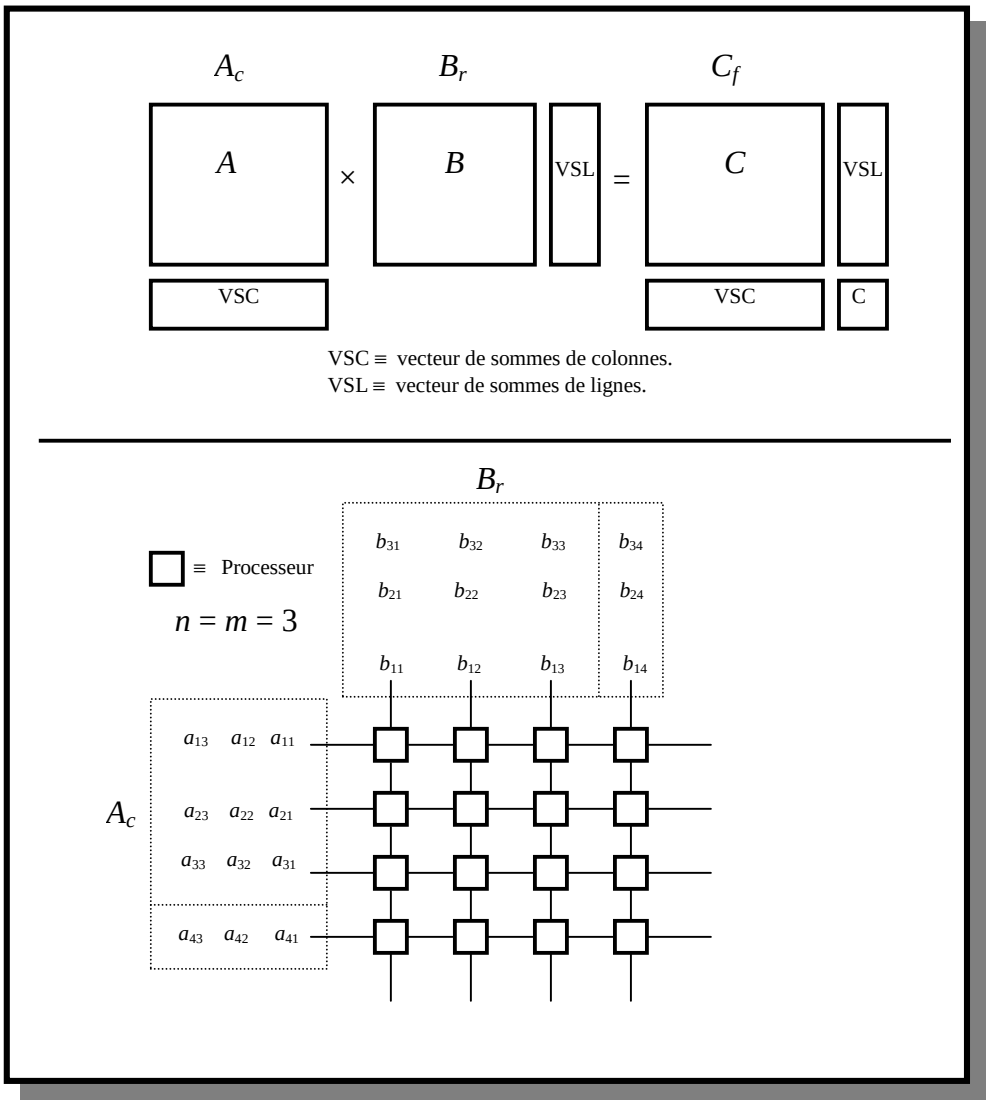


Figure 2-11. Multiplication de deux matrices dans un système multiprocesseurs

Les colonnes de matrice *B* occupent les *n* premières colonnes et le vecteur de sommes de lignes de *B* (*Bv*) occupe la $(n+1)^{\text{ème}}$ colonne.

La matrice B_r est de dimensions $m \times (n+1)$ et les éléments de vecteur de sommes de lignes de B sont générés tels que :

$$b_{j,n+1} = \sum_{k=1}^{n} b_{j,k} \quad \text{pour} \quad 1 \le j \le m.$$

Le résultat de la multiplication des matrices A_c et B_r donne théoriquement une nouvelle matrice (C_f) de vérification de sommes de lignes et de vérification de sommes de colonnes de la matrice C telle que :

$$C_f = A_c B_r = \begin{bmatrix} AB & | & ABv \\ \hline v^T AB & | & v^T ABv \end{bmatrix}$$

où les éléments de la matrice $C=AB$ occupent les premiers n colonnes et premières n lignes. Le vecteur v^TAB porte les sommes de colonnes de C et le vecteur ABv porte les sommes de lignes de C. La matrice C_f est de dimensions $(n+1)\times(n+1)$.

L'opération de multiplication ainsi que la mission de tolérance aux fautes peuvent être expliquées de la façon suivante :

Les données des matrice A_c et B_r sont fournies au réseau et les éléments de matrice C_f sont obtenus aux sorties des processeurs. A un temps donné j, le processeur $P_{i,k}$ localisé dans l'intersection de la $i^{ème}$ ligne et de la $k^{ème}$ colonne du réseau réalise la multiplication entre $a_{i,j}$ et $b_{j,k}$ et accumule le produit dans son registre. Ainsi, après m unités de temps (ou m coups d'horloge), chaque processeur calcule un élément de la matrice C_f.

Pour vérifier la validité de la matrice obtenue C, les n premiers processeurs de la ligne $(i+1)$ sont utilisés pour calculer la somme des composantes de la ligne i de la matrice obtenue C où $1 \leq i \leq n$ (la somme des n premières composantes de la ligne $(n+1)$ de la matrice C_f peut être effectuée par les n premiers processeurs de la première ligne du réseau). La somme calculée est comparée à la composante correspondante dans le vecteur de sommes de lignes de la matrice obtenue C_f (les composantes du vecteur de sommes de lignes sont disponibles dans les registres des processeurs de la $(n+1)^{ème}$ colonne du réseau). Chaque différence de la comparaison indique un ou plusieurs processeur(s) défectueux et le(s) composante(s) correspondante(s) de la matrice C_f sont erronée(s). L'identité de comparaison indique la validité de la matrice C.

Dans le cas d'une seule composante erronée dans la matrice obtenue C, c'est-à-dire dans le cas où un des processeurs calculant cette composante est défectueux, l'erreur peut être

corrigée. En réalité, une seule composante incorrecte dans la matrice C_f obtenue et localisée dans la matrice C correspond à deux différences de comparaison, dont une comparaison est obtenue avec le vecteur de sommes de lignes (comme expliqué précédemment) tandis que l'autre est obtenue avec le vecteur de sommes de colonnes. Donc, pour localiser le processeur défectueux et par conséquent localiser la composante erronée et la corriger, les processeurs de colonnes sont maintenant utilisés de la même façon que les processeurs de ligne pour calculer les sommes de colonnes de matrice obtenue C et les comparer avec les composantes de vecteur de sommes de colonnes de la matrice obtenue C_f. L'intersection entre les signatures des deux différences de comparaison localise le processeur défectueux et la composante erronée. La correction est effectuée par l'ajout d'une valeur sur le produit effectué par le processeur défectueux. Cette valeur est égale à la différence entre la somme calculée pour la comparaison et la somme correspondante dans le vecteur de sommes de la matrice C_f.

Si une seule comparaison présente une différence tandis que les autres sont identiques, alors un seul processeur calculant une composante des vecteurs de sommes de C_f est défectueux, la composante correspondante à ce processeur est incorrecte et la matrice C est valide. La correction dans ce cas, si elle est nécessaire, peut être effectuée au moyen de la remplacement de la composante erronée par la somme correspondante calculée pour la comparaison.

On trouve dans la littérature des méthodes variées qui utilisent les «check sum codes» pour tester des systèmes digitaux complexes de calcul [ChAb93, ChCh99, HuAb84, JoAb86, JoAb88, NaAb90, ReBa90].

2.4.4.2 Test des systèmes digitaux à variables d'état

Les systèmes digitaux linéaires à variables d'état représentent une sous-classe spécifique des systèmes digitaux linéaires. Ils sont construits avec des opérateurs linéaires tels que les multiplieurs linéaires (multiplication avec constante), les additionneurs et les registres [ChAb93, ChCh99]. La figure2-12 suivante représente la structure générale de ce type de systèmes.

z^{-1} représente un retard pour synchroniser le système. *L* est un circuit séquentiel synchronisé traitant des ensembles de données d'entrées qui sont des mots de données. *L* calcule une transformation linéaire des valeurs de ses *n*+*m* entrées et génère *n*+*p* valeurs de sorties.

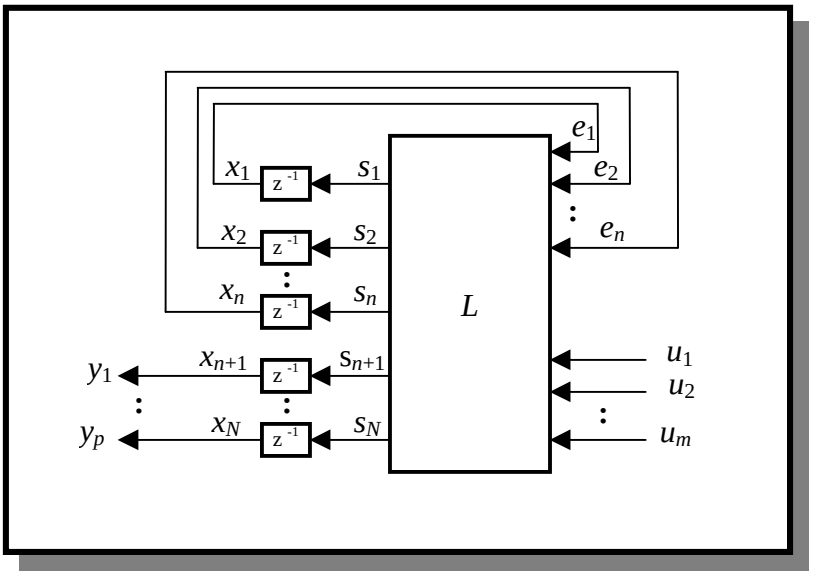


Figure 2-12. Représentation générale des systèmes digitaux linéaires à variables d'état

Les valeurs des mots de données fournies aux entrées $e_1, e_2,..., e_n,$ à un temps quelconque *t*, sont données par le vecteur :

$$[x_1(t), x_2(t),..., x_n(t)]^T \tag{2-2}$$

Les valeurs des mots de données obtenues aux sorties $s_1, s_2, \ldots, s_n,$ sont fournies au temps *t*+1 aux entrées correspondantes de *L*. Les valeurs de ces mots de données sont fournies par le vecteur :

$$[x_1(t+1), x_2(t+1),..., x_n(t+1)]^T \tag{2-3}$$

A chaque moment *t*, *L* génère *p* valeurs aux sorties $y_1, y_2, \ldots, y_p$. Les valeurs sont données par le vecteur :

$$[x_{n+1}(t), x_{n+2}(t),..., x_{n+p}(t)]^T \tag{2-4}$$

Considérant $N = n+p$, Le vecteur :

$$x(t) = [x_1(t), x_2(t),..., x_N(t)]^T \tag{2-5}$$

sera défini comme un vecteur d'état tandis que :

$$x(t+1) = [x_1(t+1), x_2(t+1),..., x_N(t+1)]^T \tag{2-6}$$

sera défini comme le vecteur d'état suivant.

$x_1, x_2, ..., x_N$: désignent les variables d'état associées aux retards, par conséquent elles sont associées aux vecteurs d'état $x(t+q)$, $q = 0, 1, 2, \ldots$ etc. $x_i(t+q)$: représente la valeur de la variable d'état x_i au temps $t+q$, et $1 \leq i \leq N$.

Le vecteur d'état $x(t)$ à un temps quelconque t est relié au vecteur d'état suivant au temps $t+1$ par l'équation d'état suivante:

$$x(t+1) = A.x(t) + B.u(t) \tag{2-7}$$

$u(t)$: est le vecteur d'entré de l'ordre m.
$x(t)$: est le vecteur d'état de l'ordre N.
$t+1$: défini l'instant suivant

Les matrices A et B sont des matrices de dimensions appropriées.
L'équation d'état peut être réécrite telle que :

$$\begin{bmatrix} x_1(t+1) \\ x_2(t+1) \\ : \\ x_N(t+1) \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & .. & a_{1N} \\ a_{21} & a_{22} & .. & a_{2N} \\ : & : & : & : \\ a_{N1} & a_{N2} & .. & a_{NN} \end{bmatrix} . \begin{bmatrix} x_1(t) \\ x_2(t) \\ : \\ x_N(t) \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} & .. & b_{1m} \\ b_{21} & b_{22} & .. & b_{2m} \\ : & : & : & : \\ b_{N1} & b_{N2} & .. & b_{Nm} \end{bmatrix} . \begin{bmatrix} u_1(t) \\ u_2(t) \\ : \\ u_m(t) \end{bmatrix} \tag{2-8}$$

La mission du test de systèmes digitaux linéaires à variables d'état peut être effectuée en calculant une nouvelle variable d'état (variable de vérification) dans un circuit supplémentaire. Le calcul peut être effectué en utilisant la technique de «real number codes » qui est une sous-classe puissante de «check sum codes». L'utilisation de «real number codes» et le principe de la méthode de test peuvent être expliqués [ChAb93, ChCh99] comme suit.

Les matrices A et B sont codées par un vecteur ligne, $CV = [\alpha_1, \alpha_2, \ldots, \alpha_N]$, dont les éléments α_i de vecteur sont des nombres réels, $1 \leq i \leq N$. L'opération de codage donne deux nouveaux vecteurs ligne La et Lb tels que :

$$La = CV.A = [la_1, la_2, \ldots, la_N] \quad (2\text{-}9)$$

$$Lb = CV.B = [lb_1, lb_2, \ldots, lb_m] \quad (2\text{-}10)$$

Les deux nouveaux vecteurs ligne spécifient une nouvelle variable d'état, *c*, et l'équation d'état (2-8) peut être réécrite pour inclure la nouvelle variable, *c* :

$$\begin{bmatrix} x_1(t+1) \\ x_2(t+1) \\ : \\ x_N(t+1) \\ c(t+1) \end{bmatrix} = \begin{bmatrix} a_{11} & a_{i2} & .. & a_{1N} & 0 \\ a_{21} & a_{22} & .. & a_{2N} & 0 \\ : & : & : & : & : \\ a_{N1} & a_{N2} & .. & a_{NN} & 0 \\ la_1 & la & .. & la_N & 0 \end{bmatrix} . \begin{bmatrix} x_1(t) \\ x_2(t) \\ : \\ x_N(t) \\ c(t) \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} & .. & b_{1m} \\ b_{21} & b_{22} & .. & b_{2m} \\ : & : & : & : \\ b_{N1} & b_{N2} & .. & b_{Nm} \\ lb_1 & lb_2 & .. & lb_m \end{bmatrix} . \begin{bmatrix} u_1(t) \\ u_2(t) \\ : \\ u_m(t) \end{bmatrix} \quad (2\text{-}11)$$

La nouvelle variable d'état à temps quelconque (*t*+1) peut être exprimée par la relation :

$$c(t+1) = La.x(t) + Lb.u(t) \quad (2\text{-}12)$$

D'autre part, en se référant aux équations 2-9 et 2-10 et en développant l'équation 2-12, la variable *c*(*t*+1) est égale à :

$$c(t+1) = CV.A.x(t) + CV.B.u(t).$$
$$c(t+1) = CV.[A.x(t) + B.u(t)].$$
$$c(t+1) = CV.x(t+1) \quad (2\text{-}13)$$

Pour tester le système, la quantité *c*(*t*+1) de l'équation 2-12 est calculée dans un circuit appelé DCC («data check-sum circuit » en anglais) et comparée à la quantité *c*(*t*+1) de l'équation 2-13 calculée par un autre circuit appelé SCC («state check-sum circuit » en anglais), figure 2-13 suivante. La différence entre les deux quantités est donnée par l'équation :

$$r(t+1) = CV.x(t+1) - [La.x(t) + Lb.u(t)] \quad (2\text{-}14)$$

Maintenant, à chaque moment *t*, la différence exprimée par la variable *r*(*t*) est égale à zéro en l'absence de fautes, sinon *r*(*t*) est différente de zéro. Pour plus de détails sur cette méthode de test on peut se référer à [ChAb93, ChCh99].

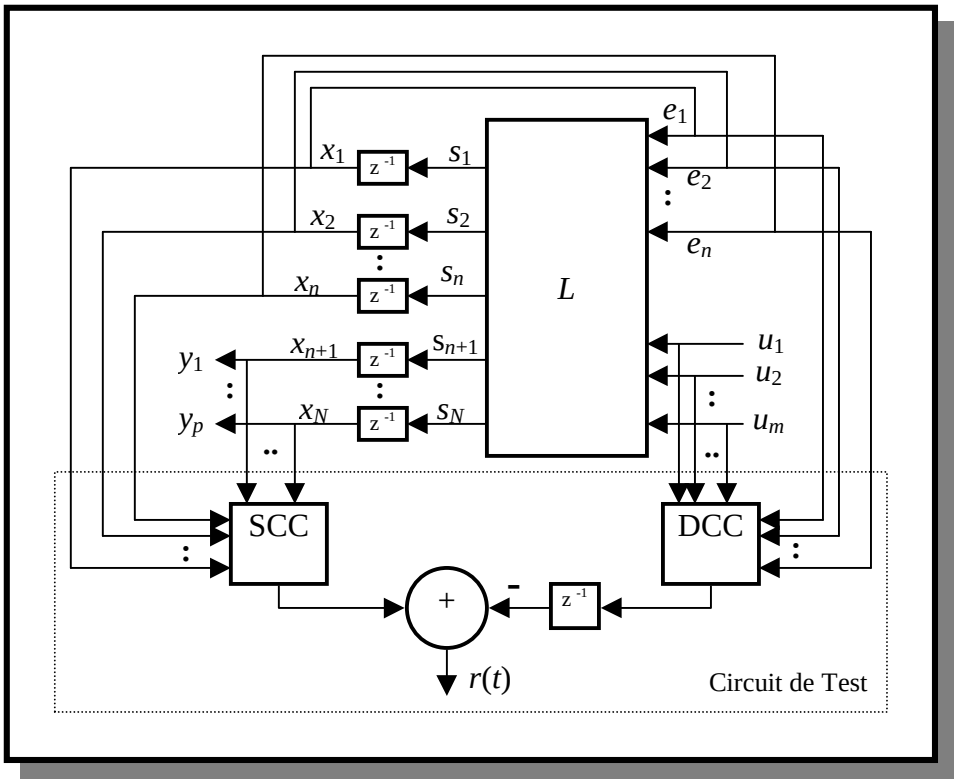


Figure 2-13. Le système digital linéaire à variables d'état avec le circuit de test

2.5 Conclusion

Le test des circuits intégrés est mis en œuvre pour détecter les fautes entraînant un dysfonctionnement dans le circuit sous test. Par conséquent, détecter le circuit en panne. Les fautes peuvent être des fautes permanentes liées au matériel ou fautes temporaires liées à l'environnement de travail. Dans ce chapitre, des techniques de test hors ligne et en ligne des systèmes digitaux ont été développées. Les techniques de test hors ligne, qui peuvent améliorer la contrôlabilité et l'observabilité des signaux des nœuds internes du circuit, sont utilisées pour tester des systèmes simples pouvant tolérer l'interruption de leur fonctionnement normal. Les techniques de test en ligne sont privilégiées pour tester des systèmes fonctionnant dans des domaines critiques d'application et ne tolèrent pas l'interruption de leur fonctionnement normal même quelques secondes par année. Pour des systèmes complexes, de nouvelles approches de test en ligne basées sur de nouvelles techniques de codage sont adoptées. Cependant, le problème de test des autres systèmes complexes, tels que les systèmes digitaux linéaires, n'est pas traité et adressé directement.

Chapitre 3 : Principe général de la méthode de test concurrent des systèmes digitaux linéaires

3.1 Introduction

Avec l'évolution des technologies VLSI facilitant l'intégration des systèmes de plus en plus complexes et à des coûts de plus en plus réduits, des systèmes intégrés spécifiques dédiés à des domaines importants d'application ont vu le jour.

Les systèmes digitaux linéaires représentent une classe spécifique de circuits utilisés dans de nombreuses applications critiques militaires, nucléaires, spatiales, pour lesquelles une défaillance en cours de fonctionnement peut entraîner des graves conséquences.

Jusqu'ici, les travaux de la recherche effectués dans le domaine du test en ligne des systèmes digitaux complexes [ChAb93, ChCh99, HuAb84, JoAb86, JoAb88, NaAb90, ReBa90, SeRa95] sont orientés vers des structures spécialisées tandis que le problème du test en ligne des systèmes digitaux linéaires n'est pas visé et adressé directement.

Ce chapitre propose une méthodologie permettant de concevoir des détecteurs de défauts en ligne pour les systèmes digitaux linéaires. La méthode utilisée est générale et applicable à tout système digital pourvu qu'il soit linéaire. Elle est basée sur l'exploitation de la redondance analytique fonctionnelle décrivant les relations entre les signaux des entrées et des sorties mesurables sur système sous test. Dans un premier temps, nous développons le principe de redondance analytique et nous en déduisons l'équation du détecteur. La conception matérielle du circuit de détection de fautes est ensuite discutée.

3.2 La méthode de détection de fautes

Avant d'aborder la méthode de détection de fautes, il nous paraît important de présenter les modèles de fautes utilisés pour cette méthode.

3.2.1 Nouveaux modèles de fautes

Généralement, les modèles de fautes représentent les moyens de description d'effets de fautes entraînant des erreurs à la sortie du circuit à tester. Ces modèles sont utilisés pour générer des tests convenables détectant le circuit atteint qui est le siège de fautes. Développer un modèle décrivant bien l'effet de faute est un problème très délicat et compliqué. Actuellement, deux classes de modèles sont utilisées : les modèles structurels au niveau de la porte logique et les modèles fonctionnels au niveau de l'unité fonctionnelle comme par exemple l'additionneur et multiplieur [AbFu86, ABFr90, Mang84, RuSa89]. Pour des circuits VLSI, les modèles fonctionnels s'imposent car les modèles structurels sont difficiles à utiliser. Pour les systèmes complexes comprenant de nombreuses unités fonctionnelles tels que les systèmes multiprocesseurs, l'utilisation de modèles structurels ou fonctionnels est très difficile, compliquée et peut être coûteuse car toutes les combinaisons binaires d'entrée de chaque porte logique ou chaque unité fonctionnelle doivent être générées. C'est la raison pour laquelle de nouveaux modèles, qui donnent une image du comportement de fautes dans les systèmes complexes, s'avèrent très important de développer.

Modèles de fautes comportementaux

Les nouveaux modèles comportementaux sont développés pour aider à tester l'efficacité des nouvelles techniques de test des systèmes complexes. Les fautes d'un système peuvent être exprimées au moyen d'un changement (déviation d'un paramètre de sa valeur nominale) des paramètres de description du système. Dans ce cas, les fautes sont modélisées au niveau du système par l'introduction de nouveaux termes dans le modèle mathématique comportemental décrivant le système à tester [Fran90]. Par exemple, si on a un multiplieur

effectuant l'opération de multiplication entre le signal d'entrée 'x' et la constante 'a', alors la sortie du multiplieur sera le signal 'y' donnée par la relation :

$$y = ax \tag{3-1}$$

L'équation 3-1 représente le modèle mathématique comportemental idéal du multiplieur. Les fautes, pouvant entraîner des erreurs 'Δy' à la sortie du multiplieur, peuvent être modélisées par l'ajout d'un nouveau terme 'Δa' dans l'équation 3-1, tel que :

$$\begin{aligned} y+\Delta y &= (a+\Delta a)x = ax + \Delta ax \\ \Delta y &= \Delta ax \end{aligned} \tag{3-2}$$

L'équation 3-2 représente le modèle réel du multiplieur et Δa modélise les fautes se reflétant dans un changement du paramètre nominal 'a'. La valeur Δax représente la valeur de l'erreur, à la sortie y, produite par les fautes. Si le multiplieur n'a pas de faute, alors Δa=0 et la sortie y est correcte, sinon $\Delta a \neq 0$ et $\Delta ax \neq 0$ et la sortie y sera erronée.

3.2.2 Redondance analytique et la méthode de détection de fautes

La redondance analytique [AbSi99, AbSi00, ChWi84, Fran90, LoWV86, SiAb01] décrit les relations entre les signaux d'entrées et de sorties accessibles à la mesure par connexion directe sur système sous test. Notre but est de pouvoir exploiter ces relations de la redondance pour fournir ou générer un signal de résidu portant des informations sur le fonctionnement du système. Nous voulons que le résidu soit nul (zéro) pour le fonctionnement sans fautes du système et diffèrent de zéro sous la condition de fautes. Pour satisfaire la condition de test concurrent, nous cherchons que ce résidu soit aussi indépendant des valeurs de signaux d'entrées. Pour les systèmes digitaux linéaires, les résultats des différentes opérations arithmétiques, additions et multiplications, doivent être codés sur un nombre fini prédéterminé de bits. Il y aura donc des arrondis et/ou troncatures à effectuer. La conséquence de ces mutilations du signal réel est qu'un signal de bruit, appelé le bruit du calcul, sera présent dans le système. Généralement, le bruit généré à l'intérieur du système n'est pas considéré comme fautes et doit être toléré. Pour cela, nous sommes intéressés à rendre le résidu seulement sensitif aux fautes même en présence de bruit dans le système.

La figure 3-1 illustre le principe de la méthode de génération du résidu, $r(t)$.

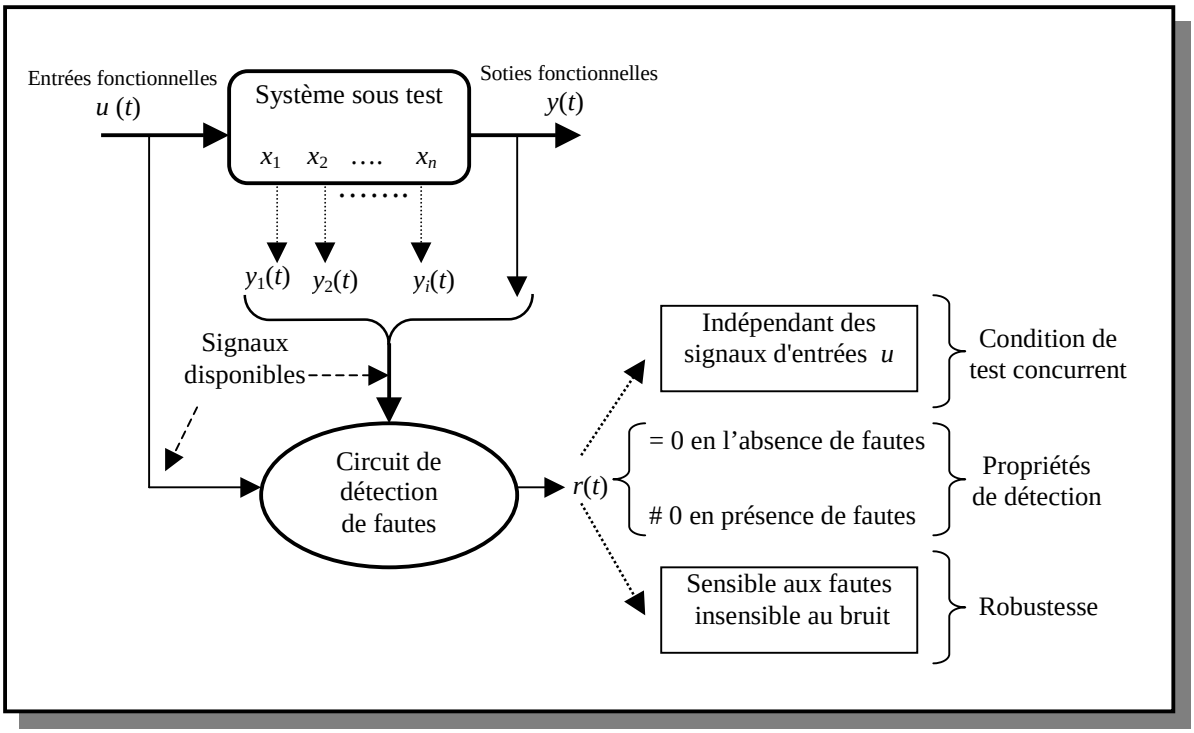


Figure 3-1. Principe de la méthode de génération du résidu

Le circuit de test (détection de fautes), implémentant la relation de la redondance et générant le résidu, est connecté aux entrées, sorties fonctionnelles et éventuellement aux autres points accessibles à la connexion dans le système sous test. Les entrées, les sorties fonctionnelles et les points de connexion accessibles représentent tout ce dont on dispose.

Un point de connexion dans le système sous test est supposé accessible si une connexion physique connectant le circuit de test sur ce point ne perturbe pas le fonctionnement correct du système. Pour le calcul de relations de la redondance, les points de connexion accessibles et utilisés pour la génération du résidu doivent être traiter comme sorties du système sous test. Si aucun point de connexion dans le système n'est accessible, alors nous sommes satisfaits des entrées/sorties fonctionnelles et c'est le cas traité dans ce chapitre. Le cas où des points de connexion sont accessibles sera traité dans le chapitre suivant.

Il existe deux formes de la redondance analytique : la redondance directe [ChWi84] et la redondance indirecte ou temporelle [AbSi99, AbSi00, ChWi84, Fran90, LoWV86].

3.2.2.1 Redondance directe

La redondance directe décrit les relations entre les signaux instantanés d'entrées et de sorties de systèmes sous test; ces relations (relations de la redondance analytique) peuvent être données par des équations mathématiques. En se référant aux équations d'état 1-7 et de sortie 1-8 (voir chapitre 1) des systèmes digitaux linéaires, la relation de la redondance analytique directe peut être exprimée par :

$$v^T y(t) = v^T Du(t) \tag{3-3}$$

où v est un vecteur colonne dont les éléments sont des nombres réels. Le vecteur v existe et l'équation 3-3 est valide s'il existe un espace vectoriel P orthogonal à C tel que :

$$P = \{ v | v^T C = 0 \} \tag{3-4}$$

où P est l'espace de parité et chaque vecteur v de P est un vecteur de parité. L'équation 3-4 peut être vérifiée si la matrice C a des lignes linéairement dépendantes, c'est-à-dire que C n'est pas de plein rang ligne (rang (C) < nombre de lignes de C). En posant :

$$r(t) = v^T y(t) - v^T Du(t) \tag{3-5}$$

le résidu $r(t)$ est nul en l'absence de fautes tandis que $r(t)$ est différent de zéro en présence de fautes car $y(t)$ est erronée. Sous la condition de fautes le résidu est donné par :

$$r(t) = v^T e(t) \tag{3-6}$$

où $e(t)$ est le vecteur de l'erreur lié au vecteur de sortie.

3.2.2.2 Redondance indirecte ou temporelle

Généralement, le cas de la redondance directe est rarement rencontré dans la pratique. Pour cela, et pour pouvoir toujours établir des relations de la redondance, nous utilisons le concept de la redondance temporelle. La redondance temporelle décrit les relations entre l'historique des signaux d'entrées et de sorties du système sous test. La relation de la redondance comprend donc les mesures instantanées et des mesures précédentes des signaux d'entrées et de sorties. Si la matrice *C* ne peut donc satisfaire les conditions de la redondance directe, alors nous retardons d'une fois les signaux d'entrées et de sorties, figure 3-2, en cherchant l'existence de la redondance temporelle dans les mesures des signaux instantanés et

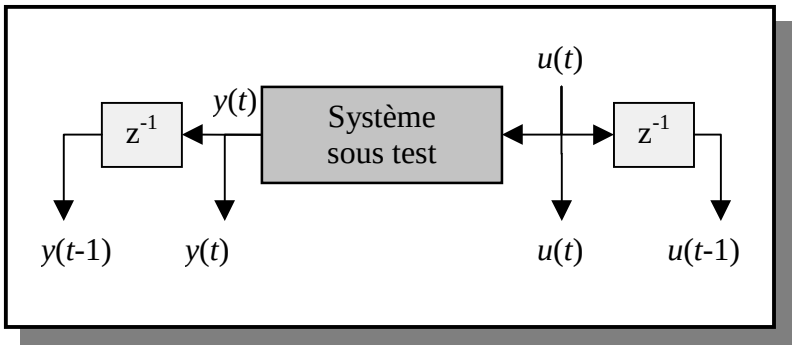


Figure 3-2: Retard d'une fois des signaux

retardés.

Les deux expressions successives du vecteur de sortie sont :

$$y(t\text{-}1)=C.x(t\text{-}1)+D.u(t\text{-}1)$$
$$y(t)=C.A.x(t\text{-}1)+C.B.u(t\text{-}1)+D.u(t) \quad (3\text{-}7)$$

Ce groupe d'équations peut se mettre sous la forme matricielle condensée suivante :

$$Y^{[1]}(t) = Ob^{[1]}.x(t\text{-}1) + H^{[1]}.U^{[1]}(t) \quad (3\text{-}8)$$

où :

$$Y^{[1]}(t)=\begin{bmatrix} y(t-1) \\ y(t) \end{bmatrix};\ Ob^{[1]}=\begin{bmatrix} C \\ CA \end{bmatrix};\ H^{[1]}=\begin{bmatrix} D & 0 \\ CB & D \end{bmatrix};\ U^{[1]}(t)=\begin{bmatrix} u(t-1) \\ u(t) \end{bmatrix}. \quad (3\text{-}9)$$

La redondance temporelle existe s'il y a un espace vectoriel P_1 tel que :

$$P_1 = \{ v | v^T Ob^{[1]} = 0 \} \qquad (3\text{-}10)$$

où P_1 est l'espace de parité de l'ordre 1, et chaque vecteur v de P_1 est un vecteur de parité.

Chaque vecteur de parité v de l'espace P_1 peut être associé à chaque instant t avec une relation de la redondance pour générer le résidu recherché, $r(t)$, tel que :

$$r(t) = v^T [Y^{[1]}(t) - H^{[1]}.U^{[1]}(t)] \qquad (3\text{-}11)$$

Si la matrice $Ob^{[1]}$ ne peut satisfaire l'équation 3-10, alors nous retardons de nouveaux les signaux, figure 3-3, en cherchons une autre matrice $Ob^{[k]}$ (k = 2, 3, …., n) qui peut vérifier l'existence de la redondance recherchée. Notons que pour $k = n$ l'existence de la redondance dans les mesures successives des signaux est toujours vérifiable.

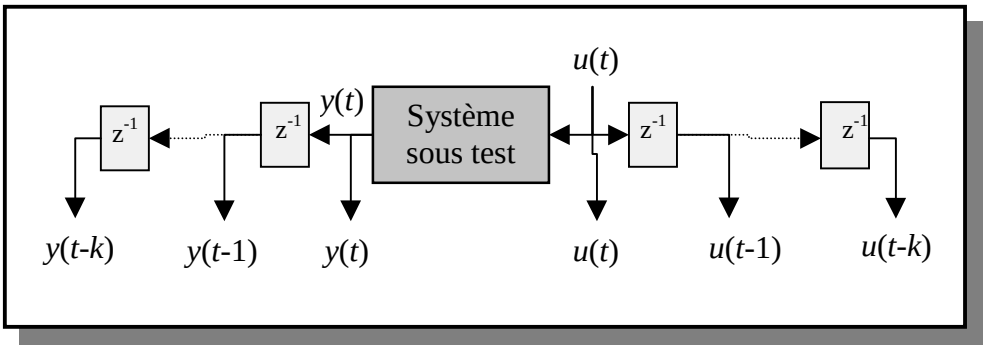


Figure 3-3: Retard de *k* fois des signaux

La méthode générale de la recherche de la redondance est détaillée dans le paragraphe suivant.

3.2.2.2.1 Relations de la redondance

Nous avons vu que les relations entre l'historique des signaux d'entrées et de sorties du système à tester peut être exprimée par une relation de redondance. Réellement, cette relation n'est pas unique et on peut avoir plusieurs relations de redondance. Celles-ci sont utilisées pour concevoir des circuits de détection de fautes et par conséquent générer des résidus servant comme indicateur d'erreur en cas de défaillance du système dans

l'accomplissement de sa mission correctement à cause de fautes. La méthode générale de détermination des relations de redondance est la suivante :

Nous avons vu que le comportement des systèmes digitaux linéaires peut être exprimé par les équations d'état 1-7 et de sortie 1-8. Pour le circuit de détection de fautes, il ne peut être connecté qu'aux points connectables du système à tester, donc le circuit de détection doit seulement utiliser les signaux accessibles directement à la mesure comme par exemple le vecteur d'entrée *u* et le vecteur de sortie *y*. Généralement, le vecteur d'état *x* n'est pas accessible directement à la mesure, il est donc considéré inconnu et ne doit pas être utilisé par le circuit de détection de fautes. Par conséquent, le vecteur d'état *x* doit être éliminé de l'équation 1-8. Le but est atteint en exprimant le vecteur de sortie *y* pour des instants successifs du temps (t-k, t-k+1, t-k+2, …, t) en termes de vecteur d'état $x(t\text{-}k)$ et la séquence d'entrée ($u(t\text{-}k)$, $u(t\text{-}k\text{+}1)$, $u(t\text{-}k\text{+}2)$, …, $u(t)$). Pour le fonctionnement sans fautes, les k+1 expressions successives du vecteur de sortie sont données par les équations suivantes :

$$y(t\text{-}k)=C.x(t\text{-}k)+D.u(t\text{-}k)$$
$$y(t\text{-}k\text{+}1)=C.A.x(t\text{-}k)+C.B.u(t\text{-}k)+D.u(t\text{-}k\text{+}1)$$
$$y(t\text{-}k\text{+}2)=C.A^2.x(t\text{-}k)+C.A.B.u(t\text{-}k))+C..B.u(t\text{-}k\text{+}1)+D.u(t\text{-}k\text{+}2)$$
$$\vdots$$
$$y(t)=C.A^k.x(t\text{-}k)+ C.A^{k\text{-}1}.B.u(t\text{-}k) +C.A^{k\text{-}2}.B.u(t\text{-}k\text{+}1)+\ldots+C..B.u(t\text{-}1)+D.u(t) \qquad (3\text{-}12)$$

Ce groupe d'équations peut se mettre sous la forme matricielle condensée suivante :

$$Y^{[k]}(t) = Ob^{[k]}.x(t\text{-}k) + H^{[k]}.U^{[k]}(t) \qquad (3\text{-}13)$$

où :

$$Y^{[k]}(t)=\begin{bmatrix} y(t-k) \\ y(t-k+1) \\ y(t-k+2) \\ \vdots \\ y(t) \end{bmatrix}; Ob^{[k]} = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^k \end{bmatrix}; H^{[k]} = \begin{bmatrix} D & 0 & .. & 0 & 0 \\ CB & D & : & 0 & 0 \\ CAB & CB & : & : & : \\ : & : & .. & D & 0 \\ CA^{k-1}B & CA^{k-2}B & .. & CB & D \end{bmatrix}; U^{[k]}(t)=\begin{bmatrix} u(t-k) \\ u(t-k+1) \\ u(t-k+2) \\ \vdots \\ u(t) \end{bmatrix}. \qquad (3\text{-}14)$$

L'entier *k* définira l'ordre de relation de la redondance, c'est-à-dire, le nombre de retards nécessaire pour chaque signal connecté au circuit de détection de fautes. Si *n* est

l'ordre du système, alors $Ob^{[k]}$ définit la matrice d'observabilité du système pour $k=n-1$. Généralement le vecteur d'état n'est pas mesurable directement à partir du système sous test, et il est donc considéré inconnu. Pour cela, pour construire un circuit de détection de fautes en utilisant seulement les variables disponibles et mesurables directement à partir du système (par exemple u et y), l'équation 3-13 peut être utilisée et le vecteur d'état $x(t-k)$ doit être éliminé. L'élimination du vecteur d'état de l'équation 3-13 donne une relation de redondance qui est une combinaison linéaire des valeurs instantanées et retardées des séquences d'entrée u et de sortie y. Le résidu associé à la relation de la redondance est égal à zéro en l'absence de fautes.

Condition 1 : l'élimination de $x(t-k)$ de l'équation 3-13 peut être réalisée s'il existe un espace vectoriel P_k tel que :

$$P_k = \{ v | v^T Ob^{[k]} = 0 \} \qquad (3\text{-}15)$$

où P_k est l'espace de parité de l'ordre 'k', et chaque vecteur 'v' de P_k est un vecteur de parité.

Condition 2 : l'équation 3-15 peut être vérifiée si la matrice $Ob^{[k]}$ a des lignes linéairement dépendantes, c'est-à-dire, P_k existe si le rang de la matrice $Ob^{[k]}$ est inférieur à son nombre de lignes $(p.(k+1))$.

Chaque vecteur de parité v de l'espace P_k peut être associé à chaque instant t avec une relation de la redondance pour générer un résidu, $r(t)$, tel que :

$$r(t) = v^T [Y^{[k]}(t) - H^{[k]}.U^{[k]}(t)] \qquad (3\text{-}16)$$

L'équation 3-16 définit une relation de la redondance et pour le fonctionnement sans fautes on aura:

$$v^T.Ob^{[k]}.x(t-k) = 0 \qquad (3\text{-}17)$$

En se référant à l'équation 3-13, on trouve que la relation simple de la redondance :

$$r(t) = 0 \qquad (3\text{-}18)$$

est satisfaite. En se référant à l'équation 3-16 on trouve que la relation de la redondance est tout simplement un modèle de 'entrée-sortie' du système.

De la discussion précédente, il est clair que le problème de la détection de fautes en ligne des systèmes digitaux linéaires peut être formulé maintenant comme suit : Trouver des vecteurs de parité $V = [v_1, v_2, \ldots, v_n]$ de l'espace de parité P_s tels que l'équation 3-15 soit vérifiée. Une fois les vecteurs trouvés, des relations de la redondance peuvent être établies, des circuits peuvent être conçus et des résidus indiquant la présence de fautes peuvent être générés.

Algorithme de détermination de relations de la redondance

Nous avons vu que le problème de détection de fautes en lignes des systèmes digitaux linéaires peut revenir à la détermination de vecteurs de parité. Pour calculer donc des vecteurs de parité et déterminer ensuite des relations de la redondance, l'algorithme de la figure 3-4 peut être utilisé. Cet algorithme propose des procédures pour calculer un groupe de vecteurs de parité qui est le fondement de toutes les solutions possibles de l'équation 3-15.

1-Initialisation $k = 0$, $Ob^{[k]} = C$.

2-Calcul du rang de la matrice $Ob^{[k]}$.

3-Test du rang, si rang < nombre de lignes ($p.(k+1)$) alors aller à 5.

4- $k = k + 1$, $Ob^{[k]} = \begin{bmatrix} Ob^{[k-1]} \\ CA^k \end{bmatrix}$, aller à 2.

5-Calcul de vecteurs de parité (V).

6-Détermination de relations de la redondance.

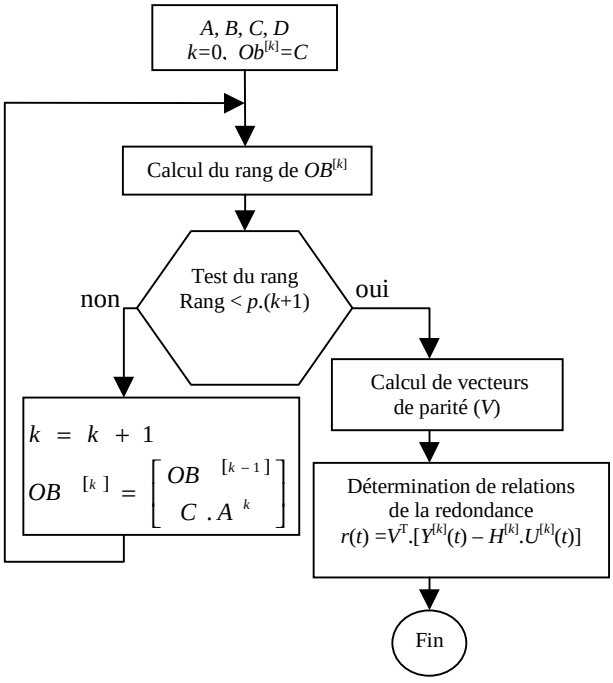


Figure 3-4. Algorithme de détermination de relations de la redondance

3.3 Conception des circuits de détection de fautes

Les vecteurs de parité peuvent être associés avec des relations de parité (relations de la redondance) comme il est indiqué dans l'équation 3-16. Chaque relation peut être utilisée pour construire un circuit de test (circuit de détection de fautes) dans lequel la sortie du circuit, à chaque instant *t*, est le résidu *r*(*t*). Le résidu est égal à zéro en l'absence de fautes. Les paramètres du circuit de test (les vecteurs, v^T et $v^T.H^{[k]}$, de relation de la redondance) peuvent être calculés en implémentant l'algorithme de la figure 3-5 dans un programme de calcul en utilisant l'un des langages de programmation.

Le circuit de test est construit avec des multiplieurs, des additionneurs et des mémoires pour la mémorisation des données d'entrées et de sorties du système à tester. Il faut noter que l'ordre de relation de la redondance, qui est en même temps l'ordre de l'espace de parité P_k, représente le nombre de fois dans lequel des valeurs d'entrées et de sorties, du système sous test, doivent être retardées (mémorisées) dans le circuit de détection de fautes. Ce nombre est représenté dans l'algorithme par la variable '*k*'. La méthode de conception est illustrée par l'exemple suivant :

3.3.1 Exemple 1

Soit le système digital linéaire, figure(3-5), suivant :

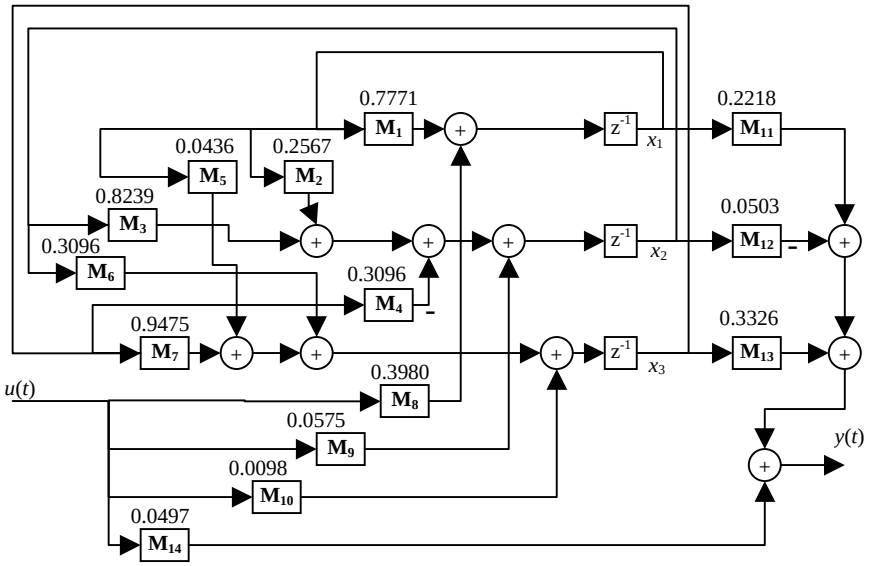


Figure 3-5. Graphe de l'écoulement de données d'un filtre digital passe-bas

Cela représente un filtre numérique passe-bas elliptique de l'ordre 3. Ce filtre a trois variables d'état (x_1, x_2, x_3), une seule entrée (u) et une seule sortie (y).

Les matrices d'état, *A*, *B*, *C*, *D*, de ce filtre sont :

$$A = \begin{bmatrix} 0.7771 & 0 & 0 \\ 0.2567 & 0.8239 & -0.3096 \\ 0.0436 & 0.3096 & 0.9475 \end{bmatrix} \qquad B = \begin{bmatrix} 0.3980 \\ 0.0575 \\ 0.0098 \end{bmatrix}$$

$$C = \begin{bmatrix} 0.2218 & -0.0503 & 0.3326 \end{bmatrix} \qquad D = \begin{bmatrix} 0.0497 \end{bmatrix}$$

Les paramètres du circuit de détection de fautes pour ce filtre ont été calculés selon les procédures de l'algorithme de détermination de relations de la redondance, ils sont :

1- L'ordre de relation de la redondance: $k = 3$.

2- Les vecteurs de parité: $V^{\mathrm{T}} = v^{\mathrm{T}} = \begin{bmatrix} -18.87 & 62.40 & -70.59 & 27.70 \end{bmatrix}$.

3- Le vecteur: $v^{\mathrm{T}}.H^{[3]} = \begin{bmatrix} 1.38 & -1.05 & -1.05 & 1.38 \end{bmatrix}$.

Pour la conception du circuit de test, la relation de la redondance suivante est utilisée :

$$r(t) = v^{\mathrm{T}}.\begin{bmatrix} y(t-3) \\ y(t-2) \\ y(t-1) \\ y(t) \end{bmatrix} - v^{\mathrm{T}}H^{[3]}.\begin{bmatrix} u(t-3) \\ u(t-2) \\ u(t-1) \\ u(t) \end{bmatrix} \qquad (3\text{-}19)$$

$$r(t) = \{-18.87y(t\text{-}3) + 62.4y(t\text{-}2) - 70.59y(t\text{-}1) + 27.7y(t)\}$$
$$-\{1.38[u(t\text{-}3) + u(t)] - 1.05[u(t\text{-}2)+u(t\text{-}1)]\} \qquad (3\text{-}20)$$

La synthèse du circuit de test (détection de fautes) peut être illustrée par la figure 3-6.

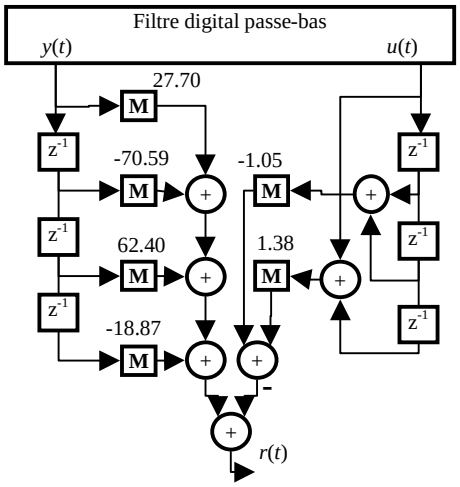


Figure 3-6. Circuit de détection de fautes

3.4 Conclusion

Dans ce chapitre, une nouvelle méthodologie de conception et d'intégration des détecteurs de défauts en ligne pour les systèmes digitaux linéaires est présentée. La méthode proposée de détection de fautes est basée sur l'exploitation de la redondance contenue dans les mesures des signaux d'entrées et de sorties du système sous test. Le circuit de test implémentant la relation de la redondance génère un signal de résidu servant comme indicateur d'erreur en présence de fautes dans le système. La méthode est générale et applicable à tout système digital pourvu qu'il soit linéaire.

Chapitre 4 : Implémentation optimale et robustesse des détecteurs concurrents de défauts

4.1 Introduction

Dans le chapitre précédent, nous avons présenté les principes de la méthode de détection concurrente de fautes. Nous avons considéré que seulement les signaux d'entrées et sorties fonctionnelles du système sont accessibles à la mesure directe.

Dans ce chapitre nous considérons le cas où des points supplémentaires de connexion sont aussi accessibles à la mesure par connexion directe sur système sous test. Nous traitons aussi le problème de la robustesse du résidu ainsi que le problème de la couverture de fautes.

4.2 Minimisation de la circuiterie de détection de fautes

L'implémentation du circuit de détection de fautes, pour un système digital linéaire, peut nécessiter une quantité considérable de matériel. En général, la taille du circuit de détection dépend de l'ordre de la relation de la redondance '*k*' (nombre de fois de retard à effectuer pour trouver des vecteurs de parité). Si ce nombre est grand, alors le circuit de test peut avoir une taille considérable. De toute façon, la taille du circuit de test peut être optimisée (minimisée) si le nombre de fois de retard (*k*) peut être réduit. Le nombre de fois de retard peut être réduit si le circuit de test peut se connecter à des nœuds internes (par exemple des variables d'état) accessibles à la mesure directement. Dans ce cas, pour calculer les paramètres d'un circuit optimisé de test, les matrices *C* et *D* à l'entrée de l'algorithme de détermination de relations de la redondance doivent être modifiées de manière à avoir de nouvelles lignes spécifiant de nouveaux nœuds accessibles directement à la mesure dans le système sous test. Ces nouveaux nœuds forment de nouvelles entrées pour le circuit de détection de fautes.

4.2.1 Exemple 2

Reprenons l'exemple du filtre passe-bas du chapitre précédant, figure 3-6. Si la variable d'état 'x_3' est mesurable directement à partir du circuit de filtre, alors les matrices *C* et *D* peuvent être remplacées par les matrices C_o et D_o :

$$C_o = \begin{bmatrix} & C & \\ 0 & 0 & 1 \end{bmatrix}; \qquad D_o = \begin{bmatrix} D \\ 0 \end{bmatrix};$$

Les deux nouvelles lignes («0 0 1» et «0») spécifient une nouvelle entrée (y_1=x_3) pour le circuit de détection de fautes. Les paramètres du circuit optimisé de détection de fautes ont été calculés. Les résultats obtenus sont :

1- L'ordre de relation de la redondance: $k = 1$.

2- Les vecteurs de parité: $V^{\mathrm{T}} = v^{\mathrm{T}} = \begin{bmatrix} -55.96 & 16.19 & 77.50 & -24.49 \end{bmatrix}$

3- Le vecteur: $v^{\mathrm{T}}.H^{[1]} = \begin{bmatrix} 3.85 & 3.85 \end{bmatrix}$

La conception du circuit de détection utilise la relation de la redondance suivante:

$$r(t) = v^{\mathrm{T}}.\begin{bmatrix} y(t-1) \\ y_1(t-1) \\ y(t) \\ y_1(t) \end{bmatrix} - v^{\mathrm{T}}H^{[1]}.\begin{bmatrix} u(t-1) \\ u(t) \end{bmatrix} \qquad (4\text{-}1)$$

$$r(t)=\{-55.96y(t-1)+16.19y_1(t-1) + 77.50y(t)-24.49y_1(t)\} - \{3.85[u(t-1)+u(t)]\} \qquad (4\text{-}2)$$

Le schéma du circuit de test est illustré par la figure 4-1.

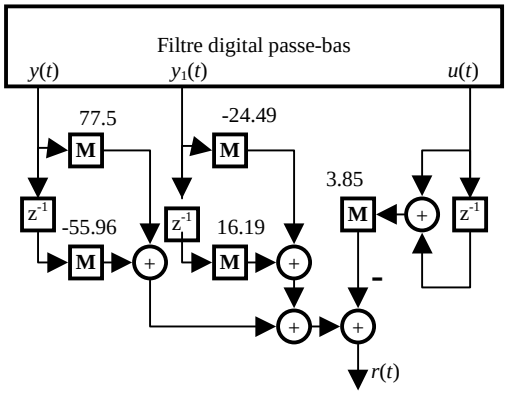


Figure 4-1. Le circuit minimisé de détection

Comparaison entre les circuits de détection de fautes

La comparaison de la surface entre le circuit minimisé de détection de fautes de la figure 4-1 et le circuit initial (non minimisé) de détection fautes de la chapitre précédent, figure 3-7, peut être résumée comme suit :

	Circuit non minimisé	Circuit minimisé	Gain
Nombre de multiplieurs	6	5	16.777%
Nombre d'additionneurs	7	5	28.572%
Nombre de mémoires	6	3	50%

4.2.2 Résultats expérimentaux

La technique proposée a été appliquée à des types différents de systèmes digitaux linéaires. Ici, les résultats de simulation, obtenus par l'outil de simulation et de calcul MATLAB, concernent le filtre numérique linéaire elliptique passe-bas indiqué dans la figure 3-6 du chapitre précédent. Le circuit minimisé de détection de fautes de la figure 4-1 a été connecté au circuit du filtre et la simulation a été faite. Le seuil de détection de fautes à la sortie du circuit de détection a été choisi par la simulation afin d'avoir la valeur de résidu la plus grande en l'absence de fautes, figure 4-2b.

Simulation de fautes

Généralement, la simulation de fautes se fait en utilisant des modèles de fautes. Ceux-ci représentent les moyens de description d'effets de fautes pouvant entraîner des erreurs à la sortie du circuit sous test. Pour les systèmes complexes, les modèles structurels et fonctionnels sont difficilement utilisables. Pour cela, les modèles à considérer ici sont des modèles comportementaux, c'est-à-dire que les fautes sont exprimées par des déviations des constantes, inclues dans le modèle qui décrit le système, de leurs valeurs nominales. La simulation de fautes a été faite en faisant des déviations des constantes nominales associées aux multiplieurs qui se trouvent dans le circuit du filtre.

La simulation de fautes d'un additionneur ou d'une mémoire est équivalente à celle du multiplieur localisé sur l'un des chemins d'entrée de l'additionneur ou de la mémoire, car les fautes d'un additionneur ou d'une mémoire peuvent être modélisées et simulées en plaçant un multiplieur sur l'un des chemins d'entrée de cet élément, additionneur ou mémoire.

Les déviations ont été faites et la réponse en fréquence de résidu pour chaque valeur de déviation a été évaluée. Les résultats présentés dans le *tableau*1 correspondent aux réponses maximales de chacune des quatre valeurs de déviation pour chaque constante nominale : +100%, +50%, -50% et -100%. Les figures 4-2a, …, 4-2f illustrent la réponse du filtre, la réponse de résidu en l'absence de fautes et les réponses pour une valeur de déviation de +50% par. Il apparaît clairement du *tableau*1 et des figures 4--2c, …, 4-2f que des fautes pouvant entraîner une déviation d'une valeur proche de +50% de la constante nominale du multiplieur $\mathbf{M}_{10}$ peuvent être indécelables tandis que toutes les autres fautes simulées sont détectées dans toute la gamme de fréquences de fonctionnement du filtre.

Table 1 : Réponse maximale en fréquence du résidu [(r/u)]

Multiplieur	**Déviation de la constante nominale**			
	+100%	+50%	-50%	-100%
$\mathbf{M_1}$	9.8	16	4.8	5.5
$\mathbf{M_2}$	1.8	0.9	0.9	1.8
$\mathbf{M_3}$	2	2.2	1	1
$\mathbf{M_4}$	8	1.9	1.8	22.5
$\mathbf{M_5}$	0.1	0.055	0.04	0.1
$\mathbf{M_6}$	3.5	0.75	0.45	1
$\mathbf{M_7}$	3.5	5	0.56	0.7
$\mathbf{M_8}$	6.8	3.4	3.4	6.8
$\mathbf{M_9}$	0.23	0.12	0.13	0.23
$\mathbf{M_{10}}$	0.02	0.012	0.015	0.022
$\mathbf{M_{11}}$	8.5	4.4	4.3	8.5
$\mathbf{M_{12}}$	3.9	1.9	1.9	3.9
$\mathbf{M_{13}}$	25.7	12.5	13	25.7
$\mathbf{M_{14}}$	1.2	0.5	0.5	1.2
Fonctionnement sans fautes	0.012			

Figure 4-2a: Réponse frequentielle du filtre passe-bas

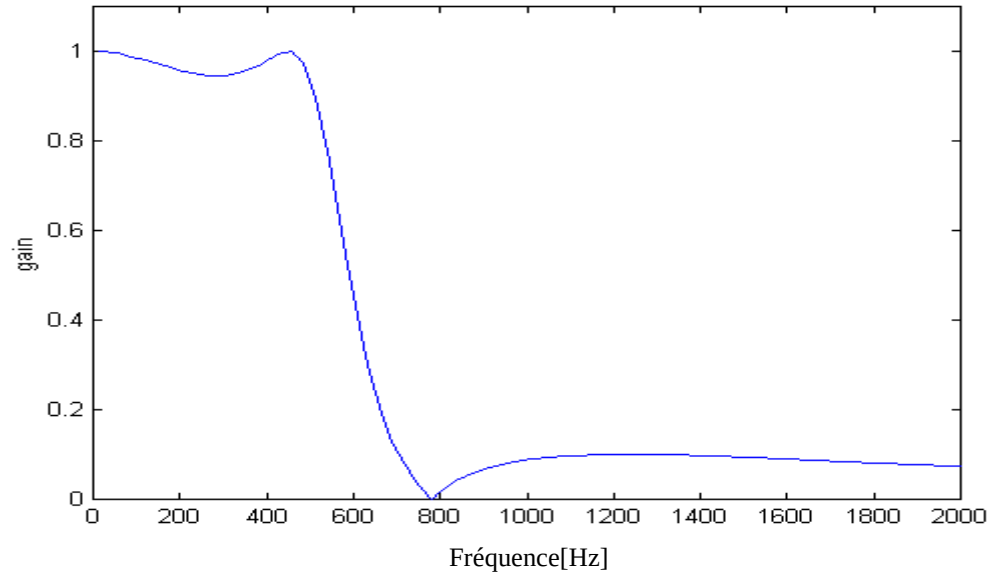


Figure 4-2b : Réponse frequentielle du résidu en l'absence de fautes

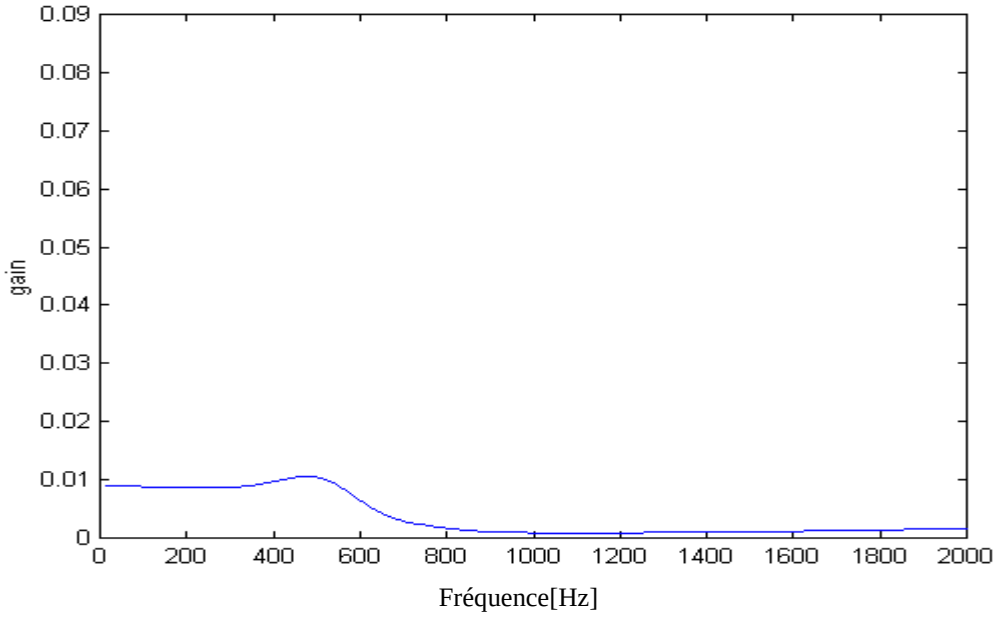


Figure 4-2c : Réponse frequentielle du résidu en présence de fautes
(déviation de constante est +50%)

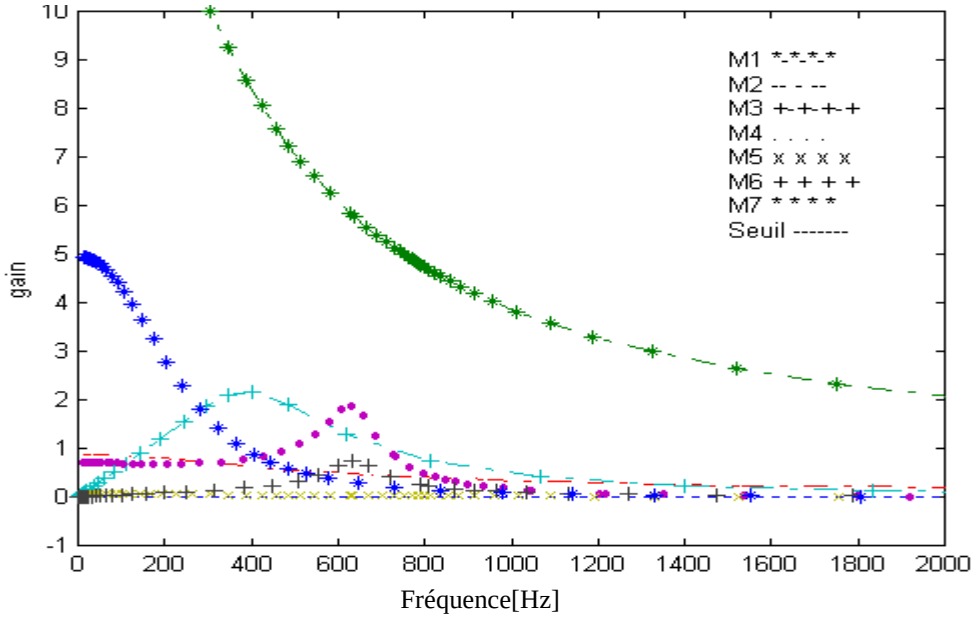


Figure 4-2d : Réponse frequentielle du résidu en présence de fautes
(déviation de constante est +50%)

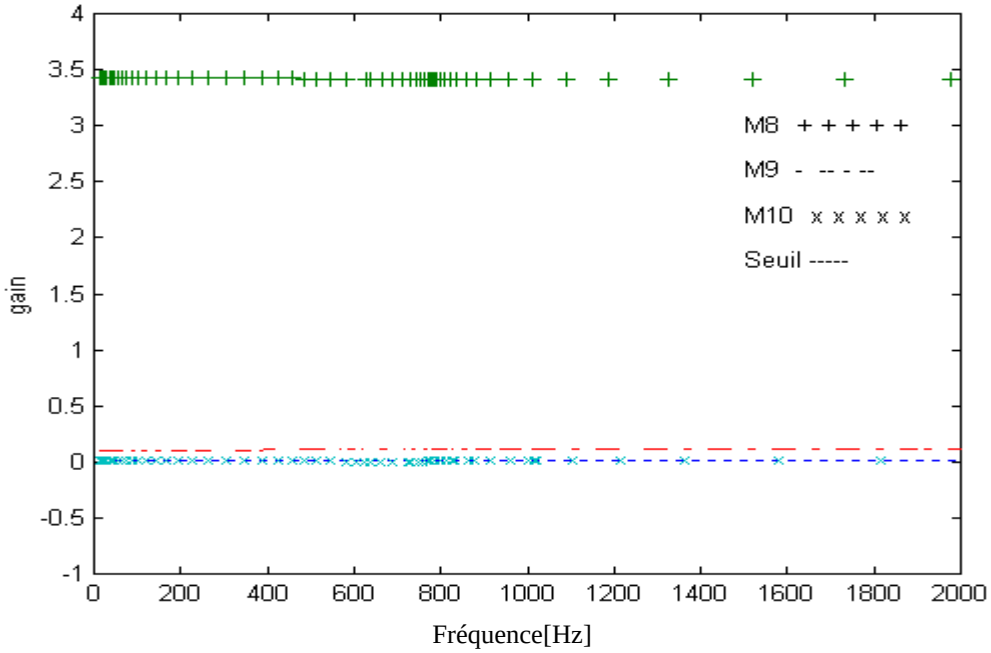


Figure 4-2e : Réponse frequentielle du résidu en présence de fautes
(déviation de constante est +50%)

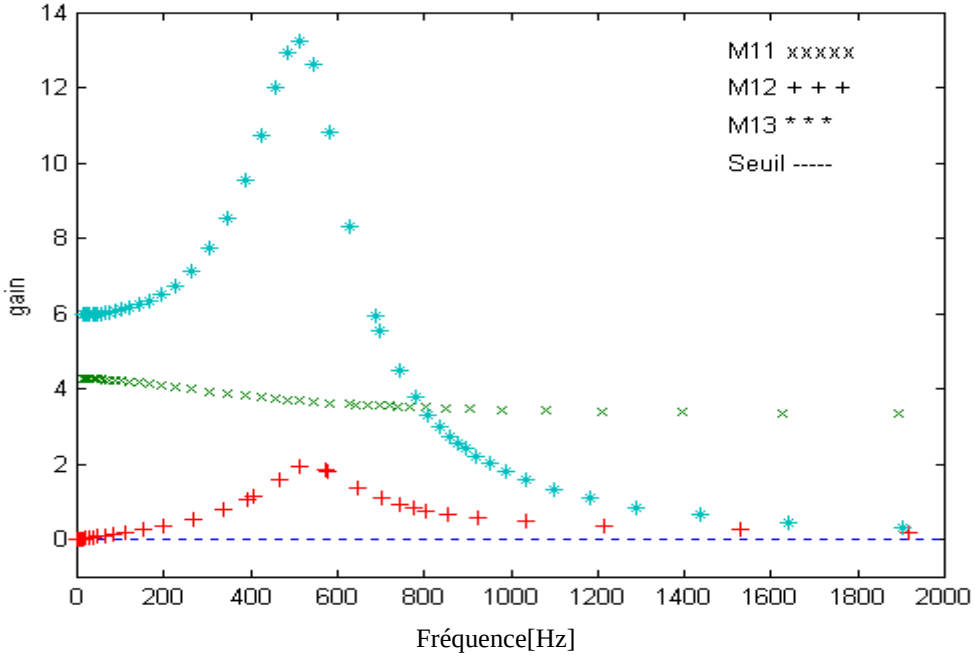
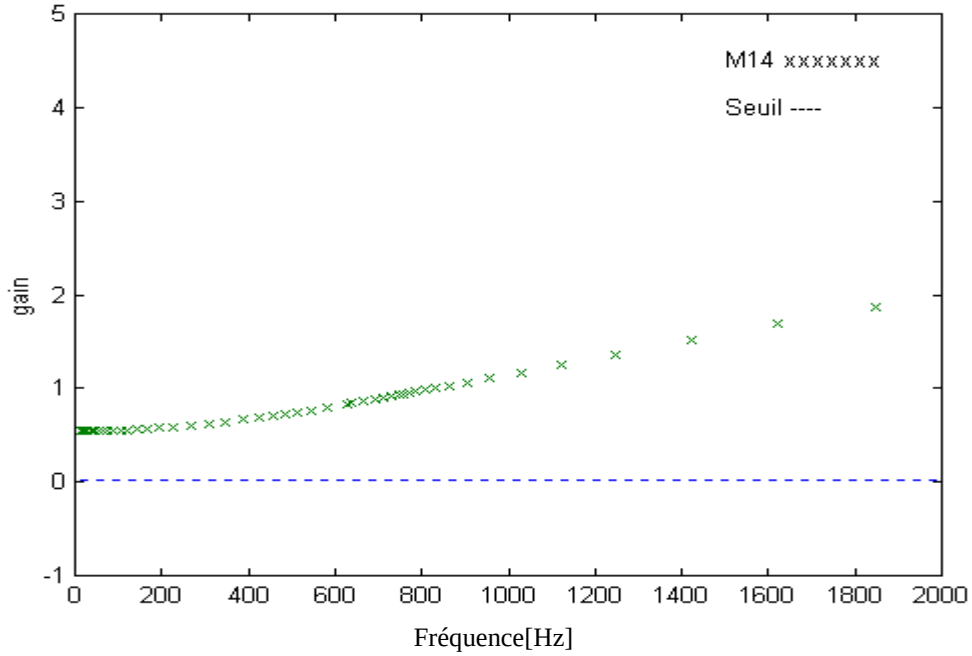


Figure 4-2f : Réponse frequentielle du résidu en présence de fautes
(déviation de constante est +50%)



4.3 Robustesse des circuits de détection de fautes

Jusqu'ici, la méthode présentée de détection de fautes ne tient pas compte du problème de bruit inévitable dans les systèmes complexes. Pour les systèmes digitaux, les résultats des différentes opérations arithmétiques, additions et multiplications, doivent être codés sur un nombre fini prédéterminé de bits. Il y aura donc des arrondis et/ou troncatures à effectuer. La conséquence de ces mutilations du signal réel est qu'un signal de bruit, appelé le bruit du calcul, sera présent dans le système. En général, le bruit généré à l'intérieur du système n'est pas considéré comme faute et doit être toléré. Pour cela, le circuit de détection de fautes doit être insensible au bruit. Par conséquent le signal de sortie du circuit de détection doit être seulement sensible aux fautes même en présence de bruit dans le système.

4.3.1 Relations robustes de la redondance

Dans cette section, le problème de conception des circuits de détection de fautes insensibles au bruit sera discuté. La conception est basée sur les relations robustes de la redondance. Nous avons vu que la détermination de relations de la redondance revient à déterminer des vecteurs de parité. Pour cela, la détermination de relations robustes de la redondance revient donc à déterminer des vecteurs optimaux de parité. Ceux-ci peuvent être déterminés comme suit :

Dans la pratique, les systèmes électroniques sont l'objet de types variés d'effets tels que les fautes et le bruit. C'est pourquoi, pour la représentation réelle des systèmes, il est important de modéliser tous les effets pouvant générer des signaux de sorties incorrectes. Pour les systèmes digitaux linéaires, le modèle mathématique donné par les équations 1-7 et 1-8 (voir chapitre1) peut être réécrit pour inclure les modèles de différents effets [Fran90, SiAb01]:

$$x(t+1) = A.x(t) + B.u(t) + E.d(t) + K.f(t) \quad (4\text{-}3)$$

$$y(t) = C.x(t) + D.u(t) + F.d(t) + G.f(t) \quad (4\text{-}4)$$

Pour cette représentation, toutes les fautes sont regroupées dans le vecteur de faute *f* tandis que tous les autres effets (bruit) sont regroupés dans un vecteur appelé le vecteur

d'entrées inconnues, *d*. Les termes *Ed* et *Fd* représentent les modèles d'entrées inconnues tandis que *Kf* et *Gf* représentent les modèles de fautes. Les matrices *A*, *B*, *C* et *D* représentent les matrices nominales de système dont les fautes se reflètent dans le changement de leurs éléments. Les matrices *K*, *G*, *E* et *F* représentent les matrices de fautes et de bruit.

Pour des instants du temps (*t-k*, *t-k*+1, *t-k*+2, …, *t*), les *k*+1 expressions successives du vecteur de sortie peuvent être données par l'équation condensée suivante :

$$Y^{[k]}(t) = Ob^{[k]}.x(t-k) + H^{[k]}.U^{[k]}(t) + H_1^{[k]}.Br^{[k]}(t) + H_2^{[k]}.Fa^{[k]}(t) \qquad (4\text{-}5)$$

où :

$$Y^{[k]}(t) = \begin{bmatrix} y(t-k) \\ y(t-k+1) \\ y(t-k+2) \\ : \\ y(t) \end{bmatrix}; \quad Ob^{[k]} = \begin{bmatrix} C \\ CA \\ CA^2 \\ : \\ CA^k \end{bmatrix}; \quad H^{[k]} = \begin{bmatrix} D & 0 & .. & 0 & 0 \\ CB & D & : & 0 & 0 \\ CAB & CB & : & : & : \\ : & : & .. & D & 0 \\ CA^{k-1}B & CA^{k-2}B & .. & CB & D \end{bmatrix}; \quad U^{[k]}(t) = \begin{bmatrix} u(t-k) \\ u(t-k+1) \\ u(t-k+2) \\ : \\ u(t) \end{bmatrix}. \qquad (4\text{-}6)$$

$$Br^{[k]}(t) = \begin{bmatrix} d(t-k) \\ d(t-k+1) \\ d(t-k+2) \\ : \\ d(t) \end{bmatrix}; \; Fa^{[k]}(t) = \begin{bmatrix} f(t-k) \\ f(t-k+1) \\ f(t-k+2) \\ : \\ f(t) \end{bmatrix}; \; H_1^{[k]} = \begin{bmatrix} F & 0 & .. & 0 & 0 \\ CE & F & : & 0 & 0 \\ CAE & CE & : & : & : \\ : & : & .. & F & 0 \\ CA^{k-1}E & CA^{k-2}E & .. & CE & F \end{bmatrix}; \; H_2^{[k]} = \begin{bmatrix} G & 0 & .. & 0 & 0 \\ CK & G & : & 0 & 0 \\ CAK & CK & : & : & : \\ : & : & .. & G & 0 \\ CA^{k-1}K & CA^{k-2}K & .. & CK & G \end{bmatrix}; \qquad (4\text{-}7)$$

En se référant aux équations 3-15, 3-16, 3-17 et 3-18 (voir chapitre 3) on trouve :

$$v^{\mathrm{T}}.Ob^{[k]}.x(t\text{-}k) = 0$$
$$v^{\mathrm{T}}.Y^{[k]}(t) = v^{\mathrm{T}} H^{[k]}.U^{[k]}(t)$$

Donc,

$$r(t) = v^{\mathrm{T}}.[H_1^{[k]}.Br^{[k]}(t) + H_2^{[k]}.Fa^{[k]}(t)] \qquad (4\text{-}8)$$

1- En l'absence de bruit et de fautes, $Br^{[k]}(t) = 0$, $Fa^{[k]}(t) = 0$ et

$$r(t) = 0 \qquad (4\text{-}9)$$

2- Si le bruit seulement est présent alors $Br^{[k]}(t) \neq 0$, $Fa^{[k]}(t) = 0$ et

$$r(t) = v^{\mathrm{T}}.H_1^{[k]}.Br^{[k]}(t) \qquad (4\text{-}10)$$

3- Si des fautes seulement sont présentes alors $Br^{[k]}(t) = 0$, $Fa^{[k]}(t) \neq 0$ et

$$r(t) = v^{\mathrm{T}}.H_2^{[k]}.Fa^{[k]}(t) \qquad (4\text{-}11)$$

4- En présence de bruit et de fautes $Br^{[k]}(t) \neq 0$, $Fa^{[k]}(t) \neq 0$ et

$$r(t) = v^{\mathrm{T}}.[H_1^{[k]}.Br^{[k]}(t) + H_2^{[k]}.Fa^{[k]}(t)] \qquad (4\text{-}12)$$

Pour que le résidu $r(t)$ soit affecté seulement par les fautes, les équations suivantes doivent être vérifiées :

$$v^{\mathrm{T}}.H_1^{[k]} = 0 \qquad (4\text{-}13)$$

$$v^{\mathrm{T}}.H_2^{[k]} \neq 0 \qquad (4\text{-}14)$$

La vérification des équations 4-13 et 4-14 implique que le résidu n'est pas affecté par le bruit, et qu'il est affecté seulement par les fautes.

Trouver une solution v^{T} satisfaisante pour les équations 3-15, 4-13 et 4-14 donne la relation robuste utilisable pour générer le résidu robuste insensible au bruit.

Malheureusement, les équations 3-15, 4-13 et 4-14 ne peuvent produire une solution. Donc, pour trouver une solution approximative optimale, un indice de performance qui prend en compte le problème de sensibilité de résidu par rapport au bruit et aux fautes peut être défini. Pour satisfaire les contraintes, l'indice peut être donné par :

$$\rho = \frac{\| v^{\mathrm{T}} H_1^{[k]} \|}{\| v^{\mathrm{T}} H_2^{[k]} \|} \qquad (4\text{-}15)$$

et sa valeur doit être minimisée en tenant compte de l'équation 3-15. Le symbole ||.|| désigne la norme euclidienne (la valeur propre la plus grande de la matrice). Pour s'assurer que v^{T} satisfera l'équation 3-15, le problème peut être formulé comme suit.

Trouver un vecteur w^{T} tel que :

$$\rho = \frac{\| w^{T} V_0 H_1^{[k]} \|}{\| w^{T} V_0 H_2^{[k]} \|} \qquad (4\text{-}16)$$

devient minimale. V_0 est la base de toutes les solutions possibles de l'équation 3-15, et w^T extrait la solution optimale (v^T optimal) de toutes les solutions possibles représentées par V_0. La solution du problème peut être obtenue par la différenciation d'indice de performance. Cela donne la relation suivante :

$$w^{T}\left(V_0 H_1^{[k]} H_1^{[k]T} V_0^{T} - \rho V_0 H_2^{[k]} H_2^{[k]T} V_0^{T}\right) = 0 \qquad (4\text{-}17)$$

Celle-ci montre que le problème est, maintenant, réduit au problème de valeurs propres et vecteurs propres généralisés. La valeur propre minimale représente la valeur optimale de l'indice de performance et le vecteur propre w^T correspondant est le vecteur sélectionnant le vecteur de parité optimal v^T. Par conséquent le vecteur optimal v^T peut ensuite être associé à une relation de la redondance pour construire le circuit robuste de détection de fautes.

<u>Détermination des matrices des modèles de fautes et de bruit</u>

Le modèle mathématique réel donné par les équations 4-3 et 4-4 peut être aussi réécrit [SiAb01].

$$x(t+1) = (A+\Delta A).[\, x(t) + \Delta x(t)] + (B+\Delta B).[u(t) + \Delta u(t)] \qquad (4\text{-}18)$$

$$y(t) = (C+\Delta C).[\, x(t) + \Delta x(t)] + (D+\Delta D).[u(t) + \Delta u(t)] \qquad (4\text{-}19)$$

où les termes ΔA, ΔB, ΔC et ΔD modélisent les fautes se reflétant principalement dans les changements dans A, B, C et D. Les termes Δx et Δu modélisent les entrées inconnues (bruit du système sous test).

Pour des instants du temps (t-k, t-k+1, t-k+2, …, t), les k+1 expressions successives du vecteur de sortie peuvent être données par l'équation condensée suivante :

$$Y^{[k]}(t) = (Ob + \Delta Ob)^{[k]}.x(t-k) + (H_3 + \Delta H_3)^{[k]}.\Delta X^{[k]}(t) + (H + \Delta H)^{[k]}.[U^{[k]}(t) + \Delta U^{[k]}(t)] \quad (4\text{-}20)$$

où :

$$Y^{[k]}(t) = \begin{bmatrix} y(t-k) \\ y(t-k+1) \\ y(t-k+2) \\ : \\ y(t) \end{bmatrix}; \; U^{[k]}(t) = \begin{bmatrix} u(t-k) \\ u(t-k+1) \\ u(t-k+2) \\ : \\ u(k) \end{bmatrix}; \; \Delta U^{[k]}(t) = \begin{bmatrix} \Delta u(t-k) \\ \Delta u(t-k+1) \\ \Delta u(t-k+2) \\ : \\ \Delta u(t) \end{bmatrix}; \; \Delta X^{[k]}(t) = \begin{bmatrix} \Delta x(t-k) \\ \Delta x(t-k+1) \\ \Delta x(t-k+2) \\ : \\ \Delta x(t) \end{bmatrix}; \quad (4\text{-}21)$$

$$(Ob + \Delta Ob)^{[k]} = \begin{bmatrix} C + \Delta C \\ (C + \Delta C)(A + \Delta A) \\ : \\ (C + \Delta C)(A + \Delta A)^k \end{bmatrix}; \qquad H_3^{[k]} = \begin{bmatrix} C & 0 & .. & 0 & 0 \\ CA & C & : & 0 & 0 \\ CA^2 & CA & : & : & : \\ : & : & .. & C & 0 \\ CA^k & CA^{k-1} & .. & CA & C \end{bmatrix}; \quad (4\text{-}22)$$

$$(H_3 + \Delta H_3)^{[k]} = \begin{bmatrix} C + \Delta C & 0 & 0 & 0 & 0 \\ (C + \Delta C)(A + \Delta A) & C + \Delta C & 0 & .. & : \\ : & : & : & C + \Delta C & 0 \\ (C + \Delta C)(A + \Delta A)^k & .. & .. & (C + \Delta C)(A + \Delta A) & C + \Delta C \end{bmatrix}. \quad (4\text{-}23)$$

$$(H + \Delta H)^{[k]} = \begin{bmatrix} D + \Delta D & 0 & 0 & 0 & 0 \\ (C + \Delta C)(B + \Delta B) & D + \Delta D & 0 & .. & : \\ : & : & : & D + \Delta D & 0 \\ (C + \Delta C)(A + \Delta A)^{k-1}(B + \Delta B) & .. & .. & (C + \Delta C)(B + \Delta B) & D + \Delta D \end{bmatrix}. \quad (4\text{-}24)$$

La relation de la redondance ou l'expression du résidu à n'importe quel temps *t*, peut être réécrite :

$$r(t) = v^{\mathrm{T}}\{Y^{[k]}(t) - (Ob + \Delta Ob)^{[k]}.x(t-k) - (H_3 + \Delta H_3)^{[k]}.\Delta X^{[k]}(t) - (H + \Delta H)^{[k]}.[U^{[k]}(t) + \Delta U^{[k]}(t)]\} \quad (4\text{-}25)$$

En se référant aux équations 3-15, 3-16, 3-17 et 3-18 (voir chapitre 3) on aura :

1- En l'absence de fautes et de bruit, $\Delta Ob = 0$, $\Delta H = 0$, $\Delta H_3 = 0$, $\Delta X^{[k]}(t) = 0$ et $\Delta U^{[k]}(t) = 0$.

$$r(t) = 0 \quad (4\text{-}26)$$

2- Si le bruit seulement est présent, alors $\Delta Ob = 0$, $\Delta H = 0$, $\Delta H_3 = 0$, $\Delta X^{[k]}(t) \neq 0$ et $\Delta U^{[k]}(t) \neq 0$.

$$r(t) = v^{T}.[\, H_3^{[k]}.\Delta X^{[k]}(t) + H^{[k]}.\, \Delta U^{[k]}(t)]$$

ou encore

$$r(t) = v^{T}.[\, H_3^{[k]} \quad H^{[k]}].\begin{bmatrix} \Delta X^{[k]}(t) \\ \Delta U^{[k]}(t) \end{bmatrix} \qquad (4\text{-}27)$$

en comparant l'équation 4-27 avec l'équation 4-10 (comparaison de signaux avec les signaux et de matrices avec les matrices) on trouve que la matrice $H_1^{[k]}$ est une concaténation de $H_3^{[k]}$ et $H^{[k]}$. $H_1^{[k]} = [\, H_3^{[k]} \quad H^{[k]}\,]$.

Ainsi, par correspondance entre $H_1^{[k]}$ de l'équation 4-7 d'un côté et $H^{[k]}, H_3^{[k]}$ des équations 4-6 et 4-22 d'un autre côté on trouve que E est une concaténation de A et B, et F est une concaténations de C et D :

$$E = [A \quad B] \quad \text{et} \quad F = [C \quad D] \qquad (4\text{-}28)$$

3- Si des fautes seulement sont présentes, $\Delta Ob \neq 0$, $\Delta H \neq 0$, $\Delta H_3 \neq 0$, $\Delta X^{[k]}(t)=0$ et $\Delta U^{[k]}(t) =0$.

$$r(t) = v^{T}.[\Delta Ob^{[k]}.x(t\text{-}k) + \Delta H^{[k]}.U^{[k]}(t)] \qquad (4\text{-}29)$$

cette équation 3- peut être réécrite:

$$r(t) = v^{T}.[\, \Delta Ob^{[k]} \quad \Delta H^{[k]}\,].\begin{bmatrix} x(t-k) \\ U^{[k]}(t) \end{bmatrix} \qquad (4\text{-}30)$$

En comparant cette équation (4-30) avec l'équation 4-11 (comparaison de signaux avec les signaux et de matrices avec les matrices) on trouve que la matrice $H_2^{[k]}$ est une concaténation de $\Delta Ob^{[k]}$ et $\Delta H^{[k]}$.

$$H_2^{[k]} = [\, \Delta Ob^{[k]} \quad \Delta H^{[k]}\,] \qquad (4\text{-}31)$$

Par comparaison entre cette équation (4-31) d'un côté et les équations 4-7, 4-22 et 4-24 d'un autre côté, on trouve que les matrices G et K sont des concaténations de ΔC, ΔD et de ΔA, ΔB respectivement.

$$G = [\, \Delta C \quad \Delta D] \quad \text{et} \quad K=[\Delta A \quad \Delta B] \qquad (4\text{-}32)$$

Détermination de ΔC, ΔD, ΔA et ΔB

Chaque composant dans les matrices *ΔA, ΔB, ΔC* et *ΔD* peut être déterminé en faisant un développement limité de l'ordre 1 autour de la valeur nominale du composant correspondant dans la matrice correspondante *A, B, C, D*, c'est-à-dire que chaque composant 'x' dans les matrices *ΔA, ΔB, ΔC* et *ΔD* est donné par une série de telle que :

$$x \rightarrow x_0 + dx \qquad (4\text{-}33)$$

ou x_0 représente la valeur nominal du composant dans les matrices *A, B, C* et *D* et dx représente une déviation donnée de cette valeur.

4.3.2 Exemple 3

Pour illustrer la méthode robuste de détection de fautes et ainsi concevoir des circuits de détection robustes insensibles au bruit, le système de la figure 4-3 est utilisé. Ceci représente un filtre numérique linéaire passe-bande elliptique de l'ordre 4. Le filtre a une seule entrée fonctionnelle (u), une seule sortie fonctionnelle (y_0), deux points de connexion internes accessibles directement à la mesure (y_1, y_2) et quatre variables d'état (x_0, …, x_3).

Les matrices d'état, *A, B, C, D*, de ce filtre sont :

$$A = \begin{bmatrix} -0.3277166 & -0.1330587 & 0.7887182 & -0.1561035 \\ 0.1330587 & -0.1847195 & 0.1561035 & 0.9564814 \\ -0.7887182 & 0.1561035 & 0.0746813 & 0.1831396 \\ -0.1561035 & -0.9564814 & -0.1831396 & -0.1221374 \end{bmatrix} \qquad B = \begin{bmatrix} 0.357846 \\ 0.070825 \\ -0.419822 \\ -0.0830915 \end{bmatrix}$$

$$C = \begin{bmatrix} 0.017702 & 0.3103486 & 0.0207678 & 0.3640988 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad D = \begin{bmatrix} 0.109414 \\ 0 \\ 0 \end{bmatrix}$$

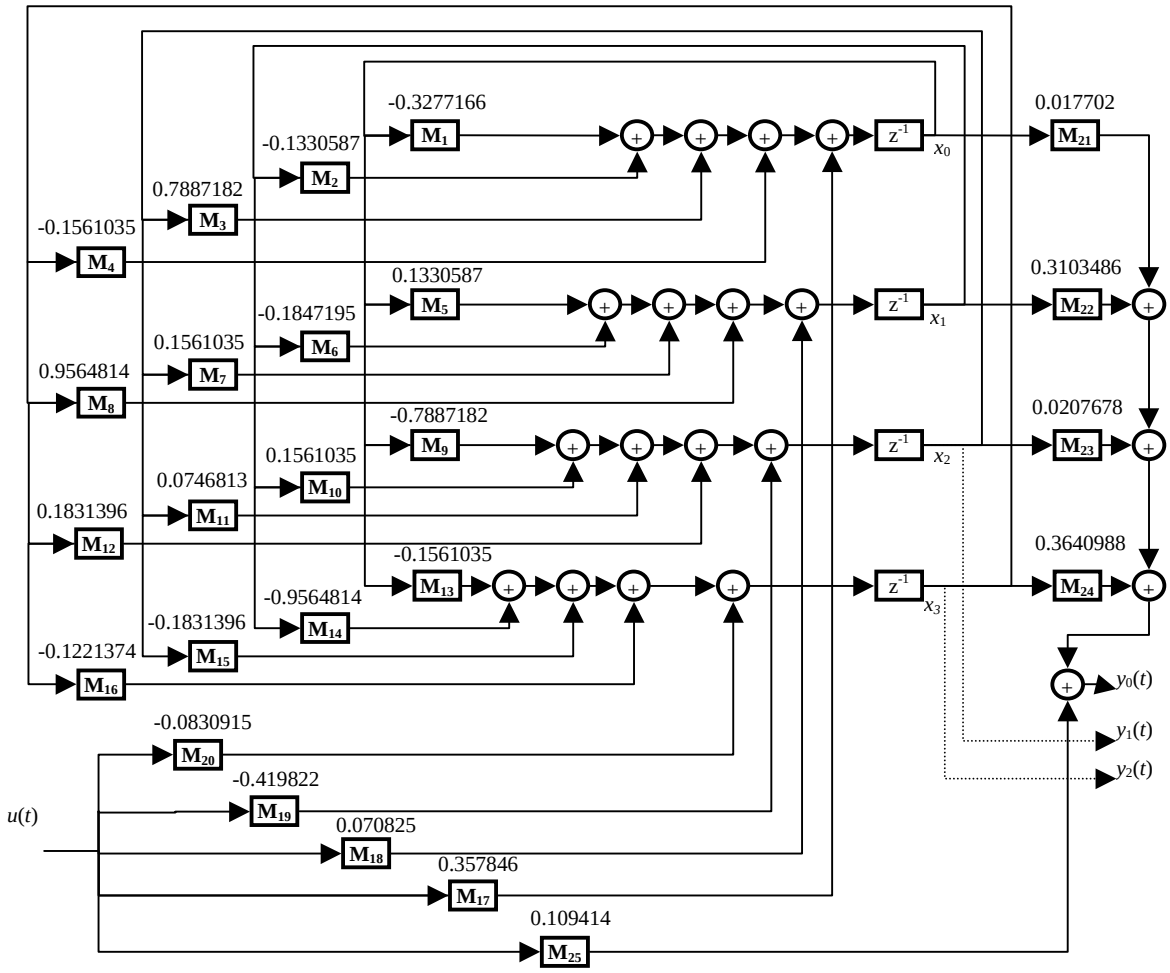


Figure 4-3 : Filtre digital linéaire elliptique passe-bande

Les paramètres du circuit robuste de détection de fautes pour ce système étaient calculés, il s'agit de :

1- L'ordre de relation de la redondance: $k = 1$.

2- Les vecteurs de parité : $V_0 = \begin{bmatrix} -91.13 & -3.47 & 29.82 & -2.74 & 3.59 & -27.82 \\ 0 & 6.75 & 27.49 & -88.46 & -3.08 & 36.93 \end{bmatrix}$

3- Le produit $V_0.H^{[1]} = \begin{bmatrix} -9.14 & -0.30 \\ -0.83 & -9.68 \end{bmatrix}$

4- Le vecteur propre généralisé optimal est : $w^T = [0.8352 \quad -0.5500]$

5- Le vecteur de parité optimal est $v^T = [-76.1061 \quad -6.6189 \quad 9.7817 \quad 46.3682 \quad 4.69297 \quad -43.5469]$

6- Le vecteur $v^T.H^{[1]} = [-7.17308 \quad 5.07333]$

Pour la conception de circuit robuste de détection de fautes, la relation de la redondance suivante est utilisée :

$$r(t) = v^{\mathrm{T}}.\begin{bmatrix} y_0(t-1) \\ y_1(t-1) \\ y_2(t-1) \\ y_0(t) \\ y_1(t) \\ y_2(t) \end{bmatrix} - v^{\mathrm{T}} H^{[1]}.\begin{bmatrix} u(t-1) \\ u(t) \end{bmatrix} \qquad (4\text{-}34)$$

$$r(t) = \{-76.1061y_0(t\text{-}1)-6.6189y_1(t\text{-}1)+9.7817y_2(t\text{-}1) + 46.3682y_0(t)+4.69297y_1(t)-43.5469y_2(t)\} - \{-7.17308u(t\text{-}1)+5.07333u(t)\} \quad (4\text{-}35)$$

La synthèse du circuit robuste de test est illustrée dans la figure suivante.

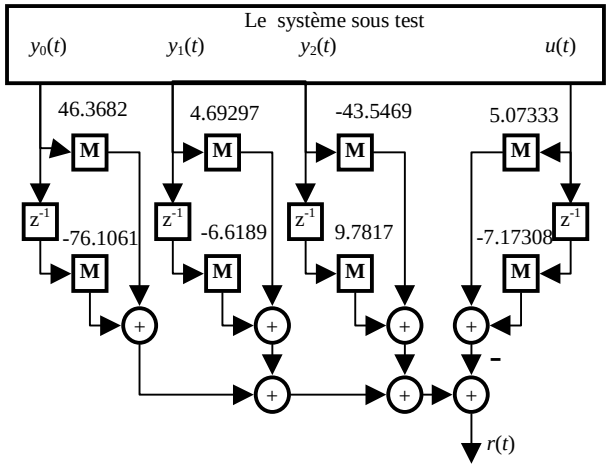


Figure 4-4. Circuit robuste de détection de fautes

4.3.3 Résultats expérimentaux

Pour tester l'efficacité du circuit robuste indiqué dans la figure 4-4, celui-ci a été simulé, en utilisant MATLAB, avec le filtre de la figure 4-3. Le seuil de détection de fautes à la sortie du circuit de détection a été choisi par simulation afin d'avoir la valeur de résidu la plus grande en l'absence de fautes, figure 4-5b.

Simulation de fautes

La simulation de fautes a été faite de la même façon que dans l'exemple précédent. Les déviations des constantes nominales ont été faites et la réponse en fréquence de résidu pour chaque valeur de déviation a été évaluée. Les résultats présentés dans le tableau 2 correspondent aux réponses maximales de chacune des deux valeurs de déviation pour chaque constante nominale : +100% et -100%. La simulation de fautes pour les additionneurs et les mémoires est équivalante à celle correspondante aux multiplieurs localisés sur les chemins d'entrées de ces additionneurs ou ces mémoires. L'évaluation complète des réponses de résidu pour une valeur de déviation de +100% par constante sont illustrées dans les figures 4-5c, …, 4-5g.

Table 2: Réponse maximale en fréquence du résidu [(r/u)]

Multiplieur	**Déviation de la constante**		**Multiplieur**	**Déviation de la contante**	
	+100%	-100%		+100%	-100%
$\mathbf{M}_1$	0.65	0.5	$\mathbf{M}_{14}$	27.5	12.5
$\mathbf{M}_2$	0.4	0.3	$\mathbf{M}_{15}$	18	13.5
$\mathbf{M}_3$	1.5	0.5	$\mathbf{M}_{16}$	7.8	7
$\mathbf{M}_4$	0.3	0.4	$\mathbf{M}_{17}$	0.3	0.3
$\mathbf{M}_5$	6	4	$\mathbf{M}_{18}$	1.1	1.1
$\mathbf{M}_6$	10	6	$\mathbf{M}_{19}$	2.5	2.5
$\mathbf{M}_7$	4.5	4	$\mathbf{M}_{20}$	2.2	2.2
$\mathbf{M}_8$	16	7	$\mathbf{M}_{21}$	3	3.2
$\mathbf{M}_9$	9.5	3	$\mathbf{M}_{22}$	62	62
$\mathbf{M}_{10}$	2	2.5	$\mathbf{M}_{23}$	2	2
$\mathbf{M}_{11}$	0.9	0.9	$\mathbf{M}_{24}$	65	65
$\mathbf{M}_{12}$	4	3.5	$\mathbf{M}_{25}$	12	12
$\mathbf{M}_{13}$	8.5	6.5			
Fonctionnement sans fautes	0.002				

Figure 4-5a : Réponse frequentielle du filtre passe-bande

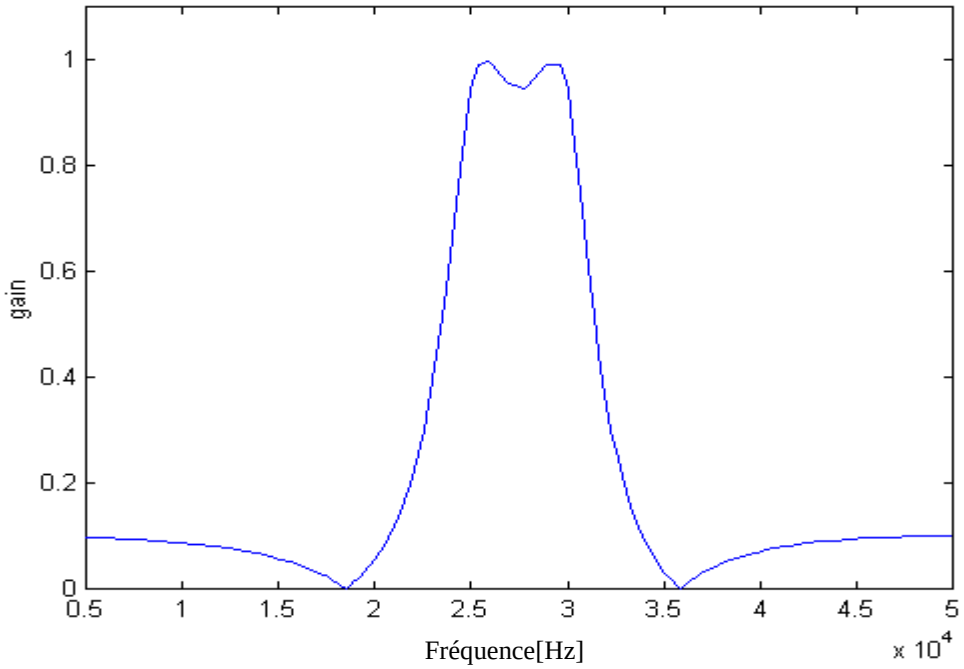


Figure 4-5b : Réponse frequentielle du résidu en l'absence de fautes

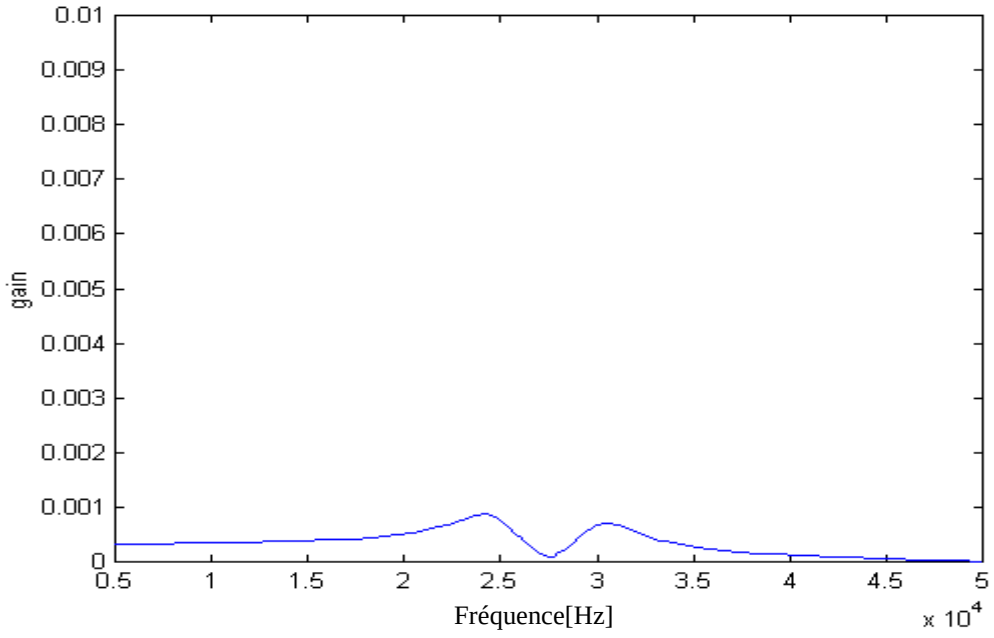


Figure 4-5c : Réponse frequentielle du résidu en présence de fautes
(déviation de constante est +100)

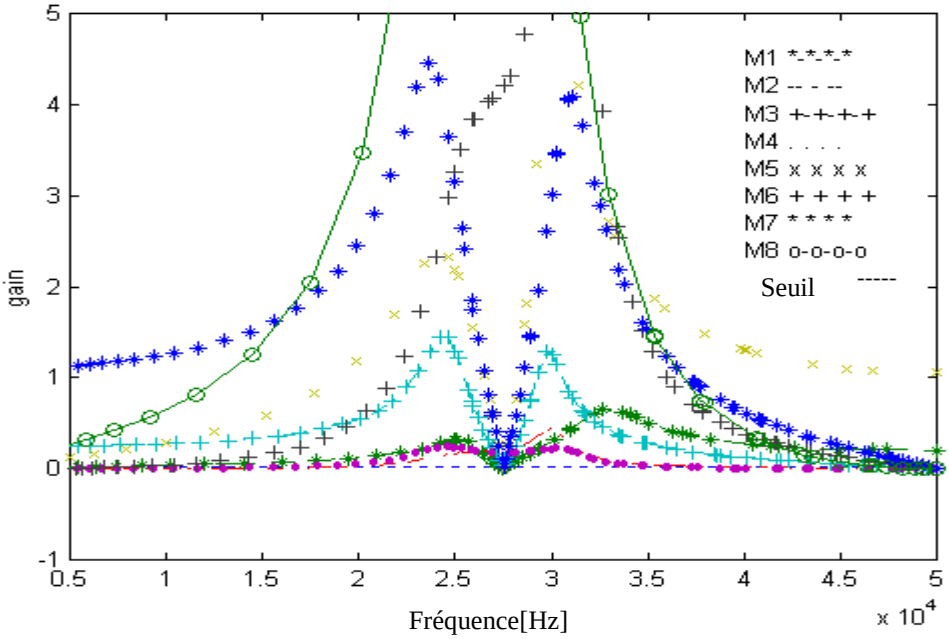


Figure 4-5d : Réponse frequentielle du résidu en présence de fautes
(déviation de constante est +100)

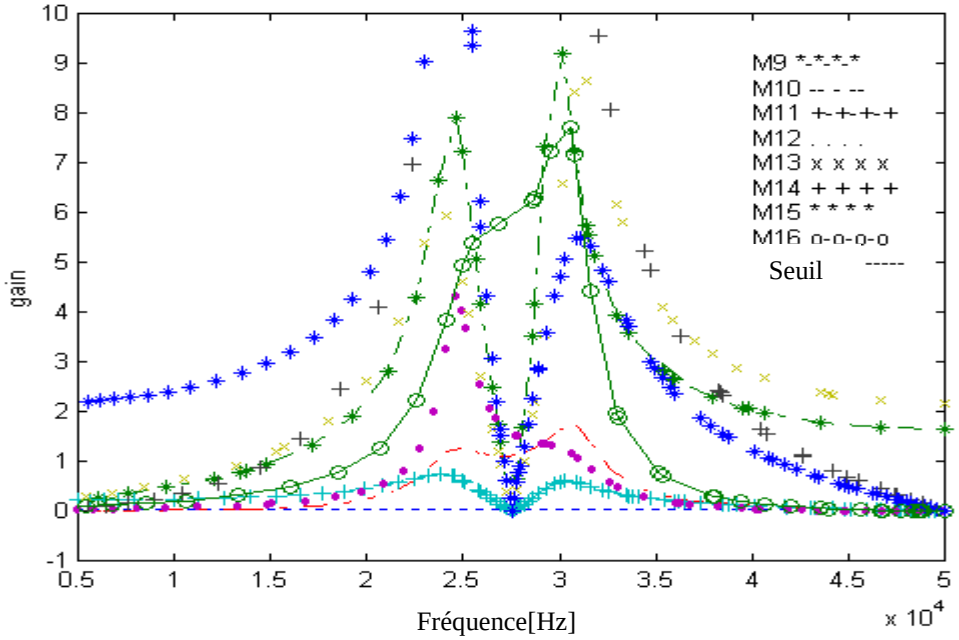


Figure 4-5e : Réponse frequentielle du résidu en présence de fautes
(déviation de constante est +100)

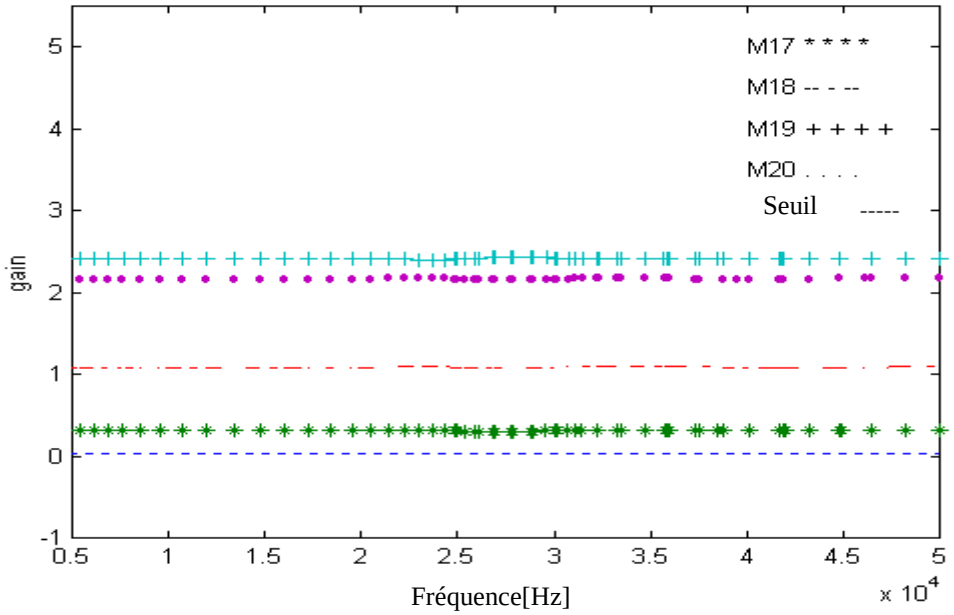


Figure 4-5f : Réponse frequentielle du résidu en présence de fautes
(déviation de constante est +100)

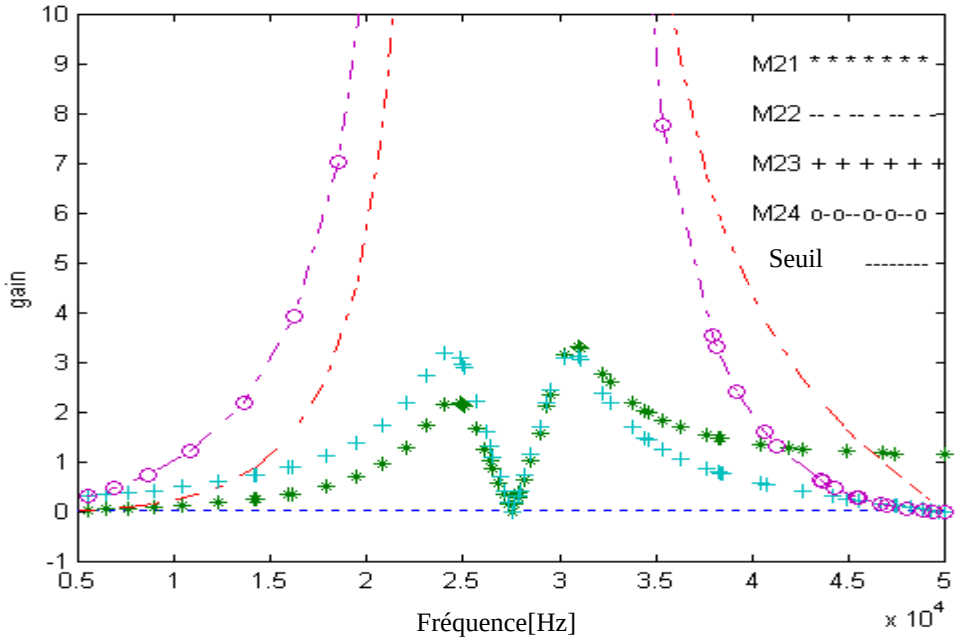
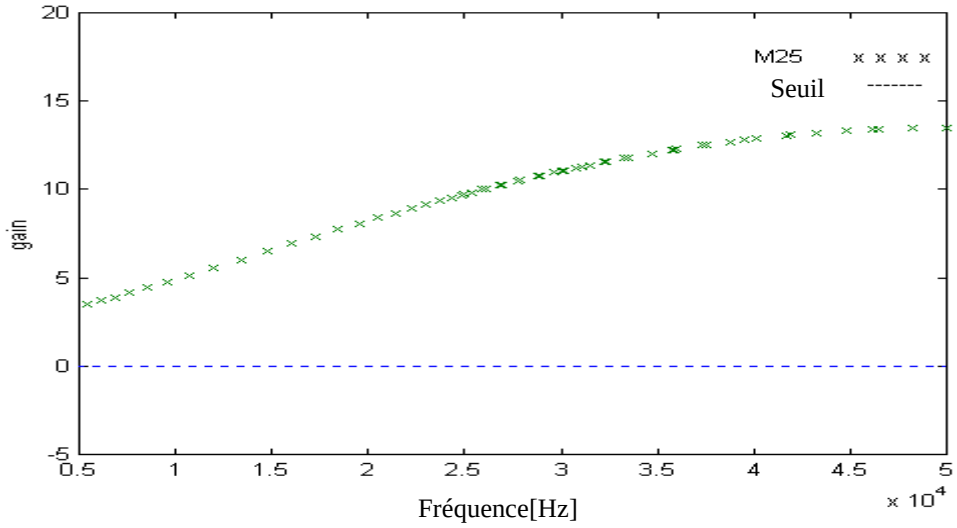
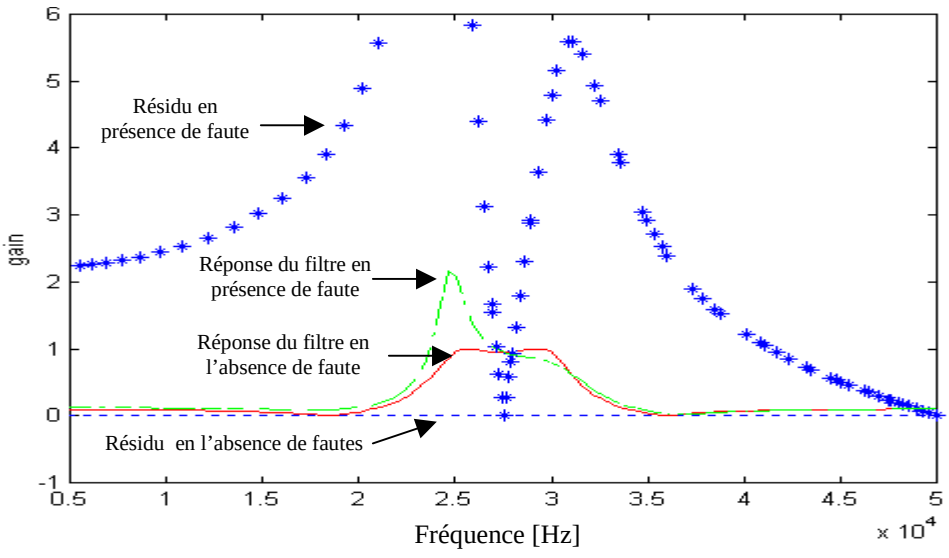


Figure 4-5g : Réponse frequentielle du résidu en présence de fautes
(déviation de constante est +100)



Du tableau 2 et des figures, 4-5c, …, 4-5g, il apparaît clairement que toutes les fautes simulées sont détectables dans toute la gamme de fréquences de fonctionnement du filtre à l'exception de quelques fautes détectables également dans toute la gamme de fréquence de fonctionnement mais pas à la fréquence centrale. Il a été constaté que les fautes non détectables à la fréquence centrale ne perturbent pas le fonctionnement correct du filtre à cette fréquence, et c'est la raison pour laquelle ces fautes ne sont pas détectables à la fréquence centrale. La figure 4-5h montre que la réponse du filtre en présence d'une faute non détectable à la fréquence centrale coïncide parfaitement avec la réponse en l'absence de fautes à cette fréquence, le fonctionnement est donc correct.

Figure 4-5h : Réponses du filtre et du résidu en l'absence et en présence de fautes



Simulation de bruit et de fautes

Pour tester la robustesse du circuit robuste de test, celui-ci a été connecté au filtre et la simulation temporelle a été faite. Les conditions de simulation sont les suivantes :

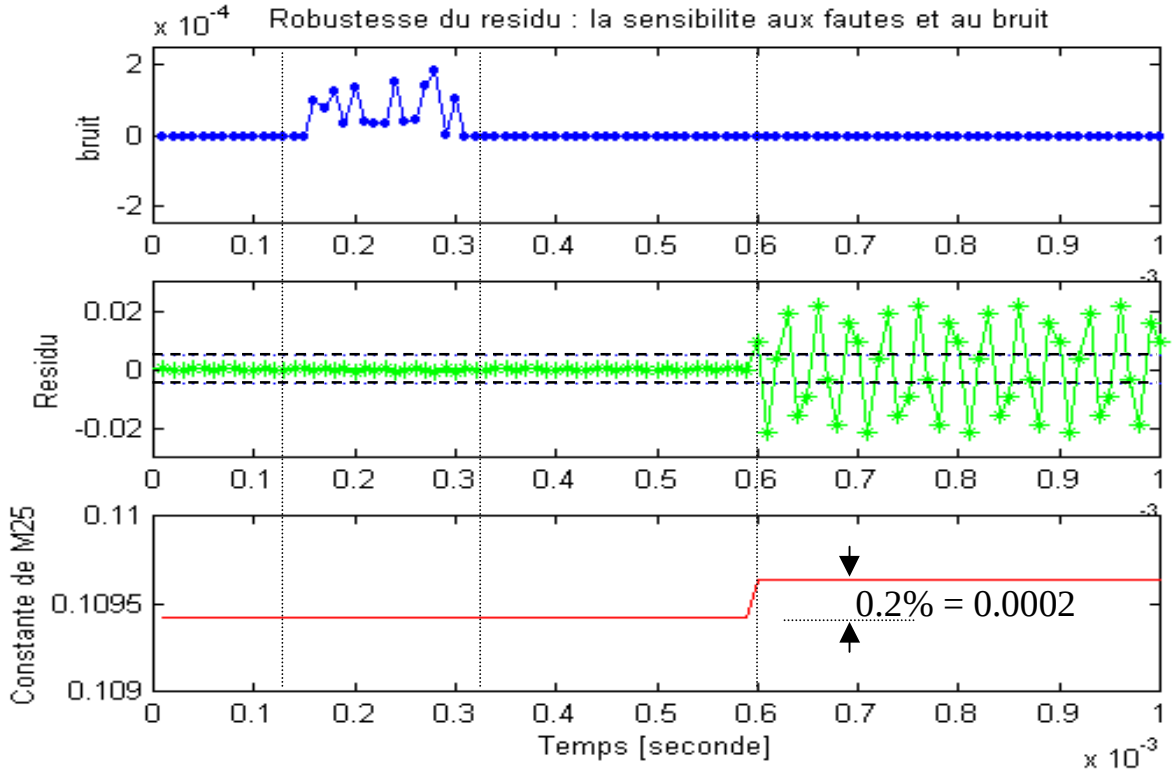
1-Un signal sinusoïdal, d'une fréquence de 30.khz et d'amplitude 1, a été échantillonné (Fs = 100.khz) et appliqué à l'entrée du filtre.

2-Un signal de bruit d'une amplitude 2.10^{-4} a été injecté pendant la simulation dans le circuit du filtre.

3-Une faute de déviation de 0.2% de la constante nominale de multiplication du multiplieur M_{25} a été ensuite injectée dans le circuit du filtre.

La réponse du résidu en présence de bruit seulement a été tout d'abord évaluée. Il a été constaté que le résidu reste stable et il ne sort pas des limites du seuil de détection de fautes, figure 4-6a. Par la suite, la faute a été injectée, ce qui entraîna un dépassement rapide des limites du seuil de détection par le résidu, figure 4-6a. Le circuit robuste est donc insensible au bruit et il est sensible seulement aux fautes.

Figure 4-6a



D'après la figure 4-6a, il est clair que la faute injectée est aussi signalée (détectée) au courant de la simulation et indépendamment des valeurs du signal d'entrée, donc le circuit robuste de test garantit aussi une détection concurrente de fautes.

Pour démontrer que le circuit robuste est optimal pour la sensibilité au bruit par rapport aux autres circuits de test (circuits conçus en utilisant les vecteurs de parité qui se trouvent dans le fondement, V_0, de l'espace de parité), le signal de bruit a été injecté dans le circuit du filtre et les réponses temporelles des résidus, obtenus des circuits de test, ont été évaluées et comparées. Les résultats de simulation ont montré que la sensibilité au bruit du résidu robuste est négligeable par rapport à la sensibilité des autres résidus. Les figures 4-6b et 4-6c montrent que le résidu robuste, qui est toujours dans les limites du seuil de détection, a une sensibilité négligeable au bruit, ce qui n'est pas le cas pour un autre résidu (non robuste) dépassant de temps en temps le seuil permis de détection de fautes.

Figure 4-6b : Réponse temporelle du résidu en l'absence de fautes et de bruit

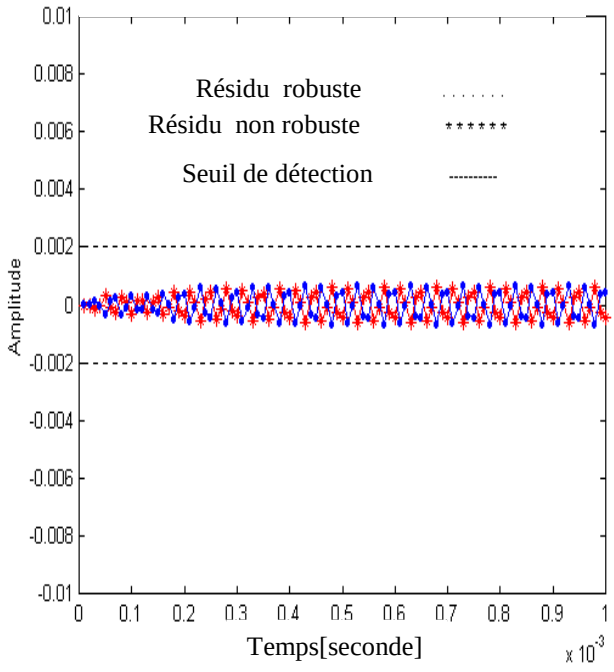
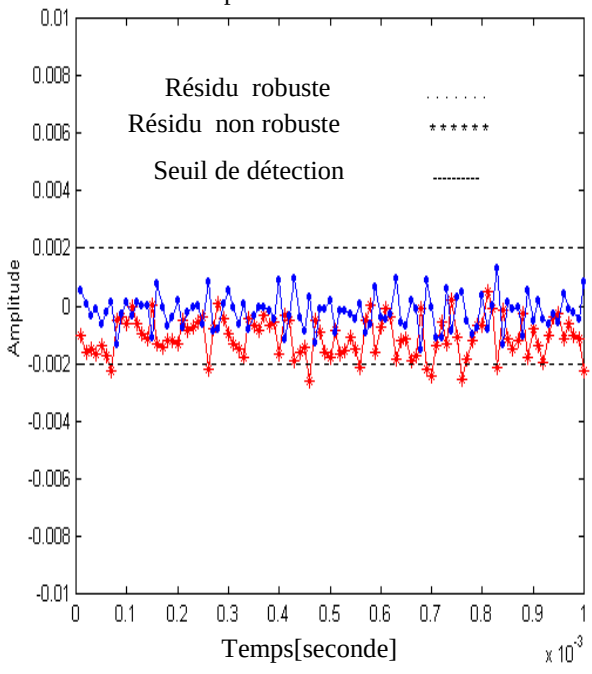


Figure 4-6c : Réponse temporelle du résidu en présence de bruit seulement



4.4 Couverture optimale de fautes

Nous avons vu que la minimisation de la surface du circuit de détection de fautes peut être réalisée en connectant le circuit de détection à des nœuds (points de connexion accessibles) supplémentaires dans le circuit sous test. Ces nouveaux points de test, qui sont choisis arbitrairement parmi les nœuds accessibles directement à la mesure, peuvent être choisis soigneusement et précisément pour augmenter la possibilité de détection de fautes.

4.4.1 Choix de points de test

Pour illustrer la méthode de détermination (choix) de points optimaux de test, le système digital linéaire de la figure suivante est utilisé.

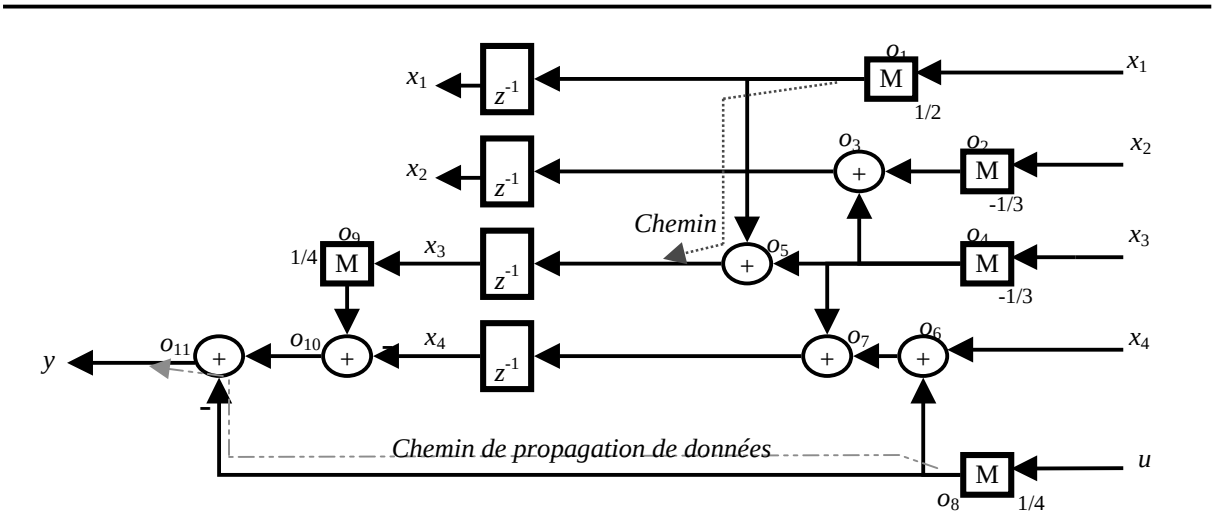


Figure 4-7: Système digital linéaire

Ce système a une entrée *u*, une sortie *y* et il est de l'ordre 4. Les matrice d'état sont :

$$A = \begin{bmatrix} 1/2 & 0 & 0 & 0 \\ 0 & -1/3 & -1/3 & 0 \\ 1/2 & 0 & -1/3 & 0 \\ 0 & 0 & 1/4 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1/4 \end{bmatrix}, \quad C = \begin{bmatrix} 0 & 0 & 1/4 & -1 \end{bmatrix}, \quad D = \begin{bmatrix} -1/4 \end{bmatrix}.$$

Avant d'exposer la méthode de choix de points, nous donnons quelques définitions :

***Définition* 1.** Un chemin de propagation de données entre deux nœuds, N_i et N_j, est une série d'opérateurs interconnectés se trouvant entre les deux nœuds.

***Définition* 2.** Le gain d'un chemin de propagation de données est le produit des gains des opérateurs interconnectés se trouvant sur le chemin. Le gain d'un additionneur est égal à '1', celui d'un soustracteur est égal à '-1' et celui d'un multiplieur est égal à la constante de multiplication.

***Définition* 3.** Le gain total des chemins qui relient N_i à N_j est égal à la somme des gains des chemins.

Pour déterminer les points optimaux, les étapes suivantes sont utilisées :

Etape 1. Construire la matrice des gains des chemins pour tous les nœuds de sorties de tous les opérateurs (additionneur, soustracteur et multiplieur) dans le système.

Cette matrice pour le système de la figure 4-7 est telle que :

Matrice des gains des chemins

↙	S_{o1}	S_{o2}	S_{o3}	S_{o4}	S_{o5}	S_{o6}	S_{o7}	S_{o8}	S_{o9}	S_{o10}	S_{o11}
S_{o1}	1	0	0	0	0	0	0	0	0	0	0
S_{o2}	0	1	0	0	0	0	0	0	0	0	0
S_{o3}	0	1	1	1	0	0	0	0	0	0	0
S_{o4}	0	0	0	1	0	0	0	0	0	0	0
S_{o5}	1	0	0	1	1	0	0	0	0	0	0
S_{o6}	0	0	0	0	0	1	0	1	0	0	0
S_{o7}	0	0	0	1	0	1	1	1	0	0	0
S_{o8}	0	0	0	0	0	0	0	1	0	0	0
S_{o9}	0	0	0	0	0	0	0	0	1	0	0
S_{o10}	0	0	0	0	0	0	0	0	1	1	0
S_{o11}	0	0	0	0	0	0	0	1	1	1	1

s_{oi} : représente le nœud de sortie du $i^{ème}$ opérateur (i = 1, 2, …, 11), et la flèche désigne le sens de propagation de données.

La matrice est remplie de '0' et de '1'. Le '0' signifie que le gain total des chemins reliant s_{oi} à s_{oj}, pour $i \neq j$, est nul ou signifie qu'il n'y a pas de chemin de propagation entre les deux nœuds. Le '1' signifie que le gain total des chemins reliant s_{oi} à s_{oj} est différent de zéro ou signifie que i=j.

***Définition* 4.** Un nœud est un point optimal de test si le nombre de 1s dans sa colonne, dans la matrice des gains, est minimal par rapport aux autres nœuds.

Si le nombre de 1s dans une colonne qui correspond à un nœud $\boldsymbol{s_{oi}}$ *est minimal, alors : Le nombre des autres nœuds (correspondant aux lignes) qui peuvent recevoir de données de* $\boldsymbol{s_{oi}}$ *est minimal ou il n'y a pas d'autre nœud* $\boldsymbol{s_{oj}}$*, où* $i \neq j$*, peut recevoir de données de* $\boldsymbol{s_{oi}}$ *(cas de nombre de 1s est égal à 1). Donc, l'erreur (fautes) de l'opérateur correspondant à* $\boldsymbol{s_{oi}}$ *ne se manifeste, peut être, que sur le nœud* $\boldsymbol{s_{oi}}$*. Pour détecter cette erreur, l'utilisation du nœud* $\boldsymbol{s_{oi}}$ *peut être donc obligatoire. En revanche, un nœud qui correspond à un nombre minimal de 1s dans sa colonne peut être un point de convergence de chemins de propagation de données partant des autres nœuds (cas de nombre de 1s dans la ligne correspondant à ce nœud est supérieur à 1). Donc, des erreurs des autres opérateurs peuvent se manifester et être détectées sur ce nœud. Par conséquent, un nœud qui correspond à un nombre minimal de 1s dans sa colonne représente un point optimal de test car il peut couvrir des erreurs qui ne sont pas couverts par les autres points et des erreurs se manifestant sur des autres nœuds.*

Donc, pour déterminer les points optimaux, les étapes qui suivent sont utilisées :

Etape 2. Calcul de la somme d'éléments de chaque colonne dans la matrice des gains.
Etape 3. Sélectionner des nœuds correspondants aux sommes minimales.

Les nœuds sélectionnés représentent ainsi les points optimaux de test pour une couverture de fautes optimale. Pour le système de la figure 3-12, les points sélectionnés sont s_{o3}, s_{o5}, s_{o7}, s_{o11}, car ces nœuds correspondent aux sommes minimales égales à 1.

Après avoir déterminé les points optimaux de test, et pour calculer les paramètres du circuit de détection devant être connecté aux points optimaux, les points sont exprimés par des lignes dans les matrices C_o et D_o. Ces nouvelles matrices, qui peuvent être différentes des matrices C et D spécifiant les sorties fonctionnelles du système, doivent remplacer C et D dans les algorithmes de détermination de relations de la redondance. Les matrices C_o et D_o, spécifiant les quatre points s_{o3}, s_{o5}, s_{o7} et s_{o11} respectivement sont telles que :

$$C_o = \begin{bmatrix} 0 & -1/3 & -1/3 & 0 \\ 1/2 & 0 & -1/3 & 0 \\ 0 & 0 & -1/3 & 1 \\ 0 & 0 & 1/4 & -1 \end{bmatrix}, \quad D_o = \begin{bmatrix} 0 \\ 0 \\ 1/4 \\ -1/4 \end{bmatrix}.$$

Minimisation du nombre de points de test sans compromettre la couverture de fautes

Compte-tenu de la complexité des systèmes actuels, et pour ne pas dégrader les performances du système en se connectant éventuellement sur des chemins critiques de propagation, le nombre de points de connexion permis pour tester un système peut être limité.

Pour cela nous sommes contraints souvent de réduire le nombre de points de test. Pour résoudre ce problème sans compromettre la couverture de fautes, nous étudions l'arbre de propagation de l'erreur par rapport aux points optimaux. Cette étude permet de choisir parmi les points optimaux des points capables, après certains retards, de couvrir toutes les fautes couvertes par les autres points. Ainsi le nombre de points de test peut être réduit tout en conservant la couverture de fautes.

Arbre de propagation de l'erreur

L'arbre de propagation de l'erreur, par rapport aux points optimaux sélectionnés pour le système de la figure 4-7, est illustré dans la figure 4-8. Les constantes associées aux flèches représentent les gains des chemins de propagation. L'effet de fautes d'un opérateur sur un nœud est indiqué par l'erreur 'e_i', où i = 1, 2, …, 11.

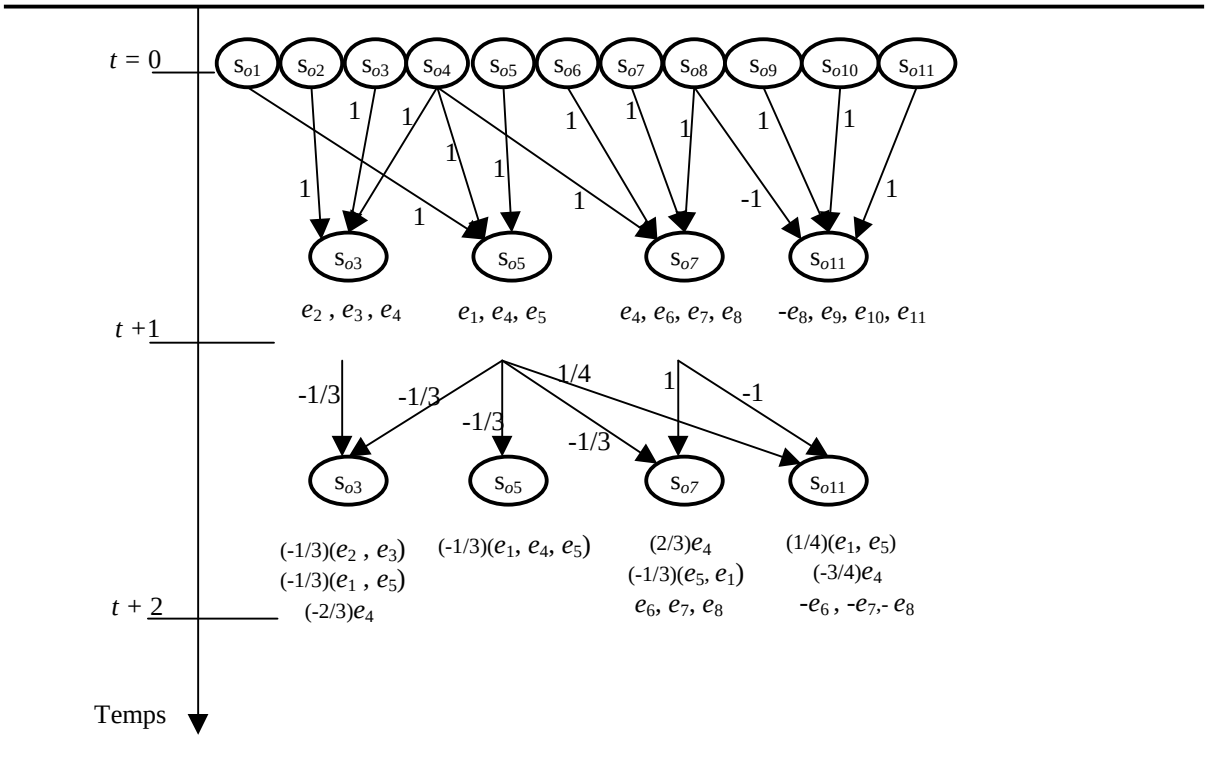


Figure 4-8 : Arbre de propagation de l'erreur

En observant l'arbre de propagation, on trouve que le point s_{o11} peut représenter s_{o5} et $s_{o7,}$ car les fautes détectées sur ces deux derniers points sont détectables également sur s_{o11} après un seul retard. Les effets de fautes (les erreurs) e_9, e_{10} et e_{11} n'apparaissent que sur s_{o11} et ne se propagent pas après avoir été détectés directement sur ce point. Certaines fautes

détectées sur s_{o3} (e_2, e_3) ne sont détectées par aucun autre point optimal. Pour cela, les points optimaux de test pouvant représenter tous les autres points (sans compromettre la couverture de fautes) sont s_{o3} et s_{o11}. Les matrices C_o et D_o correspondant aux s_{o3} et s_{o11} sont respectivement :

$$C_o = \begin{bmatrix} 0 & -1/3 & -1/3 & 0 \\ 0 & 0 & 1/4 & -1 \end{bmatrix}, \quad D_o = \begin{bmatrix} 0 \\ -1/4 \end{bmatrix}.$$

4.4.2 Optimisation de la couverture de fautes par le retard

La relation de la redondance en présence de fautes peut être aussi écrite telle que:

$$r(t) = v^{\mathrm{T}}.\{[Y^{[k]}(t) + Er^{[k]}(t)] - H^{[k]}.U^{[k]}(t)\} \qquad (4\text{-}36)$$

ou encore :

$$r(t) = v^{\mathrm{T}}.[Y^{[k]}(t) - H^{[k]}.U^{[k]}(t)] + v^{\mathrm{T}}.Er^{[k]}(t) \qquad (4\text{-}37)$$

où $Er^{[k]}(t)$ représente les expressions successives du vecteur d'erreur causé par les fautes. Mais nous avons en l'absence de fautes $v^{\mathrm{T}}Y^{[k]}(t) = v^{\mathrm{T}}.H^{[k]}.U^{[k]}(t)$, donc :

$$r(t) = v^{\mathrm{T}}.Er^{[k]}(t) \qquad (4\text{-}38)$$

Le résidu $r(t)$ est différent de 0 si la quantité $v^{\mathrm{T}}.E^{[k]}(t)$ est différente de 0, et $r(t)$ est égal à 0 ailleurs. Donc, en présence de fautes et si le produit $v^{\mathrm{T}}.E^{[k]}(t)$ est égal à 0 alors les fautes vont être non détectables.

Le problème de non détection de ce genre de fautes réside dans la spécificité du vecteur v^{T}. Si v^{T} est optimal dans son espace de parité, P_k, alors aucun vecteur de cet espace peut détecter les fautes. Pour résoudre ce problème, un autre espace de parité doit être recherché. Ce nouvel espace peut être trouvé en effectuant de nouveaux retards qui entraînent l'ajout de nouvelles lignes dans la matrice, $Ob^{[k]}$. Ainsi, un autre vecteur v^{T} optimal, ayant une nouvelle spécificité, différent de son successeur peut être trouvé. Par conséquent les fautes (erreurs) non détectables par l'ancien vecteur peuvent être détectées maintenant par le nouveau vecteur.

Exemple 4 : Soit le système digital linéaire de la figure suivant :

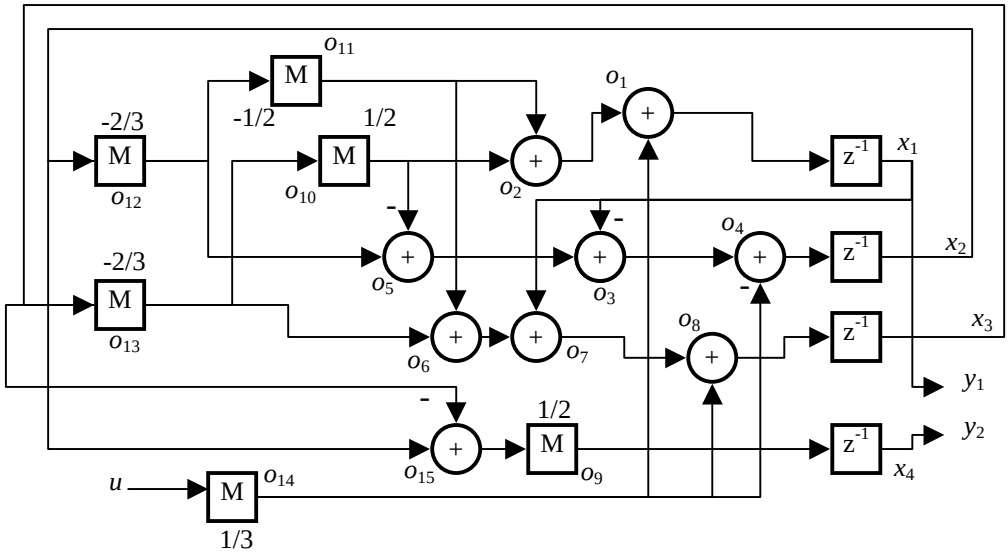


Figure 4-9 :Système digital linéaire

Ce système a une seule entrée (u) deux sorties (y_1, y_2) et quatre variables d'état (x_1, x_2, x_3, x_4). Les matrices d'état sont :

$$A=\begin{bmatrix} 0 & 1/3 & -1/3 & 0 \\ -1 & -2/3 & 1/3 & 0 \\ 1 & 1/3 & -2/3 & 0 \\ 0 & 1/2 & -1/2 & 0 \end{bmatrix};\ B=\begin{bmatrix} 1/3 \\ -1/3 \\ 1/3 \\ 0 \end{bmatrix};\ C=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix};\ D=\begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

Les paramètres du circuit de détection de fautes pour ce système ont été calculés, il s'agit de:

1- L'ordre de relation de la redondance : $k = 1$.

2- Le vecteur de parité optimal : $v^{\mathrm{T}} = \begin{bmatrix} 0 & 0 & -0.8322 & 0.5548 \end{bmatrix}$.

3- Le vecteur : $v^{\mathrm{T}} H^{[1]} = \begin{bmatrix} -0.2774 & 0 \end{bmatrix}$.

La relation de la redondance associée à v^{T} est :

$$r(t) = v^{\mathrm{T}} \begin{bmatrix} y_1(t-1) \\ y_2(t-1) \\ y_1(t) \\ y_2(t) \end{bmatrix} - v^{\mathrm{T}} H^{[1]} \begin{bmatrix} u(t-1) \\ u(t) \end{bmatrix} \tag{4-39}$$

$$r(t) = \begin{bmatrix} 0 & 0 & -0.8322 & 0.5548 \end{bmatrix} \begin{bmatrix} y_1(t-1) \\ y_2(t-1) \\ y_1(t) \\ y_2(t) \end{bmatrix} - \begin{bmatrix} -0.2774 & 0 \end{bmatrix} \begin{bmatrix} u(t-1) \\ u(t) \end{bmatrix} \tag{4-40}$$

En l'absence de fautes :

$$r(t) = -0.8322 y_1(t) + 0.5548 y_2(t) + 0.2774 u(t-1) = 0. \tag{4-41}$$

En présence de fautes :

$$r(t) = \begin{bmatrix} 0 & 0 & -0.8322 & 0.5548 \end{bmatrix} \begin{bmatrix} y_1(t-1) + e_{y1}(t-1) \\ y_2(t-1) + e_{y2}(t-1) \\ y_1(t) + e_{y1}(t) \\ y_2(t) + e_{y2}(t) \end{bmatrix} - \begin{bmatrix} -0.2774 & 0 \end{bmatrix} \begin{bmatrix} u(t-1) \\ u(t) \end{bmatrix} \tag{4-42}$$

$$r(t) = -0.8322[y_1(t) + e_{y1}(t)] + 0.5548[y_2(t) + e_{y2}(t)] + 0.2774 u(t-1). \tag{4-43}$$

$$r(t) = -0.8322 e_{y1}(t) + 0.5548 e_{y2}(t). \tag{4-44}$$

$e_y(t) = \begin{bmatrix} e_{y1}(t) \\ e_{y2}(t) \end{bmatrix}$: est le vecteur d'erreur produit par les fautes et lié aux sorties $y_1(t)$, $y_2(t)$.

Supposons maintenant qu'une faute a causé une erreur e_3 à la sortie de l'opérateur o_3 au temps t. On peut constater que cette erreur se propage et se manifeste au temps t+1 sur le point de variable d'état x_2 seulement. Donc, le vecteur d'erreur lié au vecteur d'état au temps t+1 est : $e_x(t+1) = \begin{bmatrix} 0 & e_3 & 0 & 0 \end{bmatrix}^{\mathrm{T}}$. L'erreur se propage dans le circuit selon les règles des équations suivantes, $i > 1$:

$$e_x(t+i) = A.e_x(t+i\text{-}1). \tag{4-45}$$

$$e_y(t+i\text{-}1) = C.e_x(t+i\text{-}1). \tag{4-46}$$

e_x : est le vecteur d'erreur lié au vecteur d'état x.

e_y : est le vecteur d'erreur lié au vecteur de sortie y.

A : est la matrice d'état du système, et C : Est la matrice de sortie.

Les vecteurs d'erreur successifs $e_x(t+i)$ et $e_y(t+i)$ pour i = 0, 1, 2, …, 5 sont donnés dans le *tableau* 3 suivant :

Tableau 3

		i					
		0	1	2	3	4	5
$e_x(t+i)$	$e_{x1}(t+i)$	0	0	$(1/3)e_3$	$(-1/3)e_3$	$(1/9)e_3$	$(1/9)e_3$
	$e_{x2}(t+i)$	0	e_3	$(-2/3)e_3$	$(2/9)e_3$	$(4/27)e_3$	$(-25/74)e_3$
	$e_{x3}(t+i)$	0	0	$(1/3)e_3$	$(-1/9)e_3$	$(-5/27)e_3$	$(17/74)e_3$
	$e_{x4}(t+i)$	0	0	$(1/2)e_3$	$(-1/2)e_3$	$(1/6)e_3$	$(1/6)e_3$
$e_y(t+i)$	$e_{y1}(t+i)$	0	0	$(1/3)e_3$	$(-1/3)e_3$	$(1/9)e_3$	$(1/9)e_3$
	$e_{y2}(t+i)$	0	0	$(1/2)e_3$	$(-1/2)e_3$	$(1/6)e_3$	$(1/6)e_3$

Dans le *tableau*3, on peut observer que les erreurs, $e_{y1}(t+i)$ et $e_{y2}(t+i)$, produites par l'erreur e_3 sont reliées par la relation suivante :

$$e_{y1}(t+i) = (2/3)\, e_{y2}(t+i) \qquad (4\text{-}47)$$

On peut constater aussi que cette équation est valide quelle que soit la valeur de '*i*'. En remplaçant maintenant l'équation 4-47 dans l'expression de $r(t)$ (équation 4-44), on trouve :

$$\begin{aligned} r(t+i) &= (-0.8322)(2/3)e_{y2}(t+i) + 0.5548e_{y2}(t+i). \\ r(t+i) &= -0.5548e_{y2}(t+i) + 0.5548e_{y2}(t+i) = 0. \end{aligned} \qquad (4\text{-}48)$$

Par conséquent, le résidu, *r*, est toujours nul quelle que soit la valeur de l'erreur, e_3, de l'opérateur o_3. Pour cela, l'erreur e_3 n'est pas détectable par le circuit de détection conçu avec le vecteur : $v^{\mathrm{T}} = [0 \quad 0 \quad -0.8322 \quad 0.5548]$. Pour résoudre le problème, un autre espace de parité est recherché en effectuant un nouveau retard qui entraîne l'ajout de nouvelles lignes dans la matrice $Ob^{[k]}$. Le nouveau vecteur optimal v^{T} ainsi que les autres paramètres correspondants ont été calculés. Ils sont :

1- Ordre de relation de la redondance devient : $k = 2$.

2-Le nouveau vecteur de parité est : $v^{\mathrm{T}} = [-0.2066 \;\; 0 \;\; -0.6488 \;\; 0.2259 \;\; 0.4663 \;\; -0.5175]$.

3- Le vecteur : $v^{\mathrm{T}} H^{[2]} = [-0.1474 \quad 0.1554 \quad 0]$.

La relation de la redondance associée à v^{T} est :

$$r(t) = v^{\mathrm{T}}.\begin{bmatrix} y_1(t-2) \\ y_2(t-2) \\ y_1(t-1) \\ y_2(t-1) \\ y_1(t) \\ y_2(t) \end{bmatrix} - v^{\mathrm{T}} H^{[2]} \begin{bmatrix} u(t-2) \\ u(t-1) \\ u(t) \end{bmatrix}. \tag{4-49}$$

En l'absence de fautes :

$$v^{\mathrm{T}}.\begin{bmatrix} y_1(t-2) \\ y_2(t-2) \\ y_1(t-1) \\ y_2(t-1) \\ y_1(t) \\ y_2(t) \end{bmatrix} = v^{\mathrm{T}} H^{[2]} \begin{bmatrix} u(t-2) \\ u(t-1) \\ u(t) \end{bmatrix} \Leftrightarrow r(t) = 0 \tag{4-50}$$

En présence de fautes :

$$r(t) = v^{\mathrm{T}}.\begin{bmatrix} y_1(t-2)+e_{y1}(t-2) \\ y_2(t-2)+e_{y2}(t-2) \\ y_1(t-1)+e_{y1}(t-1) \\ y_2(t-1)+e_{y2}(t-1) \\ y_1(t)+e_{y1}(t) \\ y_2(t)+e_{y2}(t) \end{bmatrix} - v^{\mathrm{T}} H^{[2]} \begin{bmatrix} u(t-2) \\ u(t-1) \\ u(t) \end{bmatrix}. \tag{4-51}$$

$$\begin{aligned} r(t) = &-0.2066 e_{y1}(t-2) - 0.6488 e_{y1}(t-1) + \\ &0.2259 e_{y2}(t-1) + 0.4663 e_{y1}(t) - 0.5175 e_{y2}(t). \end{aligned} \tag{4-52}$$

ou encore

$$\begin{aligned} r(t+i) = &-0.2066\, e_{y1}(t+i-2) - 0.6488\, e_{y1}(t+i-1) + \\ &0.2259\, e_{y2}(t+i-1) + 0.4663\, e_{y1}(t+i) - 0.5175\, e_{y2}(t+i). \end{aligned} \tag{4-53}$$

En remplaçant les valeurs qui apparaissent dans le *tableau* 3 dans l'expression de $r(t+i)$ (équation 4-53) et pour $i = 2$, on trouve :

$$\begin{aligned} &r(t+2) = 0.4663 e_{y1}(t+2) - 0.5175 e_{y2}(t+2) \\ &r(t+2) = (0.4663)(1/3)e_3 - (0.5175)(1/2)e_3 = -0.1033 e_3 \neq 0. \end{aligned} \tag{4-54}$$

Donc, l'erreur e_3 est détectable au temps t+2. En suivant la même analyse pour les erreurs des autres opérateurs dans le système, on trouve que : les erreurs des opérateurs o_3, o_4, o_5, o_6, o_7, o_8 ne sont pas détectables par le vecteur $v^{\mathrm{T}} = \begin{bmatrix} 0 & 0 & -0.8322 & 0.5548 \end{bmatrix}$ qui est optimal dans son espace de parité de l'ordre 1 tandis que le deuxième vecteur,

$v^T = [-0.2066 \quad 0 \quad -0.6488 \quad 0.2259 \quad 0.4663 \quad -0.5175]$ qui est optimal dans son espace de parité de l'ordre 2, détecte toutes les erreurs de tous les opérateurs. Comme on a 15 opérateurs dans le système, la couverture du premier vecteur est donc 9/15 = 60% tandis que pour le deuxième la couverture atteint 100%.

4.5 Relations robustes et optimales de la redondance

Le concept de recherche des relations de la redondance optimales pour la robustesse et pour la couverture de fautes peut être intégré dans l'algorithme de détermination de relations de la redondance (chapitre 3) pour obtenir l'algorithme final indiqué dans la figure suivante.

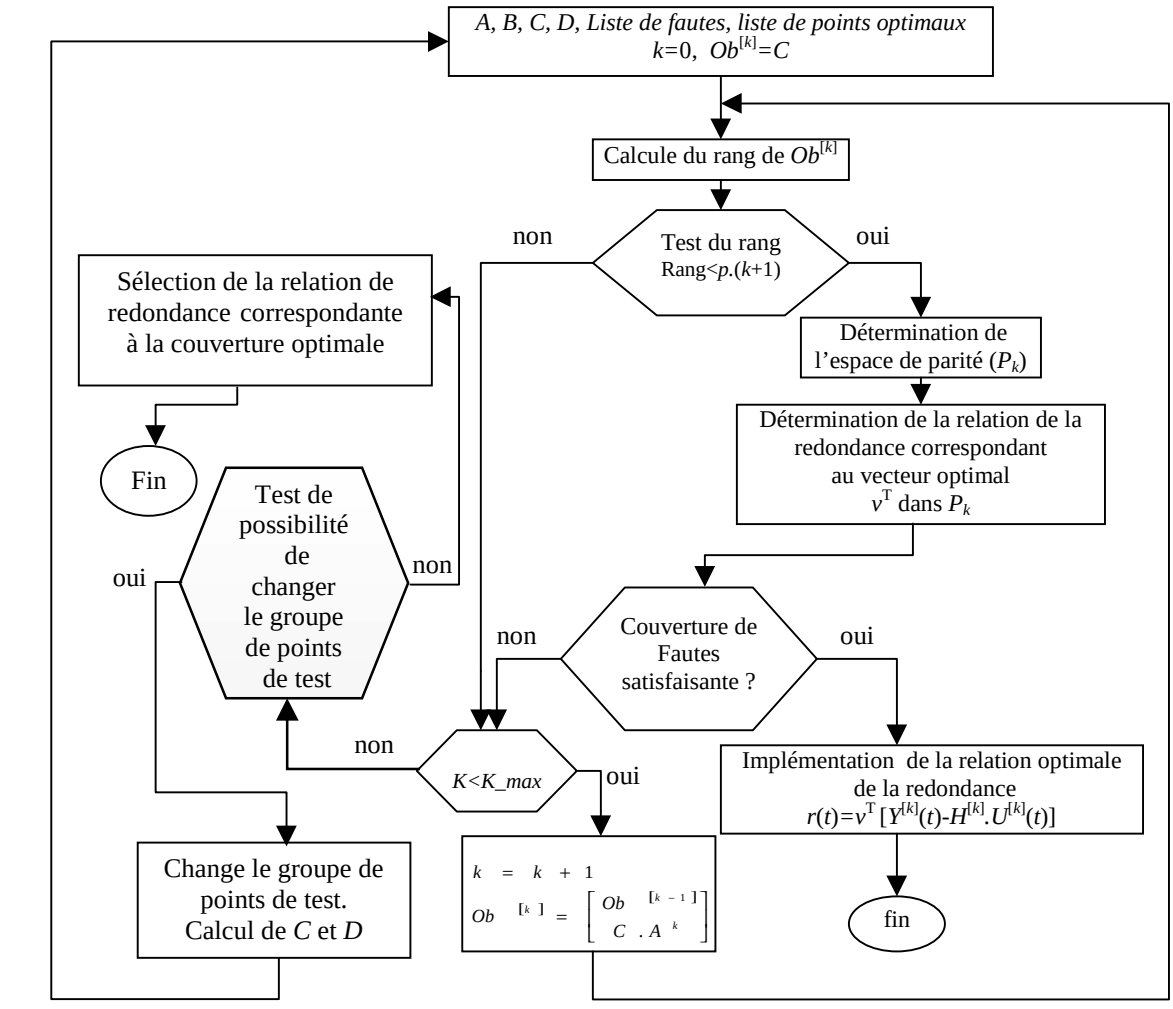


Figure 4-10 : Algorithme de détermination des relations robustes et optimales de la redondance

Les entrées de l'algorithme sont les matrices *A*, *B*, *C*, *D*, la liste de fautes (valeurs de déviations des paramètres de spécification du système à tester) et la liste de points optimaux de test correspondant aux matrices C_o et D_o. L'algorithme détermine tout d'abord le fondement de tous les vecteurs inclus dans l'espace de parité, P_k. Puis il détermine le vecteur optimal v^T dans cet espace. Ensuite, le test de l'efficacité de v^T pour la couverture de fautes suit. Si la couverture est satisfaisante, alors v^T est optimal pour tous les problèmes de l'optimisation, sinon la recherche d'un nouvel espace de parité s'impose.

4.6 Conclusion

Dans ce chapitre, une méthode de minimisation des surfaces des circuits de test concurrent des systèmes digitaux linéaires est donnée. La minimisation de la surface du circuit de test pour un système se fait en exploitant les signaux accessibles à la mesure par connexion directe sur des points de connexion dans le système. Des modèles mathématiques décrivant le comportement réel des systèmes digitaux linéaires sont proposés. Ces modèles sont exploités pour adresser le problème de la sensibilité (robustesse) des résidus utilisés pour indiquer la présence de défauts dans les systèmes à tester. Une méthode de choix de points de test est proposée pour pouvoir sélectionner les points optimaux de connexion et obtenir par conséquent une couverture optimale de fautes. Un algorithme intégrant toutes les procédures nécessaires pour la génération des résidus robustes et optimaux est proposé.

Chapitre 5 : Génération automatique des circuits de test concurrent

5.1 Introduction

Dans ce chapitre, nous présentons un outil de génération automatique des paramètres et des codes **VHDL** des circuits robustes et optimaux de détection concurrente de fautes pour les systèmes digitaux linéaires. L'outil se compose de deux parties principales : l'interface graphique et le noyau de traitement et de calcul.

5.2 Outil de génération

Pour générer automatiquement, et indépendamment des autres outils informatiques, des circuits robustes et optimaux de test concurrent pour les systèmes digitaux linéaires, un outil dédié est développé en C/Visual C++ au cours de la thèse. L'outil, figure 5-1, se compose de deux parties principales : l'interface graphique de saisie de données et de visualisation de résultats et la partie de traitement et de calcul.

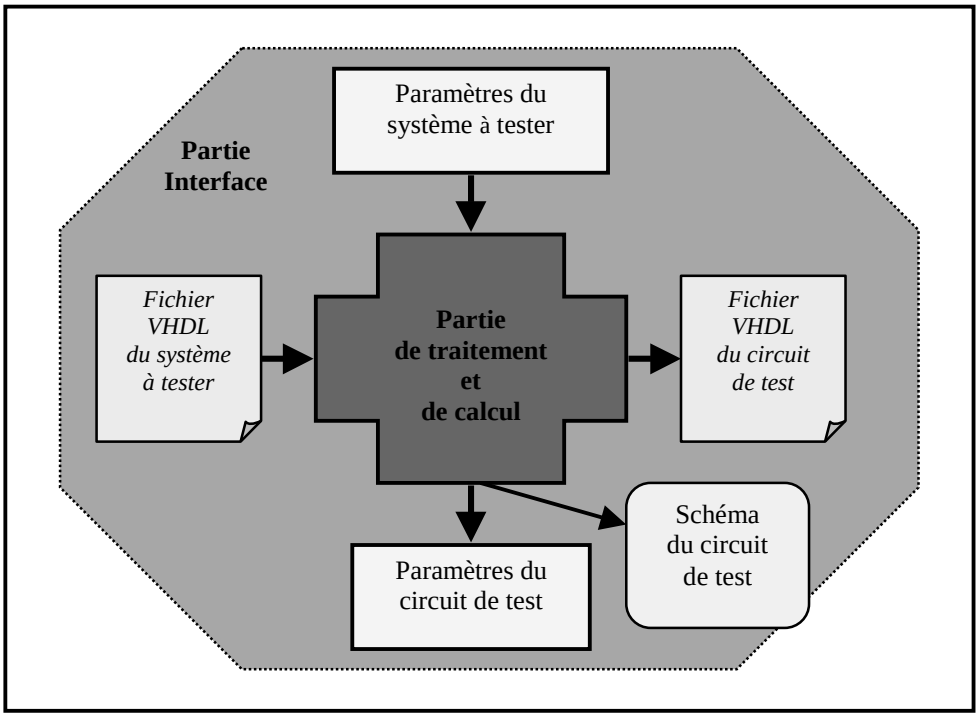


Figure 5-1 : Outil de génération automatique des circuits de test

Partie de saisie de données et de visualisation de résultats

Cette partie représente l'interface entre l'utilisateur et le noyau de traitement et de calcul. Elle permet à l'utilisateur de fournir les données nécessaires, concernant le système à tester, à la partie de traitement et de calcul. Les données peuvent être fournies soit sous une forme des paramètres (constantes et matrice de spécification) du système à tester soit sous une forme d'un fichier comportant la description VHDL du système. Cette partie permet aussi à l'utilisateur de visualiser les résultats résultant de l'opération de traitement et de tracer le schéma du circuit de test. Les résultats obtenus sont les paramètres et le fichier VHDL du circuit robuste et optimal de test.

Partie de traitement et de calcul

Cette partie implémente l'*Algorithme de détermination des relations robustes et optimales de la redondance* ainsi que les *procédures de traitement des fichiers* VHDL. La partie de traitement et de calcul se communique avec la partie de saisie de données et de visualisation pour recevoir les données à traiter. Elle génère à sa sortie les paramètres et les fichiers VHDL des circuits robustes et optimaux de test.

Les fichiers VHDL du filtre passe-bande du chapitre précédent, figure 4-3, et de son circuit robuste optimal de test (généré automatiquement par l'outil proposé) sont respectivement les fichiers LDS.vhdl et LDS_DC.vhdl suivants.

LDS.vhdl

```
Library  iee;.
Use ieee.std_logic_1164.all;
Use ieee.std_logic_arith.all;
Use ieee.std_logic_unsigned.all;
Use work.real_vector_pak.all;

Entity LDS is
        Generic(n : in integer :=4;
                 s : in integer :=3;
                m : in integer :=1);
    port(clk, reset : in std_logic;
         u : in real;
         y : out real_vector(0 to s-1));
end LDS;

architecture beh of LDS is

      signal x : real_vector(0 to n-1);

   begin

      process(clk, reset)

        begin

           if  reset = '0' then
               x <= (others => 0.0);
              y <= (others => 0.0);
         elsif clk'event and clk = '1' then

 y(0)<=0.017702*x(0)+0.3103486*x(1)+0.0207678*x(2)+0.3640988*x(3)+0.109414*u;
y(1)<=0.0*x(0)+0.0*x(1)+1.0*x(2)+0.0*x(3)+0.0*u;
y(2)<=0.0*x(0)+0.0*x(1)+0.0*x(2)+1.0*x(3)+0.0*u;

x(0)<=-0.3277166*x(0)-0.1330587*x(1)+0.7887182*x(2)-0.1561035*x(3)+0.357846*u;
x(1)<=0.1330587*x(0)-0.1847195*x(1)+0.1561035*x(2)+0.9564814*x(3)+0.070825*u;
x(2)<=-0.7887182*x(0)+0.1561035*x(1)+0.0746813*x(2)+0.1831396*x(3)-0.419822*u;
x(3)<=-0.1561035*x(0)-0.9564814*x(1)-0.1831396*x(2)-0.1221374*x(3)-0.0830915*u;

           end if;
      end process;
end beh;
```

LDS_DC.vhdl

```
Library  iee;.
Use ieee.std_logic_1164.all;
Use ieee.std_logic_arith.all;
Use ieee.std_logic_unsigned.all;
Use work.real_vector_pak.all;

Entity LDS_DC  is
        Generic(d : in integer :=1;
                  s : in integer :=3;
                 m : in integer :=1);
    port(clk, reset : in std_logic;
          u : in  real;
          y : in  real_vector(0 to s-1);
         r : out  real);
end LDS_DC;

architecture beh of LDS_DC  is
   type  z_array is array(natural range <>, natural range <>) of real;
     signal  u_in, z_u, r_in, z_uv : real;
     signal  y_in, z_yv : real_vector(0 to s-1);
          constant  VT : z_array :=((46.3682,  4.69297,  -43.5469),
                                    (-76.1061,  -6.6189,  9.7817));
         constant  VTH :  real_vector :=(5.07333,  -7.17308);
   begin
     process(clk, reset)
              variable  r_v  :  real;
          begin
          if  reset = '0' then
              z_yv <= (others => 0.0);
               z_uv <=  0.0;
               z_u  <= 0.0;
               r_in<= 0.0;
               r_v := 0.0;
        elsif clk'event and clk = '1' then
 L1 : for  j  in  0  to  s-1   loop
               z_yv(j) <= y_in(j);
               r_v := r_v+z_yv(j)*VT(1, j);
      end  loop;
L2 : for  j  in  0  to  s-1   loop
               r_v := r_v+y_in(j)*VT(0, j);
      end  loop;
                  z_u <= u_in;
                  z_uv <= z_u;
                  r_v := r_v-z_uv*VTH(1)-z_u*VTH(0);

                  r_in <= r_v;
                  r_v := 0.0;
          end if;
     end process;
       u_in <= u;
       y_in <= y;
      r <= r_in;
end beh;
```

Les fichiers LDS.vhdl et LDS_DC.vhdl ont été regroupés et le fichier résultant a été simulé au moyen de l'outil de SYNOSYS. Le résultat de simulation est illustré dans la figure5-2. Ce résultat confirme les résultats déjà obtenus dans le chapitre précédent.

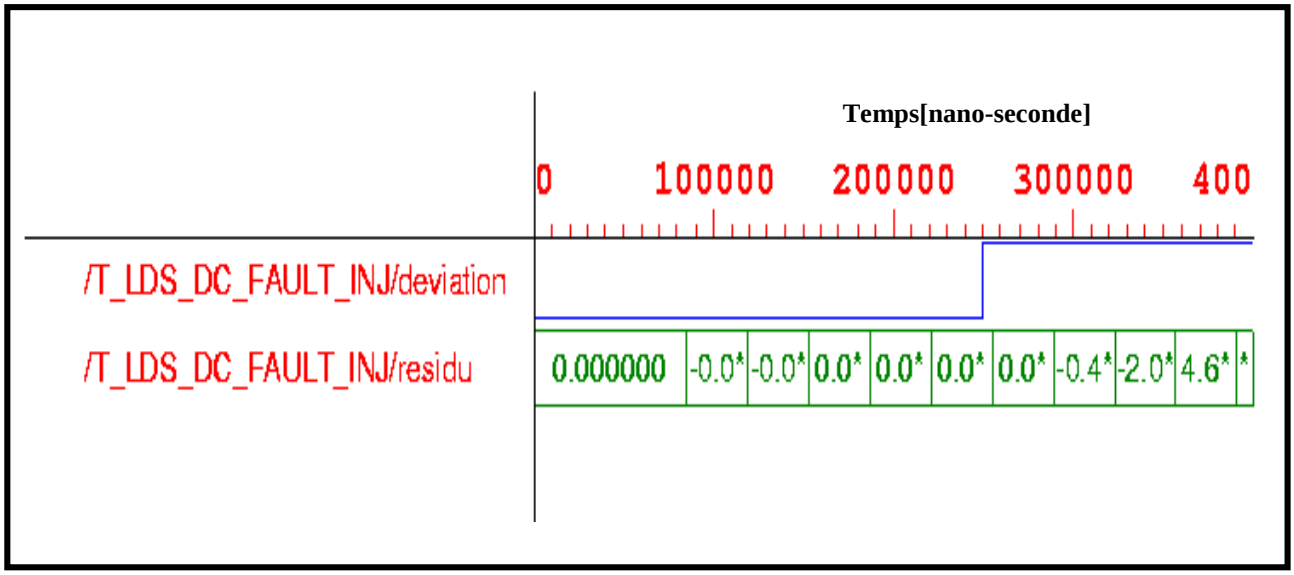


Figure 5-2. Simulation de fautes de déviation de 50% de la constante du multiplieur M_{24}

5.3 Conclusion

Dans ce chapitre, nous avons présenté un outil de génération automatique des circuits robustes et optimaux de test concurrent pour les systèmes digitaux linéaires. Les paramètres ainsi que la description VHDL du circuit robuste et optimal de test pour chaque système digital linéaire sont générés automatiquement soit à partir de la description VHDL du système à tester soit à partir des paramètres du système fournis au noyau de calcul aux moyens de l'interface graphique de l'outil.

Chapitre 6 : Conclusion et perspectives

Conclusion

Le test de circuits intégrés est mis en œuvre pour déceler la présence de dysfonctionnement dans le circuit sous test. La testabilité des circuits complexes pose un problème important depuis déjà un certain temps et il a conduit à l'adoption des techniques de test hors ligne telles que scan, BIST et de test en ligne comme les techniques de codage.

Après avoir présenté les systèmes à tester dans le premier chapitre, nous avons exposé dans le deuxième les techniques actuelles de test hors ligne et en ligne des systèmes digitaux. Nous avons discuté les limitations de ces techniques et l'exigence des applications critiques telles que les applications spatiales et nucléaires. Ce type d'applications exige, pour des raisons de sécurité, d'identifier ou tester les systèmes pendant la phase de fonctionnement normal. C'est pourquoi, il devient pratiquement impossible de tester les systèmes en utilisant les techniques de test hors ligne. Le problème de test en ligne de systèmes digitaux linéaires est très important car une erreur de donnée pendant la période de fonctionnement normal peut conduire à des conséquences funestes. Nous avons constaté que les approches qui sont développées pour le test en ligne des systèmes digitaux sont orientées vers le test des structures spécialisées tandis que le problème du test en ligne des systèmes digitaux linéaires n'est pas visé et adressé directement.

Dans le troisième chapitre de cette thèse nous avons proposé une nouvelle méthode de test en ligne pour les systèmes digitaux linéaires. La méthode proposée est basée sur la technique de la redondance analytique décrivant les relations entre l'historique des signaux d'entrée et de sortie du système à tester. Nous avons montré que cette méthode, garantissant une détection concurrente de fautes, est générale et applicable à tout système digital pourvu qu'il soit linéaire.

Les problèmes de conception des détecteurs de défauts en ligne robustes et optimaux en surfaces et en couverture de fautes étaient adressés dans le quatrième chapitre. La minimisation des surfaces de détecteurs et l'optimisation de la couverture de fautes étaient

faites par la proposition d'un algorithme sélectionnant les points optimaux de test. Des modèles réels des systèmes digitaux linéaires étaient proposés. Ceux-ci comprennent des termes modélisant tous les effets de fautes et de bruit dans le système. Ces modèles étaient exploités pour adresser le problème de la robustesse des détecteurs de défauts.

Dans le cinquième chapitre nous avons proposé un outil de génération automatique de circuits de test en ligne pour les systèmes digitaux linéaires. Les paramètres ainsi que la description VHDL du circuit de test en ligne pour chaque système digital linéaire sont générés automatiquement soit à partir de la description VHDL du système à tester soit à partir des paramètres (matrices de spécification) du système fournis directement à l'outil

Perspectives

Les points optimaux de test sont déterminés en utilisant le concept de matrice des gains des chemins de propagation de données. La matrice des gains ainsi que les matrices spécifiant les nœuds optimaux sont déterminées manuellement à partir de la structure du système à tester. La première perspective de cette étude est donc : à partir d'une description structurelle du système à tester, déterminer automatiquement tous les paramètres et toutes les matrices nécessaires à utiliser pour la génération automatique de circuits de test.

Dans cette thèse, les implémentations matérielles directes des relations de redondance sont étudiées. Celles-ci représentent en effet les implémentations les plus coûteuses en matériel par rapport aux autres implémentations telles que les implémentations programmables par exemple. La deuxième perspective importante de cette étude est donc d'étudier les implémentations programmables des relations de redondance, car pour ce type d'implémentations, le matériel supplémentaire et nécessaire à l'implémentation de test en ligne de test peut être négligeable par rapport au matériel du système à tester.

Bibliographie

[ABFr90] M. Abramovici, M. A. Breuer, A. D. Freidman, ‘‘Digital Systems Testing and Testable Design,’’ Computer science press, 1990.

[AbFu86] J. A. Abraham, K. Fuchs ‘‘Fault and error models for VLSI’’ Proceeding of the IEEE, VOL. 74, NO.5, JuneMay 1986.

[AbSh85] J. A. Abraham, H. C. Shih, ‘‘Testing of MOS VLSI Circuits,’’ Proceeding of international symposium on circuit and systems, pp. 1297-1300, 1985.

[AbSi99] A. Abdelhay, E. Simeu, ‘’Méthode généralisée de test en ligne des systèmes digitaux linéaires,’’ Colloque CAO de circuits intégrés et systemes, Aix en provence (Fuveau), France, Mai 1999.

[AbSi00] A. Abdelhay, E. Simeu, ‘’Analytical Redundancy Based Approach for Concurrent Fault Detection in Linear Digital Systems,’’ IEEE 6th IOLTW, Palma de Mallorca, Spain, July 2000.

[AKSa93a] V. D. Agrawal, C. Kime, K. Saluja, ‘’A Tutorial on Built-in Self-Test, Part 1: Principles,’’ IEEE Design and Test of computers, pp. 73-82, March 1993.

[AKSa93b] V. D. Agrawal, C. Kime, K. Saluja, ‘’A Tutorial on Built-in Self-Test, Part 2: Applications,’’ IEEE Design and Test of computers, pp. 69-77, June 1993.

[AlKi96] M. F. AlShaibi, C. R. Kime, ‘’MFBIST: A BIST Method For Random Pattern Resistant Circuits,’’ Proceeding of IEEE international test conference, pp. 176-185, 1996.

[BaAb83] P. Banerjee, J. A. Abraham, ‘‘Generating tests for physical failures in MOS logic circuits,’’ Proceeding of international test conference, pp. 554-559, 1983.

[BaVi96] G. Bavdoin, F. Virolleau ‘‘Les processeurs de traitement du signal’’ Paris : Dunod, 1996.

[BMcS87] P. H. Bardell, W. H. McAnney, J. Savir, ‘’Built-In Test for VLSI, Pseudorandom techniques,’’ JOHON WILEY & SONS, Poughkeepsie, New York, USA, 1987.

[CCCE84] P. R. Cappello, C. F. Chan, Edwards ‘‘VLSI Signal processing’’ IEEE Press, New York, 1984.

[ChAb93] A. Chatterjee, M. A. d’Abreu ‘‘The Design of Fault-Tolerant Linear Digital State Variable Systems : Theory and Techniques’’ IEEE Trans. On computers, VOL. 42, NO. 7, pp. 794-808, JULY 1993.

[ChCh99] S. Chakrabarti, A. Chatterjee ‘‘On-line Fault detection in DSP Circuits using extrapolated checksums with minimal test points’’ IEEE Internatianal Test Conference, p. 36.1, pp. 955-963, 1999.

[ChGu95] C. A. Chen, S. K. Gupta, ‘’A Methodology to Design Efficient Test Pattern Generators, ‘’ Proceeding of IEEE international test conference, pp. 814-823, 1995.

[ChMa85] Y. H. Choi, M. Malek, ‘’A tolerant FFT Processor, ‘’ Proceeding of fault tolerant computing symposium, pp. 266-271, 1985.

[ChPa96] R. Chandrmouli, S. Pateras, ‘’Testing system on a chip,’’ IEEE Spectum, pp. 42-47, November 1996.

[ChWi84] E. Y. Chow, A. S. Willsky, ‘’Analytical Redundancy and the Design of Robust Failure Detection Systems,’’ IEEE Transaction on Automatic Control, VOL. AC-29, NO. 7, pp. 603-614, July 1984.

[CIRo87] J. L. Carter, V. S. Iyengar, B. K. Rosen, ‘‘Efficient test coverage determination for delay faults,’’ Proceeding of international test conference, pp. 418-427, 1987.

[Coun93] C. Counil, ‘’Auto-test intégré des filtres numeriques,’’ Thèse, Montpellier02, France, 1993.

[Davi86] R. DAVID, ‘’Signature Analysis for Multiple Output Circuits’’, IEEE Transactions on Computers, vol. C-35, No. 9, pp. 830-837, September 1986.

[Davi98] R. DAVID, ‘’Random Testing of Digital Circuits’’, Marcel Dekker Inc., ISBN 0-8247-0182-8, 1998.

[DBHa92] B. N. Dostie, D. Burek, A. S. M. Hassan, ‘’A Multi-frequency Scan-Based BIST Method,’’ Proceeding of IEEE international test conference, pp. 506-513, 1992.

[DBTh89] R. Dekker, F. Beenker, L. Thijssen, ‘’Realistic Built-In Self Test For Static RAMSs,’’ IEEE Design and Test of Computers, pp. 26-34, February 1989.

[DCBh98] D. K. Das, I. Chaudhuri, B. B. Bhattacharga ‘’Design of an optimal Test Pattern generator for Built-in self testing of path delay faults ,’’ Proceeding of the 11^{th} international conference on VLSI Design, 1998.

[DiRo90] E. Dieulesait, D. Royer ‘‘Systèmes linéaires de commande à signaux échantillonnés’’ Paris : Masson, 1990.

[DuVi95] C. Dufaza, H. Viallon C. Chevalier, ‘’BIST Hardware Generator for Mixed Test Scheme,’’ IEEE European Test Conference, pp. 424-430, 1995.

[DuYe97] C. Dufaza, Y. Zorian, ‘’On the generation of Pseudo-Deterministic Tow-Patterns Test Sequence with LFSRs,’’ IEEE European Test Conference, pp. 69-76, 1997.

[EiBe81] B. Eitan, D. F. Bentchkowsky, ‘’Hot electron injection into the oxide in n-channel MOS devices,’’ IEEE Transaction on electron Divices, VOL. 28, pp. 328-340, 1981.

[Fant84] F. Fantini, ‘’Reliability problems with VLSI,’’ Microelectronics and Reliability, 24(2), pp. 275-296, 1984.

[FoGe87] C. Foulard, S. Centil, J. P. Sandraz ‘Commande & Regulation par Calculateur Numerique’’ Editions EYROLLES, Paris, 1987

[FoRi79] M. L. Forcier, R. E. Richardson, ‘’Microwave rectification RFI response in field effect transistors,’’ IEEE Transaction on Electromagnetic Compatibility, EMC-21(4), pp. 312-313, 1979.

[Fran90] P. M. Frank, ‘’Fault Diagnosis in Dynamic Systems using Analytical and knowledge-based Redundancy- A Survey and Some New Results,’’ Automatica, Vol. 26, NO. 3, pp. 459-474, 1990.

[GAYe95] D. Gizopoulos, A. Paschalis, Y. Zorian, ‘’An Effective BIST Scheme for Booth Multipliers,’’ Proceeding of IEEE international test conference, pp. 824-833, 1995.

[GAYe96] D. Gizopoulos, A. Paschalis, Y. Zorian, ‘’An Effective BIST Scheme for Datapaths,’’ Proceeding of IEEE international test conference, pp. 76-85, 1996.

[GiCl90] J. C. Gille, M. Clique ‘’Systemes lineaires, equations d’etat’’ Eyrolles, 1990.

[Goor91] A. J. van de Goor ‘’Testing semiconductor memories,’’ John WILEY & Sons, New York, USA, 1991.

[GoOr97] L. Goodby, A. Orailoglu, ‘‘Frequency-Domain compatibility in Digital Filter BIST,’’ Design automation conference proceedings, pp. 540-545, 1997.

[GoSo96] M. Goessel, E. S. Sogomonyan, ‘’A Parity-Preserving Multi-Input Signature Analyzer and Its Application for Concurrent Checking and BIST,’’ Journal of Electronic Testing, pp. 165-177, August 1996.

[GrNa80] J. Grason, A. W. Nagle, ‘‘Digital test generation and design for testability,’’ Design automation conference proceedings, pp. 175-189,June 1980.

[Grob95] L. Grobelny, Z. M. Dresden, ‘’ Design For Testability for asynchronous devices,’’ 1st IEEE International On-Line Testing Workshop, Novotel Nice Centre, Nice, France, July 1995.

[Hamm50] R. W. Hamming, ‘’Error detecting and error correcting codes,’’ The Bell System Technical Journal, 26(2), pp. 147-160, 1950.

[HuAb84] K. H. Huang, J. A. Abraham, ‘’Algorithm-Based Fault Tolerance for Matrix Operations,’’ IEEE Transaction on Computers, VOL. c-33, NO. 6, pp. 518-528, June 1984.

[JoAb86] J. Y. Jou, J. A. Abraham, ‘’Fault-tolerant Matrix Arithmetic and Signal Processing on Highly Concurrent Computing Structures,’’ Proceeding of the IEEE, VOL. 74, NO. 5, pp. 732-741, May 1986.

[JoAb88] J. Y. Jou, J. A. Abraham, ‘’Fault-Tolerant FFT Networks’’ IEEE Transaction on Computers, VOL. 37, NO. 5, pp. 732-741, May 1988.

[Koep86] S. Koeppe, ‘‘ Modeling and simulation of delay faults in CMOS logic circuits,’’ Proceeding of international test conference, pp. 530-536, 1986.

[KONa86] S. Y. Kung, R. E. Owen, J. G. Nash ‘‘VLSI Signal processing, II,’’ IEEE Press, New York, 1986.

[Kung85] S. Y. Kung, H. J. Whitehouse, T. Kailath ‘‘VLSI and modern Signal processing,’’ Prentice-Hall, 1985.

[LaPa83] S. Laha, J. H. Patel, ‘Error correction in arithmetic operations using time redundency,’’ Proceeding of the fault tolerant computing symposium, pp. 298-305, 1983.

[Laty95] R. Latypov, ‘’On generation of Prescribed Set of test vectors,’’ 1st IEEE International On-Line Testing Workshop, Novotel Nice Centre, Nice, France, July 1995.

[Lfer72] J. Lfermann ‘‘Systemes lineaires, variables d’état’’ Paris : Masson, 1972.

[LoWV86] X. C. Lou, A. S. Willsky, G. C. Verghese, ‘’Optimally Robust Redundancy Relations for Failure detection in Uncertain Systems,’’ Automatica, Vol. 22, No. 3, pp. 333-344, 1986.

[Madi95] V. K. Madisetti, ‘’VLSI, Digital signal processor’’ IEEE press, 1995.

[MaAv80] T. E. Mangir, A. Avzienis ‘‘Failures modes for VLSI and their effect on chip design,’’ Proceeding of the IEEE conference on Circuits and Components, pp. 658-688, 1980.

[Mang84] T. E. Mangir ‘‘Sources of failures and yield improvement for VLSI,’’ Proceeding of the IEEE, VOL. 72, NO.6, June 1984.

[MaWi98] V. K. Madisetti, D. B. Williams ‘‘Digital signal processing handbook’’ CRC press, IEEE press, 1998.

[MaWo79] T. C. May, M. H. Woods, ‘‘Alpha-particle induced soft errors in dynamic memories,’’ CRC press, IEEE Transaction on Electron Devices, ED-26(1), pp. 2-9, 1979.

[May79] T. C. May, ‘‘Soft errors in VLSI- present and future,’’ Proceeding of 29th IEEE conference on Electronic Conponenets, pp. 247-256, 1979

[McCl84] E. J. McCluskey, ‘’A survey of design for testability scan techniques,’’ VLSI design, 5(12), pp. 38-61, 1984.

[McCl85] E. J. McCluskey, ‘’Built-in self test structures,’’ IEEE design and test of computers, 2(2), pp. 29-36, 1985.

[McLe92] G. r. McLeod, ‘’BIST Techniques for ASIC Design,’’ Proceeding of IEEE international test conference, pp. 496-505, 1992.

[MoRa72] P. Monteiro, T. R. N. Rao ‘‘ A Residue Checker for Arithmetic and Logical Operations’’ Proceeding of the fault tolerant computing symposium, pp. 8-13, 1972.

[More98] V. Moreda ‘‘ Test et test intégré de pannes temporelles’’ Thèse, université montpelllier II, 1998.

[NaAb90] V. S. S. Nair, J. A. Abraham, ‘’Real-Number Codes for Fault-Tolerant Matrix Operations On Processor Arrays,’’ IEEE Transaction on Computers, VOL. 39, NO. 4, pp. 426-435, April 1990.

[Neum56] J. V. Neumann, ‘Probabilistic Logics Synthesis or Reliable Organisms from Unreliable Components,’’ Automata Studies, Annals of Mathematical Studies, No. 34, Princeton University Press, pp. 43-98, 1956.

[OpSc75] A. V. Oppenheim, R. W. Scafer ‘‘Digital signal processing’’ Englewood cliffs, NJ : Prentice-HALL, 1975.

[POBB88] M. M. Pradhan, E. J. O'Brien, S. L. Lam, J. Beausang, ''Circular BIST with Partial Scan,'' Proceeding of IEEE international test conference, pp. 719-729, 1988.

[PrRe72] D. K. Pradhan, S. M. Reddy, ''A Design Technique for Synthesis of Fault-Tolerant Adders,'' Proceeding of the fault-tolerant computing symposium, pp. 20-23, 1972 .

[RaTy93] J. Rajski, J. Tyszer, ''Accumulator-Based Compaction of Test Responses,'' IEEE Transaction on Computers, VOL. 42, NO. 6, pp. 643-650, June 1993.

[ReBa90] A. L. N. Reddy, P. Banerjee, ''Algorithm-Based Fault Detection for Signal Processing Applications,'' IEEE Transaction on Computers, VOL. 39, NO. 10, pp. 1304-1308, October 1990.

[RoBr97] Rorabaugh, C. Brilton ''Digital filters designer's handbook withe C++ algorithms'' McGraw-Hill, 1997.

[RuSa89] G. Russell, I. L. Sayers ''Advanced simulation and test methodology for VLSI design,'' Van Nostrand Reinhold, London, 1989.

[SeRa95] A. Sengupta, C. S. Raghavendra, ''Witness approach to Concurrent Error Detection in Multiprocessor Systems,'' 1st IEEE International On-Line Testing Workshop, Novotel Nice Centre, Nice, France, July 1995.

[SiAb01] E. Simeu, A. Abdelhay,''A Robust Fault Detection Scheme for Concurrent Testing of Linear Digital Systems'' IEEE 7th IOLTW, Taormina, Italy, July 2001.

[SiMc73] D. P. Siewiorek, E. J. McCluskey, ''An iterative cell switch design for hybrid redundancy,'' IEEE Transaction on computers, C-22, 290-7, 1973.

[Smit99] S. W. Smith ''The scientist and engineer's guide to Digital Signal Processing'' California technical publishing 1999.

[SRHA84] Y. Savaria, N. C. Rumin, J. F. Hayes, V. K. Agrawal ‘‘Characterization of Soft Error Sources,’’ Repport No. 84-11, McGill University, VLSI design laboratory.

[Syrz87] M. Syrzycki, ‘‘Modeling of spot defects in MOS transistors,’’ Proceeding of international test conference, pp. 148-157, 1987.

[Sze83] S. M. Sze, ‘‘VLSI Technology,’’ McGraw-hill.

[TBGH83] C. Timoc, M. Buehler, T. Griswold, C. pina, F. Scott, L. Hess, ‘‘Logical models of physical failures,’’ proceeding of international test conference, pp. 546-533, 1983.

[TMcC96] N. A. Touba, E. J. McCluskey, ‘’Altering A Pseudo-Random Bit Sequence for Scan-Based BIST,’’ Proceeding of IEEE international test conference, pp. 167-175, 1996.

[Wade94] G. Wade ‘‘Signal coding and processing’’ Cambridge university press, 1994.

[WiPa83] T. W. Williams, K. P. Parker, ‘‘Design For Testability- A survey,’’ Proceeding of the IEEE, 71(1), pp. 98-113, 1983.

[WuFu95] B. Wurth, K. Fuchs, ‘’A BIST Approch to Delay Fault Testing with reduced Test Length,’’ IEEE European Test Confrence, pp. 418-423, 1995.

[YeBe97] Y. Zorian, H. Bederr, ‘’An effective Multi-chip BIST Scheme,’’ Journal of Electronic Testing, pp. 87-95, October, 1997.