N° d'ordre : 212

présentée à

L'UNIVERSITE DES SCIENCES ET TECHNIQUES DE LILLE FLANDRES ARTOIS

pour obtenir le titre de

DOCTEUR EN ELECTRONIQUE

par

Sylvie BARANOWSKI - MAGNIEZ

UTILISATION D'UN MICROCONTROLEUR DANS UNE APPLICATION DE SECURITE

TEST ET EVALUATION DU TAUX DE COUVERTURE DE PANNES ET DE LA SECURITE

Soutenue le 29 Mars 1988 devant la Commission d'Examen

Membres du Jury :	MM.	R.	GABILLARD	Président
		F.	LOUAGE	Rapporteur
		Y.	DAVID	Rapporteur
		V.	CORDONNIER	Examinateur
		J.F.	DHALLUIN	Invité
		B.	LE TRUNG	Invité
		R.	LARDENNOIS	Invité

AVANT PROPOS

Ce travail a été réalisé au Laboratoire de RadioPropagation et Electronique de l'Université des Sciences et Techniques de Lille Flandres Artois dirigé par Monsieur le Professeur Gabillard.

Je tiens à le remercier de m'avoir offert de travailler dans son laboratoire sur un sujet aussi passionnant que la sécurité des systèmes et de l'honneur qu'il me fait en présidant ce jury.

Je remercie également Monsieur le Professeur Louage, directeur de l'ENSAM et Monsieur le Professeur Cordonnier d'accepter de juger ce travail.

Je tiens aussi à remercier Monsieur David, directeur de l'INRETS-CRESTA, de l'intérêt qu'il témoigne envers ce travail et de l'honneur qu'il me fait en le jugeant.

J'exprime ma profonde reconnaissance à Monsieur Dhalluin, ingénieur à MATRA TRANSPORT. Les nombreuses discussions que nous avons eues m'ont permis de progresser dans ce travail. Je le remercie de l'aide qu'il m'a apportée et de sa participation à ce jury.

Monsieur Le Trung, directeur de recherches à l'INRETS-CRESTA, et Monsieur Lardennois, directeur de recherches à MATRA TRANSPORT, ont accepté de juger ce travail et je les en remercie.

Je tiens aussi à remercier tout le personnel du CRESTA pour sa gentillesse.

Et enfin je remercie le personnel du laboratoire, celui de l'UFR et toutes les personnes qui, de près ou de loin, ont contribué à l'aboutissement de ce travail.

SOMMAIRE

PREMIERE PARTIE

LA MISE EN SECURITE DES SYSTEMES DE COMMANDE A MICROPROCESSEUR

I) LA SECURITE DES AUTOMATISMES DANS LES TRANSPORTS TERRESTRES

II) LE PROCEDE DE SECURITE ETUDIE

III) LES TESTS FONCTIONNELS

DEUXIEME PARTIE

EVALUATION DU TAUX DE COUVERTURE DE PANNES D'UN MICROPROCESSEUR

I) METHODES DE DETERMINATION DU TAUX DE COUVERTURE DE PANNES

II) APPLICATION AU SYSTEME ETUDIE

III) LA SIMULATION DES FAUTES

TROISIEME PARTIE

CALCUL DES PARAMETRES DE LA SURETE DE FONCTIONNEMENT DE STRUCTURES MONOPROCESSEUR ET BIPROCESSEUR

INTRODUCTION GENERALE

INTRODUCTION

Les progrès technologiques récents ont permis la mise au point de circuits intégrés très performants permettant de remplacer plusieurs cartes électroniques par un seul circuit.

Le monde des transports, et de la sécurité en général, n'est pas resté insensible à ces progrès permettant la réduction du nombre des cartes de commande, mais surtout une amélioration importante des performances et des possibilités de reconfiguration rapide des systèmes. Les microprocesseurs ont donc fait leur entrée dans les automatismes de commande de processus en sécurité.

Auparavant, les cartes de commande étaient composées de circuits électroniques discrets (transistors, résistances,...) dont les pannes étaient connues; l'étude de sécurité était relativement simple et la conception en sécurité intrinsèque possible. Dorénavant, les cartes sont composées de circuits intégrés dont la complexité est telle qu'il est impossible d'en connaitre toutes les pannes et, par là même, de faire de la conception en sécurité intrinsèque.

Les méthodes de mise en sécurité de ces systèmes à microprocesseurs ont toutes, sous diverses formes, le même but: activer la panne si elle existe, la détecter et inhiber les commandes avant qu'elles ne deviennent dangereuses.

Dans la première partie, nous présentons la méthode que nous avons choisie et qui est basée essentiellement sur l'émission périodique d'un signal déterministe et sur l'exécution de tests fonctionnels cycliques permettant l'activation de toutes les fonctions du microprocesseur.

Après une description détaillée des routines de test réalisées, nous présentons, dans la deuxième partie, une méthode d'évaluation du taux de couverture de pannes d'un microprocesseur (proportion de pannes détectées), méthode que nous appliquons au système étudié.

Dans la troisième partie, notre souci principal est de chiffrer la sécurité obtenue avec notre système. Nous développons ensuite l'apport de la redondance et l'influence des tests fonctionnels cycliques sur l'indice de sécurité et nous concluons cette partie par une comparaison avec un système existant.

1ère PARTIE

LA MISE EN SECURITE DES SYSTEMES DE COMMANDE A MICROPROCESSEUR

I) LA SECURITE DES AUTOMATISMES DANS LES TRANSPORTS TERRESTRES.

I.1) Définitions et ordres de grandeur .

La mise en service d'un système de transport nécessite d'avoir déterminé, au préalable, les valeurs numériques caractérisant sa sûreté de fonctionnement (fiabilité, sécurité, disponibilité, réparabilité...) et, bien entendu, vérifié qu'elles se situaient dans les limites souhaitées.

a) Sûreté de fonctionnement (REF.LAP).

La sûreté de fonctionnement est l'aptitude d'un système à accomplir les tâches qui lui sont confiées dans des conditions données.

Soit e(t), l'état du système à l'instant t.

$$e(t) \in \left\{E_1, E_2, E_3\right\}$$

avec

E_1 : état "en bon fonctionnement"

E_2 : état "en panne non dangereuse"

E_3 : état "en panne dangereuse".

Les principales grandeurs qui caractérisent la sûreté de fonctionnement sont définies ci-dessous:

- la fiabilité R(t).

La fiabilité est la probabilité que le système fonctionne correctement durant l'intervalle de temps $[t_o, t]$, t_o étant l'instant de première mise en service; c'est donc la probabilité qu'aucune panne ne se produise entre t_o et t .

$$R(t) = \mathcal{P}\left\{ e(\tau) = E_1 \ , \ \forall \tau \in [t_o, t] \right\}$$

- la sécurité S(t).

La sécurité est la probabilité que le système ne soit pas dans un état dangereux durant l'intervalle $[t_o, t]$; c'est donc la probabilité qu'aucune panne dangereuse ne se produise entre t_o et t .

$$S(t) = \mathcal{P}\left\{ e(\tau) \in \{E_1, E_2\}, \forall \tau \in [t_o, t] \right\}$$

- la disponibilité A(t).

La disponibilité est la probabilité que le système fonctionne correctement à l'instant t, indépendamment de sa vie antérieure.

$$A(t) = \mathcal{P}\left\{ e(t) = E_1 \right\}$$

- la réparabilité RF(t).

La réparabilité est la probabilité que le système soit réparé à l'instant t sachant qu'il était en panne durant l'intervalle de temps $[t_o, t]$.

$$RF(t) = \mathcal{P}\left\{ e(t) = E_1 / e(\tau) \in \{E_2, E_3\}, \forall \tau \in [t_o, t] \right\}.$$

b) Ordres de grandeur.

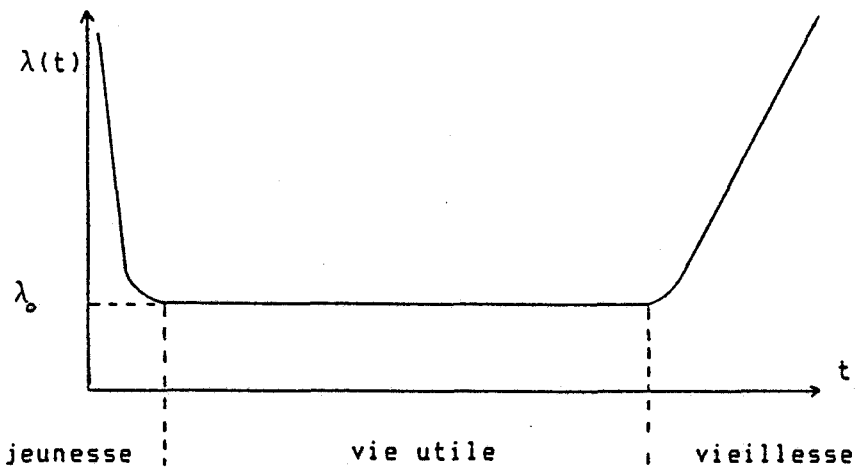
- la fiabilité

Soit f(t) la densité de probabilité de panne d'un système; on définit le taux instantané de défaillance $\lambda(t)$ comme étant égal à:

$$\lambda(t) = \frac{f(t)}{R(t)}.$$

$\lambda(t)$ a pour dimension l'inverse d'un temps.

Les équipements considérés ont une courbe de taux instantané de défaillance en fonction du temps de la forme :



Si les équipements utilisés sont déverminés (la période de jeunesse est révolue) et remplacés dès qu'ils atteignent leur période de vieillesse, le taux instantané de défaillance peut être considéré comme constant et égal à λ_o. La fiabilité s'exprime alors par une fonction exponentielle.

$$R(t) = e^{-\lambda_o t}.$$

Le taux de défaillance horaire du microprocesseur utilisé dans l'application que nous présentons dans les chapitres suivants a été évalué dans REF.DHA1.

$$10^{-5}/h < \lambda_o < 10^{-4}/h$$

- la sécurité

Le niveau de sécurité s'exprime généralement par l'indice de sécurité I_s ou taux horaire de défaillance catastrophique (contraire à la sécurité). Sa définition est la suivante:

$$I_s = - \frac{1}{S(t)} \cdot \frac{dS(t)}{dt}.$$

Pour information, dans le cas du métro VAL, l'ordre de grandeur de la valeur du taux horaire de défaillance catastrophique est:

$I_s < 10^{-6}$ /h pour les systèmes de commande de portes de véhicule

$I_s < 10^{-10}$/h pour le système de contrôle sécuritaire du véhicule (tiroir sécurité).

- la disponibilité

Si la mise en sécurité d'un système consiste à l'arréter dès que l'on constate le moindre écart entre son évolution présente et son évolution désirée, la disponibilité et la sécurité sont des grandeurs qui dépendent inversement l'une de l'autre:

Un système qui ne fait rien (la disponibilité est nulle) est parfaitement sûr (la sécurité est maximale).

On peut aussi mettre un système en sécurité en utilisant la redondance (§ I.2.2) et dans ce cas, la sécurité et la disponibilité évoluent dans le même sens (si on tolère le fonctionnement en mode dégradé).

L'étude présentée vise essentiellement l'amélioration de la sécurité.

- la réparabilité

Le système étudié est composé d'une ou plusieurs cartes de commande à microprocesseur; la réparation est içi très simple: la carte défectueuse est remplacée par une carte en bon état. Cette manière de procéder améliore aussi la disponibilité (temps de réparation plus courts). La réparabilité est une caractéristique dépendant de la structure modulaire du système et donc relativement facile à obtenir.

I.2) La mise en sécurité des systèmes de commande à microprocesseur

Lorsqu'un dispositif est réalisé en sécurité intrinsèque, il prend, en cas de défaillance de l'un de ses constituants, un état de sécurité (état non dangereux) préalablement défini. La conception en sécurité intrinsèque est remise en cause par l'utilisation des microprocesseurs pour la commande de processus en sécurité. En effet, la complexité du circuit est telle qu'il est impossible d'être sûr que le processus prendra l'état de sécurité en cas de défaillances internes; on a donc recours à la conception en sécurité probabiliste (calcul de la probabilité d'apparition des pannes résiduelles dangereuses).

Différentes méthodes de mise en sécurité de systèmes à microprocesseur existent: les tests, la redondance, les codages,...etc.

On distingue ainsi:

I.2.1) Les méthodes "hors ligne".

Les méthodes de mise en sécurité "hors ligne" consistent à prendre des précautions lors de la conception du système et/ou à effectuer des tests de ses composants, le système n'étant pas en service.

Citons quelques exemples:

- l'utilisation de composants de fiabilité élevée
- le retrait systématique des composants ayant atteint la limite de leur vie utile
- le test de fonctionnement

Ce test consiste à vérifier le fonctionnement d'un microprocesseur en observant son comportement dans un environnement donné; deux méthodes principales existent: le test déterministe et le test aléatoire.

- le test déterministe.

Il consiste à envoyer au microprocesseur un ensemble de vecteurs de test prédéterminés et à comparer le comportement du circuit testé à celui d'un circuit de référence ou à un ensemble de vecteurs de sortie issus de simulations (REF.VEL) (REF.BEL1) (REF.BEL2).

- le test aléatoire.

Il consiste à envoyer à deux circuits (celui de référence et celui sous test) une séquence de vecteurs pseudo-aléatoires de longueur déterminée et à comparer les deux sorties.(REF.THE)

Ces tests permettent de vérifier le fonctionnement d'un circuit avant de le mettre en service mais peuvent aussi servir à effectuer des tests hors-ligne périodiques si cela s'avère nécessaire.

I.2.2) Les méthodes "en ligne".

La surveillance en ligne est un moyen efficace de limiter les conséquences de l'apparition de pannes dangereuses: les pannes sont détectées (avec une certaine probabilité de détection) et les commandes émises par le microprocesseur sont inhibées avant que leurs conséquences ne deviennent dangereuses.

Plusieurs méthodes existent, citons en quelques unes:

- la conception de circuits de sécurité.

Une méthode consiste à concevoir un microprocesseur de sécurité disposant d'un maximum de protections internes: codage des informations internes (ex: parité), redondance des blocs internes sensibles (ex: la RAM) avec utilisation d'un code double-rail,...etc. (REF.BIE)

Une seconde méthode est basée sur la conception de circuits facilement testables (microprocesseurs ouverts): le nombre d'instructions et de blocs fonctionnels est limité au strict nécessaire, toutes les informations utiles au test (externe) sont accessibles à l'utilisateur,...etc. (REF.HSURF)

La nécessité de la conception et de la réalisation de circuits intégrés très spécialisés donc de petite série constituent le gros inconvénient de ces deux méthodes. En effet, ceci n'est pas accessible à beaucoup de concepteur de systèmes de sécurité qui préfèrent utiliser des composants du commerce, moins chers et dont la grande diffusion offre une garantie de bonne conformité fonctionnelle (les défauts de conception sont éliminés au fur et à mesure de leur apparition).

- les logiciels de test.

Il s'agit d'implanter en mémoire un programme, exécuté périodiquement, qui permet de tester les différentes unités du microprocesseur (l'unité arithmétique et logique, les mémoires, les bus,...etc).

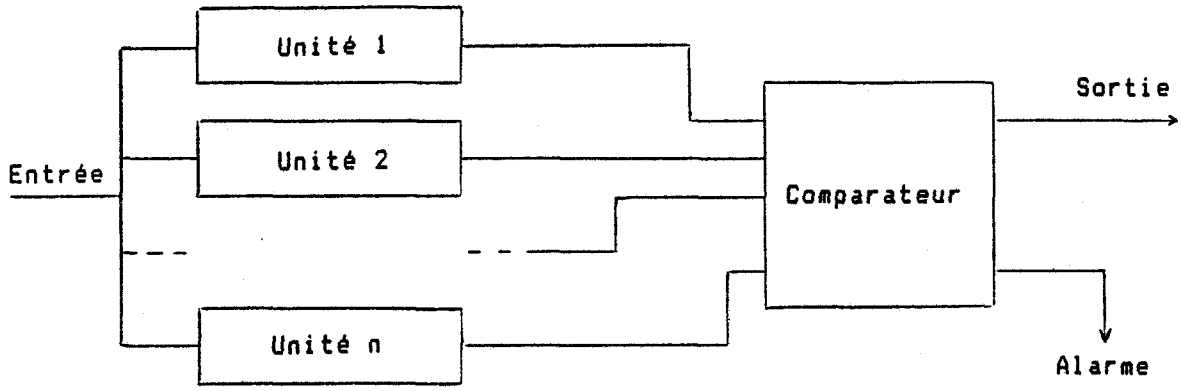
La vérification du bon fonctionnement du circuit peut être faite, par exemple, en contrôlant un signal de chien de garde émis lors du déroulement du programme ou un message codé reflétant, sous forme compacte, la valeur des vecteurs de sortie (signature).(REF.SAU)(REF.JOU)

Cette méthode requiert l'utilisation d'un contrôleur, souvent spécialisé, sachant décoder l'information de sortie (détecteur de variation temporelle de signal de chien de garde, analyseur de signature) et sur lequel une étude de sécurité doit aussi être faite (ces circuits peuvent être conçus en sécurité intrinsèque).

- la redondance.

La redondance est un terme général signifiant que l'on confie le même traitement à plusieurs unités dont on compare les résultats pour détecter leurs anomalies de fonctionnement.

Le schéma général est le suivant:



Il existe deux types de redondance:

* la redondance homogène:
  les unités et les programmes sont identiques (REF.PIL) (REF.CAS).

* la redondance hétérogène:
  les unités sont différentes et/ou les programmes sont différents (mais le traitement reste le même) (REF.SEV).

Le comparateur permet de vérifier la concordance des résultats issus des différentes unités (aspect sécurité) et, quand c'est un voteur k parmi n (avec $k > \frac{n}{2}$ et $n > 2$), il déconnecte la ou les unités défaillantes et le fonctionnement du système peut continuer en mode dégradé (aspect disponibilité).

Pour les redondances, la sécurité repose esentiellement sur:

- le comparateur, il doit être étudié avec soin
- l'impossibilité d'apparition de pannes de mode commun, ceci doit être démontré.

De plus, la synchronisation des différentes unités peut poser des problèmes, surtout pour les redondances hétérogènes.

- le codage.

Les informations à traiter sont codées. Toute anomalie de fonctionnement lors du traitement (pannes de microprocesseurs, parasites,...) transforme (avec une probabilité donnée) le mot de sortie codé correctement en un mot de sortie hors-code.

Cette méthode nécessite un encodeur en entrée et un décodeur-détecteur d'erreurs en sortie.

La sécurité repose principalement sur le décodeur qui valide la sortie et sur le choix du code. En effet, plus le code est sensible aux perturbations, meilleure est la détection (probabilité de non détection plus faible).

Un exemple de cette méthode est donné dans REF.SAC.

- les solutions mixtes.

Toutes les méthodes présentées ont des limites et il est parfois nécessaire d'associer plusieurs procédés afin d'obtenir l'indice de sécurité souhaité; parfois même, l'indice de sécurité étant suffisant, une redondance supplémentaire permet d'augmenter la disponibilité du dispositif.

Un exemple de solution mixte est donné dans REF.DAV où le système PUSH utilisé pour le métro de Hambourg associe la redondance et les logiciels de test.

- conclusion.

Nous nous sommes limités à présenter quelques exemples de solutions au problème de la mise en sécurité de systèmes de commande à microprocesseur; cette liste n'est bien entendu pas exhaustive.

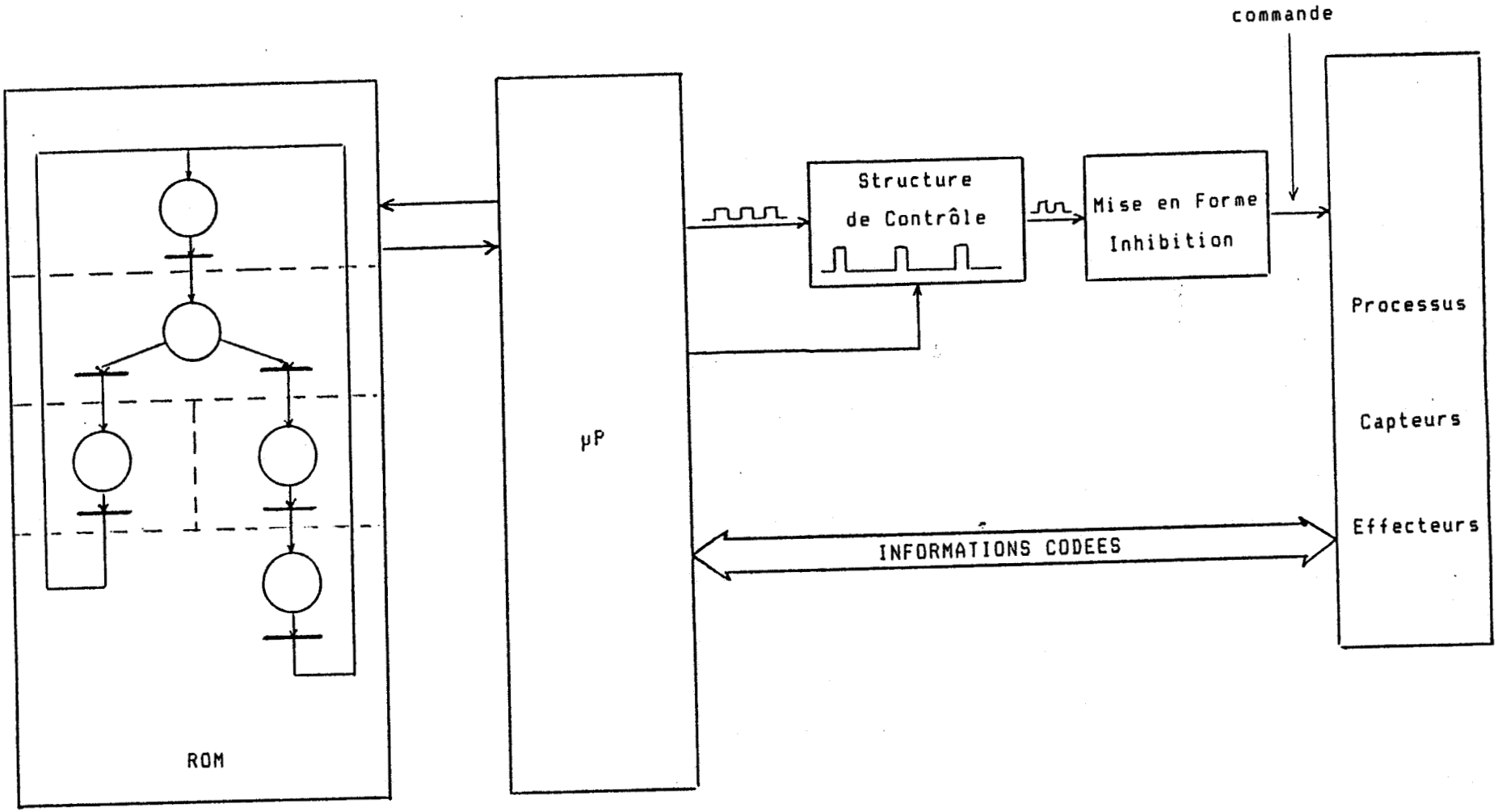
Selon les objectifs à atteindre:

- sécurité élevée au détriment de la disponibilité
- amélioration de la disponibilité
- encombrement réduit: un système redondant augmente considérablement la taille de la carte de commande
- vitesse du processus à commander: des logiciels de test exécutés cycliquement ralentissent considérablement le déroulement du programme d'application,

une méthode, ou une association de méthodes, est choisie plutôt qu'une autre; mais la mise en sécurité reste du domaine de l'imagination des concepteurs et il n'existe pas de solution unique à un problème donné.

I.3) La méthode choisie.

Le chapitre suivant présente la méthode choisie; elle est basée sur un dispositif de chien de garde amélioré et sur des logiciels de test exécutés de façon cyclique. Dans la dernière partie, nous évaluerons l'apport de la redondance sur un tel dispositif.



PRESENTATION GENERALE DU SYSTEME ETUDIE

II) LE PROCEDE DE SECURITE ETUDIE.

II.1) Présentation générale (REF.DHA2).

II.1.1) Le processus commandé.

Le processus commandé est un processus de sécurité. Sa vitesse d'évolution est très lente: le temps de changement entre deux états consécutifs n'excède pas 10 ms.

II.1.2) La commande.

Le caractère discontinu du processus à commander permet de représenter l'évolution de la commande par un graphe de PETRI. Une méthode d'implantation de ces graphes sur microprocesseurs a été mise au point au laboratoire.(REF.ELK)

L'intérêt de cette méthode est d'organiser l'écriture et le traitement du programme de commande du processus de façon très modulaire.

Chaque action (étape transition-commande) se présente sous forme d'un module de programme élémentaire.

En cours de traitement un moniteur enchaine l'exécution des différents modules correspondant au marquage du réseau de Pétri.

Cette modularité confère au logiciel de commande une grande sûreté de fonctionnement. La description de la commande par un réseau de Pétri tient compte des aspects sécurité:

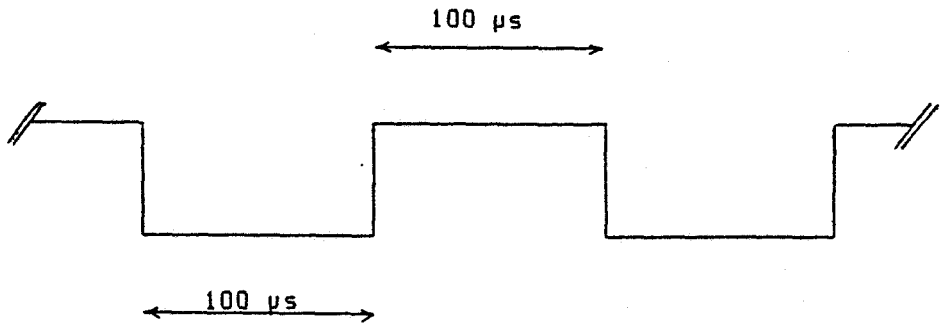
- détection de pannes sur les capteurs
- détection de pannes sur les effecteurs
- traitement "anti-rebond",....

De plus, la sécurité des transmissions est prise en compte par le codage des informations (REF.OUA).(REF.GAB).

Les aspects sécurité "processus - processeur" étant pris en compte, il ne reste plus alors que les aspects "processeur - processus".

La modularité du programme d'application a permis l'implantation d'un procédé de test temporel. Celui-ci consiste à normer temporellement les différents modules de manière à faire émettre périodiquement par le microprocesseur, un signal nommé signal équitemps.

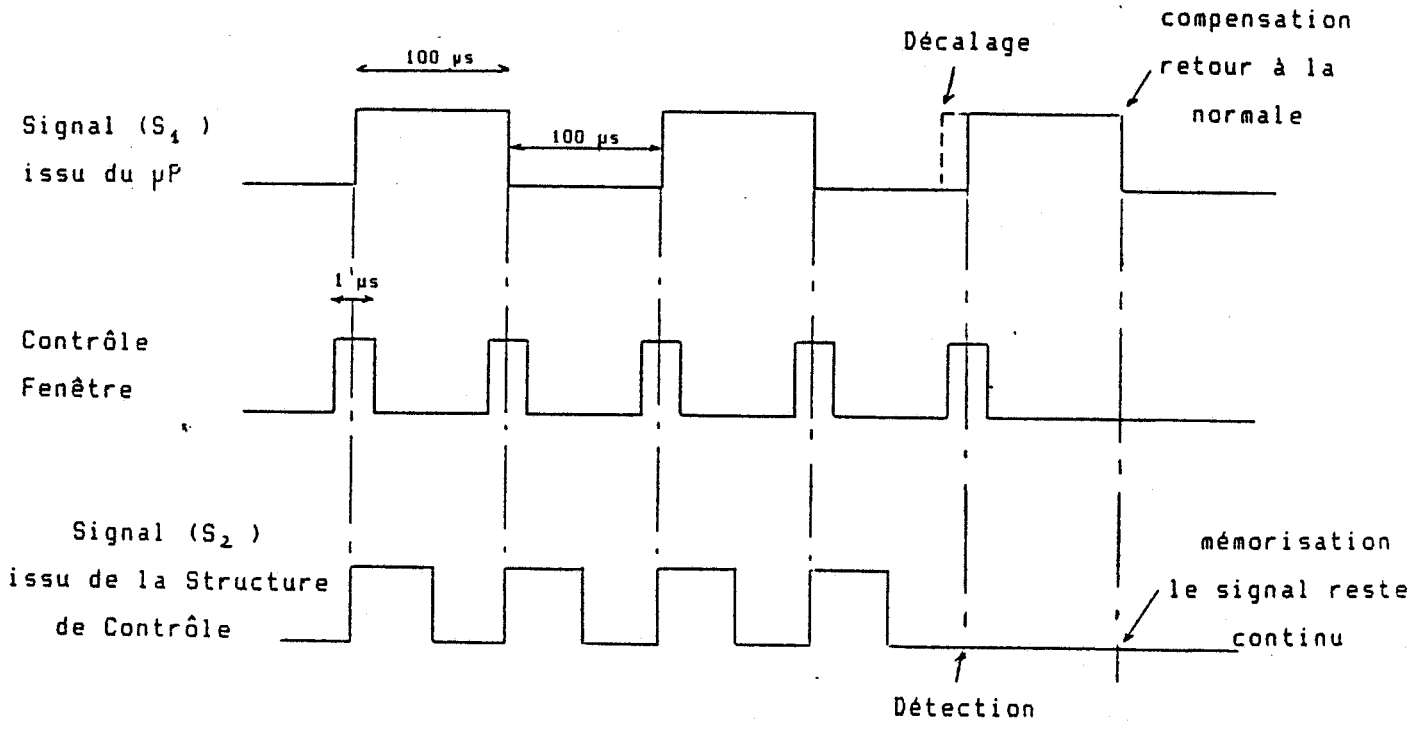
Ce dernier est un créneau de fréquence fixe 5 kHz et de rapport cyclique fixe 1/2.



Ce signal est émis en permanence durant l'exécution du programme et toute variation de fréquence et/ou de rapport cyclique traduit un mauvais fonctionnement du microprocesseur.

Le signal équitemps est surveillé par une structure de contrôle externe (REF.BAR) qui décide de la conformité ou non de ce signal. Un écart de 1µs (ou plus) est détecté; 1µs correspondant à la durée minimale d'une instruction du microprocesseur choisi (le 8031 de Intel) et fonctionnant avec une horloge de 12MHz.

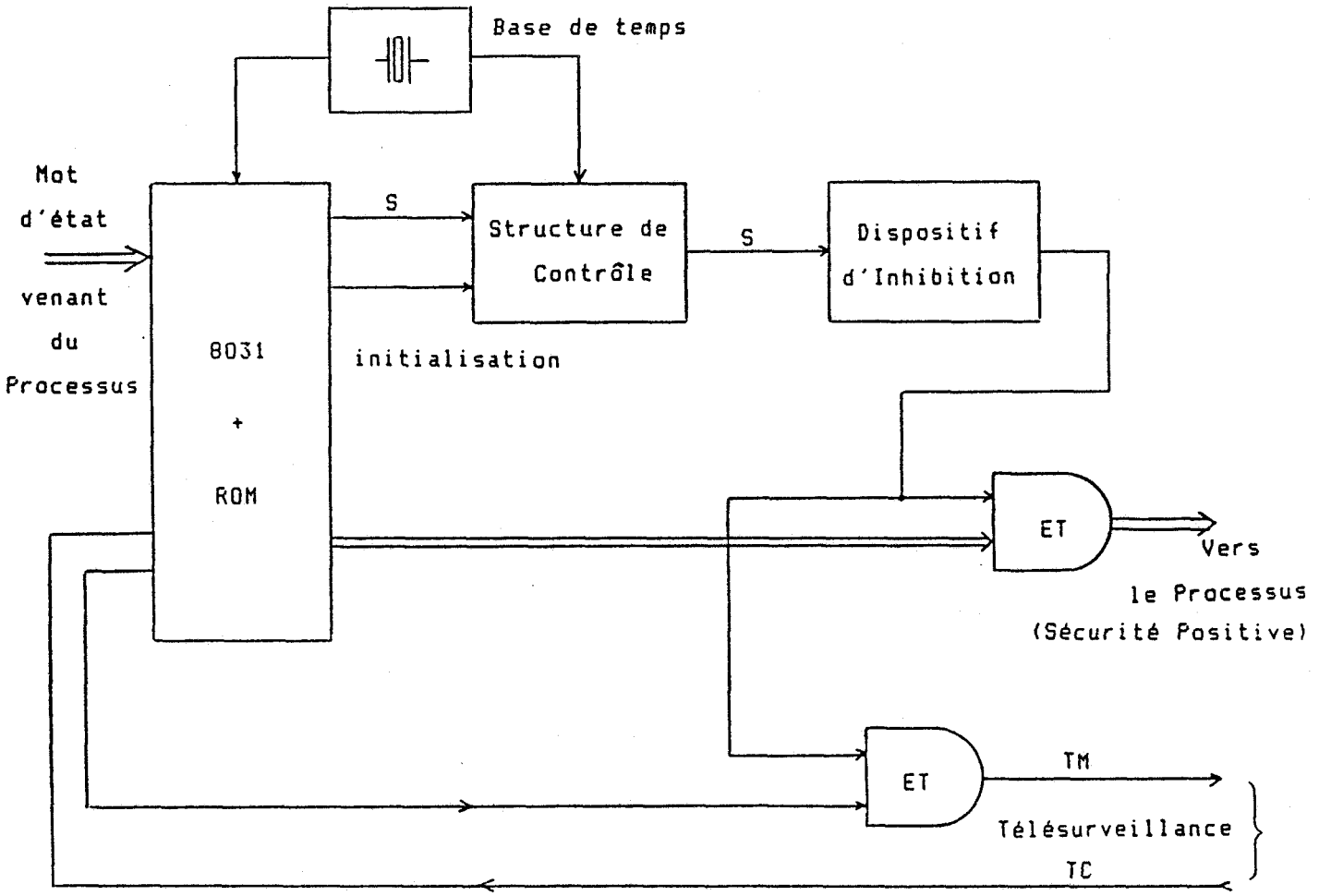
FONCTIONNEMENT DE LA STRUCTURE DE CONTROLE



Dès qu'un front du signal équitemps ne se présente pas dans la fenêtre de 1µs, il y a détection de l'erreur.

De plus, elle est mémorisée puisque la structure de contrôle ne peut plus retourner dans l'état lui permettant de délivrer à nouveau un signal de bon fonctionnement même si par ailleurs le front suivant se présente dans la fenêtre; ceci permet de s'affranchir des possibilités de compensation d'erreurs.

SYNOPTIQUE DE L'UNITE DE TRAITEMENT



TM : Télémesure TC : Télécommande

L'observation temporelle ou contrôle du signal équitemps permet de détecter les défaillances du microprocesseur si celles-ci modifient naturellement la durée d'exécution du programme.

Toutefois, il est nécessaire d'introduire des procédures de test de la partie opérative du microprocesseur, partie essentiellement logique dont les défauts n'altèrent pas forcément le signal équitemps.

En effet:

- le temps d'exécution d'une opération arithmétique ou logique défectueuse n'est pas nécessairement altéré bien que le résultat de l'opération soit faux
- le rangement d'une donnée au mauvais endroit peut durer le même temps que son rangement au bon endroit.

La contrainte majeure est d'éviter que le microprocesseur puisse, en cas de défaillance, envoyer des commandes qui rendent le processus dangereux. De ce fait, les programmes de test doivent permettre de savoir si le microprocesseur est défaillant ou non. La localisation du défaut n'est pas utile, seul l'aspect sécurité est étudié, la réparation se limitant au remplacement de la carte défectueuse.

Les routines de test sont donc organisées de manière à altérer le signal équitemps en cas de défaillance du microprocesseur.

De ce fait :

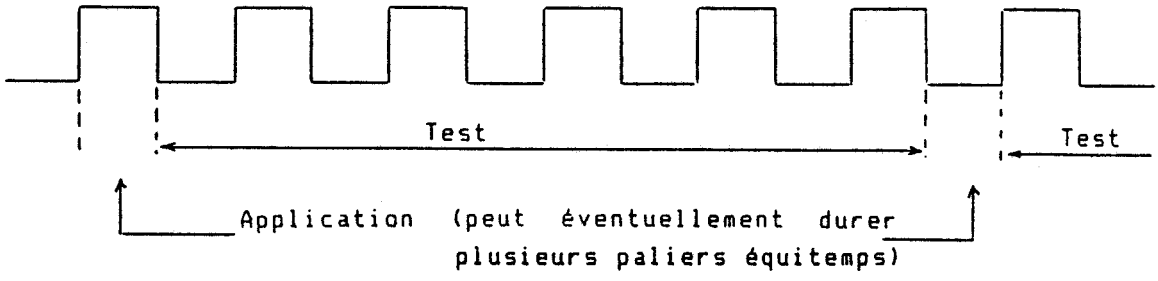
le schéma général de l'unité de traitement présenté précédemment suffit, seule la taille de la ROM est augmentée.

la structure de contrôle, en validant ou non le signal équitemps, sert donc aussi de contrôleur de logiciel de test.

Les programmes de test doivent, de plus, s'imbriquer parfaitement dans le programme principal, ne pas en géner le déroulement normal ni altérer le signal équitemps lors du passage d'un programme à l'autre.

Les programmes de test sont donc découpés en modules et un module est exécuté entre deux traitements de place de réseau de Pétri (que les transitions soient ou non franchies).

L'évolution du processus à commander étant lente vis à vis de la durée de traitement d'une place, le microprocesseur peut, par rapport au processus, rester oisif 90% du temps; ce temps disponible est mis à profit pour exécuter les routines de test.



II.2) Les tests fonctionnels.

Les tests fonctionnels comme le nom l'indique servent à tester les fonctions réalisées par le microprocesseur. Ils portent essentiellement sur la partie opérative du microprocesseur :

éléments de mémorisation: registres, ROM,RAM

opérateurs: UAL, circuits de décalage,...

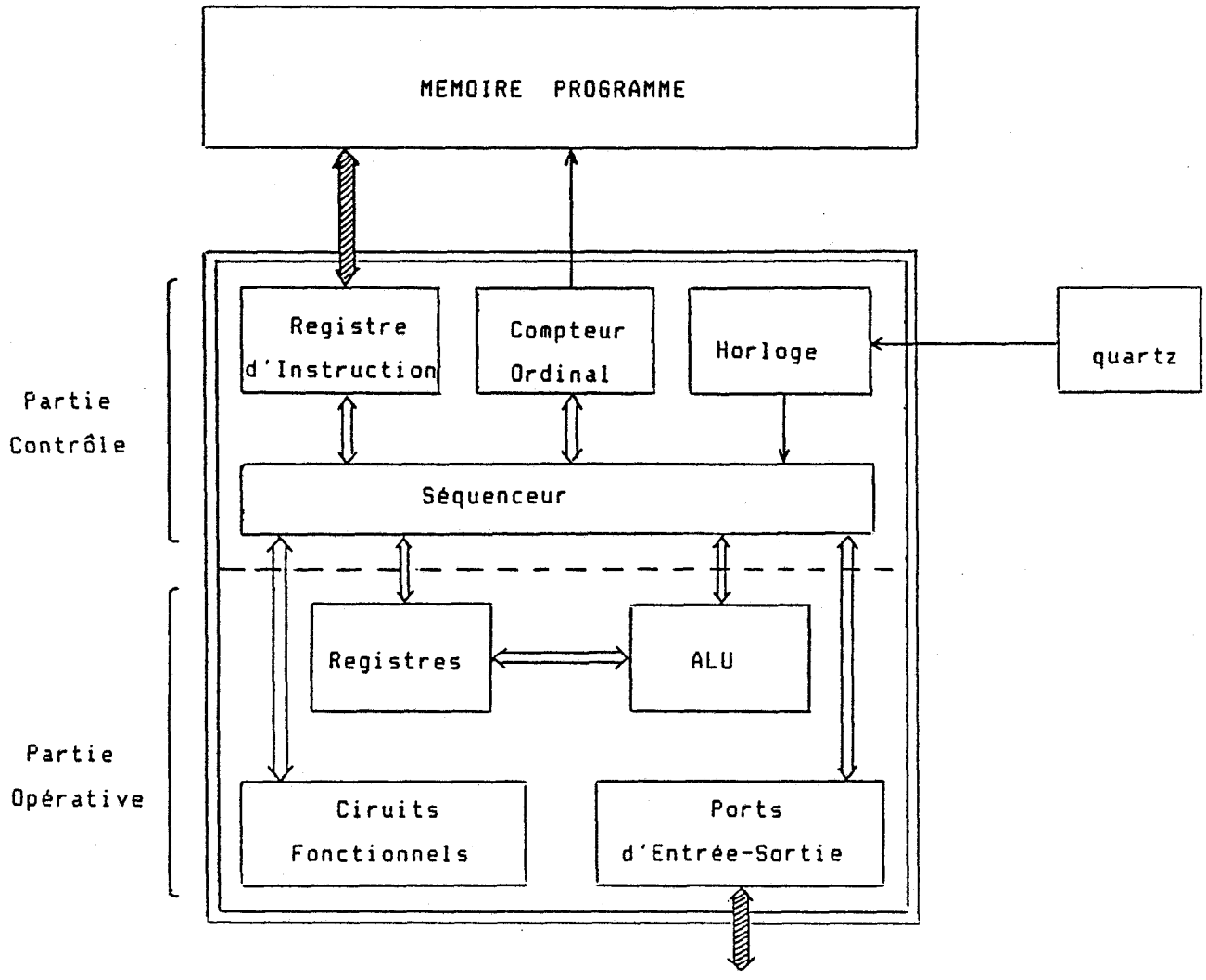
bus de données et d'adresses, circuit d'adressage...

La partie opérative est décomposée en blocs testés individuellement.

La partie contrôle du microprocesseur (séquenceur, décodeur d'instructions et d'adresses,...) n'est pas directement accessible à l'utilisateur, elle ne peut donc être testée que par ses actions à travers la partie opérative. De plus, c'est elle qui coordonne le fonctionnement du microprocesseur (séquencement) et une défaillance à ce niveau a de forte chance de se traduire par des décalages temporels.

Les hypothèses de pannes portent donc essentiellement sur la partie opérative qui est la seule directement accessible à l'utilisateur donc au test.

Le chapitre 3 présente la démarche suivie lors de l'élaboration de ces programmes de test.


MEMOIRE PROGRAMME
Registre d'Instruction
Compteur Ordinal
Horloge
quartz
Partie Contrôle
Séquenceur
Registres
ALU
Partie Opérative
Ciruits Fonctionnels
Ports d'Entrée-Sortie

III) LES TESTS FONCTIONNELS (REF MAG-S).

III.1) Les hypothèses de pannes sur la partie opérative.

La partie opérative est constituée :

- des éléments de mémorisation
- des opérateurs (UAL,...)
- des bus de données et d'adresses et circuits d'adressage

III.1.1) Les éléments de mémorisation.

Pour chaque registre et pour un registre et son voisin (test de diaphonie), on considère:

- les collages simples ou multiples de n bits quelconques
- les courts-circuits (couplage) de type ET ou OU entre un nombre quelconque de bits.

III.1.2) Les opérateurs.

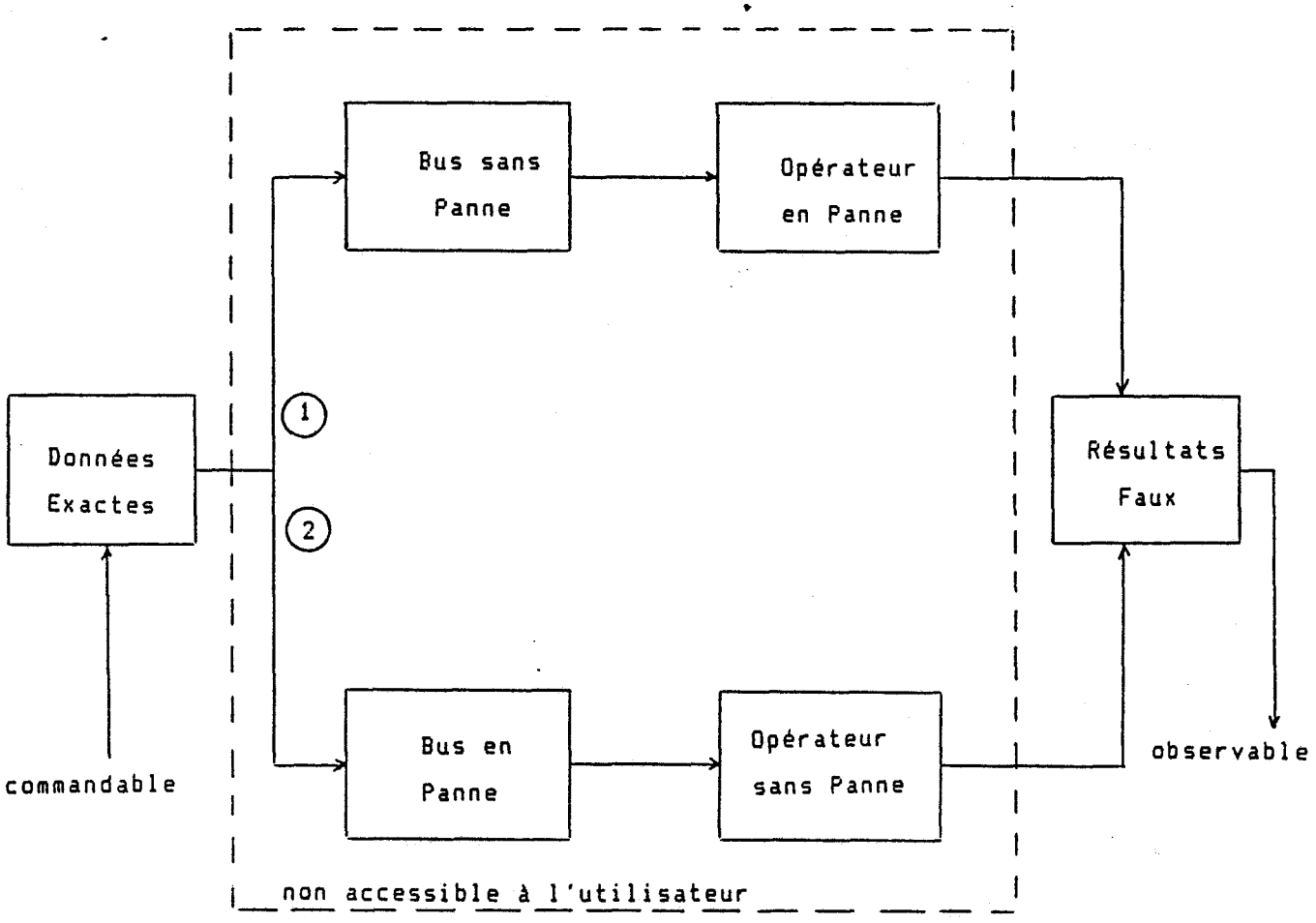
On distingue deux types de pannes dans les opérateurs:

III.1.2.a) les pannes externes

Elles agissent sur les bus situés entre l'opérateur et ses registres opérandes ou résultats. Les pannes externes sont du type collage ou couplage, simples ou multiples, sur chacun de ces bus.

Il suffit donc, pour tester le bus entre l'opérateur et le registre résultat, de vérifier qu'il n'y a pas de collage ou de couplage des bits du résultat.

Un fonctionnement défectueux du bus reliant un registre opérande à l'opérateur amène des données fausses à l'entrée de celui-ci. Ceci est équivalent, vis à vis du test, à la présence de données exactes à l'entrée d'un opérateur défectueux.



Le test sera complet à condition d'obtenir un résultat faux que les données transitent par le chemin 1 ou le chemin 2 .

Le but du test est de détecter les défaillances du microprocesseur et non pas de les localiser; les pannes affectant le bus d'entrée sont donc détectées lors des tests détectant les pannes internes à l'opérateur.

III.1.2.b) les pannes internes.

Ce sont des pannes qui influent sur l'interdépendance des bits du résultat de l'opération. Par exemple, pour un additionneur, il faut vérifier qu'il n'y a pas de collage ou de couplage affectant les retenues.

Il faut, de plus, vérifier la fonction réalisée par l'opérateur.

III.1.3) Les bus.

Comme pour les pannes externes des opérateurs, les pannes sur les bus peuvent être ramenées à des pannes sur les registres ou les opérateurs.

Les bus ne sont donc pas testés directement puisqu'un couplage ou collage, simple ou multiple est détecté lors des tests des registres ou des opérateurs.

Rappelons que le but est de détecter les pannes et non pas de les localiser; il n'y a donc pas de tests spécifiques des bus puisque ces pannes sont détectées par ailleurs.

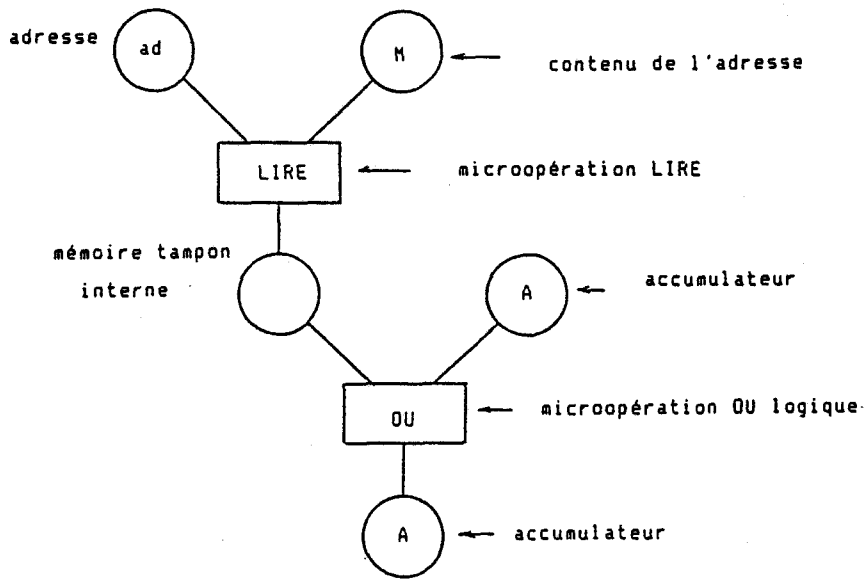
III.2) Modélisation du fonctionnement du microproceseur.

III.2.1) Les graphes d'exécution abstraite (REF.ROB).

Les fonctions et les unités fonctionnelles activées par chaque instruction peuvent être représentées par des graphes d'exécution abstraite.

Le lecteur trouvera en annexe II la description des méthodes d'obtention et d'analyse de ces graphes ainsi que la représentation graphique des instructions utilisées dans les programmes d'application.

Le graphe suivant décrit l'instruction "ORL A, adresse" qui réalise un OU logique entre l'accumulateur et le contenu d'une adresse, le résultat étant disponible dans l'accumulateur



III.2.2) Analyse des graphes.

La liste des instructions du microprocesseur utilisé dans l'application (le 8031 de Intel) est donnée en annexe I.

Une étude effectuée sur ce microprocesseur a donné les résultats suivants:

L'ensemble D des instructions structurellement dominantes est constitué de :

* CJNE @Ri,# data, rel
  comparaison et saut si inégalité
* DJNZ direct, rel
  décrémentation et saut si non nul
* ADDC A, @Ri
  addition avec retenue
* SUBB A, @Ri
  soustraction

} ——— { STRUCTURELLEMENT EQUIVALENTES

* MOV @Ri, ad
  transfert de mémoire à mémoire

A cet ensemble, il faut ajouter un certain nombre d'instructions, constituant l'ensemble I, afin d'exciter chacune des micro-opérations

* CPL complémentation
* RR rotation à droite
* RL rotation à gauche
* RRC rotation à droite à travers la retenue
* RLC rotation à gauche à travers la retenue
* MUL multiplication
* DIV division
* ANL ET logique
* ORL OU logique
* DA ajustement décimal
* XRL OU exclusif
* XCH échange des contenus de deux registres
* CLR remise à zéro d'un registre
* SWAP échange des demi octets d'un registre.

III.2.3) Restrictions nécessitées par l'application.

Le but principal de l'introduction de programmes de test dans le programme d'application est d'éliminer les mauvais fonctionnements possibles dûs à une défaillance du microproceseur.

Nous n'envisageons pas un test exhaustif du microprocesseur mais uniquement un test des parties du circuit utiles au programme d'application (et au programme de test). Il n'y a donc que les instructions utilisées qui sont testées.

Un test complet du microprocesseur nécessite le test des instructions de l'ensemble D u I (définis précédemment). Compte tenu de la remarque qui vient d'être faite, les instructions non utilisées sont éliminées de cet ensemble.

Pour traiter les graphes de Pétri avec un 8031 selon la méthode développée dans REF.ELK il ne nous faut tester que:

- les cinq instructions structurellement dominantes
- les instructions CPL, RR, RLC, ANL, ORL, XRL, XCH ET CLR de l'ensemble I.

A ceci, il faut ajouter le test des unités fonctionnelles propres au 8031: la RAM, l'EPROM, le contrôleur d'interruptions et les timers.

- les tests de la RAM et de l'EPROM sont envisagés

- le contrôleur d'interruptions n'est pas utilisé; si, lors d'un mauvais fonctionnement, une interruption est autorisée, le signal équitemps est altéré, il n'est donc pas nécessaire de tester ce contrôleur.

- un des timers est inutilisé, l'autre sert de générateur de bauds pour l'UART pour la télésurveillance du fonctionnement du processus. Le contrôle du bon fonctionnement de ces unités est effectué par ailleurs (mise en sécurité des transmissions REF.OUA, REF MAG-C).

Dans les paragraphes suivants, nous présentons les tests des différentes unités du microprocesseur ainsi que leur mise en oeuvre dans l'application étudiée.

III.3) Le test des instructions arithmétiques et logiques.

Un test exhaustif des instructions arithmétiques et logiques requiert le test de chaque fonction pour toutes les combinaisons des 256 valeurs possibles de chaque opérande (soient 65536 combinaisons de deux opérandes de 8 bits).

Un test exhaustif n'est donc pas envisageable, il faut alors trouver un algorithme de recherche des vecteurs de test qui permette l'investigation la plus complète possible de chaque fonction par un nombre raisonnable de vecteurs.

III.3.1) Décomposition des fonctions arithmétiques et logiques.

(REF.VEL)

Soit f une fonction arithmétique ou logique de deux opérandes X et Y qui donne un résultat Z:

$$Z = f\ (X,Y)$$

Par une décomposition booléenne des opérandes et du résultat, on obtient:

$$(z_0\ z_1\ z_2\ \text{------}\ z_p\ ) = f(x_0\ x_1\ x_2\ \text{------}\ x_n\ ,\ y_0\ y_1\ y_2\ \text{------}\ y_n\ )$$

avec $p \geqslant n$

Toute fonction générale peut être décomposée en un système de p fonctions simples:

$$z_0 = f_0\ (x_0\ \text{------}\ x_n\ ,\ y_0\ \text{------}\ y_n\ )$$

$$z_1 = f_1\ (x_0\ \text{------}\ x_n\ ,\ y_0\ \text{------}\ y_n\ )$$

$$\begin{array}{ccccc} . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \end{array}$$

$$z_p = f_p\ (x_0\ \text{------}\ x_n\ ,\ y_0\ \text{------}\ y_n\ )$$

On cherche à écrire les fonctions f_i de manière à mettre en évidence la dépendance entre la ième composante du résultat et les ièmes composantes des opérandes.

$$z_i = f_i\ (X,Y) = \overline{x}_i\,\overline{y}_i\,C_i^0 + \overline{x}_i\,y_i\,C_i^1 + x_i\,\overline{y}_i\,C_i^2 + x_i\,y_i\,C_i^3$$

Les fonctions C_i^j sont appelées fonctions contextes, ce sont des fonctions booléennes des variables x_k et y_k avec $k \neq i$.

Pour les fonctions logiques, les fonctions contextes sont des fonctions constantes.

ex: Pour la fonction OU exclusif :

$$z_i = f_i\ (X,Y) = \overline{x}_i\,\overline{y}_i\,0 + \overline{x}_i\,y_i\,1 + x_i\,\overline{y}_i\,1 + x_i\,y_i\,0$$

Pour les fonctions arithmétiques, les fonctions contextes correspondent à la fonction "génération de retenues".

ex: Pour l'addition :

$$z_i = f_i\ (X,Y) = \overline{x}_i\,\overline{y}_i\,C_i^0 + \overline{x}_i\,y_i\,C_i^1 + x_i\,\overline{y}_i\,C_i^2 + x_i\,y_i\,C_i^3$$

avec $\quad C_i^0 = C_i^3 = C_i = (x_{i-1}).(y_{i-1}) + C_{i-1}(x_{i-1}+y_{i-1})\ ;\ C_i^1 = C_i^2 = C_i$

Le choix des opérandes de test des fonctions arithmétiques et logiques se fait donc de cette manière:

- Toutes les combinaisons des ièmes bits des opérandes
- Si les fonctions contextes ne sont pas constantes, elles prennent successivement la valeur 1 et la valeur 0.

Le test consiste alors à regarder la valeur du ième bit du résultat (et du (i+1)ème bit si on fait une opération arithmétique) pour chacune des combinaisons des ièmes bits des opérandes (les (i-1)èmes bits des opérandes servent à affecter une valeur à la fonction contexte pour les opérations arithmétiques), la valeur des autres bits du résultat et des opérandes étant indifférente.

III.3.2) Le test des couplages.

Les vecteurs de test déterminés précédemment permettent de vérifier le fonctionnement de l'instruction au niveau de la fonction réalisée mais, ne testent pas l'absence de collage ou de couplage des bits du résultat.

Les couplages sont de quatre types:

a) le couplage C_1 : le passage à 1 d'un bit du résultat entraine le passage à 1 d'un autre bit

b) le couplage C_2 : le passage à 0 d'un bit du résultat entraine le passage à 0 d'un autre bit

c) le couplage C_3 : le passage à 1 d'un bit du résultat entraine le passage à 0 d'un autre bit

d) le couplage C_4 : le passage à 0 d'un bit du résultat entraine le passage à 1 d'un autre bit.

Pour détecter ces couplages, il suffit d'avoir un résultat du type:

a) 00000100; Seul le ième bit du résultat est à 1, tous les autres sont à 0. On vérifie ainsi que le passage à 1 du ième bit n'entraine pas le passage à 1 d'un autre bit (couplage C_1). Ce type de résultat doit être obtenu au moins une fois pour chaque i, i variant de 0 à 7.

b) 11110111; Seul le ième bit du résultat est à 0, tous les autres sont à 1. Un raisonnement analogue au précédent est tenu mais dans le cas du couplage C_2.

c) 11111111; Tous les bits sont à 1. Ceci permet de détecter les couplages du type C_3 entre un ou plusieurs bits du résultat.

d) 00000000; Tous les bits sont à 0. Ceci permet de détecter les couplages du type C_4 entre un ou plusieurs bits du résultat.

Pour révéler tous ces couplages sans augmenter le nombre de vecteurs de test, il suffit de donner une valeur précise aux bits dont la valeur était indifférente au paragraphe III.3.1 de manière à obtenir les résultats des quatre types qui viennent d'être présentés.

Les vecteurs de test ainsi déterminés permettent aussi de détecter les collages de un ou plusieurs bits du résultat.

III.4) Les autres instructions.

Le raisonnement qui vient d'être tenu est étendu aux instructions "non arithmétiques ou logiques" (CJNE, RR,...).

Ces instructions ne peuvent pas être décomposées mathématiquement de façon rigoureuse comme les instructions arithmétiques et logiques; néanmoins une méthode analogue est utilisée pour déterminer les vecteurs de test:

- ils doivent permettre de vérifier le fonctionnement de l'instruction pour toutes les combinaisons de valeurs des bits de rang i des opérandes et ce pour $0 \leqslant i \leqslant 7$.

- de plus, les valeurs des bits de rang j ($j \neq i$) sont choisies de manière à tester tous les couplages des bits du résultat.

III.5) Le test de la RAM.

III.5.1) Le test des modes d'adressage.

Les défaillances survenant lors de l'adressage d'un registre Ri peuvent avoir les conséquences suivantes:

- aucun registre n'est adressé
- un ou plusieurs registres Rj sont adressés à la place du registre Ri
- un ou plusieurs registres Rj sont adressés en plus du registre Ri.

Ces défauts ne peuvent être détectés que s'ils sont la conséquence d'une défaillance d'un mécanisme d'adressage (par comparaison de l'adressage d'un même registre par deux méthodes différentes: directe et indirecte). Par contre si c'est le registre d'adressage de la RAM qui est défaillant, l'erreur ne peut être détectée par comparaison des deux modes d'adressages (pannes de mode commun).

La vérification des mécanismes d'adressage (direct et indirect) peut se décomposer en trois parties:

- vérification de l'écriture de toutes les données
- vérification de la lecture de toutes les données
- vérification de l'adressage de tous les registres

Le test de l'adressage bit à bit d'un registre n'est pas envisagé puisque ce mode d'adressage n'est pas utilisé.

III.5.1.a) Vérification de lecture et écriture.

Pour un registre déterminé, il faut vérifier que toutes les données possibles (256 valeurs pour des registres de 8 bits) écrites de façon directe (resp. indirecte) peuvent être lues de façon indirecte (resp. directe).

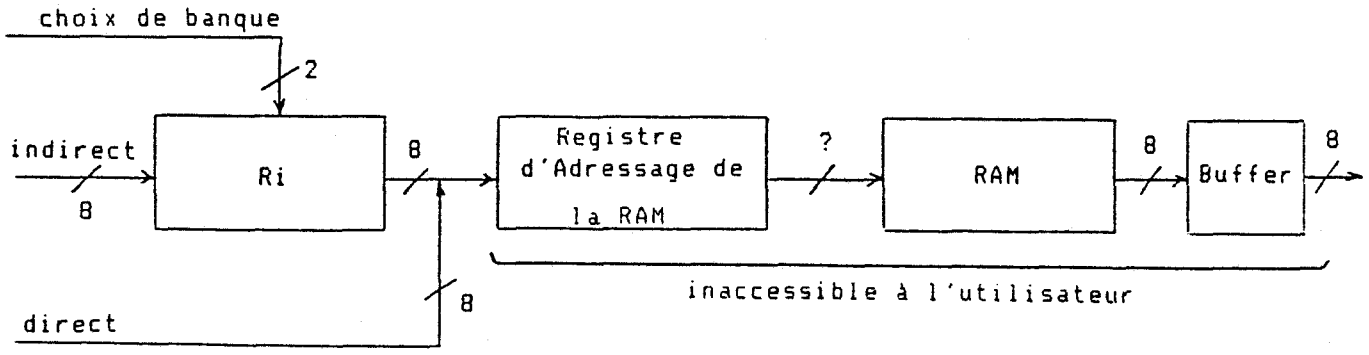
L'utilisation des deux modes d'adressage permet de s'affranchir des pannes d'un mécanisme d'adressage si l'autre est sans défaut (ou s'il a un défaut dont les conséquences n'entrainent pas la compensation des erreurs).

III.5.1.b) Vérification de l'adressage des registres.

Il faut vérifier que tous les registres de la RAM peuvent être adressés. C'est essentiellement l'adressage indirect qui est testé puisque c'est le mode d'adressage qui met en oeuvre le plus grand nombre d'éléments du microprocesseur.

Le registre adressé indirectement est comparé au registre adressé directement (en l'absence de panne, c'est le même). Le test exhaustif de l'adressage de tous les registres de la RAM n'est pas utile. En effet, lors de la comparaison des deux mécanismes d'adressage, il apparait qu'il suffit de vérifier l'absence de collage ou de couplage sur les 8 bits d'adresse, le reste étant commun aux divers modes d'adressage (c'est la panne de mode commun évoquée précédemment).

MECANISME D'ADRESSAGE



La vérification de l'adressage par la comparaison des deux modes d'adressage direct et indirect se limite donc aux cas suivants:

adresses testées		
00 H	———	couplage 0 ---> 1 (couplage C_1)
01 H 02 H 04 H 08 H 10 H 20 H 40 H	———	couplage 1 ---> 1 (couplage C_2)
7F H	———	couplage 1 ---> 0 (couplage C_3)
7E H 7D H 7B H 77 H 6F H 5F H 3F H	———	couplage 0 ---> 0 (couplage C_4)

La définition des différents couplages ainsi que la détermination des octets correspondants sont présentées au paragraphe III (test des instructions arithmétiques et logiques).

La méthode est identique mais ici le bit de rang 7 n'est pas pris en compte car il est inaccessible en adressage indirect (seule la zone RAM d'adresse 00 H à 7F H est accessible par adressage indirect sur le 8031).

III.5.1.c) Adressage indirect par d'autres registres.

Il faut nécessairement tester l'adressage indirect par tous les registres possibles. Dans le cas du 8031, il n'y a que deux registres (R_0 et R_1) qui permettent l'adressage indirect.

Le test se limite aux zones RAM effectivement adressées par ces registres dans le programme d'application. Il s'effectue par comparaison des registres indirectement adressés par les deux registres R_0 et R_1 autorisés (en l'absence d'erreur, ce sont les mêmes).

III.5.2) Test du contenu de la RAM.

Le test du contenu de la RAM est effectué de façon exhaustive. Il est constitué de la vérification suivante: toutes les données (256 valeurs pour des registres de 8 bits) peuvent être écrites ou lues et ce, pour chaque registre de la RAM.

Il est évident que dans le cas présent, il était posssible de limiter le nombre de vecteurs de test au nombre nécessaire à la détection des collages et des couplages des zones élémentaires de 1 bit constituant chaque registre. Compte tenu de l'algorithme de test utilisé (décrémentation et comparaison avec émission du signal équitemps), le test exhaustif est beaucoup moins couteux en octets et à peine plus couteux en temps que le test avec un nombre restreint de vecteur; le test exhaustif a donc été retenu.

III.5.3) Le test de la diaphonie.

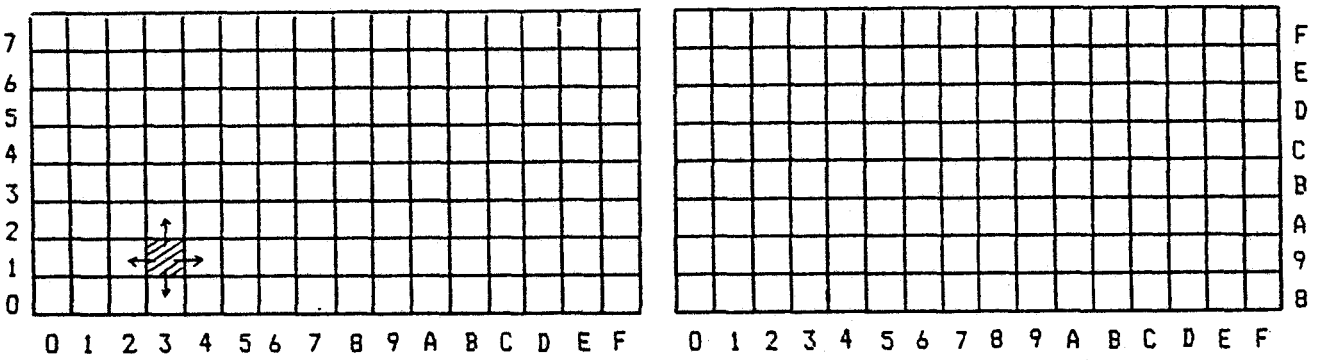
Ce test permet de vérifier l'absence de couplage entre deux bits de deux registres voisins. Le problème consiste à connaitre l'implantation des cases mémoires dans la RAM afin de déterminer quel est le registre voisin d'un autre registre donné.

La RAM du 8031 est divisée en deux zones de 128 octets:

une première zone, dont les adresses sont comprises entre 00 H et 7F H, entièrement accessible à l'utilisateur et ,

une deuxième zone nommée RAM-SFR (Random Access Memory for Special Function Registers) dont une partie seulement est accessible à l'utilisateur: c'est dans cette zone que se situent les registres spéciaux (accumulateur, registre B, le pointeur de données, le pointeur de pile,...); ceux-ci sont testés individuellement.

L'implantation des cases mémoires de la RAM peut être déduite des informations contenues dans REF.INT, elle pourrait donc être représentée par le schéma ci-dessous:



Sur cette figure, les flèches indiquent les différents couplages à tester pour la case d'adresse 13 H :

13 H <-----> 23 H

13 H <-----> 03 H

13 H <-----> 12 H

13 H <-----> 14 H

Il faut alors tester les couplages entre les registres:

- d'adresse X et X ± 10 H
- d'adresse X et X ± 01 H.

En fait, seule la moitié de ces couplages est testée puisque le couplage entre les registres d'adresses X et X+10 H est lè même que celui entre les registres d'adresses (X+10 H) et (X+10 H)-10 H.

III.5.4) Difficultés rencontrées.

- Le mécanisme d'adressage indirect utilise la pile du microprocesseur lors de son fonctionnement et le pointeur de pile, utile au programme d'application, peut contenir une adresse quelconque de la RAM. Il a donc fallu résoudre le problème de la sauvegarde de ce pointeur.

- Le test de la RAM s'effectue "en ligne", les registres contiennent donc des données nécessaires au bon fonctionnement du programme. Il est, de ce fait, indispensable de sauvegarder ces données avant de tester les registres qui les contiennent et de les réécrire correctement dans le registre d'origine avant leur utilisation par le programme principal.

III.6) Le test de la ROM.

La ROM testée est une 27128 (16 k Octets).

Le test consiste en 16 "cheksums". Chaque cheksum est la somme arithmétique de tous les octets contenus dans 1 k Octets de mémoire, les valeurs justes des résultats sont mises sous la forme de 48 octets (3 octets par résultat) contenus dans un même k Octets de mémoire.

Ces tests permettent de détecter les changements d'un octet de la ROM, les problèmes de lecture, d'adressage,...et toutes les erreurs multiples qui ne se recombinent pas, et ceci par k Octets de programme.

Le test étant divisé en 16 cheksums distinctes, la probabilité de compensation d'erreurs est plus faible que s'il était constitué d'une seule cheksum sur 16 k Octets.

Ces cheksums sont exécutées périodiquement à la même fréquence que les autres tests fonctionnels.

III.7) Organisation du programme de test.

III.7.1) Durée des modules de test.

Le programme de test est découpé en modules de 1ms maximum pour ne pas géner le fonctionnement du programme principal.

Le test d'une instruction ou d'une zone mémoire est effectué en un module et éventuellement en plusieurs modules partiels si la durée de ce test excède 1 ms.

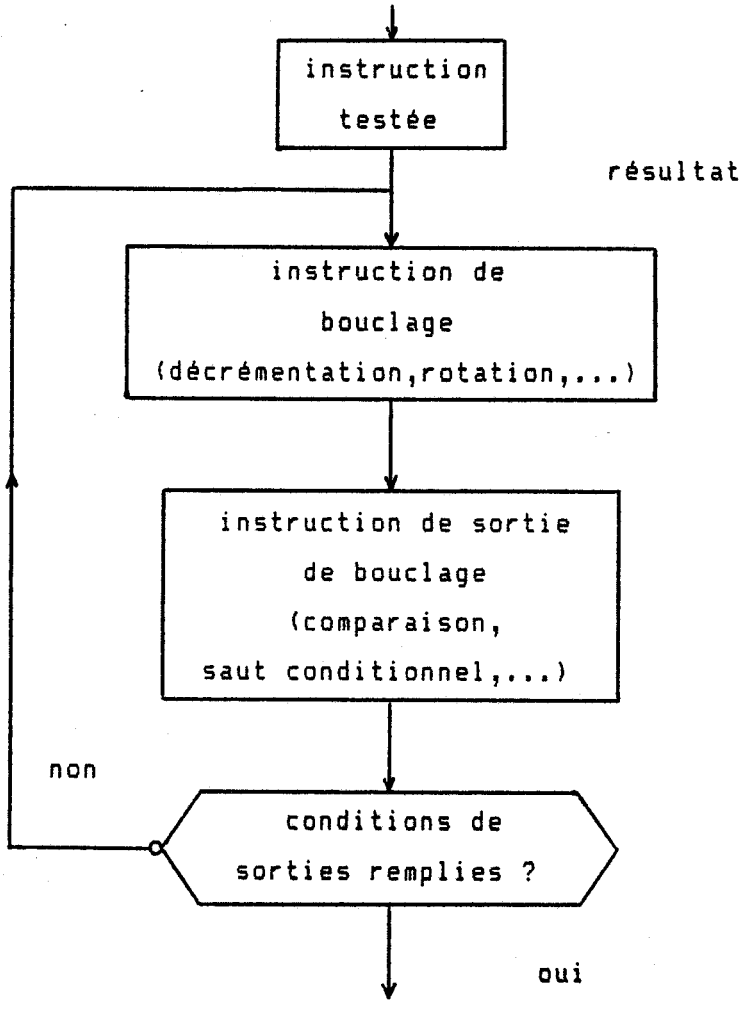
III.7.2) Le contrôle des résultats.

> Il ne faut en aucun cas que ce soit le microprocesseur qui se déclare exempt de défaillance.
> C'est le signal équitemps qui doit refléter le fonctionnement du microprocesseur.

Le programme de test est donc organisé de manière telle que les résultats conditionnent les temps d'exécution des différentes routines; ceci est obtenu au moyen d'instructions de bouclage :

le nombre de boucles exécutées reflète la validité du résultat qui, de ce fait, n'est pas basée sur une seule instruction pouvant elle-même, s'avérer défaillante et donc déclarer juste un résultat faux.

Ce principe est illustré par le schéma suivant:



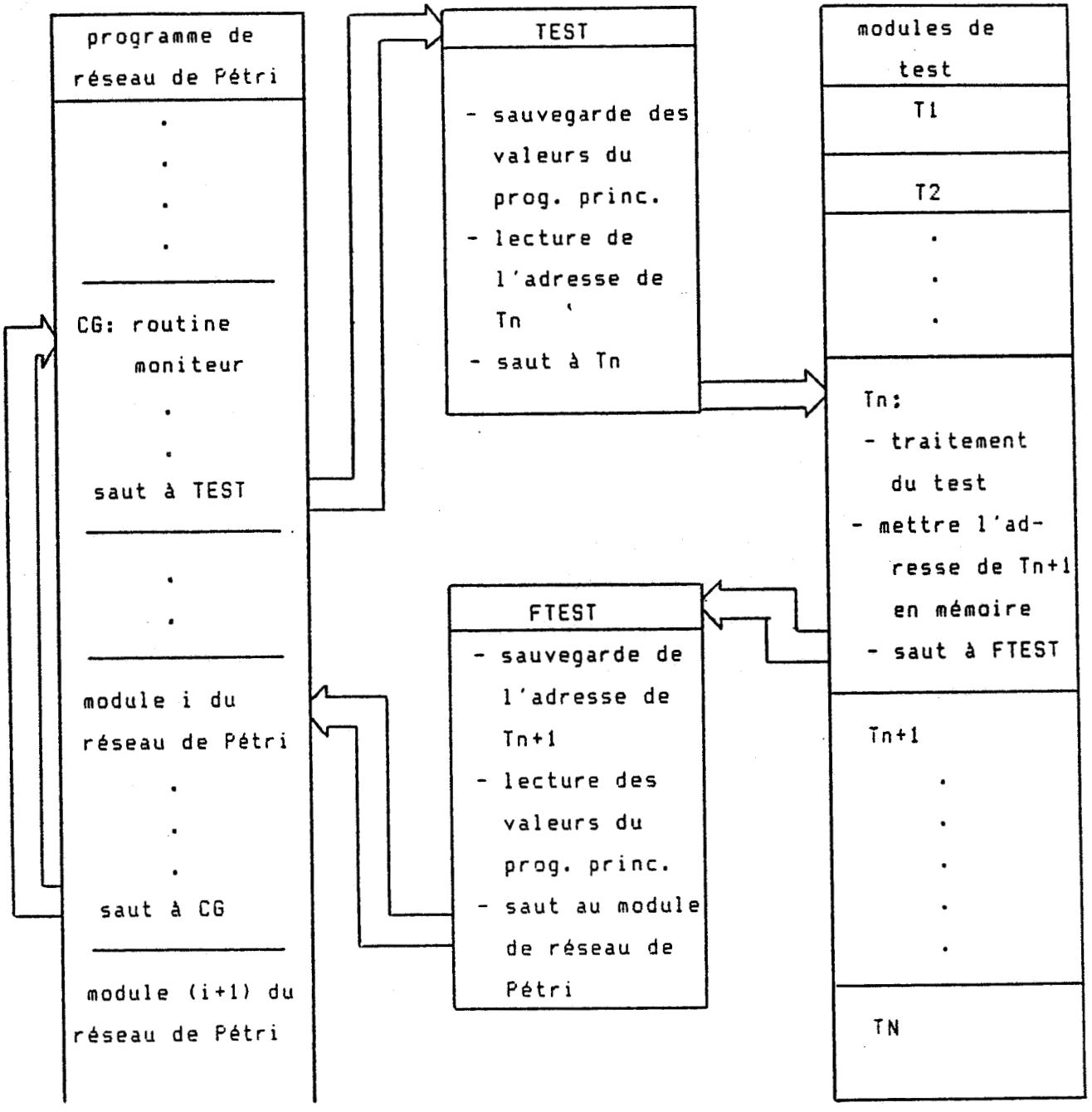
Le nombre de boucles conditionne l'instant d'apparition du front du signal équitemps émis, s'il ne concorde pas au signal de fenêtre de la structure de contrôle, l'écart temporel est détecté et elle inhibe les commandes envoyées par le microprocesseur (Cf chap.II).

III.7.3) Le programme.

Le programme de test dont la conception vient d'être décrite occupe un espace mémoire de 6 k Octets et le test complet du microprocesseur est exécuté en 650 ms.

Le programme est cyclique: lorsque le test est terminé, le nouveau module exécuté est le premier.

ORGANISATION GENERALE DES PROGRAMMES D'APPLICATION ET DE TEST



prog.princ correspond au programme principal de réseau de Pétri

Ti : ième module de test

N : nombre total de modules de test.

III.8) Conclusion.

Un test complet et exhaustif d'un microprocesseur nécessite l'application d'un nombre considérable de vecteurs de test. La méthode que nous venons d'exposer permet de réduire très fortement ce nombre tout en offrant la possibilité de faire un test complet.

Elle possède en outre l'avantage d'être applicable à d'autres microprocesseurs (moyennant une étude de ceux-ci) et notamment aux circuits 16 bits , 32 bits,... et ceci sans que le nombre de vecteurs de test ne devienne prohibitif.

Prenons un exemple:

Le test complet d'une opération à deux opérandes de n bits nécessite 2^{2n} vecteurs pour un test exhaustif et 10n+2 vecteurs par la méthode exposée; de même pour le test d'un mécanisme d'adressage sur n bits : 2n+2 vecteurs au lieu de 2^n pour un test exhaustif, etc....

Cette méthode permet donc de faire le test de circuits travaillant sur des champs de données de n bits de plus en plus grands (microprocesseurs 16, 32 bits,....) sans se heurter au problème de "l'explosion combinatoire" du nombre de vecteurs de test.

2ème PARTIE

EVALUATION DU TAUX DE COUVERTURE DE PANNES D'UN MICROPROCESSEUR

INTRODUCTION

La grande difficulté de la mise en sécurité des microprocesseurs est l'évaluation de cette sécurité.

En effet, les structures internes de ces composants sont généralement inconnues de l'utilisateur et les méthodes classiques d'évaluation de la sécurité ne sont plus applicables.

Nous avons essayé de donner un ordre de grandeur du taux de couverture de pannes du système de sécurité étudié. La démarche a été la suivante :

1) Evaluation grossière et pessimiste de la sécurité par analyse du schéma topographique du circuit.

2) Evaluation par la mesure.

I) METHODES DE DETERMINATION DU TAUX DE COUVERTURE DE PANNES.

La définition du taux de couverture de pannes (τ) que nous adoptons dans ce mémoire est la suivante:

Le taux de couverture de pannes est égal à la probabilité de détecter une panne lorsque celle-ci a eu lieu.

Si l'hypothèse d'équiprobabilité des pannes peut être émise, le taux de couverture de pannes se détermine simplement par le rapport du nombre de pannes détectées sur le nombre de pannes possibles.

$$\tau = \frac{\text{nombre de pannes détectées}}{\text{nombre de pannes possibles}}$$

Selon le niveau de connaissance que l'on a de la structure interne du microprocesseur, la détermination du taux de couverture de pannes est plus ou moins précise.

La connaissance fine de la structure interne du microprocesseur au niveau des transistors ou même des portes logiques est souvent inaccessible à l'utilisateur qui ne dispose en général que de manuels d'utilisation, et de plus elle est difficilement exploitable.

En effet, de part la combinatoire (nombre de pannes possibles par élément et nombre d'éléments), on conçoit mal de faire le bilan de toutes les pannes possibles à ce niveau puis d'en déduire les influences sur les phénomènes observables de l'extérieur du microprocesseur afin de vérifier que toutes ces pannes sont détectables.

La couverture absolue de toutes les pannes est impossible; une approche microscopique n'est envisageable que lors de la conception du microprocesseur si l'on implante des contrôleurs dans le silicium (REF.BIE).

Notre approche sera donc relative, notre objectif étant de chiffrer une valeur minimale de la proportion de défaillances détectées. Les défaillances sont observées d'un point de vue logique ou fonctionnel.

Une solution qui est proche de la structure réelle du microprocesseur tout en restant exploitable est basée sur la connaissance du "schéma-bloc" de celui-ci c'est à dire, un schéma représentant tous les blocs fonctionnels ainsi que les liaisons entre eux (bus).

Ce schéma doit être suffisamment précis: chaque bloc fonctionnel doit pouvoir être considéré comme élémentaire. Ceci suppose une collaboration étroite avec le constructeur qui fournit malheureusement rarement ce genre de document néanmoins, quelques explications, notamment sur le fonctionnement de la partie contrôle permettraient d'améliorer considérablement la connaissance du circuit.

En l'absence de ces informations, il faut, pour résoudre le problème admettre l'hypothèse d'équiprobabilité des pannes en regard de la proportion de surface occupée.

Cette hypothèse découle des remarques suivantes:

- les transistors constituant un circuit intégré peuvent être considérés comme uniformément répartis sur la surface de celui-ci; un phénomène extérieur tel qu'une perturbation électromagnétique peut produire un défaut n'importe où sur la puce.
- lors de la réalisation des circuits, des malfaçons peuvent se produire, certains transistors peuvent avoir des faiblesses qui se révèleront de façon aléatoire durant la vie utile du composant. Ceci peut se produire n'importe où sur le wafer (sauf sur les circuits périphériques éliminés) et donc n'importe où sur un microprocesseur de ce wafer.

Un schéma topographique détaillé du circuit intégré s'avère utile pour évaluer, de façon relative, la surface occupée par chaque partie constituant le microprocesseur.

Et enfin, si aucune de ces informations n'est disponible, une solution consiste à faire un bilan des pannes détectées de façon sûre mais le taux de couverture total ne pourra pas en être déduit.

I.1) Le schéma bloc du microprocesseur est connu.

Hypothèses:

- Le constructeur a fourni un schéma bloc fonctionnel détaillé du microprocesseur.

- Le schéma bloc fonctionnel est suffisamment précis pour que chaque unité puisse être considérée comme élémentaire: le microprocesseur se présente sous la forme d'un assemblage d'unités fonctionnelles indépendantes et reliées entre-elles par des bus.

- Le modèle de pannes est localisé au bloc:

  * collage ou couplage des entrées d'un bloc
  * collage ou couplage des sorties d'un bloc
  * la table de vérité du bloc est modifiée (mauvais fonctionnement)
  * collage ou couplage sur un bus de liaison entre blocs.

Le bilan des pannes possibles devient simple à faire et de cardinalité maitrisable. Il faut alors analyser chacune de ces pannes de manière à déterminer si elles sont détectées ou non (analyse de la propagation des erreurs dues aux défauts des blocs jusqu'à une sortie observable et vérification de la détection de chacune de ces erreurs lors des tests).

La liste des pannes possibles et des pannes détectées étant dressée, le taux de couverture de pannes est alors le rapport suivant:

$$\tau = \frac{\text{nombre de pannes détectées}}{\text{nombre de pannes possibles}}$$

si on admet que toutes les pannes sont équiprobables.

Dans le cas contraire, il faut attribuer un coéfficient à chaque panne, celui-ci correspond à sa probabilité d'apparition.

Cette méthode nécessite toutefois de connaitre précisément le schéma bloc du microprocesseur: il doit refléter la réalité interne du circuit et ne pas comporter d'omission (dans un but de protection du secret industriel par exemple).

Lorsque ces informations ne sont pas disponibles, une deuxième méthode peut être appliquée: elle est moins précise mais plus accessible à l'utilisateur.

I.2) La structure interne est inconnue.

Dans ce cas, il faut exploiter les informations contenues dans les manuels d'utilisation éventuellement complétées par des explications de fonctionnement ou des indications de topographie du circuit fournies par le constructeur.

Si cette information est complétée par des précisions sur le fonctionnement de la partie contrôle (les renseignements fournis par les manuels d'utilisation sont très succints dans ce domaine), cela permet de faire des hypothèses de pannes de cette partie basées sur un fonctionnement réel du circuit et non sur un fonctionnement possible.

Dans ce cas, la définition des pannes et le calcul du taux de couverture de pannes présentés au paragraphe précédent ne sont plus applicables.

Une solution possible consiste alors en une décomposition du microprocesseur en blocs généraux et non plus élémentaires (ex: les registres, l'unité arithmétique et logique, la RAM, la ROM,...etc), chaque bloc étant considéré séparément pour définir les hypothèses de pannes. La connaissance du circuit est ici trop faible pour adopter des hypothèses de pannes générales et applicables à l'ensemble des blocs.

Des hypothèses de pannes sont donc faites pour chaque bloc afin de vérifier si les tests fonctionnels couvrent bien l'ensemble des pannes du bloc considéré et en déterminer un taux partiel de couverture de pannes.

Lorsque la topographie du circuit est connue, la proportion de la surface totale occupée par chaque bloc peut alors être déterminée et si les hypothèses suivantes sont vérifiables :

- les évènements du type: pannes du bloc i sont indépendants. L'apparition d'une panne dans un bloc i (i; 0 < i < nombre total de blocs) est un évènement parfaitement aléatoire,

- deux blocs occupant la même surface ont la même probabilité de tomber en panne: équiprobabilité des pannes par rapport à la surface occupée,

le taux de couverture de pannes se définit par:

$$\tau = 1 - P_{NDT}$$

P_{NDT} étant la probabilité de non détection d'une panne du microprocesseur soit:

$$P_{NDT} = \frac{\sum_{i=1}^{N} \alpha_i \, P_{NDi}}{\sum_{i=1}^{N} \alpha_i}$$

avec: α_i, taux d'occupation en surface du bloc i (proportion de surface)

N, nombre total de blocs du microprocesseur

on a: $$\sum_{i=1}^{N} \alpha_i = 1$$

P_{NDi} , probabilité de non détection d'une panne du bloc i

On a donc :

$$\boxed{\tau = 1 - \sum_{i=1}^{N} \alpha_i \, P_{NDi}}$$

RESTRICTIONS:

- Si l'hypothèse d'équiprobabilité des pannes par rapport à la surface occupée n'est pas complètement vérifiée notamment lorsqu'on considère un bloc fonctionnel central et un bloc périphérique au circuit (REF.COU), le taux d'occupation en surface doit être pondéré par un coefficient traduisant la situation du bloc considéré (centrale ou périphérique) ou alors la probabilité P_{NDi} doit être majorée pour les blocs périphériques afin de déterminer le taux de couverture de pannes compte-tenu de la remarque précédente.

Les coéfficients de pondération ou de majoration ne sont pas simples à déterminer; nous avons donc choisi de calculer la proportion de surface occupée en tenant compte de cette remarque. La surface des blocs périphériques est calculée en incluant dans les blocs les plots de sortie et les "zones interplots" faiblement actives.

La proportion de surface occupée par un bloc périphérique est donc surestimée ceci permet alors de se rapprocher de l'hypothèse émise: la probabilité de pannes est plus importante mais la surface est surestimée, le rapport est donc peu différent de celui obtenu pour un bloc central.

- Le chiffre du taux de couverture de pannes obtenu est très grossier mais volontairement pessimiste. La connaissance du circuit étant incomplète, beaucoup de pannes sont déclarées comme non détectées par manque d'informations (un grand nombre de ces pannes est très probablement détecté).

Cette façon de procéder, même si elle ne permet pas de déterminer la valeur exacte du taux de couverture de pannes, en donne toutefois une borne inférieure (C aura probablement une valeur (très) supérieure au chiffre obtenu).

- Quand la topographie exacte du circuit est inconnue, la détermination du taux partiel de couverture de pannes de chaque bloc est possible mais on ne peut en déduire le taux global du microprocesseur.

II) APPLICATION AU SYSTEME ETUDIE.

Ce chapitre est consacré à l'application des méthodes qui viennent d'être exposées au système considéré.

Celui-ci se compose d'un microprocesseur 8031 et d'une mémoire EPROM 27128 (16 k Octets) surveillés par les tests temporel et fonctionnel.

II.1) Connaissance de la structure interne du 8031.

L'application de la première méthode nécessite la connaissance de la structure interne du circuit.

Le dialogue auprès des constructeurs (Intel MHS France) afin d'obtenir ces informations s'étant avéré impossible, cette méthode ne pourra pas être appliquée. De plus, elle possède l'inconvénient d'être très proche de la structure réelle du circuit et donc très dépendante des évolutions technologiques rapides des différentes versions du circuit.

La seconde méthode, quant à elle, permet d'évaluer la sécurité d'un système sans être trop liée à une structure matérielle donnée.

Compte tenu de l'absence d'informations complémentaires, nos connaissances du microprocesseur utilisé se limitent aux données fournies par les manuels d'utilisation. Dans REF.INT se trouve un schéma topographique du 8051 (8031 avec ROM intégrée), ce schéma quoique insuffisamment précis, nous permet de déterminer l'ordre de grandeur du taux de couverture de pannes ($\mathcal{C}$).

II.2) Proportion de surface occupée par chaque bloc fonctionnel.

Le schéma est en fait une microphotographie du circuit sur laquelle les différents blocs fonctionnels sont délimités.

L'imprécision de ce document résulte de la taille des blocs considérés : ils sont parfois très généraux et remplissent plusieurs fonctions (un exemple: les interruptions et le port série sont dans le même bloc fonctionnel).

La proportion de surface occupée par chaque bloc fonctionnel a été déterminée. Les résultats sont donnés dans le tableau suivant ainsi que les moyens de mise en sécurité adoptés pour chacun des blocs (voir dans la 1^{e} partie le détail du procédé de sécurité étudié).

BLOCS FONCTIONNELS CONSIDERES	PROPORTION DE SURFACE (en %)	PROCEDURES DE TEST
port 0	8,32	Test de la ROM
port 1	4,78	Sécurité liaisons, RDP
port 2	4,89	Test de la ROM
port 3	4,58	Sécurité liaisons, RDP
port control	2,84	Sécurité liaisons, test ROM
ROM (4 kO)	15,20	Test de la ROM
RAM	5,84	Test de la RAM
RAM SFR	5,84	Test des SFR, test temporel
RAM control	4,36	Test de la RAM
Interrupt et Port série	7,44	Test temporel et Sécurité des transmissions
PC / DPTR	3,78	Test temporel
Control logic	8,24	Tests fonctionnel, temporel
PLA	10,57	Tests fonctionnel, temporel
Timer	1,97	Sécurité des transmissions
Timing	2,25	Test temporel
Clock	5,80	Test temporel
ALU	3,28	Test ALU

*RDP: sécurité introduite dans le réseau de Pétri de la commande (traitement anti rebond par exemple)

II.3) Détermination des taux partiels de couverture de pannes.

La détermination des taux partiels de couverture de pannes est faite en considérant chaque bloc fonctionnel séparément. Pour cela, il faut émettre des hypothèses de pannes et vérifier que celles-ci sont détectées par une ou plusieurs composantes du procédé de mise en sécurité du microprocesseur (test fonctionnel, test temporel, codage des transmissions,...).

Les paragraphes qui suivent vont donc reprendre le détails de ces déterminations.

II.4) Le bloc ROM.

II.4.1) hypothèses de pannes.

- changement du contenu d'une case mémoire
- mauvais adressage : la case mémoire effectivement adressée n'est pas celle attendue
- problème de lecture: la case adressée est la bonne, son contenu n'a pas été modifié mais la donnée transmise lors de la lecture n'est pas celle attendue.

D'un point de vue macroscopique, toutes ces pannes sont équivalentes à la première puisque la valeur lue n'est pas celle attendue quelle qu'en soit la raison.

II.4.2) Les tests.

Les pannes de la ROM sont détectées lors de son test: la cheksum (Cf 1ère partie).

La vérification du test de la ROM a été effectuée en modifiant tour à tour chacun des octets mémorisés (Cf 2ème partie Chap.III). Il ne subsiste aucun cas non détecté; toutes les pannes simples décrites précédemment sont détectées, nous dirons donc que

$$P_{ND\ ROM} = 1 - \mathcal{C}_{ROM} = 0$$

II.4.3) Remarques.

Les hypothèses de pannes présentées sont des cas de pannes simples, elles sont toutes détectées.

Examinons les cas de pannes multiples. La ROM contient les octets constituant les instructions du programme. Des modifications multiples des octets de la ROM vont donc aussi entrainer des modifications du programme exécuté. Il faudra donc, pour qu'une panne multiple ne soit pas détectée, qu'elle entraine:

- une compensation lors de chacune des 16 sommes arithmétiques soient donc 16 résultats justes

et,

- aucune altération du signal équitemps lorsque les instructions seront exécutées.

Un exemple simple de panne multiple détectée est le suivant:

16 pannes simultanées entrainant un octet faux dans chacun des 16 k Octets.

II.5) Les blocs Port 0 et port 2.

Les port 0 et 2 servent d'intermédiaires entre la ROM extérieure et le microprocesseur (le port 0 pour les octets bas des adresses et les données, le port 2 pour les octets hauts des adresses).

Lors du test de la ROM, les adresses émises et les données lues transitent nécessairement par les ports 0 et 2; leur fonctionnement est donc indirectement vérifié par la checksum ainsi que par d'autres tests puisque les instructions du programme se trouvent dans la ROM.

II.5.1) Hypothèses de pannes.

Ce sont les collages à 0 et 1 de chacun des bits d'un port et les couplages entre eux.

II.5.2) Taux partiel de couverture de pannes.

Une panne simple du port 0 ou du port 2 est équivalente, vue du microprocesseur, à des pannes multiples de la ROM, dispersées sur les 16 k Octets.

La détermination du taux de couverture de pannes est, dans ce cas, difficile à faire puisqu'il dépend du contenu de la ROM.

Exemple: Le couplage de deux bits du port 0 entraine des modifications dans les adresses émises ainsi que dans les données reçues: les données sont lues au mauvais endroit et sont, de plus, modifiées.

La valeur du taux de couverture de pannes ne sera donc pas déterminée dans le cas d'une étude générale.

Néanmoins, lors d'une panne du port 0 ou du port 2, deux cas peuvent se présenter:

- le nombre d'octets modifiés est relativement faible et la détection est faite de la même façon que lors des pannes multiples de la ROM.
- le nombre d'octets modifiés est important et le programme n'est plus structuré, dans ce cas, la probabilité d'altération du signal équitemps est très grande.

On peut donc estimer que la probabilité de non détection d'erreurs du fonctionnement des ports 0 et 2 tend très fortement vers zéro.

$$P_{ND\ P0} \longrightarrow 0 \quad \text{et} \quad P_{ND\ P2} \longrightarrow 0$$

II.6) Le bloc ALU.

II.6.1) Les hypothèses de pannes.

- Une fonction arithmétique ou logique de l'ALU ne fonctionne plus correctement

- Les fonctions arithmétiques et logiques sont sans défaillance mais les opérandes d'entrée ou les résultats observés sont erronés (les opérandes ne sont pas lus au bon endroit, les résultats ne se situent pas dans les bons registres, panne du bus entre les registres internes et l'ALU proprement dite,...etc).

Vu de l'utilisateur, le deuxième groupe de pannes est équivalent au premier puisque les registres internes ne lui sont pas accessibles et le seul constat qu'il peut faire est que le résultat n'est pas juste.

Une décomposition mathématique des fonctions arithmétiques et logiques a permis de faire des hypothèses de pannes au niveau des fonctions booléennes élémentaires constituant chaque fonction :
(voir 1ère partie III)

- mauvais fonctionnement d'une fonction booléenne
- mauvaise transmission de contexte
- couplage ou collage des bits du résultat
- collage ou couplage des bits des opérandes.

II.6.2) Taux partiel de couverture de pannes.

Par un choix particulier d'opérandes, toutes les pannes citées précédemment peuvent être détectées lors des tests. Des essais (voir le chapitre suivant) ont permis de confirmer l'idée que toutes les pannes sont détectées, on en déduit donc que:

$$P_{ND\ ALU} = 1 - \mathcal{Z}_{ALU} = 0$$

II.7) Le bloc RAM.

II.7.1) Hypothèses de pannes.

- collage à 0 ou 1 d'un ou plusieurs bits de la RAM
- diaphonie entre cases booléennes voisines:
  * couplages entre bits voisins d'un même registre
  * couplage entre bits voisins de registres voisins

II.7.2) La RAM proprement dite.

Les tests fonctionnels font les vérifications suivantes:

- absence de collage d'un ou plusieurs bits
- absence de couplage entre bits voisins (d'un même registre ou de registres voisins)

La probabilité de non détection d'une panne de la RAM est donc presque nulle:

$$P_{ND\ RAM} = 1 - Z_{RAM} \neq 0$$

Il est bien évident que cette détection ne concerne que les pannes constatées lors d'une écriture ou d'une lecture dans la RAM; des parasites modifiant de façon fugitive certains bits de la RAM ne peuvent être détectés avec certitude.

II.7.3) La RAM SFR (Special Function Register).

Cette partie de la RAM contient les registres internes utiles au fonctionnement du microprocesseur. Certains sont connus et accessibles à l'utilisateur, d'autres non : leur rôle dans le fonctionnement du microprocesseur n'est pas explicité dans le manuel d'utilisation.

II.7.3.1) Remarques sur les registres connus.

- l'accumulateur et le registre B sont testés
- le PSW (registre d'état) est indirectement testé, son utilisation fréquente lors du fonctionnement normal du microprocesseur entraine la détection de ses défaillances (mauvais choix de banque de registre, conditions mal interprétées, erreurs arithmétiques,...etc)
- le SP (pointeur de pile) et le DPTR (pointeur de données) servent essentiellement à la sauvegarde des adresses, leurs défaillances sont détectées par les tests temporels
- les défaillances des ports "latch" sont détectées au même titre que les défaillances des ports eux mêmes
- les registres utilitaires pour les interruptions et le port série (IE, IP, TMOD, TCON, TH1, TL1, SCON, SBUF, PCON) sont intimement liés au fonctionnement des interruptions et du port série et donc, leurs défaillances sont détectées
- le timer 0 (TH0, TL0) est inutilisé. Ses défaillances sont sans effet (ce cas n'est pas dangereux) ou affectent la liaison série et sont détectées comme une défaillance de celle-ci.

II.7.3.2) Remarques sur les registres inconnus.

Il est fort probable qu'une partie de ces registres sert de registres temporaires (pour l'unité arithmétique et logique par exemple) ou de buffer. Leurs défaillances sont détectées comme celles des blocs fonctionnels (ALU, bus interne,...).

Les défaillances des registres inconnus inutilisés doivent probablement être sans effet.

La méconnaissance de cette partie de RAM ne permet pas d'en déterminer la probabilité de non détection de pannes; on suppose simplement que ces registres sont utiles au fonctionnement du microprocesseur (optimisation de la surface de silicium occupée par composant) et donc que les effets de leurs défaillances se font sentir au niveau du fonctionnement de zones accessibles à l'utilisateur.

Compte tenu du manque d'informations, la probabilité de non détection est grossièrement estimée à:

$$P_{ND\ RAMSFR} = 1 - \zeta_{RAMSFR} = 0,5$$

valeur certainement très supérieure à la réalité

II.8) Le bloc PC/DPTR.

II.8.1) Hypothèses de pannes du PC (Program Counter).

Ne connaissant pas le fonctionnement interne du PC, les hypothèses de pannes se situent au niveau du fonctionnement externe:

- mauvaise incrémentation du PC au cours du déroulement normal du programme (quelle qu'en soit la raison)
- lors de la rupture de séquence (saut), le PC n'est pas chargé par la valeur exacte (quelle qu'en soit la raison)

II.8.2) Détection de pannes.

Le fonctionnement du PC est intimement lié au déroulement du programme, lui même lié à la génération du signal équitemps de test temporel.

Lors de l'exécution du programme, le cheminement, quoique très structuré, n'est absolument pas linéaire et l'émission du signal équitemps est permanente (un décalage de 1 µs correspondant à la durée d'exécution de l'instruction la plus rapide est détecté); ceci nous amène à la constatation suivante:

Un disfonctionnement du PC ne pourra pas constamment donner lieu à une génération correcte de ce signal de test temporel d'où, une très forte présomption de détection.

II.8.3) Hypothèses de pannes du DPTR (Data Pointer Register).

Le DPTR est un registre 16 bits et donc toutes les hypothèses de pannes relatives aux registres sont retenues à savoir:

- collage ou couplage d'un ou plusieurs bits
- panne du (ou des) bus de liaison entre le DPTR et un bloc utilisateur de la donnée contenue dans ce registre.

Pour l'utilisateur, toutes ces pannes sont équivalentes à une donnée fausse dans le DPTR.

II.8.4) Détection de pannes.

Le DPTR sert essentiellement (dans le cas étudié) à la sauvegarde des adresses des routines à exécuter (traitement des places de réseau de Pétri, routines de test fonctionnel). Si l'adresse sauvegardée est inexacte, le signal de test temporel n'est pas émis correctement ou, si l'adresse de la place à traiter est remplacée par une autre, le réseau de Pétri n'évolue pas correctement et la détection est faite au niveau du programme de traitement.

II.8.5) Taux partiel de couverture de pannes.

La probabilité de non détection des pannes de ce bloc PC/DPTR tend donc très fortement vers zéro, le chiffre exact reste cependant impossible à déterminer.

$$P_{ND\ PC/DPTR} = 1 - \zeta_{PC/DPTR} \longrightarrow 0$$

II.9) Les blocs PLA - Control Logic - Timing.

Ces blocs constituent l'essentiel de la partie contrôle du microprocesseur dont le fonctionnement est très peu explicité dans les manuels d'utilisation.

La modélisation de son fonctionnement ainsi que les hypothèses de pannes font référence à REF.BRA et REF.BEL1 et sont plutôt de nature intuitive.

II.9.1) Hypothèses de pannes.

- l'exécution d'une instruction est constituée d'un ensemble de microopérations à activer dans un ordre établi. Suite à une défaillance du circuit de décodage des instructions ou lors de leurs exécutions, des microopérations supplémentaires peuvent être activées et/ou des microopérations qui devaient l'être ne le sont pas.

- le circuit "timing" est en panne et les microopérations, bien qu'en ordre correct, ne sont pas activées aux instants voulus.

- lors de l'exécution d'une instruction conditionnelle, la condition est mal interprétée.

II.9.2) Détection de pannes.

- Toutes les pannes de la partie contrôle qui se traduisent par un décalage temporel sont nécessairement détectées: le signal équitemps n'est plus correctement émis (défaillance du circuit "timing", activation de microopérations supplémentaires, non activation de microopérations nécessaires,...).

- D'autres pannes n'entrainent pas directement de décalages temporels (par exemple une microopération est activée à la place d'une autre) mais leurs influences sont ressenties au niveau du déroulement du programme comme certaines pannes du PC et modifient le signal équitemps (il y a dans ce cas détection par les tests temporels), de plus elles altèrent le fonctionnement des instructions et sont donc détectées par les tests fonctionnels.

- Le mauvais fonctionnement du circuit de calcul des prédicats est détecté lors des tests des instructions conditionnelles.

II.9.3) Taux partiels de couverture de pannes.

Il semble évident que la probabilité de non détection de pannes du circuit "timing", dont les défaillances ont une influence directe sur le temps, tend fortement vers zéro.

$$P_{ND\ Timing} = 1 - \zeta_{Timing} \longrightarrow 0$$

Par contre, une telle constatation est plus difficile à faire pour les blocs PLA et Control Logic. S' il est fort probable que la majeure partie des pannes de ces blocs est détectée, le taux partiel de couverture de pannes ne peut être considéré comme tendant vers l'unité (manque de connaissance de ce bloc). La probabilité de non détection de pannes est prise comme étant inférieure à 0,1

$$P_{ND\ PLA + Control\ Logic} = 1 - \zeta_{PLA + Control\ Logic} < 0,1$$

valeur qui peut être considérée comme une borne supérieure compte tenu des remarques précédentes.

II.10) Le bloc CLOCK.

L'incidence temporelle d'une défaillance du bloc CLOCK semble évidente puisque c'est l'horloge qui régit tous les phénomènes temporels du fonctionnement du microprocesseur.

Il est possible qu'une défaillance qui se limite à une dérive de l'horloge ne soit pas immédiatement détectée: le décalage n'est pas suffisant pour déclarer le microprocesseur comme étant en panne. Cette latence n'est pas dangereuse puisque dans ce cas l'horloge n'influence pas directement les actions réalisées et n'engendre qu'une homothétie temporelle (toutes les durées sont multipliées par un même coefficient) sans modifier le fonctionnement du microprocesseur.

La probabilité de non détection de pannes est donc presque nulle

$$P_{ND\ CLOCK} = 1 - \zeta_{CLOCK} \neq 0$$

II.11) Les blocs Port Control et RAM Control.

N'ayant aucune information sur le fonctionnement exact de ces blocs, il est difficile de leurs attribuer un taux de couverture de pannes. On admettra donc que le bloc RAM Control a le même P_{ND} que la RAM et le bloc Port Control celui des ports.

II.11.1) RAM Control.

La RAM est divisée en deux parties:

- la RAM de $P_{ND\ RAM} = 0$
- la RAM SFR de $P_{ND\ RAMSFR} = 0,5$

La probabilité de non détection de pannes du bloc RAM Control est, d'après la remarque précédente, égal à

$$P_{ND\ RAM\ CONTROL} = \frac{P_{ND\ RAM} + P_{ND\ RAMSFR}}{2} \quad 0,25$$

II.11.2) Port Control.

Le microprocesseur possède quatre ports d'entrées-sorties:

- les ports 0 et 2 servant à la liaison microprocesseur <--> ROM

$$P_{ND\ P0} \longrightarrow 0 \quad P_{ND\ P2} \longrightarrow 0$$

- les ports 1 et 3 servant aux liaisons microprocesseur<-->processus la probabilité de non détection de ces ports est fortement liée à l'application, elle est arbitrairement prise comme étant égale à 1 (valeur très pessimiste puisque c'est le cas ou aucune panne n'est détectée). On obtient donc:

$$P_{ND\ Port\ Control} = \frac{P_{ND\ P0} + P_{ND\ P1} + P_{ND\ P2} + P_{ND\ P3}}{4} = 0,5$$

II.12) Les blocs Timer, Interrupt and Serial Port.

Ce sont les organes directs de liaison avec le processus. Ils ne peuvent être testés que par des procédures de codage des informations, de commande-contrôle,...etc, procédures qui dépendent du programme d'application.

II.12.1) Les interruptions.

Les interruptions ne sont pas utilisées. Si, suite à une défaillance, une interruption venait à être autorisée, elle entrainerait une rupture du déroulement du programme et serait donc détectée par le test temporel.

$$P_{ND\ \ Interrupt} = 0$$

II.12.2) Le port série.

Le port série est utilisé pour émettre ou recevoir des messages codés. Suite à une défaillance de celui-ci, les mots émis ou reçus ont une probabilité donnée d'être hors-code, cette probabilité dépend du code choisi.

Il est impossible de faire un test en ligne du port série puisqu'il peut recevoir des messages à tout instant; il n'y a donc que le codage des informations et leur traitement (critère de plausibilité du message reçu) qui assurent la mise en sécurité du port série.

La probabilité de non détection d'erreurs du codage est inférieure à 10^{-10} (REF.OUA) dans le cas étudié, la probabilité de non détection de pannes du port série est donc faible.

II.12.3) Les timers.

Il y a deux timers dans le 8031, l'un des deux est inutilisé et l'autre sert de générateur de baud pour l'UART. Les défaillances des timers se font essentiellement sentir au niveau de la liaison série et donc, là aussi, leur mise en sécurité est celle des transmissions (codage, tests de vraisemblance, vitesse de transmission,...).

Il semble donc que la probabilité de non détection de pannes des timers soit elle aussi faible.

II.12.4) Les ports 1 et 3.

Les ports 1 et 3 sont les organes d'entrée et de sortie du système. Ils permettent la liaison entre le microprocesseur et le processus à commander.

Le test en ligne des ports n'est pas possible car les données transitent dans les deux sens (certaines broches sont configurées en entrée d'autres en sortie) de manière asynchrone, bien qu'elles soient traitées de façon synchrone.

La mise en sécurité des ports 1 et 3 est donc fonction de celle du traitement des données qui transitent par eux. La détection des pannes ne peut se faire qu'au niveau du programme d'application (le réseau de Pétri) par des procédés de commande-contrôle et des tests de vraisemblance (vérification de la conséquence de l'envoi d'une commande, plausibilité des données venant du processus).

il est donc très difficile de déterminer un taux partiel de couverture de pannes des ports 1 et 3, il dépend pour beaucoup de l'application.

II.13) Détermination du taux de couverture de pannes.

On s'aperçoit, à la lecture des paragraphes précédents, que la détermination des taux partiels de couverture de pannes n'est pas aisée lorsque l'on ne connait le microprocesseur qu'à travers les informations fournies dans les manuels d'utilisation.

La détermination du taux de couverture de pannes du microprocesseur est donc très grossière et volontairement pessimiste. Le chiffre obtenu est faible mais constitue une borne inférieure du taux réel.

$$\tau_{déterminé} \leqslant \tau_{réel} \leqslant 1$$

Le tableau suivant résume les différents paramètres pris en compte pour la détermination de ce taux.

II.13.1) Remarques.

- Dans le cas étudié, la ROM n'est pas interne et occupe 16 k octets au lieu de 4, sa surface sera donc multipliée par 4.

Cette manière de procéder n'est pas rigoureuse : la proportion de surface occupée réellement par la ROM et la probabilité d'occurrence d'une panne sont sous estimées. Ceci n'est pas génant, car le taux partiel de couverture de pannes de la ROM étant très proche de 1, l'erreur commise sur la valeur du taux général est négligeable.

- Les probabilités de non détection de pannes des ports 1 et 3 dépendent de l'application. Il semble évident que, dans un système de sécurité, les protections prises au niveau du programme d'application soient comparables à celles prises au niveau du microprocesseur: un programme inapte à détecter et traiter les défaillances du processus qu'il commande ou des liaisons processus $\longleftrightarrow$ processeur ne pourra jamais être utilisé pour la commande d'un système en sécurité même s'il est implanté sur un microprocesseur de sécurité.

Il nous a donc semblé qu'une probabilité de non détection de pannes des ports 1 et 3 de 10^{-2} était raisonnable.

- Pour effectuer les calculs, les probabilités de non détection de pannes des blocs serial port et timer, qui dépendent de l'application (codage, transmissions, etc...) ont, elles aussi, été prises comme égale à 10^{-2} .

II.13.2) Résultats.

La valeur du taux de couverture de pannes ainsi déterminée est de 0,95, cette valeur est très inférieure à celle correspondant à un microprocesseur en sécurité. Elle constitue néanmoins une base sur laquelle nous pouvons nous appuyer pour déterminer l'indice de sécurité du système étudié (voir la dernière partie) sachant que les valeurs trouvées sont nettement plus pessimistes que les valeurs réelles.

Il apparait vraiment dommage que les constructeurs n'aient pas voulu nous donner de renseignements, cela nous aurait permis de chiffrer de façon beaucoup plus rigoureuse le taux de couverture de pannes et d'obtenir un chiffre nettement meilleur et plus significatif.

Le chapitre suivant est consacré à une méthode de détermination du taux de couverture de pannes par simulation de fautes du microprocesseur. Nous verrons qu'elle n'est pas exempte de problèmes et que les résultats obtenus ne sont pas nécessairement plus significatifs.

Détermination des taux partiels (Résultats obtenus)

BLOCS FONCTIONNELS CONSIDERES	PROPORTION DE SURFACE α_i en %	P_{NDi}	REMARQUES
port 0	8,32	0	
port 1	4,78	10^{-2}	dépend de l'application
port 2	4,89	0	
port 3	4,58	10^{-2}	dépend de l'application
port control	2,84	0,5	
ROM (16 kO)	15,20 x 4	0	
RAM	5,84	0	
SFR	5,84	0,5	
RAM control	4,36	0,25	
Interrupt et Port série	7,44	10^{-2}	dépend de la P_{ND} du codage
PC / DPTR	3,78	0	
Control logic	8,24	0,1	
PLA	10,57	0,1	
Timer	1,97	10^{-2}	dépend de la P_{ND} du codage
Timing	2,25	0	
Clock	5,80	0	
ALU	3,28	0	
$\sum_{i=1}^{N} \alpha_i$	145,48		
$\sum_{i=1}^{N} \alpha_i P_{NDi}$		7,5	

On obtient donc: $P_{NDT} = \dfrac{\sum_{i=1}^{N} \alpha_i P_{NDi}}{\sum_{i=1}^{N} \alpha_i} = 0,05$

et donc, $\boxed{\zeta = 0,95}$

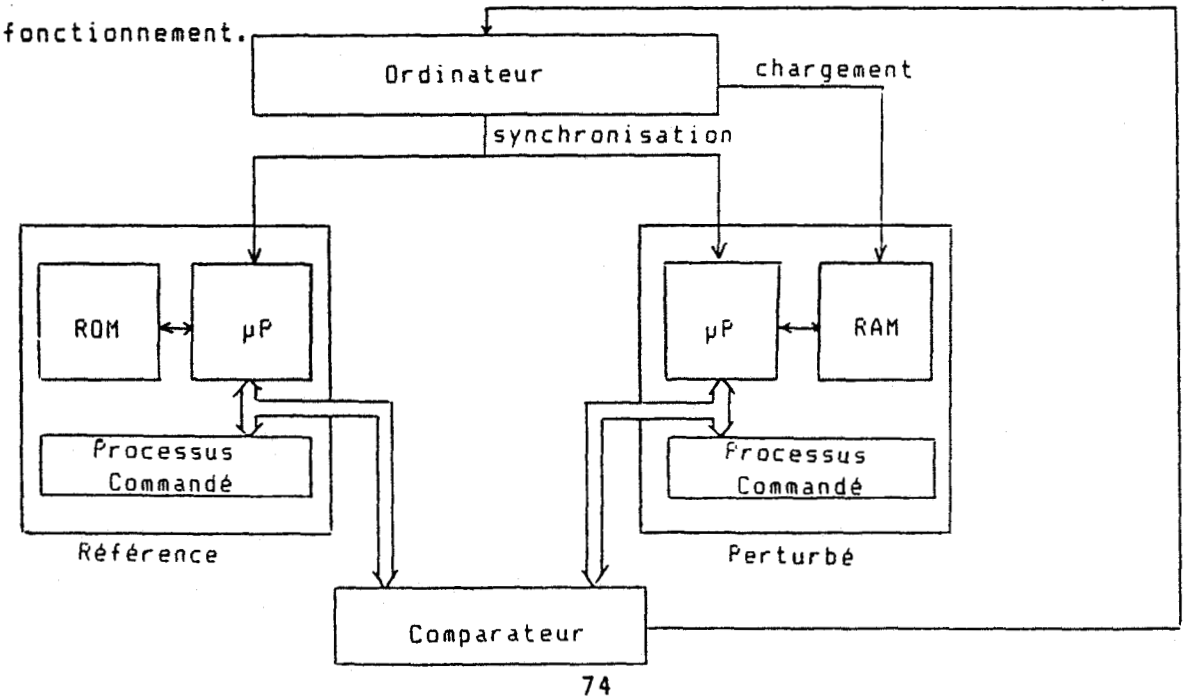
III) LA SIMULATION DES FAUTES.

La simulation de fautes peut sembler un moyen simple de détermination du taux de couverture de pannes, on simule toutes les fautes, on observe si elles sont détectées ou pas et on fait le rapport permettant de déterminer ζ . Il est vrai que, vues sous cet angle, les difficultées que représentent une simulation, la liste de fautes à simuler, les moyens d'observation,...etc n'apparaissent pas.

La liste des fautes à simuler nécessite la connaissance du circuit et le problème du chapitre précédent, à savoir le manque d'informations, demeure; il est moins crucial mais néanmoins présent. De plus, selon le niveau de simulation, un nombre plus ou moins grand de fautes sera simulé.

La simulation complète du fonctionnement d'un microprocesseur n'est pas envisageable si on ne le connait que par les manuels d'utilisation.

Un testeur de logiciel basé sur la comparaison des fonctionnements de deux microprocesseurs identiques dont l'un exécute le programme testé sans modification et l'autre un programme identique à quelques octets près a été mis au point au laboratoire (REF.MAH). Une utilisation particulière de cet appareil nous a permis de simuler des fautes du microprocesseur. La figure suivante illustre son principe de fonctionnement.



III.1) Utilisation de l'appareil.

Ce testeur permet certaines modifications des octets d'un programme, le déroulement de ce programme modifié, ainsi qu'une comparaison des adresses, des commandes envoyées et des signaux équitemps émis par les deux microprocesseurs.

La simulation de fautes dues à des défaillances de ROM, des ports 0 et 2 (adressage de la ROM et données),...etc est donc simple à faire.

Les premiers essais effectués consistaient en la modification d'un seul octet sur tout le programme et la vérification de la détection par les procédures de test fonctionnel de la ROM. Tous les octets ont été modifiés tour à tour et remplacés par les 255 autres valeurs possibles.

Le module de cheksum ayant été vérifié (toutes ces fautes ont été détectées) il n'a plus été introduit dans les programmes lors des essais suivants puisque la simulation des fautes internes est faite au moyen de modifications du contenu de la ROM et donc susceptibles d'être détectées par la cheksum mais pas par le test approprié.

III.2) Simulation de fautes internes.

Le second type de modifications qu'il est possible de faire sur le contenu de la mémoire est la substitution de tous les octets d'une valeur donnée X par une autre Y (plusieurs substitutions peuvent être faites lors d'un même essai).

Nous nous sommes donc servi de ce genre de modifications afin de simuler des fautes du registre d'instruction, de l'unité arithmétique et logique,...etc. Il est bien évident que les instructions modifiées ont été remplacées par d'autres instructions de même longueur (en nombre d'octets) et de même durée d'exécution afin que la détection soit due au programme de test fonctionnel et non pas à une altération naturelle du signal de test temporel.

1ère série d'essais

Les instructions arithmétiques et logiques sont permutées entre-elles. Ceci permet de simuler certaines fautes de l'UAL, notamment celles qui consistent en la modification de la fonction réalisée ainsi que des fautes au niveau du registre résultat (transfert UAL ⟷ registre tampon,
transfert registre tampon ⟷ registre résultat,...).

2ème série d'essais

Les instructions réalisent leur fonction initiale, mais les registres opérandes ou résultat sont modifiés. Ceci permet de simuler des fautes au niveau des bus reliant l'UAL aux différents registres.

3ème série d'essais

Les instructions conditionnelles sont remplacées, dans la mesure du possible, par des instructions dont la condition est : soit opposée, soit toujours vraie, soit toujours fausse. Ceci permet de simuler des fautes au niveau de l'exécution de ces instructions, du circuit de calcul des prédicats,....

4ème série d'essais

Les instructions de transfert sont modifiées de manière à inverser les registres source et destination, à mal interpréter les adresses et les données,.... Les instructions d'empilement et de désempilement (push et pop) sont, elles aussi, modifiées: inactives, inversées,....

Ces essais permettent la simulation de défauts sur les bus, les registres,... ainsi que des erreurs au niveau du fonctionnement des instructions.

5ème série d'essais

Elle concerne les instructions réalisant des fonctions booléennes, les modifications sont analogues à celles des instructions arithmétiques et logiques: permutations des fonctions booléennes entre-elles, modifications des opérandes ou résultats,.... Les fautes simulées se situent au niveau de la fonction réalisée, des bus reliant le processeur booléen et les registres résultats ou opérandes, l'adressage booléen....

6ème série d'essais

Cette série d'essais est moins liée à la simulation de fautes précises que les précédentes. Les modifications sont systématiques: on remplace chaque code opération par un code opération distant de 1 du premier. Ceci permet de simuler certaines erreurs de décodage des instructions.

Remarques:

* Tous ces essais permettent en outre de simuler un certain nombre de défauts au niveau de la mémoire externe et de ses liaisons avec le microprocesseur et des fautes de la partie contrôle : registre d'instruction, PLA,....

* Compte tenu des limites de l'appareil utilisé, seules les fautes qui viennent d'être présentées ont été simulées.

III.3) Résultats.

Tous ces essais ont amené un résultat simple :

Tout est détecté et dans ce cas $\tau = 1$

Il est bien évident que ce résultat nécessite des commentaires.

- Le nombre de fautes simulées est relativement limité compte tenu de la méthode de simulation choisie: le taux de couverture de pannes ne peut pas être raisonnablement considéré comme égal à 1 mais plutôt comme tendant vers 1.

- Cette simulation, quoique imparfaite, montre l'aptitude des tests fonctionnels à détecter les pannes du microprocesseur et renforce l'idée que le taux de couverture de pannes de 95% trouvé précédemment est faible devant le taux réel.

IV) CONCLUSION.

L'évaluation du taux de couverture de pannes d'un système microprogrammé n'est pas facile à faire lorsque l'on ne connait pas en détail la structure du microprocesseur utilisé.

L'utilisation d'un lot de microprocesseurs défectueux afin de déterminer ce taux, procédé couramment utilisé pour la qualification de méthodes de test hors-ligne, n'est pas envisageable pour un test en ligne, la nature des défaillances à détecter n'étant pas la même (pannes à caractère aléatoire au cours de la vie utile du composant et non pannes de jeunesse).

Une méthode statistique de détermination du taux de couverture de pannes à partir d'un très grand nombre d'essais en fonctionnement normal n'est pas envisageable à notre niveau: le nombre d'essais doit être très grand pour être significatif, cela nécessite donc un nombre d'équipement qui dépasse le cadre d'une étude de prototype et de plus le temps nécessaire à ces essais est prohibitif (sans procédure de mise en sécurité, le taux de défaillance horaire du microprocesseur est de l'ordre de 10^{-4}/h soit donc, en moyenne, une panne après plus d'un an de fonctionnement)

Ces deux méthodes classiques n'étant pas utilisables, nous avons essayé d'en déterminer d'autres.

Comme nous l'avons vu dans les chapitres précédents, les méthodes utilisées ne donnent pas le taux de couverture de pannes réel du système. L'une permet la détermination d'une borne inférieure à ζ, l'autre permet de vérifier que la borne déterminée précédemment est nettement inférieure à la réalité et peut donc être utilisée par les calculs de sécurité sachant que les résultats obtenus sont très pessimistes par rapport à la réalité.

Les méthodes présentées, quoique très imparfaites, permettent de faire un pas en avant dans la détermination du taux de couverture de pannes d'un système microprogrammé: les procédés de test décrits dans la littérature sont souvent des tests hors ligne et lorsque ce sont des tests en ligne, leur taux de couverture n'est jamais déterminé.

Il est bien évident, qu'une parfaite collaboration avec les constructeurs est une condition sine qua non pour la détermination de la valeur réelle de C. Le concepteur d'un système basé sur le notre devra donc, au préalable, s'assurer de l'existence de cette collaboration s'il veut mener à bien son projet et obtenir des chiffres significatifs.

La méthode de mise en sécurité de systèmes microprogrammés que nous venons de présenter est transposable sur d'autres systèmes si les conditions rencontrées tout au long des pages précédentes sont respectées, principalement:

- la durée d'exécution de chaque instruction est constante
- le programme est exécuté avec un seul flux d'instruction
- la dynamisation du microprocesseur ne met en oeuvre qu'une seule horloge
- la vitesse d'évolution du processus à commander est faible devant la vitesse d'exécution du programme de commande
- les interruptions sont interdites.

La troisième partie de ce mémoire présente le calcul des paramètres de la sûreté de fonctionnement du système étudié; il est basé sur le taux de couverture de pannes précédemment déterminé.

Nous chiffrons ensuite l'apport de la redondance et nous faisons une comparaison avec un système existant.

3ème PARTIE

CALCUL DES PARAMETRES DE LA SURETE DE FONCTIONNEMENT DE STRUCTURES MONOPROCESSEUR ET BIPROCESSEUR

I) L'UTILISATION DES GRAPHES D'ETATS.

L'ensemble des états d'un système est défini par toutes les combinaisons des états des éléments qui le constituent. L'évolution de ce système peut être décrite par un processus stochastique si l'on connait la probabilité qu'a le système d'être dans un état donné à l'instant t (et ce, pour tous les états possibles).

Un processus stochastique est markovien si son évolution future ne dépend pas de son passé mais dépend uniquement de l'état présent, il est donc parfaitement défini par la donnée des éléments suivants:

distribution initiale et distribution conditionnelle

Les éléments des systèmes étudiés ont des lois de défaillance et de réparation qui peuvent être considérées comme exponentielles, ils peuvent donc être représentés par des processus de Markov (REF.PIL) (REF.CAS) (REF.LIG).

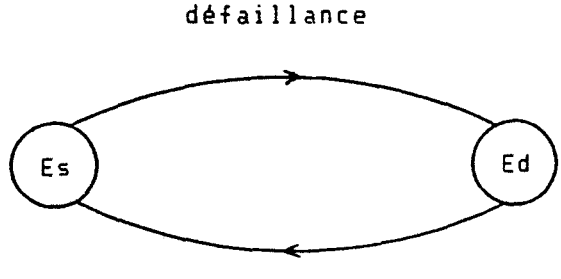
Méthodes de calcul (REF.LAP).

Soit E l'ensemble des états que peut prendre un système.

E peut être partitionné en deux sous-ensembles:

Es : sous-ensemble des états "succès"

Ed : sous-ensemble des états "défaillants"



Soient $P_i(t)$ la probabilité d'être dans un état e_i à l'instant t et λij le taux de transition de l'état e_i vers l'état e_j .

Le calcul de la probabilité d'être dans un état donné est mené en résolvant l'équation de Chapmann-Kolmogorov pour les processus de Markov:

$$\dot{\mathbb{P}}(t) = \mathbb{P}(t)\,\mathbb{A} \qquad (1)$$

avec

$$\mathbb{P}(t) = \left[P_i(t)\right] \quad \text{et} \quad \mathbb{A} = \left[\lambda ij\right] \qquad \begin{matrix} 0<i\leqslant N \\ 0<j\leqslant N \end{matrix}$$

N: nombre d'états possibles.

Cette équation a pour solution:

$$\mathbb{P}(t) = \mathbb{P}(0)\ \exp\ (\mathbb{A}\,t)$$

et en utilisant les transformées de Laplace,

$$\tilde{\mathbb{P}}(s) = \mathbb{P}(0)\ (s\,\mathbb{I} - \mathbb{A})^{-1}$$

avec $\mathbb{I}$ la matrice identité et s la variable de Laplace

La partition effectuée précédemment sur l'ensemble E se retrouve sur la matrice $\mathbb{A}$ appelée matrice de transition.

$$\mathbb{A} = \begin{array}{c} \overbrace{}^{Es} \quad \overbrace{}^{Ed} \\ \left[\begin{array}{c:c} \mathbb{A}_{SS} & \mathbb{A}_{SD} \\ \hdashline \mathbb{A}_{DS} & \mathbb{A}_{DD} \end{array}\right] \end{array} \begin{array}{l} \} Es \\ \} Ed \end{array}$$

La partition de l'ensemble E diffère suivant la grandeur de la sûreté de fonctionnement à déterminer:
par exemple, un état "arrêt" est un état de succès pour la sécurité et un état défaillant pour la fiabilité.

Une méthode de détermination d'une grandeur X(t) de la sûreté de fonctionnement est la suivante:

- les états défaillants sont rendus absorbants, c'est à dire qu'ils ne peuvent plus être quittés une fois atteints soit donc, $\lambda_{ij} = 0$ pour i tel que $e_i \in Ed$.

La matrice $\mathbb{A}$ prend donc la forme suivante:

$$\mathbb{A} = \left[\begin{array}{c|c} \mathbb{A}_{SS} & \mathbb{A}_{SD} \\ \hline 0 & 0 \end{array}\right]$$

- si on partitionne $\mathbb{P}(t)$: $\mathbb{P}(t) = \left[\mathbb{P}_S(t) \quad \mathbb{P}_D(t)\right]$, l'équation (1) devient:

$$\begin{cases} \dot{\mathbb{P}}_S(t) = \mathbb{P}(t) \cdot \mathbb{A}_{SS} \\ \dot{\mathbb{P}}_D(t) = \mathbb{P}(t) \cdot \mathbb{A}_{SD} \end{cases}$$

d'où $\mathbb{P}_S(t) = \mathbb{P}_S(0) \exp(\mathbb{A}_{SS} t)$

or $X(t) = \sum_{i=1}^{\text{card Es}} P_i(t)$

soit $X(t) = \mathbb{P}_S(0) \exp(\mathbb{A}_{SS}.t).\mathbb{1}_S$ avec $\mathbb{1}_S = \left[\begin{array}{c} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{array}\right]$ } Card Es

et en utilisant la transformée de Laplace

$$\boxed{\tilde{X}(s) = \mathbb{P}_S(0).(s\mathbb{I} - \mathbb{A}_{SS})^{-1}.\mathbb{1}_S} \qquad (2)$$

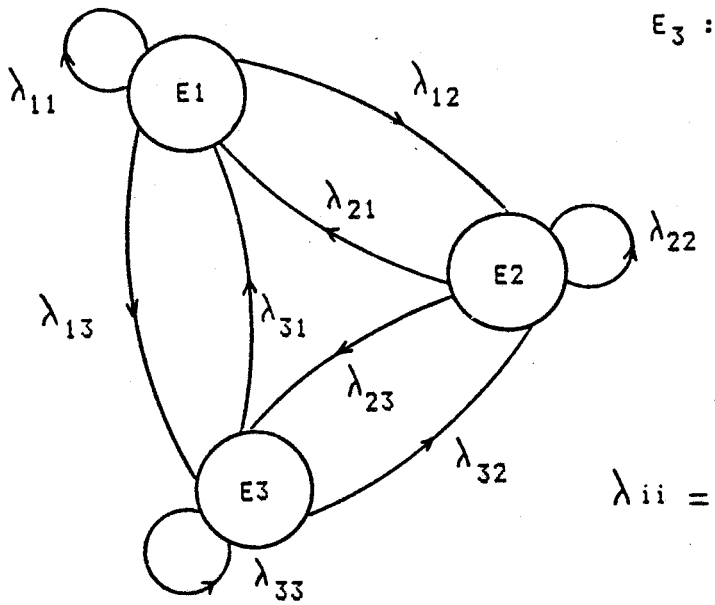
Les principales grandeurs de la sûreté de fonctionnement définies au chapitre I de la première partie : fiabilité, sécurité, disponibilité et réparabilité peuvent être déterminées à l'aide de cette formule en définissant correctement les états "succès" et "défaillants".

Exemple:

E_1 : état "en fonctionnement"

E_2 : état "en panne non dangereuse"

E_3 : état "en panne dangereuse"



$$\begin{bmatrix} \lambda_{11} & \lambda_{12} & \lambda_{13} \\ \lambda_{21} & \lambda_{22} & \lambda_{23} \\ \lambda_{31} & \lambda_{32} & \lambda_{33} \end{bmatrix} = \mathbb{A}$$

$$\lambda_{ii} = -\sum_{i \neq j} \lambda_{ij}$$

Le tableau suivant résume les valeurs des différents paramètres à prendre en compte

	équations	E_S	E_D	$\mathbb{P}_S(0)$	$\mathbb{A}_{SS}$	$\mathbb{1}_S$
fiabilité	$R(t)=\mathbb{P}_S(0).\exp(\mathbb{A}_{SS}t).\mathbb{1}_S$	E_1	E_2, E_3	$\begin{bmatrix} 1 \end{bmatrix}$	$\begin{bmatrix} \lambda_{11} \end{bmatrix}$	$\begin{bmatrix} 1 \end{bmatrix}$
sécurité	$S(t)=\mathbb{P}_S(0).\exp(\mathbb{A}_{SS}t).\mathbb{1}_S$	E_1, E_2	E_3	$\begin{bmatrix} 1 & 0 \end{bmatrix}$	$\begin{bmatrix} \lambda_{11} & \lambda_{12} \\ \lambda_{21} & \lambda_{22} \end{bmatrix}$	$\begin{bmatrix} 1 \\ 1 \end{bmatrix}$
réparabilité *	$RF(t)=1-\mathbb{P}_S(0).\exp(\mathbb{A}_{SS}t).\mathbb{1}_S$	E_2, E_3	E_1	dépend des conditions initiales considérées	$\begin{bmatrix} \lambda_{22} & \lambda_{23} \\ \lambda_{32} & \lambda_{33} \end{bmatrix}$	$\begin{bmatrix} 1 \\ 1 \end{bmatrix}$
disponibilité * *	$A(t)=\mathbb{P}_S(0).\exp(\mathbb{A}_{SS}t).\mathbb{1}_S$			dépend des conditions initiales considérées	$\begin{bmatrix} \lambda_{11} & \lambda_{12} & \lambda_{13} \\ \lambda_{21} & \lambda_{22} & \lambda_{23} \\ \lambda_{31} & \lambda_{32} & \lambda_{33} \end{bmatrix}$	$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$

*

L'équation (2) permet de déterminer la probabilité de rester dans un état "en panne" sachant que l'on y était auparavant; cet évènement est complémentaire de celui dont on cherche la probabilité lors du calcul de la réparabilité, on obtient donc:

$$RF(t) = 1 - \mathbb{P}_S(0).\exp(\mathbb{A}_{SS}\,t).\mathbb{1}_S.$$

$\mathbb{P}_S(0)$ peut prendre deux valeurs suivant l'état initial défaillant considéré:

$\mathbb{P}_S(0) = [1\ 0]$: état "en panne non dangereuse"
$\mathbb{P}_S(0) = [0\ 1]$: état "en panne dangereuse".

**

La partition de l'ensemble E des états possibles n'est plus à faire, la matrice de transition est içi prise dans son intégralité.

La matrice $\mathbb{1}_S$ a alors une forme particulière puisque la probabilité cherchée n'est plus celle de rester dans un état donné mais d'être dans un état de fonctionnement correct quel que soit l'état antérieur.

$\mathbb{P}_S(0)$ peut prendre plusieurs valeurs selon l'état initial considéré:

$\mathbb{P}_S(0) = [1\ 0\ 0]$: état "en bon fonctionnement"
$\mathbb{P}_S(0) = [0\ 1\ 0]$: état "en panne non dangereuse"
$\mathbb{P}_S(0) = [0\ 0\ 1]$: état "en panne dangereuse".

II) STRUCTURE MONOPROCESSEUR.

Nous allons utiliser les résultats du chapitre précédent afin de déterminer les paramètres de la sûreté de fonctionnement du système étudié (version monoprocesseur).

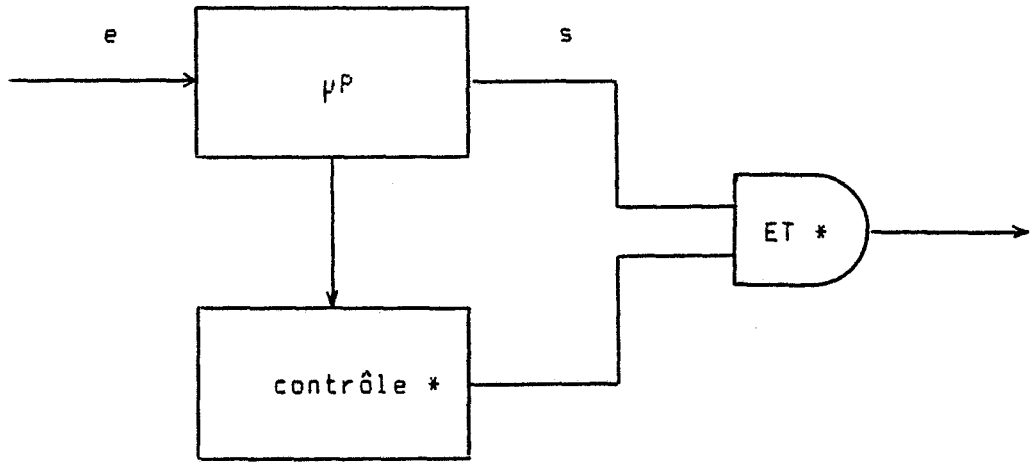
La valeur du taux de couverture de pannes utilisée pour ces calculs est celle déterminée dans la deuxième partie, soit 0,95.

Deux cas sont envisagés selon l'utilisation du système étudié:

- matériel au sol ⟶ réparable : les équipes de maintenance peuvent intervenir dès la détection de la panne.

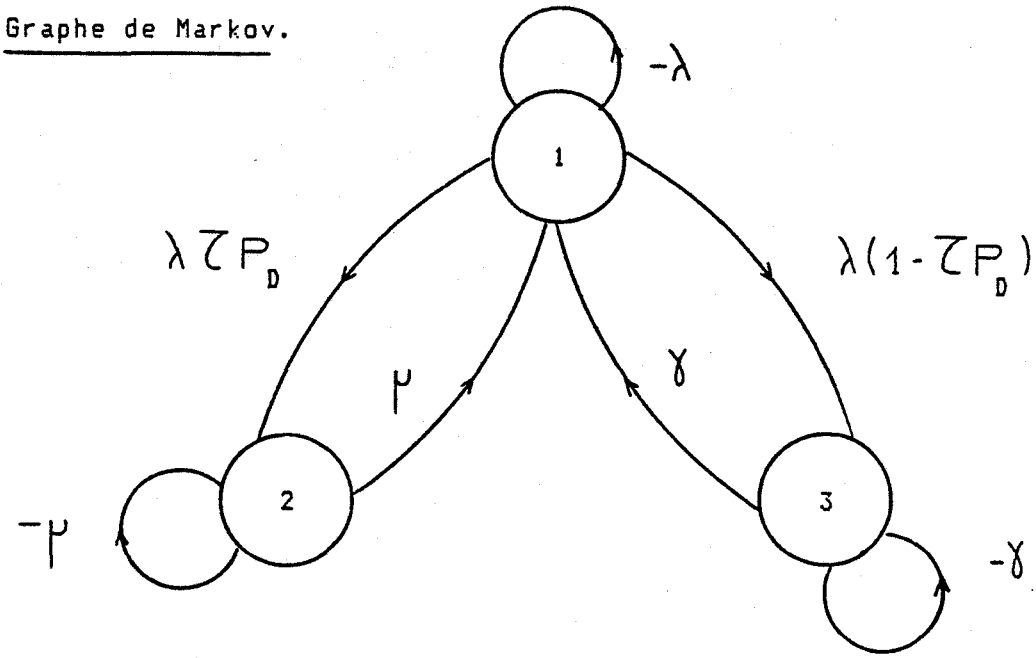
- matériel embarqué ⟶ non réparable : la réparation ne peut avoir lieu durant la mission.

II.1) Principe général.



* réalisé en sécurité intrinséque

II.2) Graphe de Markov.



état 1 : bon fonctionnement

état 2 : arrêt (panne détectée)

état 3 : danger après une panne non détectée

λ : taux de défaillance du microprocesseur

μ : taux de réparation après une panne détectée donc non dangereuse

γ : taux de réparation après un évènement catastrophique (panne non détectée donc dangereuse).

On remarque que, sur le graphe, le taux de défaillance est modulé par ζ et P_D :

ζ : taux de couverture de pannes

P_D : probabilité temporelle de détection

Le paramètre P_D a été introduit pour tenir compte du phénomène suivant:

Si une erreur détectable se manifeste au cours de l'exécution des programmes de test, elle est détectée.

Si une erreur détectable se manifeste au cours de l'exécution du programme d'application, elle peut engendrer une situation dangereuse.

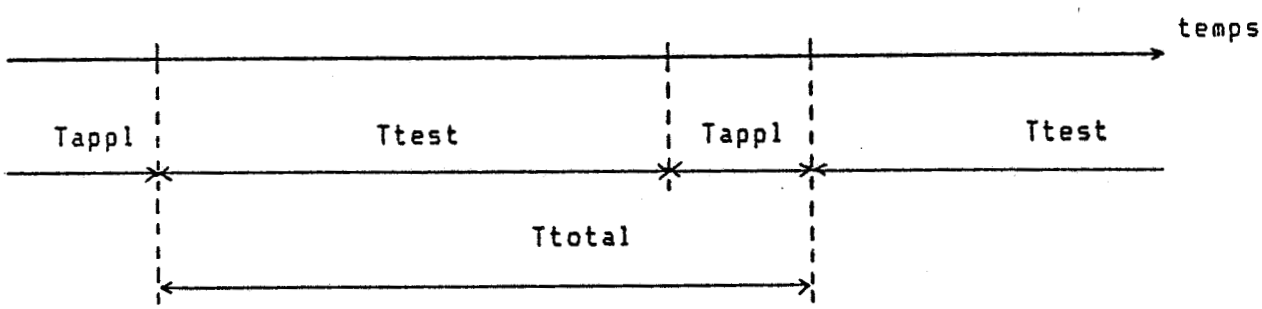
Si l'on peut considérer que l'apparition d'une erreur est aléatoire dans le temps, la probabilité de manifestation d'une erreur à l'instant t_e pendant le déroulement du programme d'application est égale à la probabilité que l'instant t_e se trouve dans l'intervalle

$$\left[Ttotal - Ttest\ ,\ Ttotal\right]$$

avec

Ttotal = durée d'un cycle
Ttest = durée d'un test
Tappl = durée de l'application

} ⟶ pour un cycle



La probabilité de détection est donc égale à

$$P_D = \frac{T\,test}{T\,total} = \text{proportion de temps utilisée pour les tests}$$

On remarque donc qu'il n'y aura détection que si:

- l'erreur est détectable par les tests

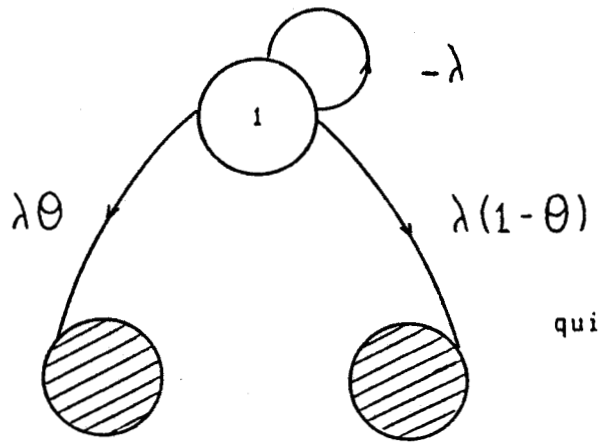
et

- l'erreur se manifeste durant l'exécution des programmes de test

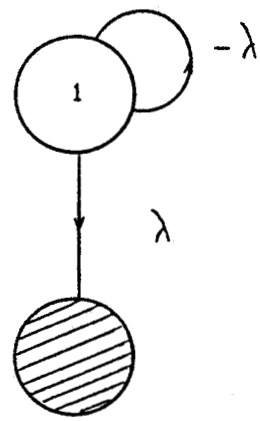
Le taux de couverture de pannes global devient donc égal au produit $\zeta . P_D$ qui sera noté Θ pour la simplification des expressions obtenues dans tous les développements mathématiques de ce chapitre.

II.3) Calcul de la fiabilité.

Le graphe de fiabilité se déduit du graphe précédent en rendant les états défaillants (défaillance détectée ou non) absorbants:



qui se simplifie:



De la matrice de transition générale

$$\mathbb{A} = \begin{bmatrix} -\lambda & \lambda\theta & \lambda(1-\theta) \\ \mu & -\mu & 0 \\ \gamma & 0 & -\gamma \end{bmatrix}$$

on déduit la matrice $\mathbb{A}_{SS}$

$$\mathbb{A}_{SS} = \begin{bmatrix} -\lambda \end{bmatrix}$$

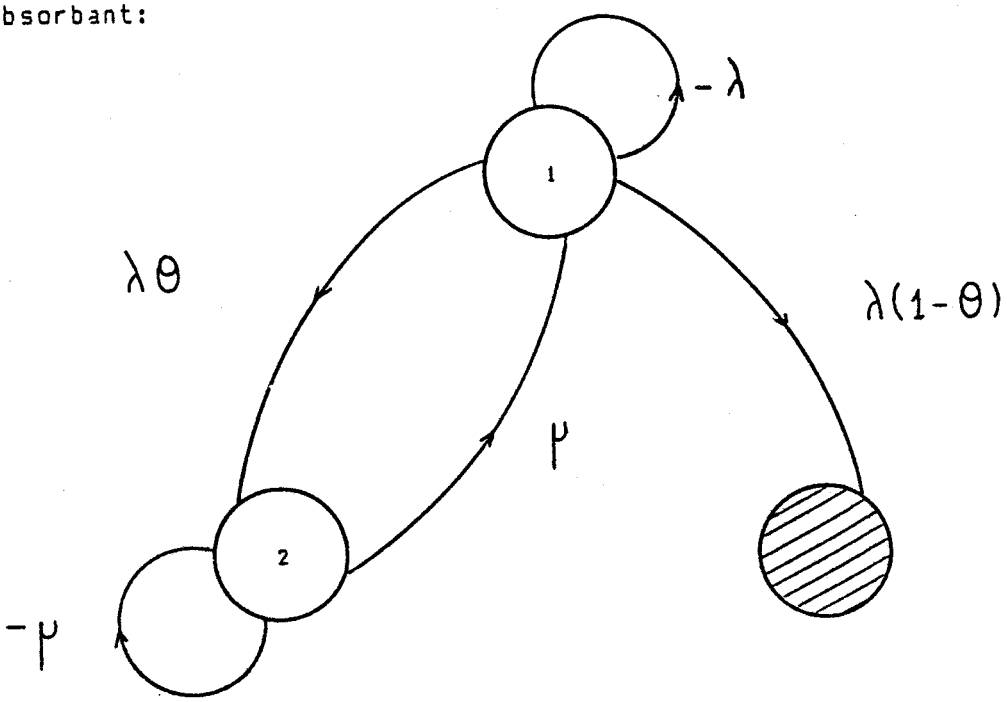
La fiabilité est donc égale à:

$$\boxed{R(t) = e^{-\lambda t}} \quad ,$$

ce résultat est classique.

II.4) Calcul de la sécurité.

Dans le graphe de sécurité, l'état "panne dangereuse" est un état absorbant:



La matrice $\mathbb{A}_{SS}$ est donc égale à:

$$\mathbb{A}_{SS} = \begin{bmatrix} -\lambda & \lambda\theta \\ \mu & -\mu \end{bmatrix}$$

$$\mathbb{S}(s) = \underbrace{\begin{bmatrix} 1 & 0 \end{bmatrix}}_{\mathbb{P}_S(0)} \cdot \underbrace{\begin{bmatrix} s+\lambda & -\lambda\theta \\ -\mu & s+\mu \end{bmatrix}^{-1}}_{[\,s\mathbb{I} - \mathbb{A}_{SS}\,]^{-1}} \cdot \underbrace{\begin{bmatrix} 1 \\ 1 \end{bmatrix}}_{\mathbb{1}_S}$$

Le déterminant de $(s\,\mathbb{I} - \mathbb{A}_{ss})$ est égal à:

$$(s+\lambda)(s+\mu) - \lambda\Theta\mu = (s-s_1)(s-s_2)$$

avec

$$s_1 = \frac{-(\lambda+\mu) + \sqrt{(\lambda+\mu)^2 - 4\lambda\mu(1-\Theta)}}{2}$$

$$s_2 = \frac{-(\lambda+\mu) - \sqrt{(\lambda+\mu)^2 - 4\lambda\mu(1-\Theta)}}{2}$$

On obtient donc

$$\tilde{S}(s) = \frac{(s+\mu) + \lambda\Theta}{(s-s_1)(s-s_2)} = \frac{s_1+\mu+\lambda\Theta}{(s_1-s_2)(s-s_1)} + \frac{s_2+\mu+\lambda\Theta}{(s_2-s_1)(s-s_2)}$$

Essayons de simplifier cette expression

$$2s_1 = -\lambda - \mu + \sqrt{\mu^2\left(1+\left(\frac{\lambda}{\mu}\right)^2 - \frac{2\lambda}{\mu}(1-2\Theta)\right)}$$

$$\frac{2s_1}{\mu} = -1 - \frac{\lambda}{\mu} + \left\{1 + \left(\frac{\lambda}{\mu}\right)^2 - \frac{2\lambda}{\mu}(1-2\Theta)\right\}^{1/2}$$

on pose $\bar{\Theta} = 1 - \Theta$

$$\frac{2s_1}{\mu} = -1 - \frac{\lambda}{\mu} + \left\{1 + \left(\frac{\lambda}{\mu}\right)^2 + \frac{2\lambda}{\mu}(1-2\bar{\Theta})\right\}^{1/2}$$

or $\lambda << \mu$, faisons un développement limité au 2^e ordre:

$$\frac{2s_1}{\mu} \cong -1 - \frac{\lambda}{\mu} + 1 + \frac{1}{2}\left\{\left(\frac{\lambda}{\mu}\right)^2 + \frac{2\lambda}{\mu}(1-2\bar{\Theta})\right\} - \frac{1}{8}\left\{\left(\frac{\lambda}{\mu}\right)^2 + \frac{2\lambda}{\mu}(1-2\bar{\Theta})\right\}^2$$

$$\frac{2s_1}{\mu} \cong -\frac{2\lambda}{\mu}\bar{\Theta} + \left(\frac{\lambda}{\mu}\right)^2\left\{2\bar{\Theta}(1-\bar{\Theta})\right\} - \frac{1}{2}\left(\frac{\lambda}{\mu}\right)^3(1-2\bar{\Theta}) - \frac{1}{8}\left(\frac{\lambda}{\mu}\right)^4$$

$$s_1 \cong -\lambda(1-\Theta)$$

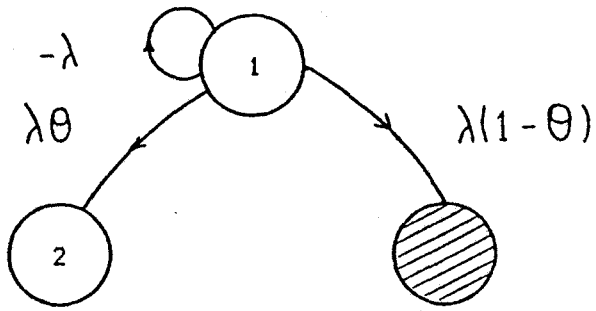
de même pour s_2, $\quad s_2 \cong -\mu - \lambda\Theta$

On obtient donc

$$S(t) \cong e^{-\lambda(1-\Theta)t}$$

II.5) Calcul de la sécurité (structure non réparable).

Le graphe de sécurité de la structure non réparable est le suivant:



Le calcul de sécurité de la structure non réparable est identique à celui de la structure réparable avec un taux de réparation nul ($\mu = 0$)

On obtient donc

$$S(t) = \Theta + (1 - \Theta).e^{-\lambda t}$$

II.6) Calcul de l'indice de sécurité.

$$I_s(t) = - \frac{1}{S(t)} \cdot \frac{dS(t)}{dt}$$

On obtient donc, pour la structure réparable:

$$I_s(t) \cong \lambda(1 - \Theta)$$

et pour la structure non réparable:

$$I_s(t) \cong \frac{\lambda(1 - \Theta)}{(1 - \Theta) + \Theta e^{\lambda t}}$$

II.7) Résultats numériques obtenus.

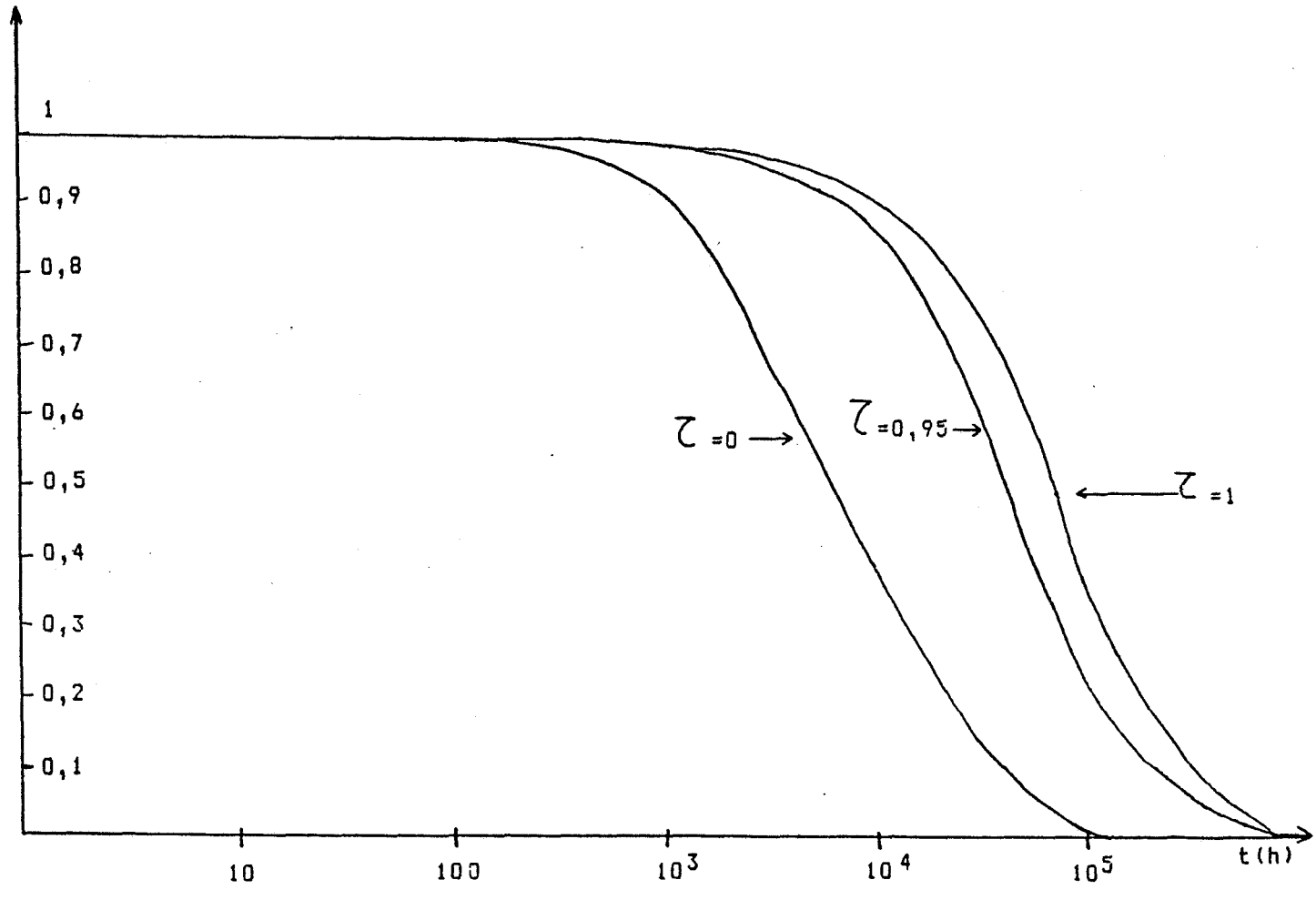
Les courbes suivantes illustrent les expressions mathématiques obtenues dans les trois cas suivants:

$\zeta = 0$: pas de procédure de test

$\zeta = 0,95$: structure étudiée

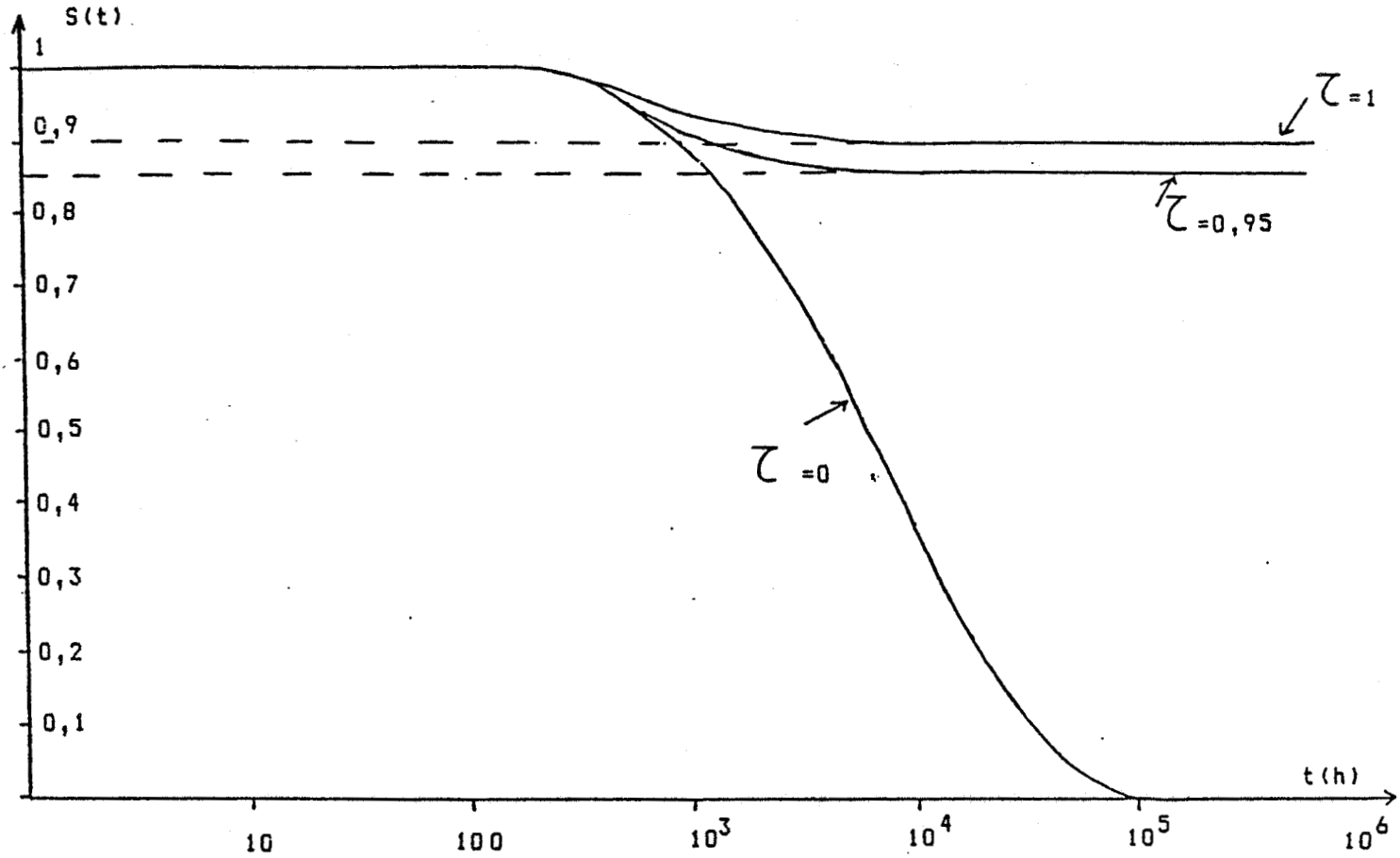
$\zeta = 1$: toutes les pannes sont détectées par les procédures de test.

Ces courbes sont tracées dans les conditions étudiées c'est à dire 10% du temps consacré au programme d'application et 90% au test, soit donc $P_D = 0,9$ avec un taux horaire de défaillance $\lambda = 10^{-4}\ h^{-1}$.



Courbes (1) de sécurité - Structure réparable

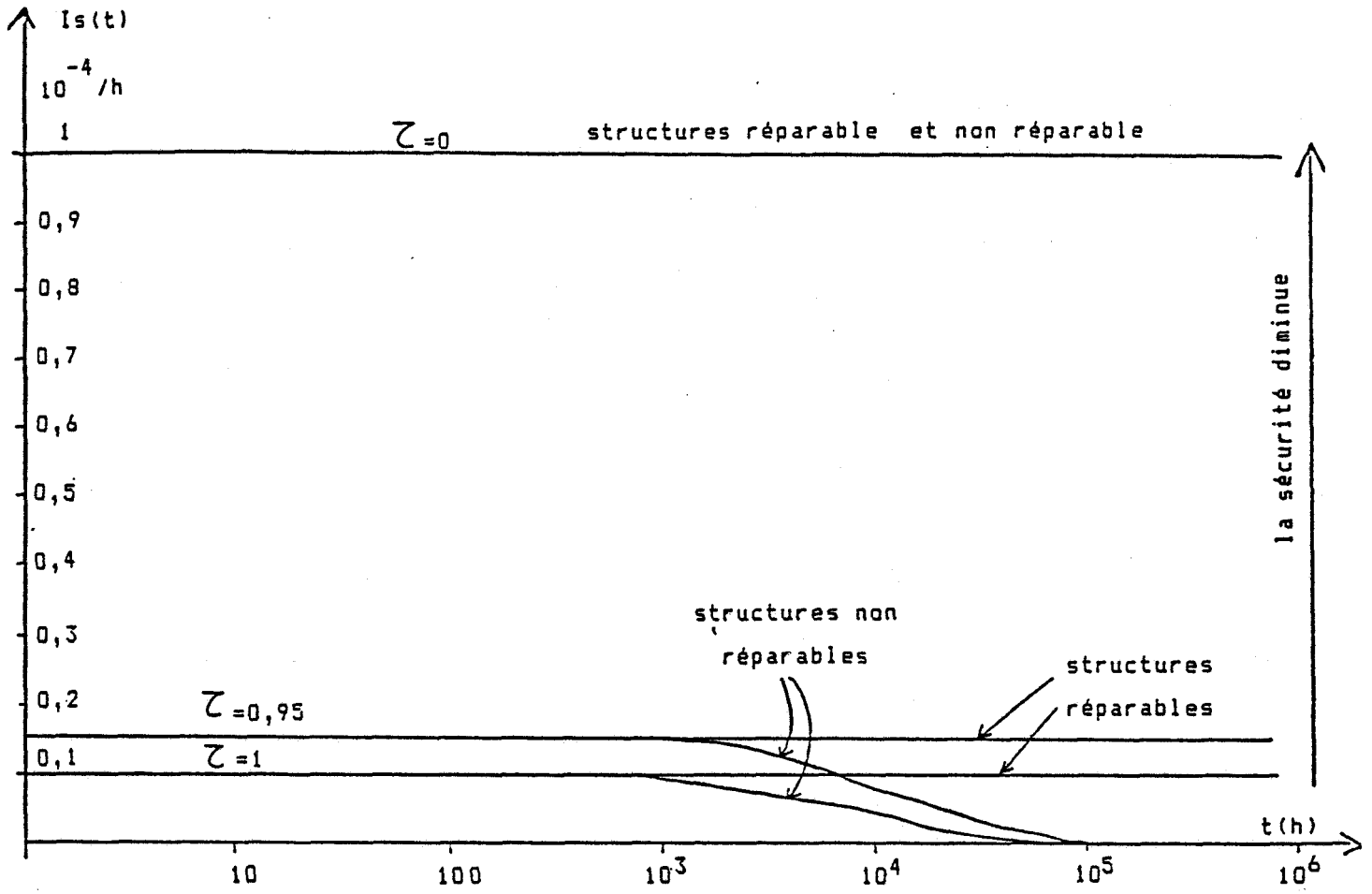
La courbe de fiabilité correspond à la courbe de sécurité lorsque $\zeta = 0$.



Courbes (2) de sécurité - Structure non réparable

On peut remarquer que ces courbes 2, contrairement aux courbes précédentes 1, ne tendent pas vers zéro lorsque le temps t devient tel que $t > \frac{10}{\lambda(1 - \tau P_D)}$

Ceci s'explique facilement par le fait que la structure est non réparable et que lorsque la durée de fonctionnement est supérieure à $\frac{1}{\lambda}$, on tend vers une certitude de panne et donc la sécurité devient égale à la probabilité de détecter la panne soit τP_D .



Courbes (3) Indice de sécurité
(ou taux horaire de défaillance catastrophique)

On peut remarquer, en observant ces courbes 3, que la structure non réparable semble plus sûre que la structure réparable, ce jugement est à nuancer en considérant ces deux points:

- lorsque la structure est réparable, elle est remplacée dès l'apparition d'une défaillance (au temps de réparation près); le taux horaire de défaillance catastrophique est constant puisque l'on considère un taux de défaillance horaire constant (le système ne "vieillit" pas).

- lorsque la structure n'est pas réparable, le système reste en arrêt après toute défaillance (catastrophique ou non). Un arrêt après une défaillance bénigne (défaillance détectée) est un état de sécurité, le système défaillant n'étant pas réparé, il reste dans cet état de sécurité "arrêt" alors que dans le cas des structures réparables, le système défaillant est réparé et peut donc de nouveau tomber en panne catastrophique d'où, une diminution de la sécurité.

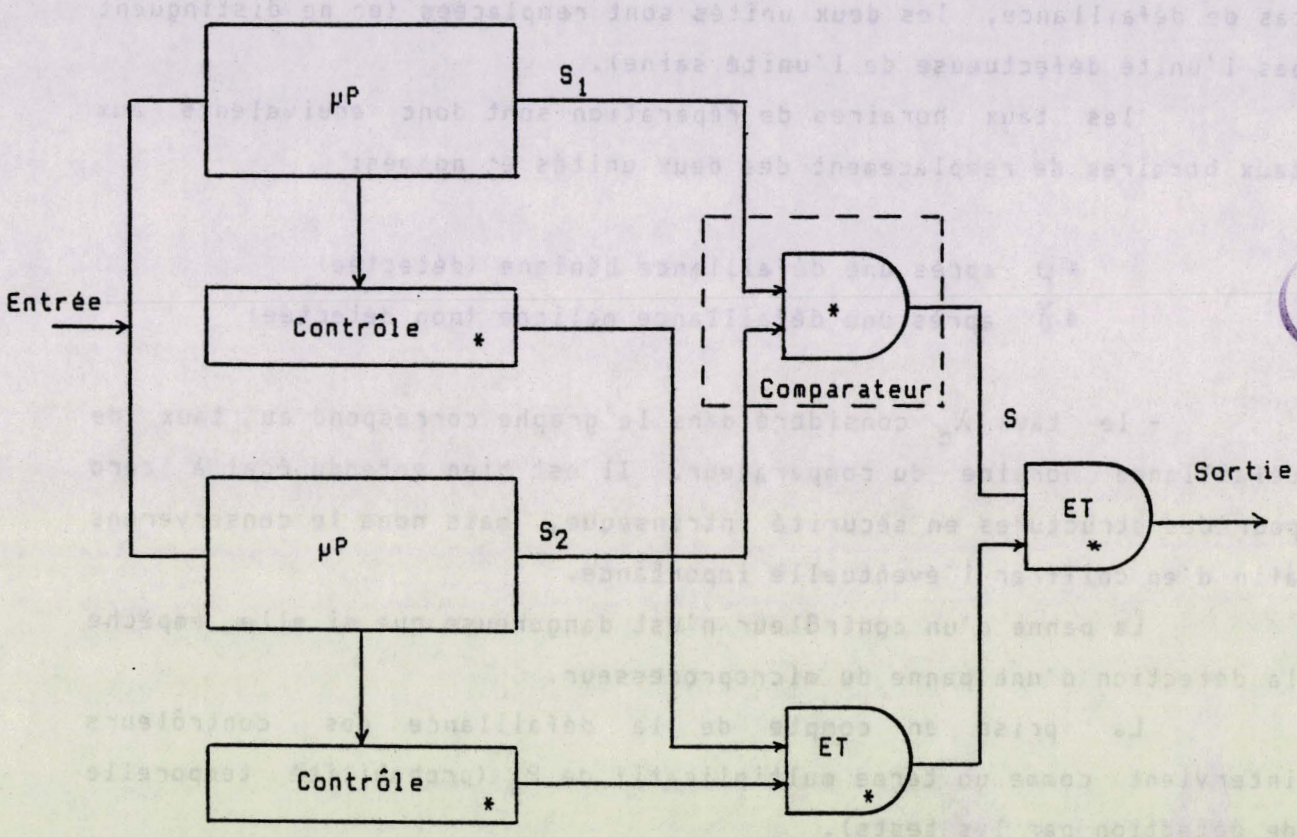
II.8) Conclusion.

L'examen des résultats numériques obtenus, montre que la structure monoprocesseur n'est pas satisfaisante pour atteindre un niveau de sécurité suffisant (le taux horaire de défaillance catastrophique est au maximum égal à 10^{-5}/h pour les structures réparables avec une proportion temporelle de test de 90%).

Nous avons donc envisagé l'étude d'une version biprocesseur, réparable ou non, de notre procédé, les résultats sont présentés dans les paragraphes suivants.

III) STRUCTURE BIPROCESSEUR.

III.1) Principe général (en sécurité positive):



les éléments marqués d'une * sont réalisés en sécurité intrinsèque.

A l'issue du comparateur, le signal s prend la valeur:

- $s = s_1 = s_2$ si $s_1 = s_2$
- $s = 0$ si $s_1 \neq s_2$

III.2) Graphe de Markov.

- les circuits redondants sont considérés comme identiques soit donc:

$$\lambda_1 = \lambda_2 = \lambda \quad \text{(taux de défaillance égaux)}$$

- un seul taux horaire de réparation est considéré puisque, en cas de défaillance, les deux unités sont remplacées (on ne distinguent pas l'unité défectueuse de l'unité saine).

les taux horaires de réparation sont donc équivalents aux taux horaires de remplacement des deux unités et nommés:

* μ après une défaillance bénigne (détectée)
* γ après une défaillance maligne (non détectée)

- le taux λ_c considéré dans le graphe correspond au taux de défaillance horaire du comparateur. Il est bien entendu égal à zéro pour des structures en sécurité intrinsèque, mais nous le conserverons afin d'en chiffrer l'éventuelle importance.

La panne d'un contrôleur n'est dangereuse que si elle empêche la détection d'une panne du microprocesseur.

La prise en compte de la défaillance des contrôleurs intervient comme un terme multiplicatif de P_D (probabilité temporelle de détection par les tests).

Dans les expressions, P_D est remplacé par $P_D(1- P_{def})$ avec P_{def} la probabilité de défaillance du contrôleur.

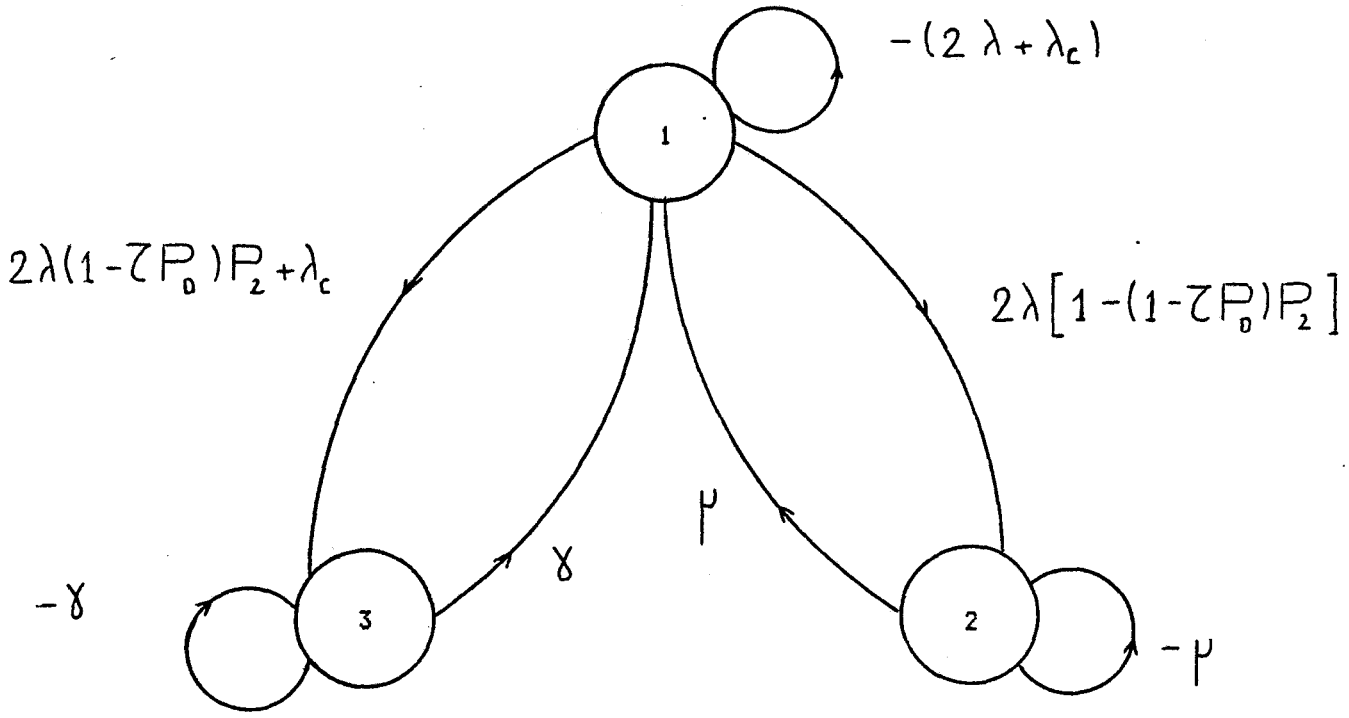
$$P_{def} = \lambda_{control} \cdot T$$

avec $\lambda_{control}$: taux de défaillance horaire du contrôleur ($\lambda_{control}$ est de l'ordre de 10^{-9} /h REF.BAR).

et T : durée pendant laquelle le contrôleur peut tomber en panne après la défaillance du microprocesseur sans altérer le signal de sortie (entrée du comparateur)

pour $T = 10^5$ h (environ 10 ans), $P_{def} = 10^{-4}$ /h, son influence est donc négligeable sur P_D .

Le graphe obtenu est le suivant:



état 1 : fonctionnement correct des deux unités

état 2 : défaillance détectée (Arrêt)

* détection par la comparaison des sorties
  ou
* détection par les tests fonctionnels (panne de mode commun détectable)

état 3 : défaillance non détectée (Danger)

* panne du circuit de comparaison
* panne de mode commun non détectable des deux microprocesseurs

Remarques

- Toute panne des circuits de comparaison est considérée comme dangereuse (hypothèse pessimiste).

- La mise en sécurité par la redondance est basée sur la comparaison des sorties des deux unités effectuant la même tâche. Il est bien évident que la comparaison n'est efficace que lors de la modification des vecteurs de sortie, celle-ci faisant suite à l'exécution d'instructions d'élaboration des sorties.

Il existe un laps de temps (intervalle séparant deux émissions de vecteur de sortie) pendant lequel le second microprocesseur peut tomber en panne alors que le premier était défaillant.

L'hypothèse généralement émise:
" toute panne simple est détectée avant l'apparition d'une seconde panne", n'est plus valable dans le cas présent.

Nous avons donc introduit le terme P_2 correspondant à la probabilité d'apparition d'une panne dans le second microprocesseur durant le temps Δt . (Δt : latence de propagation de la panne à la sortie du 1er microprocesseur) et ce pour des pannes non détectables par les tests fonctionnels.

On obtient donc :

$$P_2 = \lambda(1-\overline{c}P_D)\Delta t$$

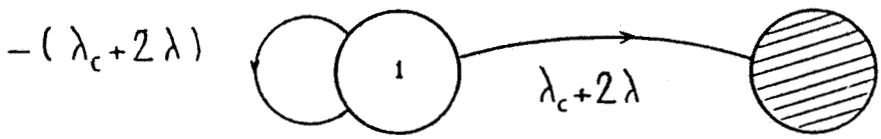
Une hypothèse pessimiste est émise :

Les sorties sont identiques dès lors que les deux microprocesseurs sont en panne non détectable par les tests fonctionnels.

Pour les développements mathématiques, l'expression $\overline{c}P_D$ sera remplacée par Θ .

III.3) Calcul de la fiabilité.

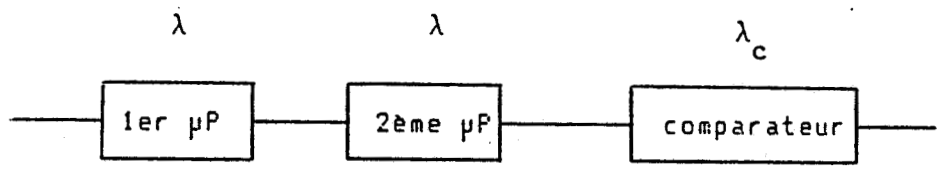
Le graphe de fiabilité se déduit simplement du graphe précédent:



On trouve le résultat classique:

$$R(t) = e^{-(2\lambda+\lambda_c)t}$$

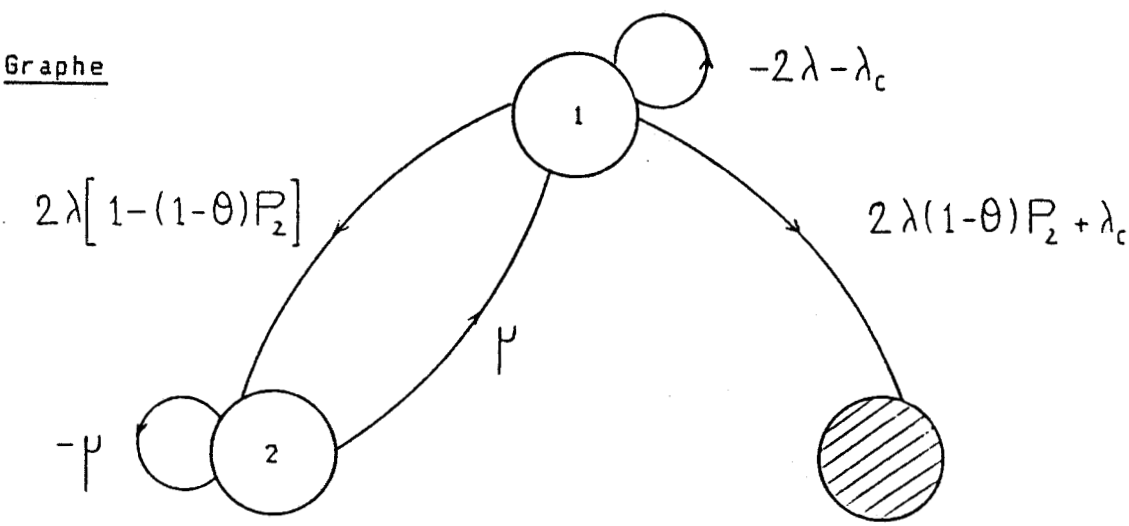
correspondant aux trois éléments en série au sens de la fiabilité:



$$R(t) = \prod_{i=1}^{3} R_i = e^{-\lambda t}.e^{-\lambda t}.e^{-\lambda_c t}$$

III.4) Calcul de la sécurité (structure réparable).

Graphe



De la matrice de transition générale :

$$\mathbb{A} = \begin{bmatrix} -2\lambda-\lambda_c & 2\lambda[1-(1-\theta)P_2] & \lambda_c+2\lambda(1-\theta)P_2 \\ \mu & -\mu & 0 \\ \gamma & 0 & -\gamma \end{bmatrix}$$

on déduit la matrice $\mathbb{A}_{SS}$

$$\mathbb{A}_{SS} = \begin{bmatrix} -2\lambda - \lambda_c & 2\lambda\left[1-(1-\theta)P_2\right] \\ \mu & -\mu \end{bmatrix}$$

$$\tilde{S}(s) = \mathbb{P}_S(0)\left[s\,\mathbb{I} - \mathbb{A}_{SS}\right]^{-1} \mathbb{1}_S$$

le déterminant de $\left[s\,\mathbb{I} - \mathbb{A}_{SS}\right]$ est égal à:

$$(s + 2\lambda + \lambda_c)(s + \mu) - 2\lambda\mu\{1 - (1-\theta)P_2\} = (s - s_1)(s - s_2)$$

avec

$$s_1 = \frac{-2\lambda - \lambda_c - \mu + \sqrt{(2\lambda + \lambda_c + \mu)^2 - 4\lambda_c\mu - 8\lambda\mu(1-\theta)P_2}}{2}$$

$$s_2 = \frac{-2\lambda - \lambda_c - \mu - \sqrt{(2\lambda + \lambda_c + \mu)^2 - 4\lambda_c\mu - 8\lambda\mu(1-\theta)P_2}}{2}$$

on obtient donc

$$\tilde{S}(s) = \frac{s + \mu + 2\lambda\{1 - (1-\theta)P_2\}}{(s - s_1)(s - s_2)}$$

soit donc

$$S(t) = \sum_{\substack{i=1 \\ i\neq j}}^{2} \frac{s_i + \mu + 2\lambda\{1-(1-\theta)P_2\}}{s_i - s_j} \cdot e^{s_i t}$$

Simplifions cette expression en faisant un développement limité comme dans l'étude de la version monoprocesseur (on nomme $\overline{\Theta} = 1 - \Theta$).

$$\frac{2s}{\mu}i = - \frac{2\lambda}{\mu} - \frac{\lambda}{\mu}c - 1 \pm \{ 1 + (\frac{2\lambda}{\mu} + \frac{\lambda}{\mu}c)^2 - \frac{2\lambda}{\mu}c + \frac{4\lambda}{\mu} (1 - 2\overline{\Theta}P_2) \}^{1/2}$$

$\frac{\lambda}{\mu} \ll 1$ et $\frac{\lambda}{\mu}c \ll 1$, effectuons un développement limité au 2eme ordre

$$\frac{2s}{\mu}i \cong - \frac{2\lambda}{\mu} - \frac{\lambda}{\mu}c - 1 \pm \{1 + \frac{1}{2}(\frac{2\lambda}{\mu} + \frac{\lambda}{\mu}c)^2 - \frac{\lambda}{\mu}c + \frac{2\lambda}{\mu} (1 - 2\overline{\Theta}P_2)$$

$$- \frac{1}{8} \{ \frac{2\lambda}{\mu} + \frac{\lambda}{\mu}c)^2 - \frac{2\lambda}{\mu}c + 4\frac{\lambda}{\mu} (1 - 2\overline{\Theta}P_2) \}^2 \}$$

$$\frac{2s}{\mu}i \cong - \frac{2\lambda}{\mu} - \frac{\lambda}{\mu}c - 1 \pm \{1 - \frac{\lambda}{\mu}c + \frac{2\lambda}{\mu}(1 - 2\overline{\Theta}P_2) + \frac{4\lambda}{\mu}(1 - \overline{\Theta}P_2)(\frac{2\lambda}{\mu} \overline{\Theta}P_2 + \frac{\lambda}{\mu}c)\}$$

or $P_2 \ll 1$

$$\frac{2s}{\mu}i \cong - \frac{2\lambda}{\mu} - \frac{\lambda}{\mu}c - 1 \pm \{1 - \frac{\lambda}{\mu}c + \frac{2\lambda}{\mu} (1 - 2\overline{\Theta}P_2) + 4 \frac{\lambda\lambda c}{\mu^2}\}$$

soit donc $s_1 \cong - \lambda_c - 2\lambda (\overline{\Theta}P_2 - \frac{\lambda}{\mu}c)$ et $s_2 \cong - \mu - 2\lambda (1 - \overline{\Theta}P_2 + \frac{\lambda}{\mu}c)$

soit

$$S(t) = \frac{-\lambda_c + 2\lambda(1 - 2\overline{\Theta}P_2 + \frac{\lambda}{\mu}c) + \mu}{-\lambda_c + \mu + 2\lambda(1 - 2\overline{\Theta}P_2 + \frac{2\lambda}{\mu}c)} \cdot e^{s_1 t} + \frac{\frac{2\lambda\lambda}{\mu}c}{-\lambda_c + \mu + 2\lambda(1 - 2\overline{\Theta}P_2 + \frac{2\lambda}{\mu}c)} \cdot e^{s_2 t}$$

$$S(t) = A\, e^{s_1 t} + B\, e^{s_2 t}$$

avec $$A = \frac{-\lambda_c + 2\lambda(1 - 2\overline{\Theta}P_2 + \frac{\lambda}{\mu}c) + \mu}{-\lambda_c + \mu + 2\lambda(1 - 2\overline{\Theta}P_2 + \frac{2\lambda}{\mu}c)} \neq 1$$

et $$B = \frac{\frac{2\lambda\lambda}{\mu}c}{-\lambda_c + \mu + 2\lambda(1 - 2\overline{\Theta}P_2 + \frac{2\lambda}{\mu}c)} \neq 0$$

comme de plus $e^{s_2 t}$ tend très rapidement vers 0 quand t croit, on obtient

$$\boxed{S(t) \cong e^{-\left[\lambda_c + 2\lambda\left[(1-\Theta)P_2 - \frac{\lambda}{\mu}c\right]\right] t}}$$

III.5) Calcul de la sécurité (structure non réparable).

Le graphe de sécurité et la détermination de S(t) sont identiques à ceux de la structure réparable en prenant $\mu = 0$

On obtient donc

$$s_1 = -(2\lambda+\lambda_c) \qquad s_2 = 0$$

$$S(t) = \frac{2\lambda(1-\Theta)P_2+\lambda_c}{(2\lambda+\lambda_c)} . e^{-(2\lambda+\lambda_c)\,t} + \frac{2\lambda\{1-(1-\Theta)P_2\}}{(2\lambda+\lambda_c)}$$

III.6) Calcul de l'indice de sécurité $I_s(t)$.

$$I_s(t) = -\frac{1}{S(t)} \cdot \frac{dS(t)}{dt}$$

Structure réparable:

$$I_s(t) = \lambda_c + 2\lambda\{(1-\Theta)P_2 - \frac{\lambda}{\mu}c\} \quad .$$

Structure non réparable:

$$I_s(t) = \frac{\{\lambda_c + 2\lambda(1-\Theta)P_2\}.\{2\lambda + \lambda_c\}}{\lambda_c + 2\lambda(1-\Theta)P_2 + 2\lambda\{1-(1-\Theta)P_2\}e^{(\lambda_c+2\lambda)t}}$$

III.7) Résultats obtenus.

III.7.1) Influence du comparateur.

Dans un premier temps, nous allons essayer de chiffrer l'importance du taux de défaillance du comparateur sur les résultats de sécurité obtenus.

Pour la structure réparable, l'influence de λ_c est facile à déterminer:

$$\boxed{I_s(t) = \lambda_c + 2\lambda(1-\Theta)P_2 - \frac{2\lambda\lambda_c}{\mu}}$$

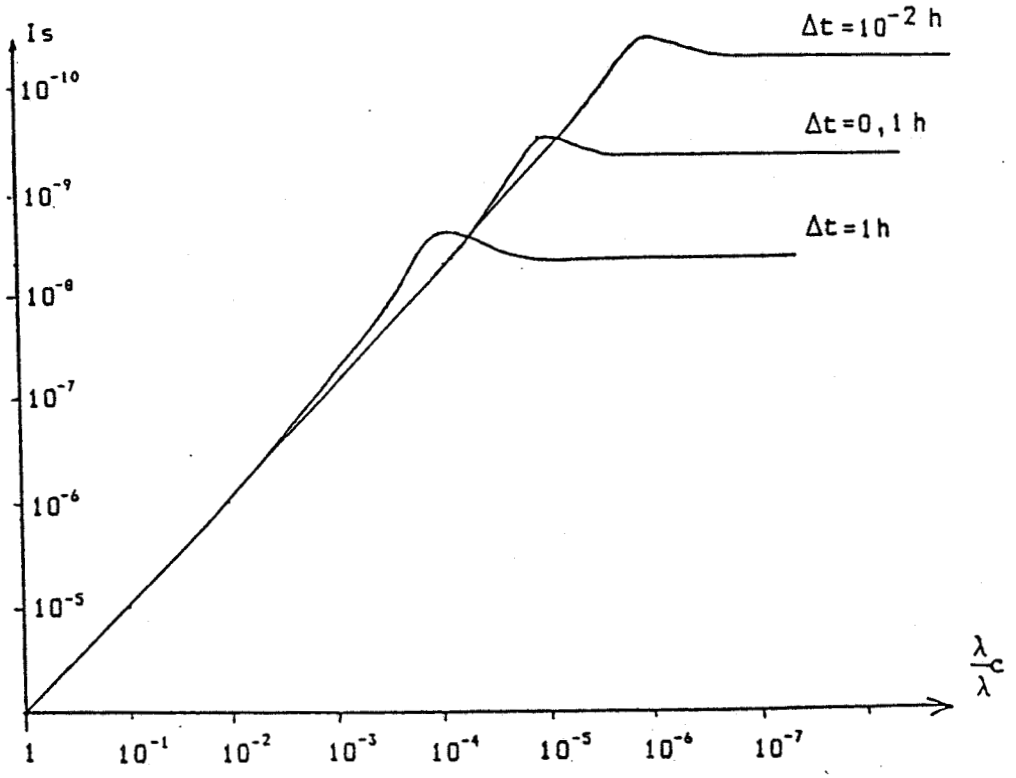
$$I_s(t) = \underbrace{\lambda_c}_{\text{défaillance du comparateur}} - \underbrace{\frac{2\lambda\lambda_c}{\mu}}_{\text{défaillance du comparateur et de la redondance}} + \underbrace{2\lambda(1-\Theta)P_2}_{\text{défaillance de la redondance}}$$

- λ_c exprime la défaillance du comparateur quel que soit l'état de la redondance (défaillante ou non)
- $2\lambda(1-\Theta)P_2$ exprime la défaillance de la redondance quel que soit l'état du comparateur
- on soustrait donc le terme $\frac{2\lambda\lambda_c}{\mu}$ qui exprime la défaillance simultanée du comparateur et de la redondance. Celle ci s'exprime en fait par une expression beaucoup plus complexe dont les termes d'ordre supérieur à 2 ont été éliminés lors du développement limité.

On remarque donc la prédominance du terme λ_c dès qu'il devient du même ordre de grandeur que λ , ce qui démontre la conclusion que nous avions faite dans la première partie lors de la présentation de la redondance: la sécurité des structures redondantes est essentiellement fonction du taux de défaillance du comparateur et, à un niveau moindre, de la probabilité d'apparition d'une panne sur chacun des microprocesseurs (les sorties sont identiques mais fausses).

Dans un premier temps, nous ne considérons que la redondance sans la mise en sécurité par les tests fonctionnels ($\Theta = 0$). On peut donc tracer les courbes de l'indice de sécurité $I_s = f(\frac{\lambda}{\mu}c)$ pour différentes valeurs de $P_2 = \lambda \Delta t$.

(les courbes sont tracées pour $\mu = 1/h$ et $\lambda = 10^{-4}/h$)



Courbes (4) de Sécurité en fonction du rapport λ_c / λ pour divers P_2

On remarque deux zones distinctes sur ces courbes :

- une zone où l'indice de sécurité est essentiellement fonction de λ_c

$$I_s \cong \lambda_c \quad \text{lorsque} \quad \lambda_c > \lambda^2 \Delta t$$

- une zone où l'indice de sécurité devient constant quelle que soit la valeur de λ_c:

$$I_s \cong 2\lambda^2 \Delta t \quad , \text{ lorsque} \quad \lambda_c < \lambda^2 \Delta t$$

d'où on peut tirer la conclusion suivante:

La redondance n'est efficace que si le taux de défaillance horaire du comparateur est inférieur au produit du taux de défaillance horaire du microprocesseur et de la probabilité d'apparition d'une seconde panne avant la détection de la première:

$$\lambda_c < \lambda P_2$$

Pour la structure non réparable, l'indice de sécurité n'est plus constant en fonction du temps.

En effet, lorsque $\lambda t > 1$, la probabilité que les microprocesseurs soient tombés en panne tend vers 1; le système n'étant pas réparable, il demeure à l'arrêt après l'apparition d'une panne (dangereuse ou non) et ne peut plus devenir dangereux; l'indice de sécurité tend vers zéro lorsque $\lambda t > 1$.

Pour l'étude de l'influence du taux de défaillance horaire du comparateur, nous allons nous placer dans le cas où $\lambda t << 1$, c'est à dire dans l'intervalle de temps où l'indice de sécurité est maximal.

Un système non réparable n'est de toute façon pas prévu pour fonctionner pendant une durée supérieure à $\frac{1}{\lambda}$.

On note $\lambda_c = x\,\lambda$

$$I_s(t) = \frac{\lambda\{x + 2(1 - \Theta)P_2\}\,(2 + x)}{x + 2(1 - \Theta)P_2 + 2\{1 - (1 - \Theta)P_2\}.e^{(x+2)\lambda t}}$$

On considère que $x \leqslant 1$ sinon la structure redondante est moins sécuritaire que la structure non redondante et son utilisation ne se justifie plus.

Pour $\lambda t << 1$, $e^{(x+2)\lambda t} \cong 1$

On obtient donc:

$$I_s(t) \# \lambda(x + 2(1 - \Theta)P_2)$$

Si on trace les courbes I_s en fonction de λ_c/λ pour diverses valeurs de P_2 et en ne considérant pas les tests fonctionnels ($\Theta = 0$), on obtient les mêmes courbes que pour la structure réparable (courbes 4) et on peut donc émettre les mêmes conclusions à savoir que la redondance n'est efficace que si le comparateur est tel que $\lambda_c < \lambda P_2$.

Lorsque l'on introduit les tests fonctionnels, il faut remplacer λ par $\lambda(1 - \Theta)$.

Conclusion :

Pour les structures réparables et pour les structures non réparables durant l'intervalle de fonctionnement:

$$I_s \cong \lambda_c \quad \text{——————} \quad \text{pour} \quad \lambda_c > \lambda^2 (1 - \Theta)^2 \Delta t$$

$$I_s \cong 2\lambda^2 (1 - \Theta)^2 \Delta t \quad \text{———} \quad \text{pour} \quad \lambda_c < \lambda^2 (1 - \Theta)^2 \Delta t$$

III.7.2) Influence de la latence de propagation Δt .

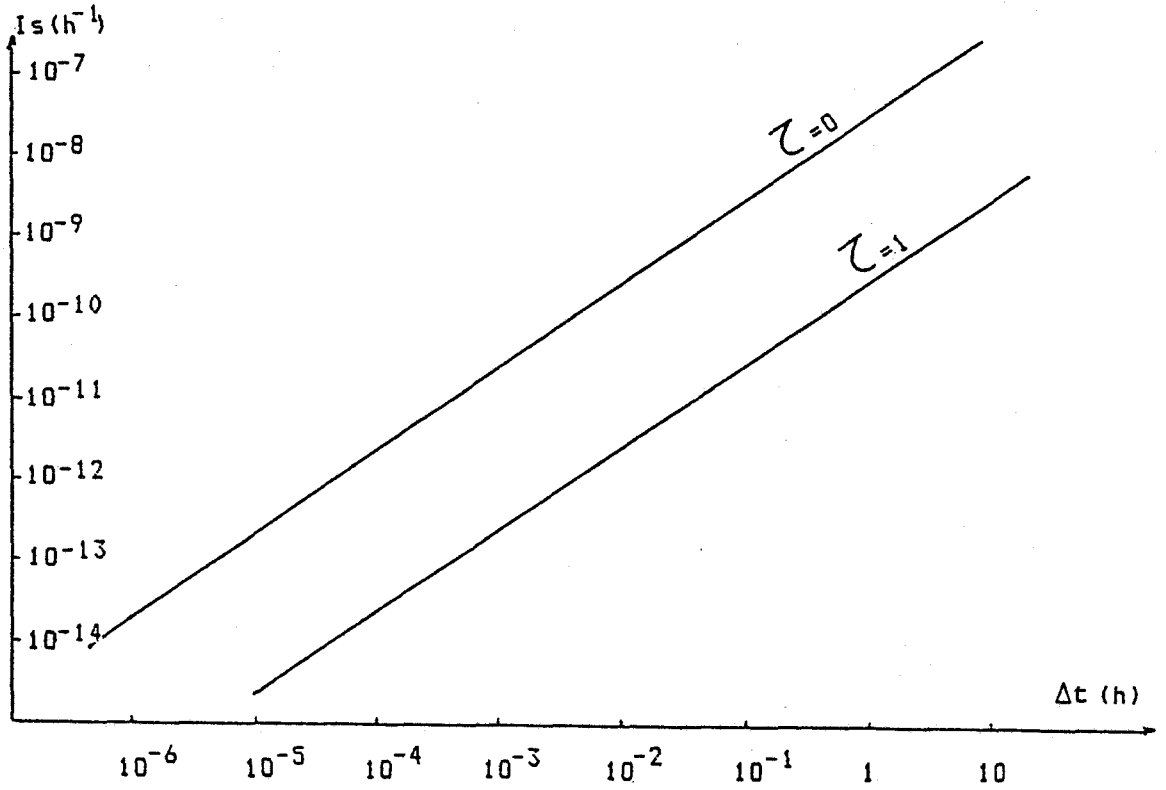
Afin de chiffrer l'importance de la latence de propagation Δt, nous allons considérer que le circuit de comparaison est tel que $\lambda_c < \lambda^2 (1 - \Theta)^2 \Delta t$, c'est à dire que la sécurité ne dépend que de la redondance. Ceci revient à considérer que $\lambda_c = 0$ dans les expressions de $S(t)$ et $I_s(t)$.

Dans l'intervalle de temps de fonctionnement, la structure non réparable possède le même indice de sécurité que la structure réparable:

$$I_s(t) = 2\lambda^2 (1 - \Theta)^2 \Delta t = 2\lambda^2 (1 - \tau P_D)^2 \Delta t$$

on remarque donc la linéarité de I_s en fonction de Δt.

On trace les courbes de $I_s(t)$ en fonction de Δt pour diverses valeurs du taux de couverture de pannes (pour une valeur de $P_D = 0,9$ correspondant à notre procédé).



Indice de Sécurité en fonction de Δt pour les valeurs extrêmes de τ

Courbes (5)

Lorsque l'on observe ces courbes 5 , on remarque le faible apport des tests sur la valeur de l'indice de sécurité pour un Δt donné : l'indice de sécurité d'une structure non testée n'est multiplié que par 10^{-2} lorsqu'elle est entièrement testée (pour $P_D = 0,9$).

On peut alors se poser la question suivante:

A quoi servent les tests ?

Nous allons y répondre au paragraphe suivant.

III.7.3) Influence des tests fonctionnels.

La valeur de Δt correspond à la latence de propagation de la première panne à la sortie du microprocesseur (soit donc à l'entrée du comparateur).

Cette latence correspond à l'intervalle de temps séparant deux émissions de signaux de sortie lorsque la panne se produit sur une partie du microprocesseur effectivement utilisée pour élaborer les vecteurs de sortie.

Mais, lorsque la panne se produit sur une partie rarement utilisée, cette latence devient extrèmement grande :

par exemple, une panne affectant une zone de la ROM qui n'est explorée que par des procédures d'urgence aura une latence correspondant à la période de ces procédures (situations relativement peu fréquentes).

L'introduction de procédures de test permet de vérifier le bon fonctionnement du microprocesseur que les zones soient fréquemment ou rarement utilisées.

La latence de propagation des pannes en sortie sera donc, au maximum, égale à la période des tests. Dans l'exemple cité précédemment, la panne sur la ROM sera détectée lors de la cheksum durant la période de test qui suit son apparition.

III.8) Conclusion .

Pour notre procédé :

- $\lambda = 10^{-4}/h$ $\quad C = 0,95$ $\quad P_D = 0,90$

- la période d'un test est de 650 ms soit $\Delta t = 1,8\ 10^{-4}$ h

- le comparateur est en sécurité intrinséque $\lambda_c = 0$

- l'influence du taux de défaillance des contrôleurs est négligeable

L'indice de sécurité ou taux horaire de défaillance catastrophique est égal à

$$I_S = 7,6\ 10^{-14}/h$$

La valeur de l'indice de sécurité (ou taux horaire de défaillance catastrophique) donne entière satisfaction.

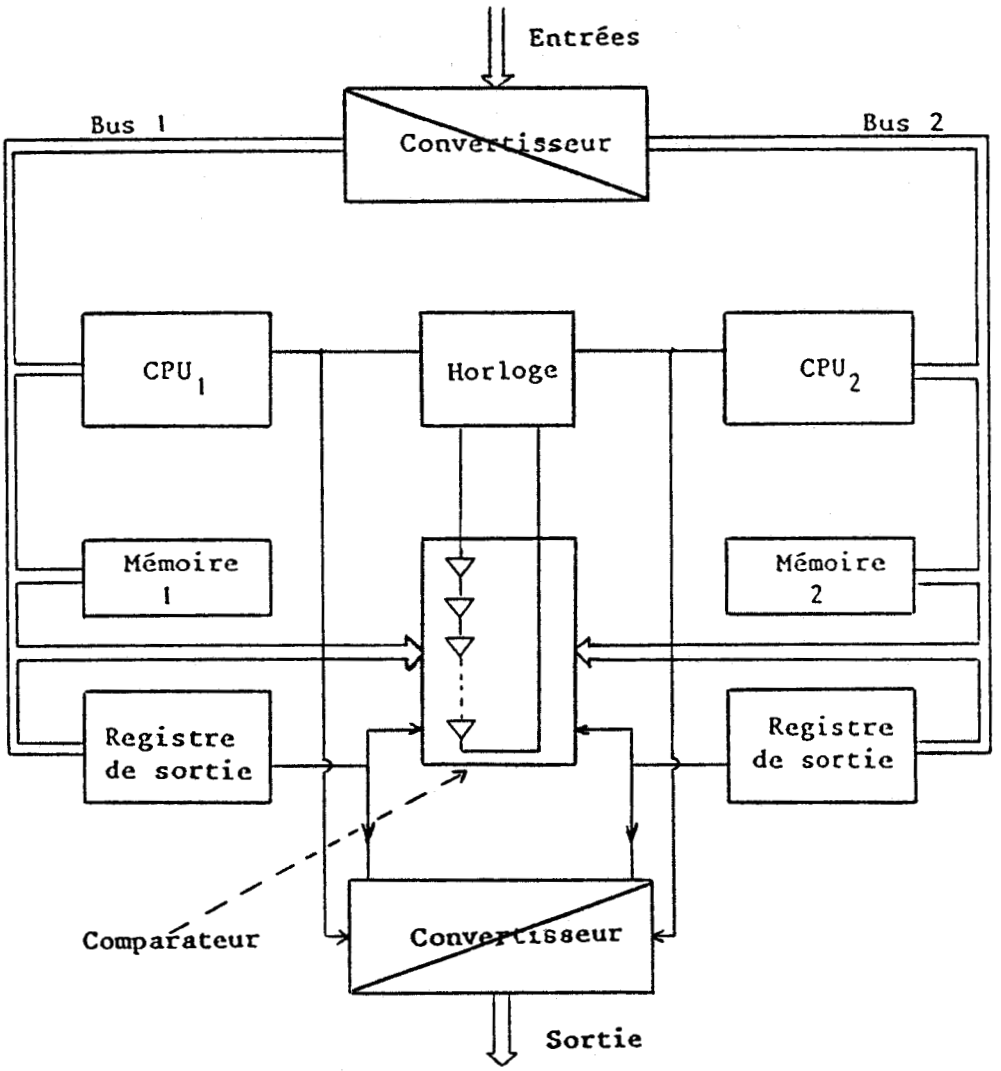
Cette structure biprocesseur peut donc être utilisée pour des applications de sécurité.

III) COMPARAISON AVEC UN SYSTEME EXISTANT.

Nous allons comparer la structure que nous venons d'étudier avec un système existant:

Le système SIMIS utilisé dans le système "PUSH" (Prozesz rechnergesteuertes U-Bahn Automatisierung - System Hambourg) de la société SIEMENS pour le pilotage automatique du métro de Hambourg.

Les données sur lesquelles nous allons fonder notre comparaison ainsi que le schéma de principe présenté ci-dessous sont tirés de REF.DAV.



Détaillons un peu ce système:

- il est basé sur l'utilisation de deux microprocesseurs 8080 de INTEL en redondance.

- Les deux chaînes sont indépendantes (comme notre système)

- des tests cycliques sur chaque unité sont exécutés périodiquement toutes les dix minutes, la durée d'un test étant de 2,5 minutes (25% du temps est consacré aux tests).

Ces tests permettent le contrôle de la mémoire, des registres, du bus et des instructions.

La méthode de test n'est pas détaillée mais les éléments testés sont ceux que nous avons testés également.

- pour effectuer la comparaison nous considérons que les tests couvrent la totalité des pannes du microprocesseur soit $\zeta = 1$.

- le test se déroule toutes les 10 minutes, nous considérons donc que Δt = 10 mn = 1/6 h.

- Le taux de défaillance horaire du 8080 est lui aussi considéré comme égal à 10^{-4}/h.

La différence essentielle entre les deux systèmes réside dans leur mode de commande:

* notre système est destiné à la commande de processus lents et continus. La proportion de temps consacré aux tests est importante mais les tests sont morcellés

* le système SIMIS est destiné à la commande de processus plus rapides possédant des périodes d'oisiveté importantes puisque le test est effectué en 2,5 minutes toutes les 10 minutes.

La comparaison des deux systèmes est résumée dans le tableau suivant.

Valeurs numériques considérées	notre système	le système PUSH
taux de défaillance du microprocesseur λ	10^{-4}/h	10^{-4}/h
taux de couverture de pannes C	0,95	1
probabilité temporelle de détection de pannes P_D	0,90	0,25
taux de défaillance des contrôleurs	négligeable	négligeable
taux de défaillance du comparateur λ_c	négligeable	négligeable
latence de propagation Δt	$1,8\ 10^{-4}$ h	0,17 h
taux horaire de défaillance catastrophique $I_S(t)$	$7,6\ 10^{-14}$/h	$1,8\ 10^{-9}$/h

Tableau récapitulatif des valeurs numériques de chaque paramètre considéré pour effectuer la comparaison

On remarque que le système que nous venons d'étudier possède un taux horaire de défaillance catastrophique nettement inférieur à un système mis en service.

Il est bien évident qu'une étude approfondie, au cas par cas, des pannes multiples, peut réduire la valeur trouvée précédemmnt puisque nous avons considéré que toute panne multiple est dangereuse alors qu'elle ne l'est que si les sorties produites sont identiques.

Conclusion:

La structure proposée est donc largement compétitive et donne entière satisfaction quant aux chiffres obtenus.

CONCLUSION GENERALE

CONCLUSION

L'étude qui vient d'être faite montre les difficultés de l'introduction des circuits intégrés dans les systèmes nécessitant une grande sécurité.

L'enjeu économique et l'évolution très rapide des technologies et des processus de fabrication imposent aux constructeurs de garder une certaine discrétion sur les schémas internes de ces circuits. Les microprocesseurs sont vus sous un aspect "boite noire" et ceci ne facilite pas les études de sécurité: les modes de panne et surtout les conséquences des pannes internes sur les sorties sont mal connus.

Il s'avère nécessaire d'avoir recours à des tests périodiques du circuit afin d'activer successivement toutes les composantes du microprocesseur et de propager les défauts vers les sorties afin de permettre leur détection.

Il faut ensuite chiffrer les résultats obtenus par ces tests, c'est à dire déterminer quelle est la proportion de pannes détectées (taux de couverture de pannes) et faire ainsi une évaluation probabiliste de la sécurité.

Le lecteur a pu s'apercevoir, dans la deuxième partie, que la détermination du taux de couverture de pannes n'était pas simple lorsque la structure interne du circuit était mal connue et que, dans ces conditions, il était nécessaire de faire une évaluation pessimiste afin de chiffrer le taux horaire de pannes contraire à la sécurité.

Après avoir déterminé l'indice de sécurité de la structure étudiée, nous avons envisagé l'étude d'une version redondante de notre système.

Le calcul de l'indice de sécurité de cette nouvelle structure nous a permis de trouver les conditions dans lesquelles la redondance était efficace: taux de défaillance du comparateur négligeable, faible latence de propagation de pannes, utilité des tests fonctionnels. Nous avons vu que la redondance améliorait nettement la sécurité si la latence de propagation était faible et que celle-ci était diminuée par des tests fonctionnels qui exploraient périodiquement les zones du processeur rarement activées par le processus.

Compte-tenu des chiffres obtenus, nous pouvons conclure que la version d'origine du procédé étudié n'est pas entièrement satisfaisante mais que la version redondante permet d'atteindre les valeurs exigées pour une application de sécurité; ceci a été confirmé par une comparaison avec un système existant.

REFERENCES BIBLIOGRAPHIQUES

/BAR/ BARANOWSKI F
"Etude et conception de contrôleurs de sécurité en logique dynamisée. Evaluation probabiliste de l'insécurité résiduelle".
Thèse de doctorat USTL Lille Décembre 1987.

/BEL1/ BELLON C
"Le test fonctionnel de circuits intégrés complexes"
Thèse d'état INPG Grenoble Octobre 1983.

/BEL2/ BELLON C. KOLOKITHAS E. VELAZCO R.
"Le système GAPT :une chaine de test pour microprocesseurs"
Revue Onde Electrique Vol 65 N° 6 Novembre 1985.

/BIE/ BIED CHARRETON D
"Etude de faisabilité d'un microprocesseur autotestable"
Rapport INRETS CRESTA N° 17 Décembre 1986.

/BRA/ BRAHME D. ABRAHAM J A.
"Functional testing of microprocessors"
IEEE Transactions on computers Vol C33 N° 6 Juin 1984.

/CAS/ CASPI P. PILAUD E.
"Conception et évaluation descendante d'un système à haute sécurité"
4ème Colloque international sur la fiabilité et la maintenabilité. France 1984.

/COU/ COURTOIS B.
"Test et LSI"
Thèse d'état INPG Grenoble Juin 1982.

/DAV/ DAVID Y.
Développement et homologation de deux systèmes de conduite de métros en Allemagne Fédérale"
Note de lecture Revue RTS N° 11 1986

/DHA1/ DHALLUIN J F.
"Commande-contrôle de processus en sécurité. Application à la commande d'un ensemble de portes véhicule d'une rame de métro type VAL"
Thèse de Docteur ingénieur USTL Lille Décembre 1983.

/DHA2/ DHALLUIN J F. GABILLARD R. EL KOURSI E M. BARANOWSKI S.
"Présentation d'un procédé de mise en sécurité de microprocesseurs par les tests temporels et fonctionnels"
4ème Séminaire Européen sur la sécurité des systèmes
3SF Deauville Juin 1986.

/ELK/ EL KOURSI E M.
"Une méthode sûre de développement des logiciels destinés à la commande de processus en sécurité: application à la commande de portes véhicule du métro VAL"
Thèse de doctorat USTL Lille Mai 1985.

/GAB/ GABILLARD R.
"Sécurité de la transmission. Application aux systèmes de transport"
Revue Onde Electrique Vol 67 N° 3 Mai 1987.

/HSURF/ SAUCIER G. JAY C.
"Conception et utilisation d'un microprocesseur spécialisé de sécurité"
4ème Colloque international sur la fiabilité et la maintenabilité. France 1984.

/INT/ INTEL
Manuels d'utilisation du 8031:
- MCS-51 Macro-assembler user's guide 1979
- microcontroler Handbook 1984

/JOU/ JOURNEAU M.
"Méthode de l'analyse de signatures".
Journées SEE "La testabilité et le test des circuits intégrés (VLSI)." 5 - 6 Juin 1986 Evry

/LAP/ LAPRIE J C.
"Prévision de la sûreté de fonctionnement et architecture de structures numériques temps réel réparables"
Thèse d'état université P Sabatier de Toulouse Juin 1975.

/LIG/ LIGERON J C. DELAGE A.
"Modélisation de la disponibilité d'un système de transport par chaines de Markov"
Revue de Statistique Appliquée Vol XXVIII N° 3 1980

/MAH/ MAHMALGI A.
"Etude et réalisation d'un testeur de logiciel pour microcontrôleur: contribution à la mise en sécurité d'un système microprogrammé"
Thèse de doctorat USTL Lille Juillet 1987.

/MAG-C/ MAGNIEZ C.
"Commande-contrôle de processus par traitement hiérarchisé. Etude des aspects sécurité-disponibilité".
Thèse de 3ème cycle USTL Lille Décembre 1985.

/MAG-S/ MAGNIEZ S.
"Test fonctionnel de la partie opérative du microcontrôleur 8051".
DEA USTL Lille Juillet 1985.

/OUA/ OUADGHIRI A H.
"Sur les conditions d'utilisation des codes détecteurs d'erreurs dans les transmissions numériques nécessitant une sécurité quasi-absolue. Applications aux systèmes de transports et en particulier aux métros souterrains"
Thèse de doctorat USTL Lille Juin 1986.

/PIL/ PILAUD E.
"Conception et validation de systèmes informatiques à haute sûreté de fonctionnement"
Thèse de Docteur ingénieur INPG Grenoble Novembre 1982.

/ROB/ ROBACH C. SAUCIER G.
"Le test des microprocesseurs et des systèmes à microprocesseur: état de l'art et perspective"
Revue Onde Electrique Vol 61 1981.

/SACEM/
"Processeur codé ou monoprocesseur de sécurité"
Journées : "Sécurité des applications des automatismes numériques dans les transports"
Villeneuve d'Ascq Octobre 1984.

/SAU/ SAUCIER G.
"Conception d'un microprocesseur spécialisé pour la haute sécurité"
Journées : "Sécurité des applications des automatismes numériques dans les transports"
Villeneuve d'Ascq Octobre 1984.

/SEV/ SEVESTRE M.
"Validation d'un système de sécurité ferroviaire à base de microprocesseurs"
4ème Colloque international sur la fiabilité et la maintenabilité. France 1984.

/THE/ THEVENOT FOSSE P.
"Test aléatoire de microprocesseurs 8 bits. Application au Motorola 6800"
Thèse d'état INPG Grenoble Octobre 1983.

/VEL/ VELAZCO R.
"Test comportemental de microprocesseurs"
Thèse de Docteur ingénieur INPG Grenoble Mars 1982.

ANNEXES

ANNEXE 1

Liste des instructions du 8031

. MCS®-51 Instruction Set Description

ARITHMETIC OPERATIONS

Mnemonic		Description	Byte	Cyc
ADD	A,Rn	Add register to Accumulator	1	1
ADD	A,direct	Add direct byte to Accumulator	2	1
ADD	A,@Ri	Add indirect RAM to Accumulator	1	1
ADD	A,#data	Add immediate data to Accumulator	2	1
ADDC	A,Rn	Add register to Accumulator with Carry	1	1
ADDC	A,direct	Add direct byte to A with Carry flag	2	1
ADDC	A,@Ri	Add indirect RAM to A with Carry flag	1	1
ADDC	A,#data	Add immediate data to A with Carry flag	2	1
SUBB	A,Rn	Subtract register from A with Borrow	1	1
SUBB	A,direct	Subtract direct byte from A with Borrow	2	1
SUBB	A,@Ri	Subtract indirect RAM from A w/ Borrow	1	1
SUBB	A,#data	Subtract immed. data from A w/ Borrow	2	1
INC	A	Increment Accumulator	1	1
INC	Rn	Increment register	1	1
INC	direct	Increment direct byte	2	1
INC	@Ri	Increment indirect RAM	1	1
DEC	A	Decrement Accumulator	1	1
DEC	Rn	Decrement register	1	1
DEC	direct	Decrement direct byte	2	1
DEC	@Ri	Decrement indirect RAM	1	1
INC	DPTR	Increment Data Pointer	1	2
MUL	AB	Multiply A & B	1	4
DIV	AB	Divide A by B	1	4
DA	A	Decimal Adjust Accumulator	1	1

LOGICAL OPERATIONS

Mnemonic		Destination	Byte	Cyc
ANL	A,Rn	AND register to Accumulator	1	1
ANL	A,direct	AND direct byte to Accumulator	2	1
ANL	A,@Ri	AND indirect RAM to Accumulator	1	1
ANL	A,#data	AND immediate data to Accumulator	2	1
ANL	direct,A	AND Accumulator to direct byte	2	1
ANL	direct,#data	AND immediate data to direct byte	3	2
ORL	A,Rn	OR register to Accumulator	1	1
ORL	A,direct	OR direct byte to Accumulator	2	1
ORL	A,@Ri	OR indirect RAM to Accumulator	1	1
ORL	A,#data	OR immediate data to Accumulator	2	1
ORL	direct,A	OR Accumulator to direct byte	2	1
ORL	direct,#data	OR immediate data to direct byte	3	2
XRL	A,Rn	Exclusive-OR register to Accumulator	1	1
XRL	A,direct	Exclusive-OR direct byte to Accumulator	2	1
XRL	A,@Ri	Exclusive-OR indirect RAM to A	1	1
XRL	A,#data	Exclusive-OR immediate data to A	2	1
XRL	direct,A	Exclusive-OR Accumulator to direct byte	2	1
XRL	direct,#data	Exclusive-OR immediate data to direct	3	2
CLR	A	Clear Accumulator	1	1
CPL	A	Complement Accumulator	1	1
RL	A	Rotate Accumulator Left	1	1
RLC	A	Rotate A Left through the Carry flag	1	1
RR	A	Rotate Accumulator Right	1	1
RRC	A	Rotate A Right through Carry flag	1	1
SWAP	A	Swap nibbles within the Accumulator	1	1

DATA TRANSFER

Mnemonic		Description	Byte	Cyc
MOV	A,Rn	Move register to Accumulator	1	1
MOV	A,direct	Move direct byte to Accumulator	2	1
MOV	A,@Ri	Move indirect RAM to Accumulator	1	1
MOV	A,#data	Move immediate data to Accumulator	2	1
MOV	Rn,A	Move Accumulator to register	1	1
MOV	Rn,direct	Move direct byte to register	2	2
MOV	Rn,#data	Move immediate data to register	2	1
MOV	direct,A	Move Accumulator to direct byte	2	1
MOV	direct,Rn	Move register to direct byte	2	2
MOV	direct,direct	Move direct byte to direct	3	2
MOV	direct,@Ri	Move indirect RAM to direct byte	2	2
MOV	direct,#data	Move immediate data to direct byte	3	2
MOV	@Ri,A	Move Accumulator to indirect RAM	1	1
MOV	@Ri,direct	Move direct byte to indirect RAM	2	2
MOV	@Ri,#data	Move immediate data to indirect RAM	2	1
MOV	DPTR,#data16	Load Data Pointer with a 16-bit constant	3	2

DATA TRANSFER (cont.)

Mnemonic		Description	Byte	Cyc
MOVC	A,@A+DPTR	Move Code byte relative to DPTR to A	1	2
MOVC	A,@A+PC	Move Code byte relative to PC to A	1	2
MOVX	A,@Ri	Move External RAM (8-bit addr) to A	1	2
MOVX	A,@DPTR	Move External RAM (16-bit addr) to A	1	2
MOVX	@Ri,A	Move A to External RAM (8-bit addr)	1	2
MOVX	@DPTR,A	Move A to External RAM (16-bit addr)	1	2
PUSH	direct	Push direct byte onto stack	2	2
POP	direct	Pop direct byte from stack	2	2
XCH	A,Rn	Exchange register with Accumulator	1	1
XCH	A,direct	Exchange direct byte with Accumulator	2	1
XCH	A,@Ri	Exchange indirect RAM with A	1	1
XCHD	A,@Ri	Exchange low-order Digit ind. RAM w/A	1	1

BOOLEAN VARIABLE MANIPULATION

Mnemonic		Description	Byte	Cyc
CLR	C	Clear Carry flag	1	1
CLR	bit	Clear direct bit	2	1
SETB	C	Set Carry flag	1	1
SETB	bit	Set direct Bit	2	1
CPL	C	Complement Carry flag	1	1
CPL	bit	Complement direct bit	2	1
ANL	C,bit	AND direct bit to Carry flag	2	2
ANL	C,/bit	AND complement of direct bit to Carry	2	2
ORL	C,bit	OR direct bit to Carry flag	2	2
ORL	C,/bit	OR complement of direct bit to Carry	2	2
MOV	C,bit	Move direct bit to Carry flag	2	1
MOV	bit,C	Move Carry flag to direct bit	2	2

PROGRAM AND MACHINE CONTROL

Mnemonic		Description	Byte	Cyc
ACALL	addr11	Absolute Subroutine Call	2	2
LCALL	addr16	Long Subroutine Call	3	2
RET		Return from subroutine	1	2
RETI		Return from interrupt	1	2
AJMP	addr11	Absolute Jump	2	2
LJMP	addr16	Long Jump	3	2
SJMP	rel	Short Jump (relative addr)	2	2
JMP	@A+DPTR	Jump indirect relative to the DPTR	1	2
JZ	rel	Jump if Accumulator is Zero	2	2
JNZ	rel	Jump if Accumulator is Not Zero	2	2
JC	rel	Jump if Carry flag is set	2	2
JNC	rel	Jump if No Carry flag	2	2
JB	bit,rel	Jump if direct Bit set	3	2
JNB	bit,rel	Jump if direct Bit Not set	3	2
JBC	bit,rel	Jump if direct Bit is set & Clear bit	3	2
CJNE	A,direct,rel	Compare direct to A & Jump if Not Equal	3	2
CJNE	A,#data,rel	Comp. immed. to A & Jump if Not Equal	3	2
CJNE	Rn,#data,rel	Comp. immed. to reg. & Jump if Not Equal	3	2
CJNE	@Ri,#data,rel	Comp. immed. to ind. & Jump if Not Equal	3	2
DJNZ	Rn,rel	Decrement register & Jump if Not Zero	2	2
DJNZ	direct,rel	Decrement direct & Jump if Not Zero	3	2
NOP		No operation	1	1

Notes on data addressing modes:

- Rn — Working register R0-R7
- direct — 128 internal RAM locations, any I/O port, control or status register
- @Ri — Indirect internal RAM location addressed by register R0 or R1
- #data — 8-bit constant included in instruction
- #data16 — 16-bit constant included as bytes 2 & 3 of instruction
- bit — 128 software flags, any I/O pin, control or status bit

Notes on program addressing modes:

- addr16 — Destination address for LCALL & LJMP may be anywhere within the 64-Kilobyte program memory address space.
- addr11 — Destination address for ACALL & AJMP will be within the same 2-Kilobyte page of program memory as the first byte of the following instruction.
- rel — SJMP and all conditional jumps include an 8-bit offset byte. Range is +127, -128 bytes relative to first byte of the following instruction.

ANNEXE 2

LES GRAPHES D'EXECUTION ABSTRAITE

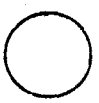
I) Obtention des graphes.

Un graphe d'exécution abstraite est un graphe biparti:

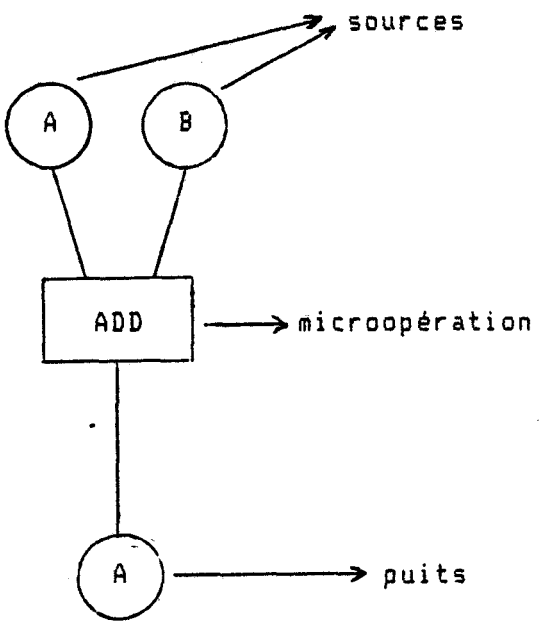
* aux noeuds du premier type sont associés les éléments de mémorisation utilisés par l'instruction, aux noeuds du deuxième type sont associés les microopérations activées par l'instruction.

* il y a un arc entre un élément de mémorisation et une microopération si la microopération traite la donnée contenue dans l'élément de mémorisation et, un arc de la microopération vers l'élément de mémorisation si la microopération envoie le résultat dans l'élément de mémorisation.

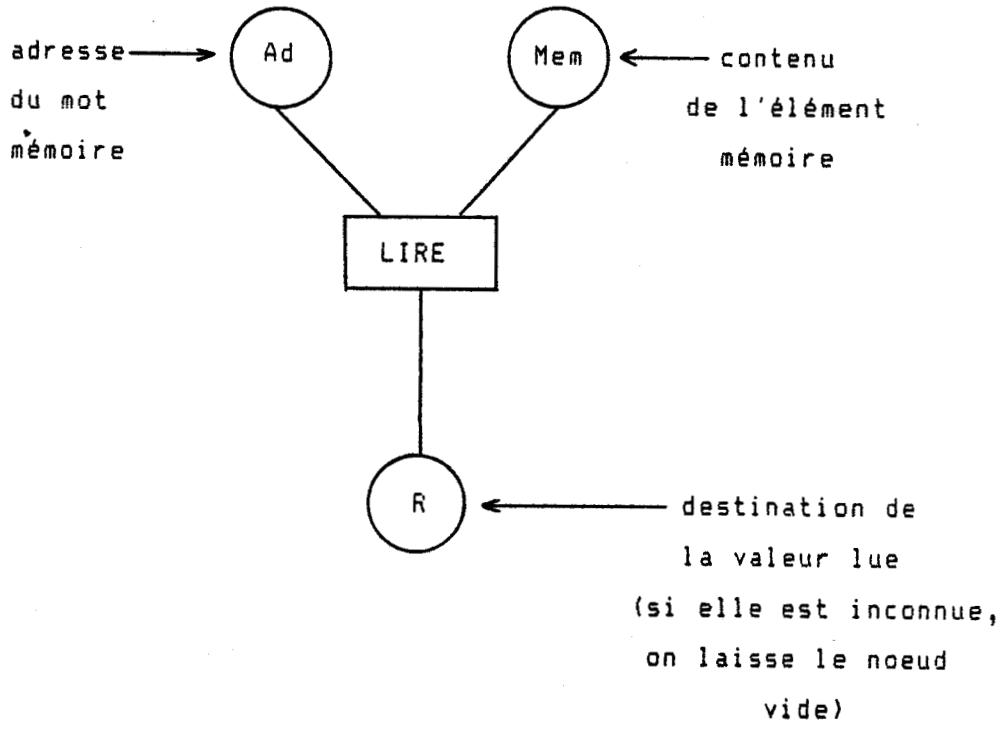
* les feuilles du graphe (sources et puits) sont toutes des noeuds du premier type.

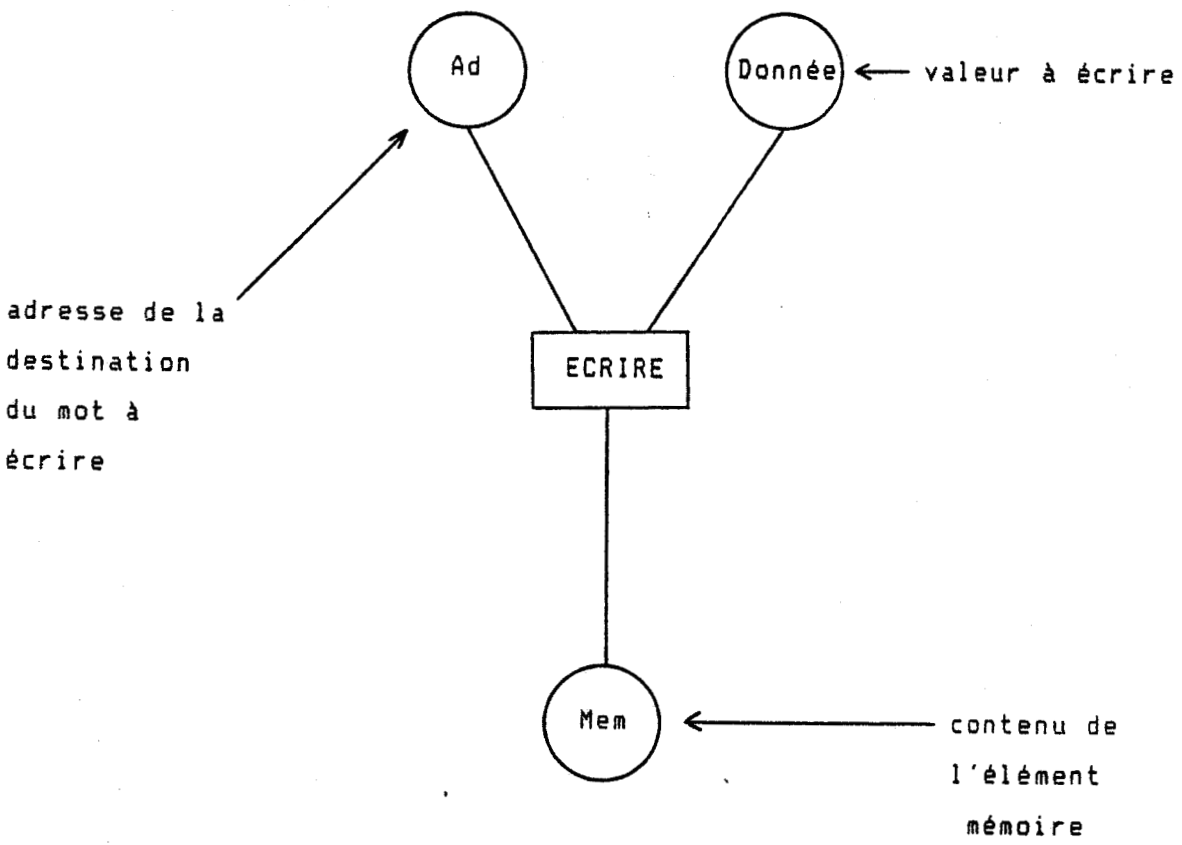
noeud du 1er type: 

noeud du 2ème type : 



convention d'écriture:





II) Analyse des instructions.

On utilise deux niveaux d'analyse:

* sur les graphes non renseignés: analyse structurelle

* sur les graphes renseignés : analyse fonctionnelle

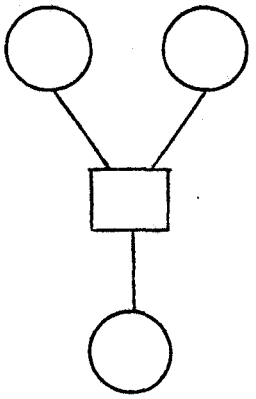
a) analyse structurelle:

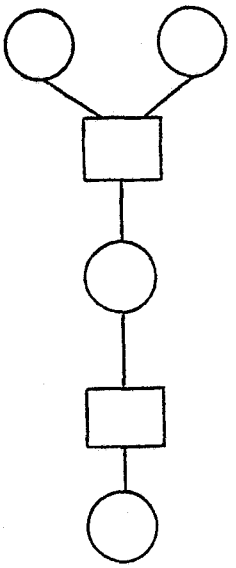
Définition :

Une instruction I1 est structurellement dominéepar une instruction I2 si le graphe I1 peut être immergé dans le graphe de I2 .

Exemple:

I1





I2

On peut en déduire l'ensemble, nommé D, des instructions non structurellement dominées.

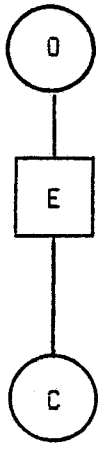
b) analyse fonctionnelle:

Cette analyse se fait sur les graphes renseignés. Ayant déterminé l'ensemble D des instructions non structurellement dominées, il faut déterminer un ensemble complémentaire I de manière à activer au moins une fois chaque élément de mémorisation et chaque microopération pour faire un test complet.

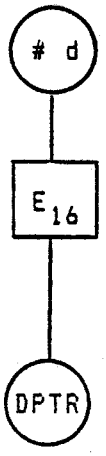
Les tests sont effectués sur l'ensemble D ∪ I.

Graphes d'exécution abstraite des instructions utilisées par le RDP

Classe 1:

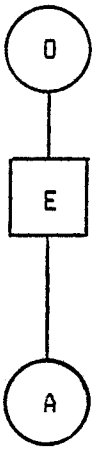


CLR C

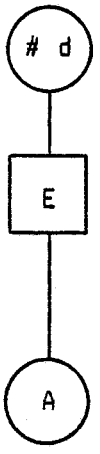


MOV DPTR, # d

RR A

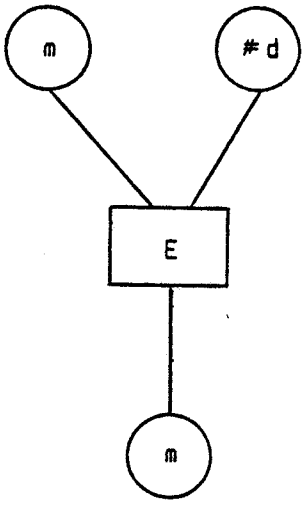


CLR A

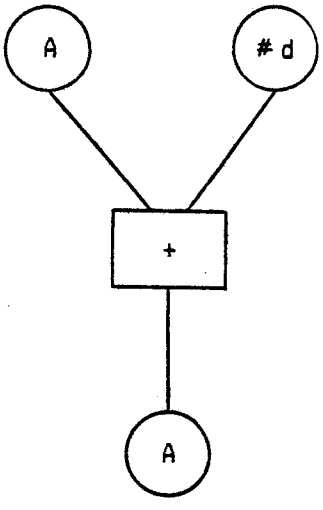


MOV A, # d

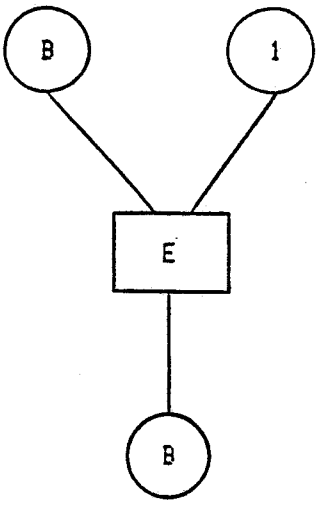
Classe 2:



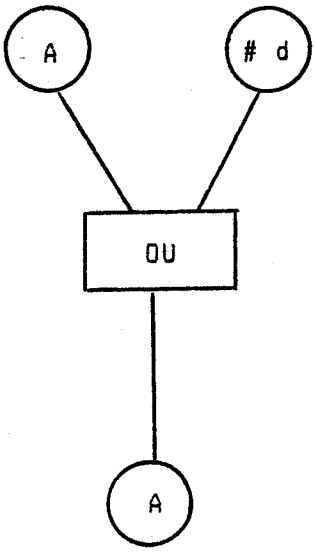
MOV direct, # d



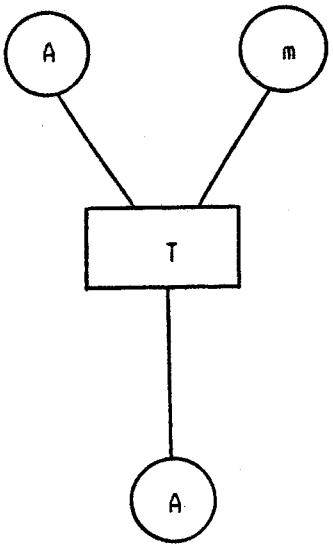
ADD A, # d



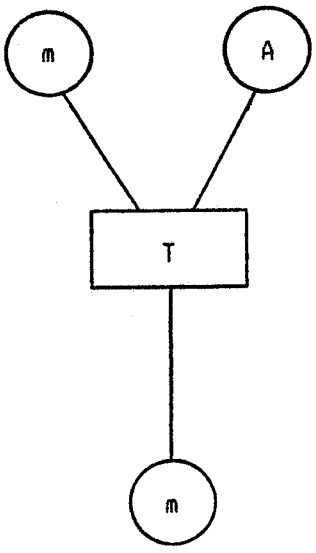
Set B bit



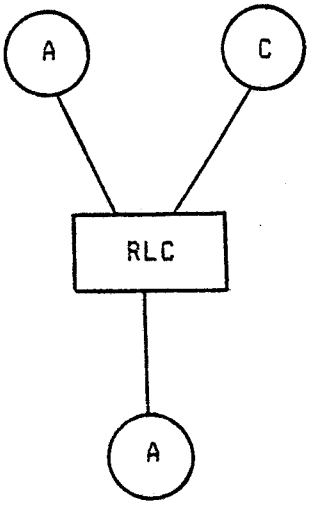
ANL A, # d



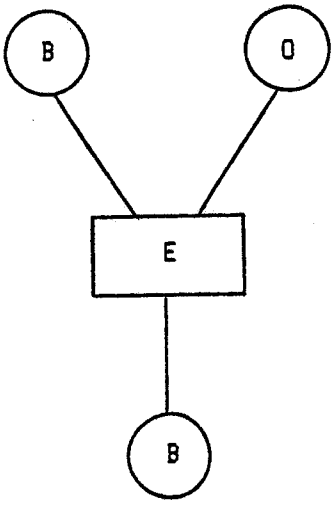
MOV A, direct



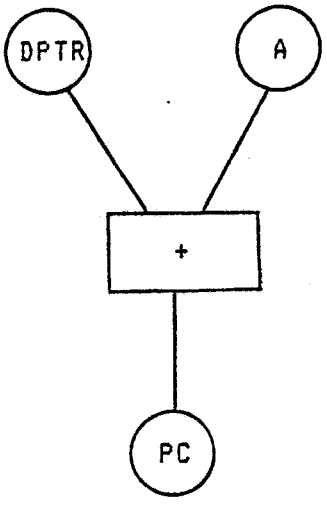
MOV direct, A



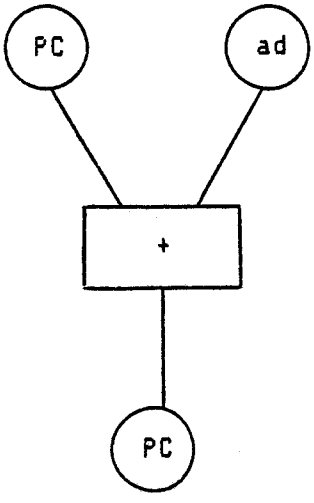
RLC A



CLR bit

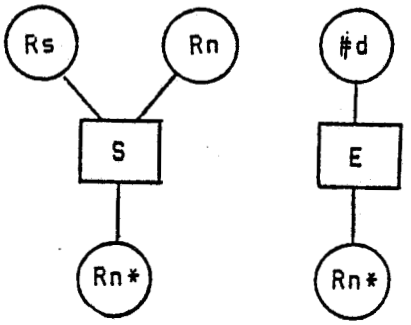


JMP @A + DPTR

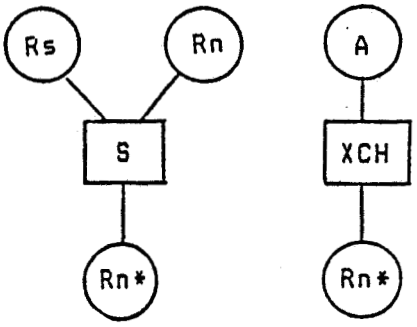


SJMP adresse

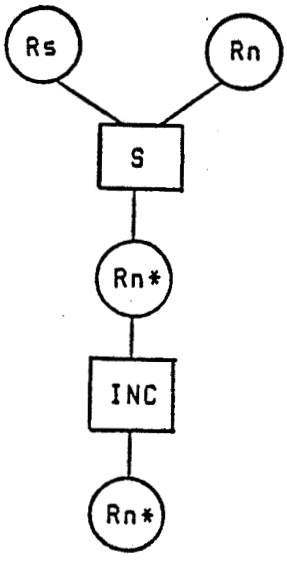
Classe 3:



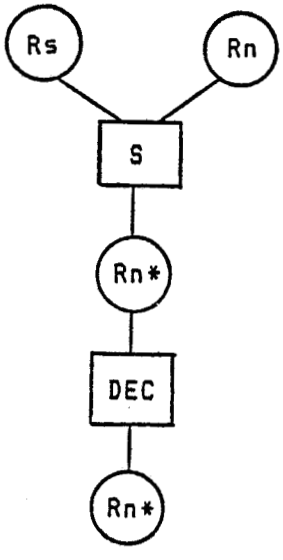
MOV Rn, #d



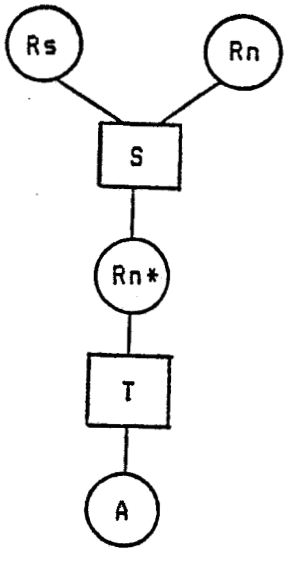
XCH A, Rn



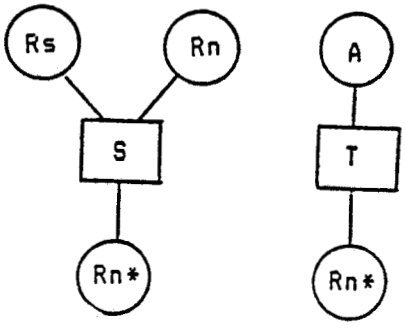
INC Rn



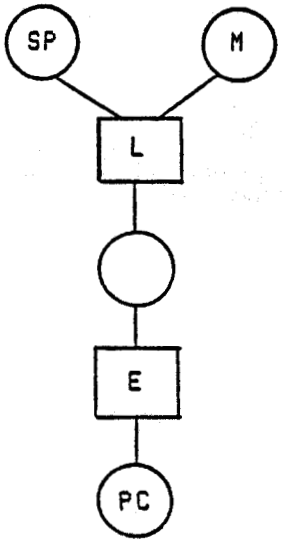
DEC Rn



MOV A, Rn

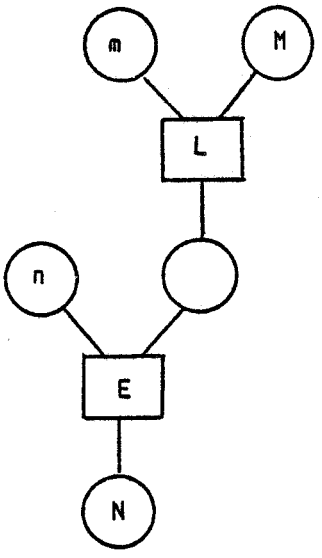


MOV Rn, A

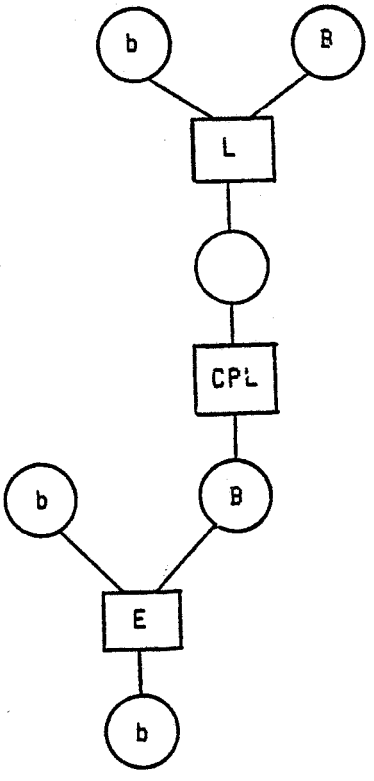


RET

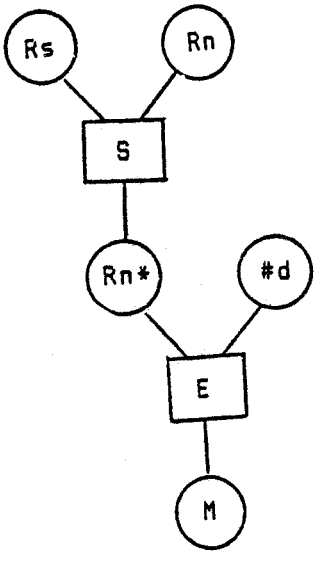
Classe 4:



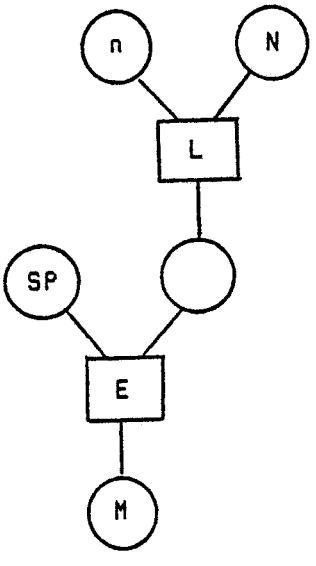
MOV n, m



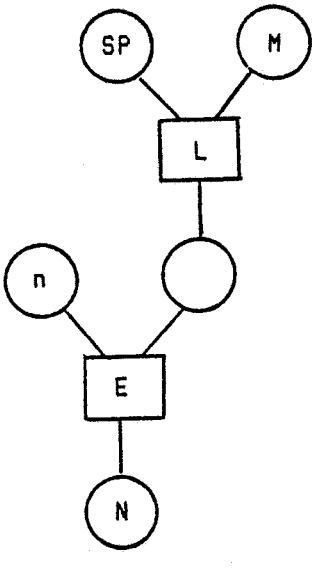
CPL bit



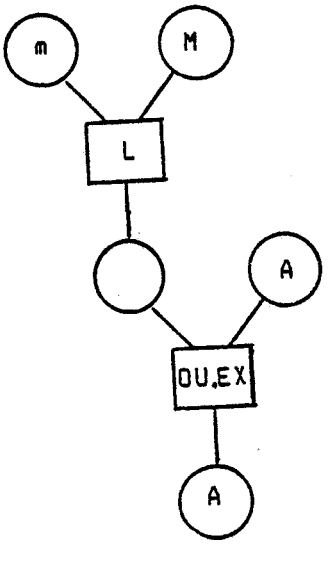
MOV @Rn, #d



PUSH N

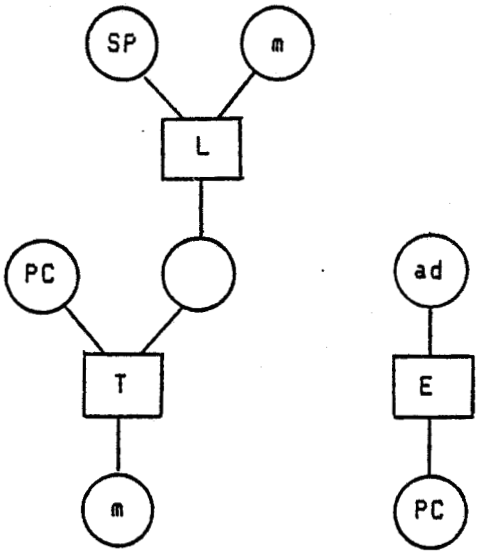


POP N

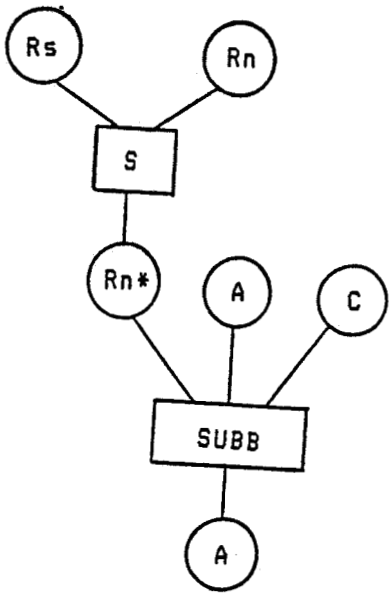


XRL A, direct

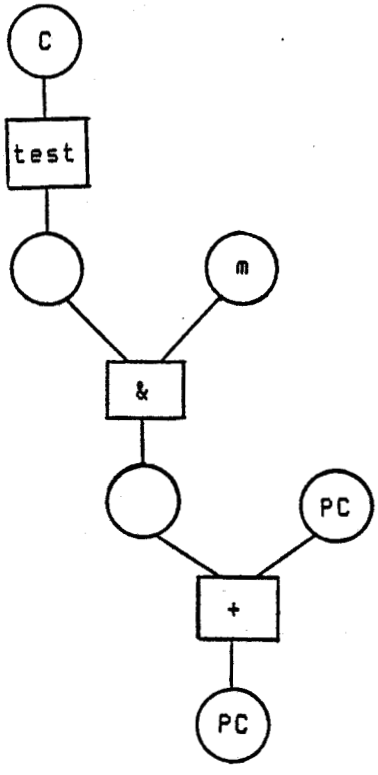
Classe 5:



CALL ad

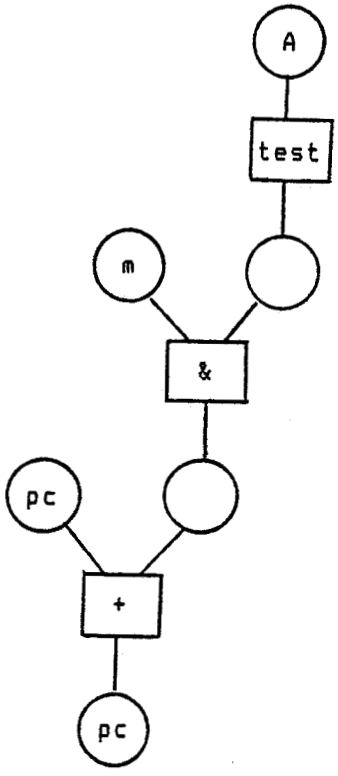


SUBB A, Rn



JC bit

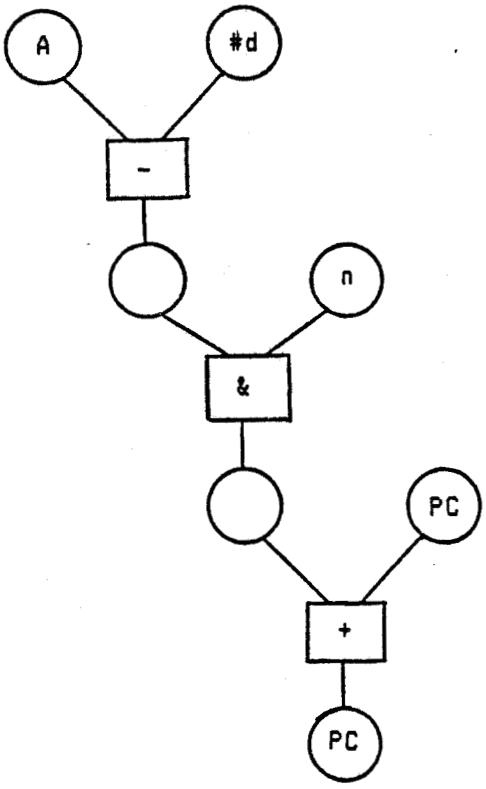
JNC bit



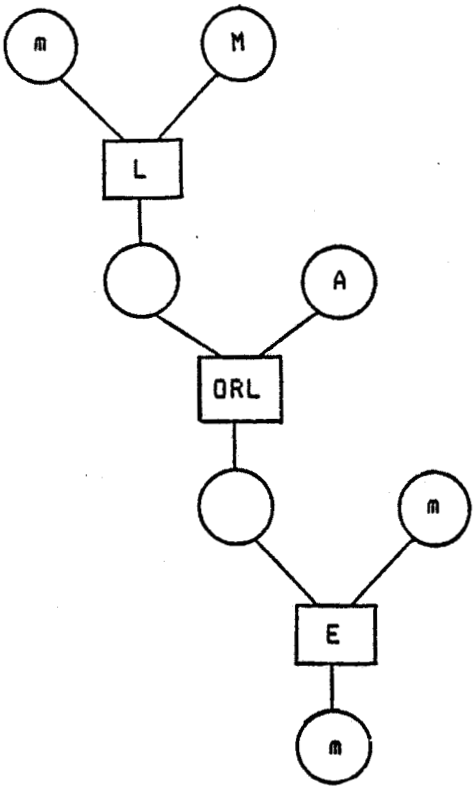
JZ m

JNZ m

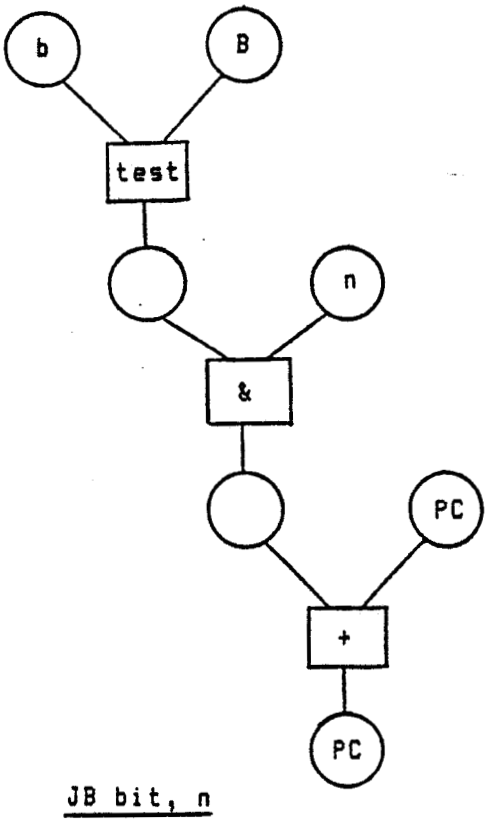
Classe 6:



CJNE A, #d,n

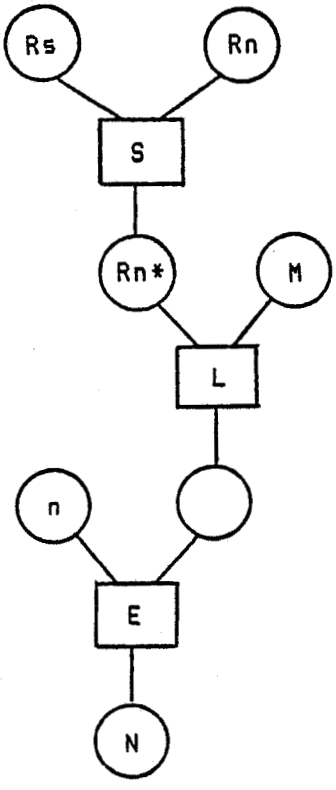


ORL direct, A

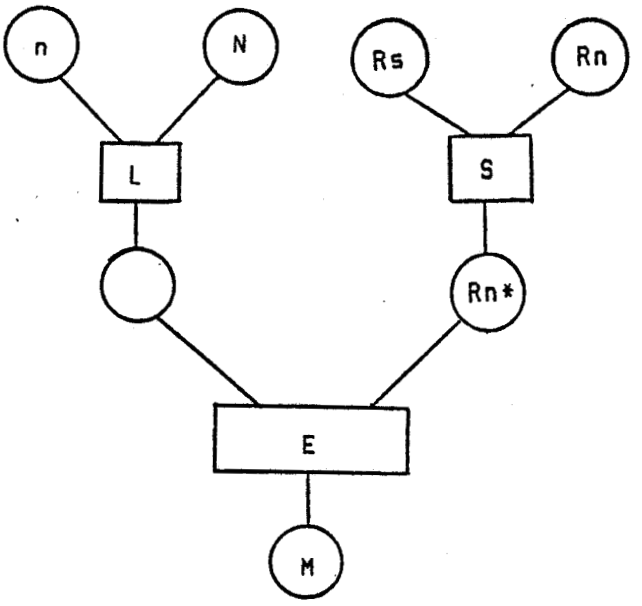


JB bit, n

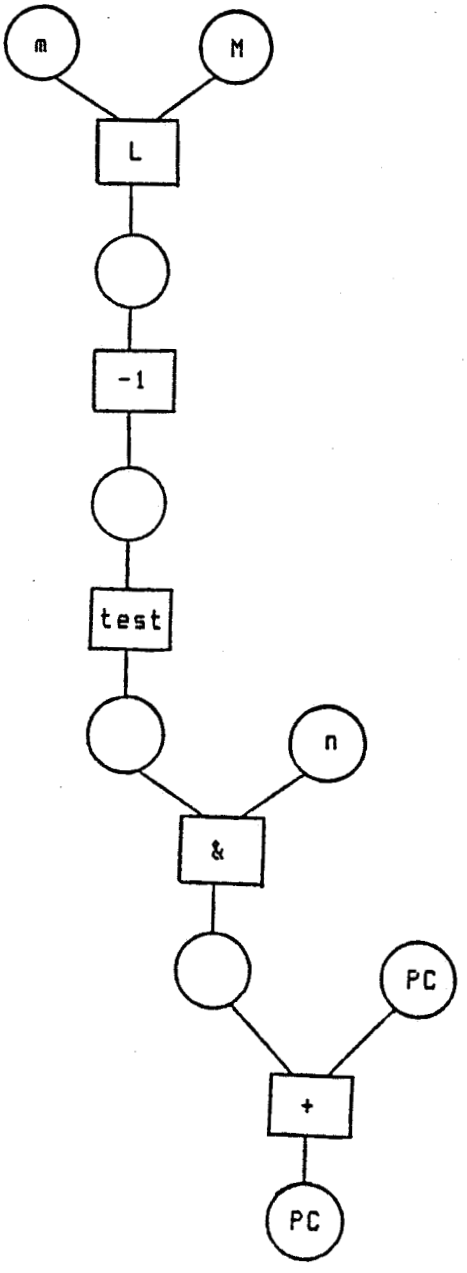
JNB bit, n



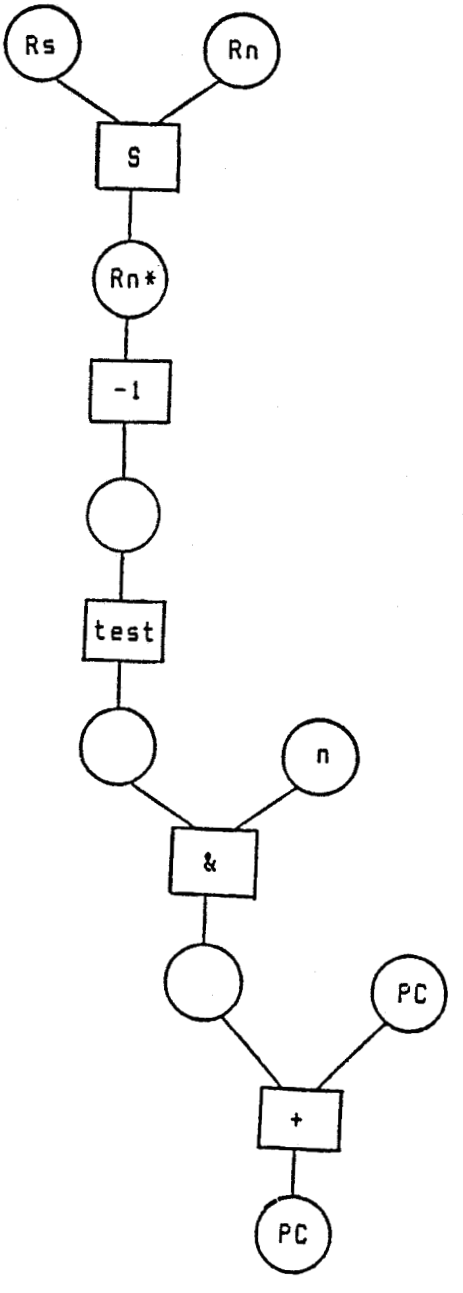
MOV n,@Rn



MOV @Rn, n

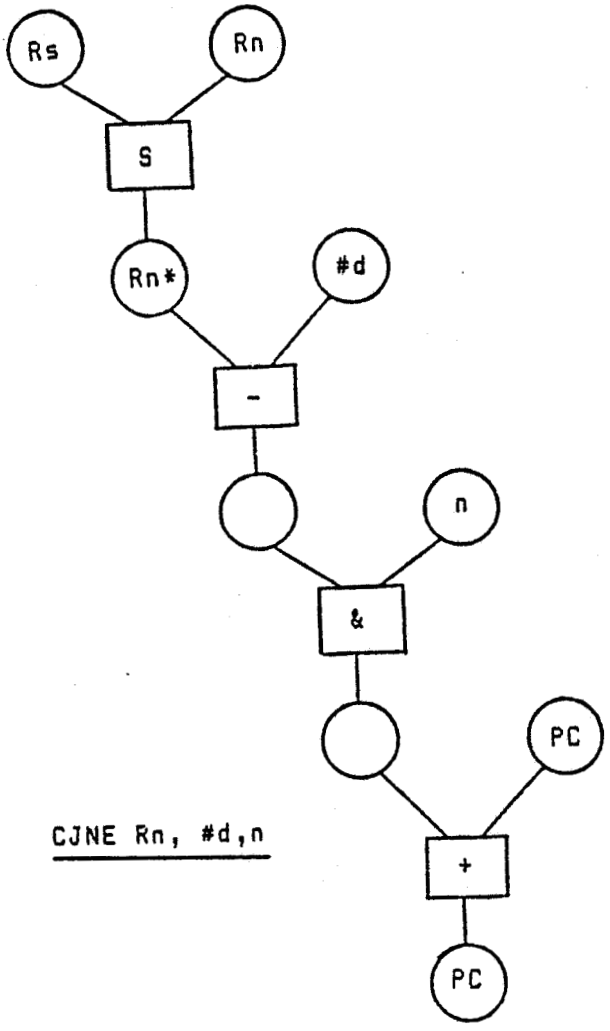


DJNZ m, n



DJNZ Rn,n

Classe 7:



CJNE Rn, #d,n

RESUME

Les microprocesseurs sont des composants complexes qui constituent les éléments sensibles de l'étude de sécurité d'un système les utilisant et nécessitant une grande sécurité de fonctionnement.

Dans une première partie, l'auteur présente une étude de la mise en sécurité d'un microcontroleur (8031) afin de l'intégrer dans une application de sécurité. Celle-ci est basée sur l'émission périodique d'un signal et l'exécution en ligne de procédures de test fonctionnel qui, en cas de pannes, permettent l'inhibition des commandes émises par le microprocesseur.

Une méthode particulière de détermination des vecteurs de test a permis de réduire leur nombre tout en assurant un test complet du circuit.

Dans une seconde partie, l'auteur propose une méthode de détermination du taux de couverture de pannes d'un microprocesseur et l'applique au système étudié afin de chiffrer la sécurité.

La détermination du taux d'insécurité résiduelle est réalisée de façon probabiliste à l'aide de graphes de Markov. L'étude paramétrée de la sécurité d'une version redondante du système précédent a permis de quantifier l'influence du taux de défaillance horaire du comparateur et celle de la période des tests sur le taux horaire de défaillance catastrophique d'un système redondant.

MOTS-CLES : MICROPROCESSEUR _ SECURITE _ DETECTION DE PANNES _ TEST FONCTIONNEL _ TAUX DE COUVERTURE DE PANNES _ STRUCTURE REDONDANTE _ GRAPHES DE MARKOV.