Cohérence de copies multiples avec latence de détection d'erreur et test fonctionnel de micro-processeurs

Carlos Postigo

THESE

présentée à

l'Institut National Polytechnique de Grenoble

pour obtenir le grade de

DOCTEUR DE 3ème Cycle
«Informatique»

par

Carlos POSTIGO

COHERENCE DE COPIES MULTIPLES AVEC LATENCE DE DETECTION D'ERREUR ET TEST FONCTIONNEL DE MICROPROCESSEURS

Thèse soutenue le 14 juin 1983 devant la commission d'examen.

L. BOLLIET	Président
J.P. BANATRE	
A. COSTE	
B. COURTOIS	Examinateurs
A. MEINGUSS	
J.L. PAUL	

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Année universitaire 1982-1983

Président de l'Université : D. BLOCH

Vice-Président : René CARRE
Hervé CHERADAME
Marcel IVANES

PROFESSEURS DES UNIVERSITES :

ANCEAU François	E.N.S.I.M.A.G.
BARRAUD Alain	E.N.S.I.E.G.
BAUDELET Bernard	E.N.S.I.E.G.
BESSON Jean	E.N.S.E.E.G.
BLIMAN Samuel	E.N.S.E.R.G.
BLOCH Daniel	E.N.S.I.E.G.
BOIS Philippe	E.N.S.H.G.
BONNETAIN Lucien	E.N.S.E.E.G.
BONNIER Etienne	E.N.S.E.E.G.
BOUVARD Maurice	E.N.S.H.G.
BRISSONNEAU Pierre	E.N.S.I.E.G.
BUYLE BODIN Maurice	E.N.S.E.R.G.
CAVAIGNAC Jean-François	E.N.S.I.E.G.
CHARTIER Germain	E.N.S.I.E.G.
CHENEVIER Pierre	E.N.S.E.R.G.
CHERADAME Hervé	U.E.R.M.C.P.P.
CHERUY Arlette	E.N.S.I.E.G.
CHIAVERINA Jean	U.E.R.M.C.P.P.
COHEN Joseph	E.N.S.E.R.G.
COUMES André	E.N.S.E.R.G.
DURAND Francis	E.N.S.E.E.G.
DURAND Jean-Louis	E.N.S.I.E.G.
FELICI Noël	E.N.S.I.E.G.
FOULARD Claude	E.N.S.I.E.G.
GENTIL Pierre	E.N.S.E.R.G.
GUERIN Bernard	E.N.S.E.R.G.
GUYOT Pierre	E.N.S.E.E.G.
IVANES Marcel	E.N.S.I.E.G.
JAUSSAUD Pierre	E.N.S.I.E.G.
JOUBERT Jean-Claude	E.N.S.I.E.G.
JOURDAIN Geneviève	E.N.S.I.E.G.
LACOUME Jean-Louis	E.N.S.I.E.G.
LATOMBE Jean-Claude	E.N.S.I.M.A.G.

.../...

LESSIEUR Marcel	E.N.S.H.G.
LESPINARD Georges	E.N.S.H.G.
LONGEQUEUE Jean-Pierre	E.N.S.I.E.G.
MAZARE Guy	E.N.S.I.M.A.G.
MOREAU René	E.N.S.H.G.
MORET Roger	E.N.S.I.E.G.
MOSSIERE Jacques	E.N.S.I.M.A.G.
PARIAUD Jean-Charles	E.N.S.E.E.G.
PAUTHENET René	E.N.S.I.E.G.
PERRET René	E.N.S.I.E.G.
PERRET Robert	E.N.S.I.E.G.
PIAU Jean-Michel	E.N.S.H.G.
POLOUJADOFF Michel	E.N.S.I.E.G.
POUPOT Christian	E.N.S.E.R.G.
RAMEAU Jean-Jacques	E.N.S.E.E.G.
RENAUD Maurice	U.E.R.M.C.P.P.
ROBERT André	U.E.R.M.C.P.P.
ROBERT François	E.N.S.I.M.A.G.
SABONNADIERE Jean-Claude	E.N.S.I.E.G.
SAUCIER Gabrielle	E.N.S.I.M.A.G.
SCHLENKER Claire	E.N.S.I.E.G.
SCHLENKER Michel	E.N.S.I.E.G.
SERMET Pierre	E.N.S.E.R.G.
SILVY Jacques	U.E.R.M.C.P.P.
SOHM Jean-Claude	E.N.S.E.E.G.
SOUQUET Jean-Louis	E.N.S.E.E.G.
VEILLON Gérard	E.N.S.I.M.A.G.
ZADWORNY François	E.N.S.E.R.G.

PROFESSEURS ASSOCIES

BASTIN Georges	E.N.S.H.G.
BERRIL John	E.N.S.H.G.
CARREAU Pierre	E.N.S.H.G.
GANDINI Alessandro	U.E.R.M.C.P.P.
HAYASHI Hirashi	E.N.S.I.E.G.

PROFESSEURS UNIVERSITE DES SCIENCES SOCIALES (Grenoble II)

BOLLIET Louis
Chatelin Françoise

PROFESSEURS E.N.S. Mines de Saint-Etienne

RIEU Jean
SOUSTELLE Michel

CHERCHEURS DU C.N.R.S.

FRUCHART Robert	Directeur de Recherche
VACHAUD Georges	Directeur de Recherche

.../...

ALLIBERT Michel	Maître de Recherche
ANSARA Ibrahim	Maître de Recherche
ARMAND Michel	Maître de Recherche
BINDER Gilbert	
CARRE René	Maître de Recherche
DAVID René	Maître de Recherche
DEPORTES Jacques	
DRIOLE Jean	Maître de Recherche
GIGNOUX Damien	
GIVORD Dominique	
GUELIN Pierre	
HOPFINGER Emil	Maître de Recherche
JOUD Jean-Charles	Maître de Recherche
KAMARINOS Georges	Maître de Recherche
KLEITZ Michel	Maître de Recherche
LANDAU Ioan-Dore	Maître de Recherche
LASJAUNIAS J.C.	
MERMET Jean	Maître de Recherche
MUNIER Jacques	Maître de Recherche
PIAU Monique	
PORTESEIL Jean-Louis	
THOLENCE Jean-Louis	
VERDILLON André	

CHERCHEURS du MINISTERE de la RECHERCHE et de la TECHNOLOGIRE (Directeurs et Maîtres de Recherches, ENS Mines de St. Etienne)

LESBATS Pierre	Directeur de Recherche
BISCONDI Michel	Maître de Recherche
KOBYLANSKI André	Maître de Recherche
LE COZE Jean	Maître de Recherche
LALAUZE René	Maître de Recherche
LANCELOT Francis	Maître de Recherche
THEVENOT François	Maître de Recherche
TRAN MINH Canh	Maître de Recherche

PERSONNALITES HABILITEES à DIRIGER des TRAVAUX de RECHERCHE (Décision du Conseil Scientifique)

ALLIBERT Colette	E.N.S.E.E.G.
BERNARD Claude	E.N.S.E.E.G.
BONNET Rolland	E.N.S.E.E.G.
CAILLET Marcel	E.N.S.E.E.G.
CHATILLON Catherine	E.N.S.E.E.G.
CHATILLON Christian	E.N.S.E.E.G.
COULON Michel	E.N.S.E.E.G.
DIARD Jean-Paul	E.N.S.E.E.G.
EUSTAPOPOULOS Nicolas	E.N.S.E.E.G.
FOSTER Panayotis	E.N.S.E.E.G.

.../...

GALERIE Alain	E.N.S.E.E.G.
HAMMOU Abdelkader	E.N.S.E.E.G.
MALMEJAC Yves	E.N.S.E.E.G. (CENG)
MARTIN GARIN Régina	E.N.S.E.E.G.
NGUYEN TRUONG Bernadette	E.N.S.E.E.G.
RAVAINE Denis	E.N.S.E.E.G.
SAINFORT	E.N.S.E.E.G. (CENG)
SARRAZIN Pierre	E.N.S.E.E.G.
SIMON Jean-Paul	E.N.S.E.E.G.
TOUZAIN Philippe	E.N.S.E.E.G.
URBAIN Georges	E.N.S.E.E.G. (Laboratoire des ultra-réfractaires ODEILLON)
GUILHOT Bernard	E.N.S. Mines Saint Etienne
THOMAS Gérard	E.N.S. Mines Saint Etienne
DRIVER Julien	E.N.S. Mines Saint Etienne
BARIBAUD Michel	E.N.S.E.R.G.
BOREL Joseph	E.N.S.E.R.G.
CHOVET Alain	E.N.S.E.R.G.
CHEHIKIAN Alain	E.N.S.E.R.G.
DOLMAZON Jean-Marc	E.N.S.E.R.G.
HERAULT Jeanny	E.N.S.E.R.G.
MONLLOR Christian	E.N.S.E.R.G.
BORNARD Guy	E.N.S.I.E.G.
DESCHIZEAU Pierre	E.N.S.I.E.G.
GLANGEAUD François	E.N.S.I.E.G.
KOFMAN Walter	E.N.S.I.E.G.
LEJEUNE Gérard	E.N.S.I.E.G.
MAZUER Jean	E.N.S.I.E.G.
PERARD Jacques	E.N.S.I.E.G.
REINISCH Raymond	E.N.S.I.E.G.
ALEMANY Antoine	E.N.S.H.G.
BOIS Daniel	E.N.S.H.G.
DARVE Félix	E.N.S.H.G.
MICHEL Jean-Marie	E.N.S.H.G.
OBLED Charles	E.N.S.H.G.
ROWE Alain	E.N.S.H.G.
VAUCLIN Michel	E.N.S.H.G.
WACK Bernard	E.N.S.H.G.
BERT Didier	E.N.S.I.M.A.G.
CALMET Jacques	E.N.S.I.M.A.G.
COURTIN Jacques	E.N.S.I.M.A.G.
COURTOIS Bernard	E.N.S.I.M.A.G.
DELLA DORA Jean	E.N.S.I.M.A.G.
FONLUPT Jean	E.N.S.I.M.A.G.
SIFAKIS Joseph	E.N.S.I.M.A.G.
CHARUEL Robert	U.E.R.M.C.P.P.
CADET Jean	C.E.N.G.
COEURE Philippe	C.E.N.G. (LETI)

.../...

DELHAYE Jean-Marc	C.E.N.G. (STT)
DUPUY Michel	C.E.N.G. (LETI)
JOUVE Hubert	C.E.N.G. (LETI)
NICOLAU Yvan	C.E.N.G. (LETI)
NIFENECKER Hervé	C.E.N.G.
PERROUD Paul	C.E.N.G.
PEUZIN Jean-Claude	C.E.N.G. (LETI)
TAIEB Maurice	C.E.N.G.
VINCENDON Marc	C.E.N.G.

LABORATOIRES EXTERIEURS

DEMOULIN Eric	C.N.E.T.
DEVINE	C.N.E.T. (R.A.B.)
GERBER Roland	C.N.E.T.
MERCKEL Gérard	C.N.E.T.
PAULEAU Yves	C.N.E.T.
GAUBERT C.	I.N.S.A. Lyon

REMERCIEMENTS

Je tiens à remercier particulièrement M. L. BOLLIET, Professeur à l'Université de Grenoble, qui m'a fait l'honneur de présider le jury de cette thèse et qui a examiné avec une attention toute particulière ce manuscrit.

M. B. COURTOIS, Chargé de recherche CNRS, trouvera ici l'expression de ma gratitude pour avoir dirigé cette thèse. Il est à l'origine des idées de base de cette étude; ses critiques et encouragements ont largement contribué à améliorer ce travail.

Je remercie également,

M. J. P. BANATRE, Ingénieur de recherche INRIA, qui a bien voulu lire et critiquer cette thèse, tâche qu'il a fait avec compétence et efficacité. Ses remarques pertinentes m'ont aidé à améliorer la rédaction de cette thèse;

M. A. COSTES, Professeur à l'Institut National Polytechnique de Toulouse, qui a bien voulu participer au jury de cette thèse malgré ses nombreuses occupations et qui a toujours manifesté un vif intérêt à nos recherches sûr la sûreté de fonctionnement. Qu'il soit remercié pour l'attention et le soutien qu'il a accordés à ce travail;

M. A. MEINGUSS, Ingénieur à la Société Electronique Serge Dassault, qui a bien voulu juger ce travail et qui l'a examiné avec une attention toute particulière. Ses critiques constructives ont largement contribué à améliorer le texte initial;

M. J. L. PAUL, Ingénieur à la Société CII-ALCATEL, avec qui j'ai pu discuter de cette étude, pour avoir accepté de participer au jury et pour l'intérêt amical qu'il témoigne à nos travaux;

Ces remerciements ne seraient pas complets si je n'associais pas les membres de l'Equipe de Recherche en Architecture des Ordinateurs, notamment:

M. F. ANCEAU, qui a bien voulu m'accepter au sein de l'equipe qu'il dirige;

M. J. P. SCHOELLKOPF, grâce à qui le système d'édition de texte a pu être utilisé pour cette thèse;

M. M. NICOLAIDIS, avec qui j'ai pu discuter nombre de points de cette étude.

J' associe dans mes remerciements:

Les membres du GSI MINI MICRO, spécialement M. J. P. EYNARD avec qui j'ai eu des échanges fructueux;

Mon ami, L. M. SANTANA pour ses conseils et encouragements;

Le Service de Reprographie de l'IMAG qui a assuré, avec gentillese, le tirage de ce document.

AVERTISSEMENT AU LECTEUR

Bien que le contenu total de cette thèse suive une démarche logique et concrète, il est possible d'omettre certains chapitres ou sous-chapitres sans perte de continuité. Les lecteurs qui connaissent le contexte de chaque partie peuvent lire seulement les libellés marqués avec un *.

SOMMAIRE GENERAL

Introduction à la thèse 1

Première Partie 3

- Résumé 5
- Sommaire Première Partie 7
- Première Partie 9

Seconde Partie 85

- Résumé 87
- Notations 89
- Sommaire Seconde Partie 91
- Seconde Partie 93

Introduction à la thèse

Cette thèse est constituée de deux parties bien différenciées, bien qu'elles soient concernées, toutes les deux, par la sûreté de fonctionnement de systèmes informatiques.

Diverses approches utilisées dans ce domaine sont representées de manière non-exhaustive dans le schéma de la figure 1. Elles sont décrites brièvement dans les sous-chapitres II.1 et II.2 de la première partie et dans le chapitre II de la seconde partie. Ce schéma est une synthèse des classifications trouvées dans [AVI76], [RAN78a], [AND81] et [COU82], où l'on peut reconnaître la place des deux sujets étudiés dans cette thèse.

La première partie concerne particulièrement les systèmes répartis et traite de l'inclusion de la notion de latence de détection de pannes dans des algorithmes de maintien de la cohérence d'information à copies multiples.

La seconde partie concerne le test fonctionnel hors-ligne de la Partie Contrôle du microprocesseur MOTOROLA M6800. Une méthodologie est appliquée et une adaptation est réalisée à partir des algorithmes conçus, en principe, pour le test en-ligne.

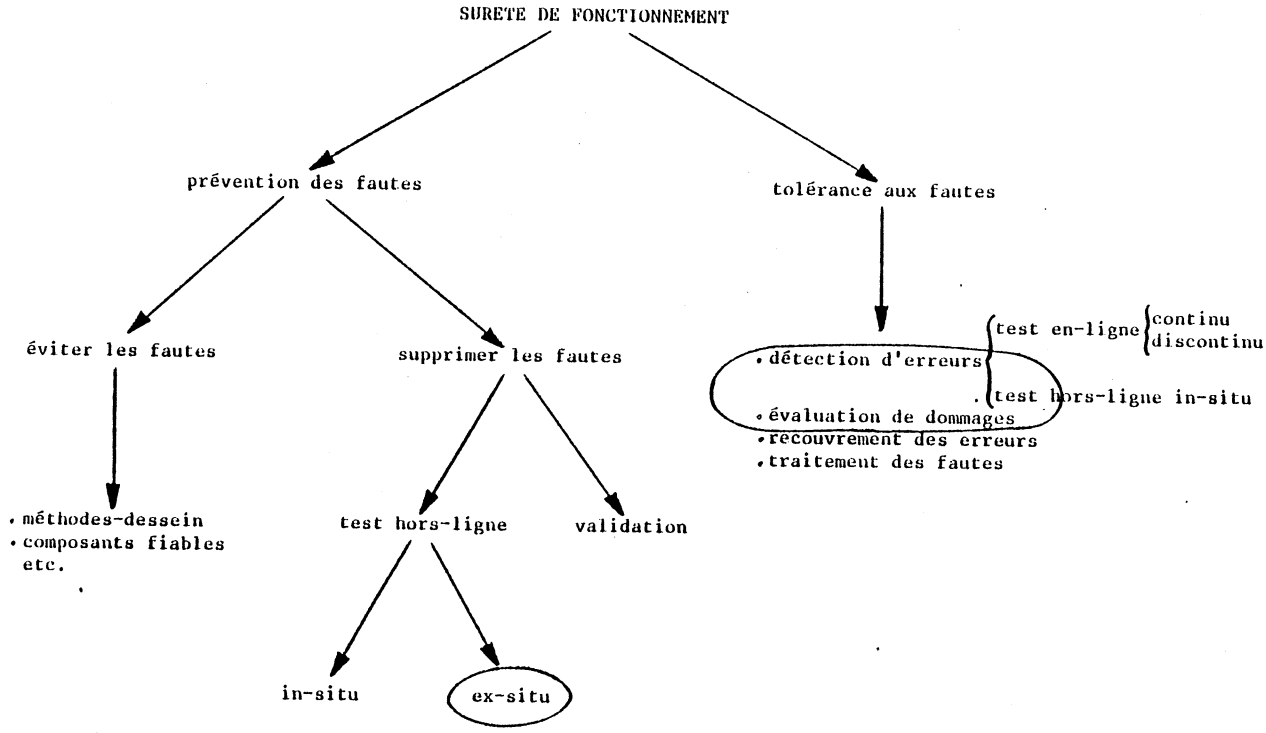


Figure 1.- Représentation (non-exhaustive) de différentes approches pour l'obtention de systèmes sûrs de fonctionnement

PREMIERE PARTIE

RESUME

Cette partie décrit une méthode de maintien de la cohérence de copies multiples d'information dans un système réparti, avec prise en compte de fautes latentes. Cette prise en compte est incluse, au moyen d'une temporisation de messages, dans la méthode de maintien de la cohérence de KANEKO et al. La méthode est ultérieurement modifiée en vue de rétablir la cohérence, détruite par cette temporisation. L'originalité de la méthode dérive du fait que le problème du mauvais fonctionnement non-détecté d'un système réparti (fautes latentes) n'a pas été jusqu'à maintenant pris en compte dans le problème du maintien de la cohérence. Une analyse comparative est établie entre cette démarche et une autre basée sur la méthode de HERMAN et VERJUS.

Sommaire Première Partie

I.- Introduction 9

*I.1.- Environnement 9

I.2.- Plan de la Première Partie 13

II.- Concepts de base 15

II.1.- La sûreté de fonctionnement 15

II.2.- La tolérance aux fautes 16

II.3.- Les systèmes répartis 16

III.- Cadre général 18

*III.1.- Le support 18

III.2.- Application 19

III.3.- Le programme de recherche 21

III.3.1.- Première phase 21

*III.3.2.- Deuxième phase 22

III.3.3.- Troisième phase 23

*IV.- Terminologie 25

*V.- Modèles de latence et de confinement 28

V.1.- Modèle de latence de détection 28

V.2.- Modèle général de confinement d'erreur 31

VI.- Horloges logiques 33

*VII.- La cohérence de copies multiples 37

VIII.- Algorithmes de cohérence 42

*IX.- Confinement et temporisation 45

*X.- Hypothèses établies 49

*XI.- Algorithme de Kaneko avec temporisation 53

XI.1.- Hypothèses supplémentaires 53

XI.2.- Inclusion de la temporisation (Partie I) 54

XI.3.- Exemple de fonctionnement sans panne 56

XI.4.- Panne contrôlée d'une station (Partie II) 60

XI.5.- Exemple de synchronisation 66

XI.6.- Panne non-contrôlée d'une station 67

*XII.- Algorithme de Herman et Verjus avec temporisation 70

XII.1.- Hypothèses supplémentaires 70

XII.2.- Inclusion de la temporisation 71
XII.3.- Panne contrôlée d'une station 72
XII.4.- Panne non-contrôlée d'une station 73
*XIII.- Analyse comparée des algorithmes 74
*XIV.- Conclusion 78
Références 80

I.-Introduction

I.1.- Environnement

Cette première partie est concernée par la prise en compte de la notion de latence de détection de fautes, dans des algorithmes de maintien de la cohérence de copies multiples d'information dans les systèmes répartis.

L'information et son état correcte constituent une des clés du bon fonctionnement de tout système informatique. Des règles ont été établies pour contrôler l'utilisation et la diffusion de l'information. Parmi ces règles, celles du contrôle d'accès, ont été mises en vigueur par le logiciel de base, au moyen de mécanismes de protection et de synchronisation. Les procédures de maintien de la cohérence font partie des mécanismes de synchronisation. Ils ont été créés, initialement, pour résoudre le problème créé par l'accès concurrent dans les cas de systèmes multi-utilisateurs (dans le domaine de Bases de Données).

Les copies multiples ont surgi, initialement, comme une solution au problème du partage de l'information. On peut trouver plusieurs raisons qui justifient l'utilisation de copies multiples de la même information: accroissement de la fiabilité, accroissement de la performance, etc. Dans le cas de systèmes répartis, ces raisons se trouvent renforcées par les avantages fournis par ce type de système par rapport aux systèmes centralisés [BAN80], [COR81], [ENS78], [KIM79], [KIM82]. On considère qu'un système réparti est celui constitué de plusieurs sites ou stations autonomes, où le seul moyen de liaison logique est l'ensemble de messages qui sont échangés. Un site, étant pris dans le sens d'hôte: système intelligent (terminal ou ordinateur, avec mémoire) utilisé pour réaliser un traitement. Dans ce type de système on peut trouver des copies de la même information sur plusieurs sites du système. A moins d'y prendre garde, cette information peut atteindre des états incohérents dûs aux erreurs

non-détectées.

L'existence de copies multiples dans les systèmes répartis a demandé une révision des algorithmes de maintien de la cohérence réalisés pour les systèmes centralisés. En effet, la duplication de l'information a introduit un nouveau type de cohérence: la cohérence mutuelle.

L'utilisation de systèmes répartis qui garantissent la continuité du bon fonctionnement, dans le service fourni (TOLERANCE AUX FAUTES), oblige ces algorithmes à considérer les problèmes de défaillance de sites, la partition du réseau, la coupure de communication, etc.. La plupart des algorithmes devéloppés (une vingtaine) ont plus ou moins pris en compte ce problème en atteignant divers degrés de robustesse en participant à la détection de fautes, à la reprise de données, etc.

Les algorithmes de maintien de la cohérence en systèmes répartis resolvent, avec des degrés d'efficacité divers et avec des méthodes différentes, le problème engendré par l'arrêt intempestif d'un site. Dans ce cas précis ils garantissent le maintien de la cohérence. Mais ils ne considèrent pas le cas de mauvais fonctionnement non-détecté d'un site, qui peut endommager la cohérence du système. Ce mauvais fonctionnement non-détecté (faute latente) d'un site étant précurseur d'un arrêt définitif du site. Généralement, on considère que tout ce qui a été fait avant l'arrêt est correct, ou on décide d'un point de reprise avant lequel tout est considéré correct.

Des nouvelles applications exigeant une fiabilité encore plus grande, nous amènent à considérer les cas de mauvais fonctionnement non-détecté. Le souci principal de cette étude, est que les algorithmes satisfassent le fonctionnement normal (sans faute) du système réparti. Il faut aussi, qu'ils le préparent à l'éventualité d'une défaillance des sites, de manière à maintenir le système en fonctionnement continu, en utilisant le maximum d'informations correctes. En plus ils doivent permettre à

un site qui était à l'arrêt, de devenir cohérent lorsqu'il devient opérationnel [SHA78].

La fiabilité du système est maintenue, généralement, par le niveau de base qui prend en charge la détection d'erreur, la reconfiguration du système, etc. Il essaie aussi, de limiter les dommages causés par la défaillance de composantes de systèmes en utilisant divers mécanismes tels que: structure modulaire [RAN78], [HOR73], opérations atomiques [LOM77], blocs de recouvrement [HOR74], etc., mais sans y arriver complétement et obligeant à l'utilisation de points de réprise.

Certains types de systèmes abordent ce problème de la tolérance aux fautes d'une manière particulière: dégradation progressive, redondance minimale, etc., tel que SKALP en [COU81], ce qui permet un certain degré de fiabilité très économique et acceptable pour certaines applications. En présence d'une faute, le système est reconfiguré en supprimant le composant défaillant et en acceptant une diminution de sa performance (ses tâches sont réparties entre les composants survivants).Ce type de système posséde des mécanismes de détection de fautes qui se produisent, généralement, avec une latence de détection.

En admettant que l'échange de messages n'est pas instantané, mais qu'il nécessite un certain temps, on observe que la connaissance qu'a chaque site, à un instant donné, de l'état du système est relativement obsolète. Ceci nous oblige à admettre l'existence de fautes latentes. En conséquence, l'utilisation que fait chaque site de messages provenant des autres sites et qui est dépendante de l'état (bon ou mauvais) du système, devra être anticipée d'une temporisation. Au bout de celle-ci, on peut dire si un message peut être ou non utilisé. On parlera alors de confinement d'erreur. La temporisation doit permettre à chaque station d'utiliser (exécuter) un message correct et de refuser un message erroné ou suspect de l'être.

En bref, dans notre approche, des méthodes de test doivent

permettre, aux sites qui constituent un système, de détecter un certain nombre de fautes qui leur sont propres, avec une latence de détection, et de les communiquer aux autres sites (cela implique pour certaines fautes d'origine matérielle, des sites multi-processeurs) avant son arrêt définitif. Il faut remarquer que cette approche se révèle utile dans la mesure où l'intervalle moyen d'arrivée des messages de mise à jour vers un site quelconque est plus petit que l'intervalle moyen de temporisation; c'est à dire que, dans le cas de défaillance d'un site, il y aura plusieurs messages, provenant de lui, à prendre en compte. Son application est justifiée dans les systèmes qui exigent un haut degré de sûreté de fonctionnement et dans les applications de contrôle en temps réel où les commandes une fois exécutées ne peuvent pas être défaites [SHE78].

Des algorithmes de maintien de la cohérence, appliqués à ce type de système doivent prendre en compte cette latence afin de garantir la fiabilité du système. Cette étude essaie de prouver la faisabilité de cette approche sur un algorithme en particulier; celui de KANEKO .et al., [KAN79], un des plus élégants conçus dernièrement et qui ajoute à son élégance, son économie de coût de communication et sa robustesse. Bien que son application soit destinée à une synchronisation de typé écriture-écriture dans les Bases de Données [BER81], il peut être également utilisé dans les cas de systèmes de commande répartie avec des copies d'information dans divers sites.

La modification de l'algorithme de Kaneko et al. pour prendre en compte la latence de détection détruit partiellement la "proprieté première" de l'algorithme: la cohérence mutuelle n'est plus garantie dans le cas de défaillance d'un site. Une deuxième modification nous permettra de résoudre ce problème et de garantir la cohérence totale des copies dans tous les cas. Les hypothèses choisies ont été les plus réalistes possibles et elles ont pour objectif de simplifier les algorithmes. Par conséquence, un élargissement des hypothèses doit impliquer simplement une augmentation relative de la complexité des algorithmes et non la

mise en cause de cette approche.

I.2.- Plan de la première partie·

Le chapitre suivant est un rappel des notions de surêté de fonctionnement, de tolérance aux fautes et de la participation des systèmes répartis dans ce domaine.

Le chapitre III essaie de placer avec précision le problème et sa solution, ses paramètres, ses alternatives, etc. et la phase du programme de recherche qui nous concerne.

Le chapitre IV est un chapitre nécessaire dans tout ouvrage traitant de tolérance aux fautes; il concerne la terminologie employée.

Un modèle de latence de détection dérivé du modèle de [SHE78] est esquissé dans le chapitre V. Il constitue la base de ce rapport.

Un rappel de la notion d'horloge logique et sa participation dans la synchronisation des événements dans les systèmes repartis est fait au chapitre VI.

L'objectif de notre approche étant le maintien de la cohérence de copies multiples, une brève analyse de la notion de cohérence et un survol des principaux algorithmes de maintien de la cohérence utilisant des estampilles font l'objet des chapitres VII et VIII respectivement.

Les notions de confinement et de temporisation sont définies avant d'entrer dans le détail du mécanisme (chapitre IX).

Le chapitre X définit les hypothèses générales établies, qui délimitent notre solution.

L'algorithme, avec prise en compte de la latence de détection, des hypothèses supplémentaires et quelques cas et exemples types,

font l'objet du chapitre XI.

L'algorithme de Herman et Verjus modifié et les hypothèses de base sont décrits dans le chapitre XII. Les divers types de panne y sont considérés.

Le chapitre XIII établit un bref parallèle entre les deux algorithmes modifiés. Les conclusions font l'objet du dernier chapitre (XIV).

II.- Concepts de base

II.1.- La sûreté de fonctionnement

L'utilisation de plus en plus diversifiée des systèmes informatiques a eu comme conséquence leur accès dans des secteurs jusqu'à maintenant inabordables en raison de leur complexité ou du degré de fiabilité exigé. En même temps, cela a crée une dépendance accrue des utilisateurs par rapport aux systèmes. Dans certains cas, il ne peut pas être permis à un système de fonctionner incorrectement ou de n'est pas être disponible plus d'un certain période de temps. Par conséquence, le degré de fiabilité qu'exige ce type de systèmes est de plus en plus grand.

Les fabricants sont alors obligés de fournir des systèmes dans lesquels plusieurs et diverses approches ont été utilisées pour pallier à ce problème. Ces approches sont classées en deux stratégies [AVI76]:

- la prévention des fautes; et
- la tolérance aux fautes

Tel qu'il a été montré dans la figure 1, chaque stratégie peut être réalisée par divers mécanismes, qui, bien appliqués, peuvent fournir un degré et un type de fiabilité convenable. C'est cela qu'on appelle l'acceptabilité d'un système. Ainsi, dans un système tel que SKALP [COU81] le type de fiabilité à garantir est concentré sur la disponibilité du système, et le degré souhaité est celui exigé par un commutateur téléphonique (par exemple: indisponibilité totale de 3 jours/30 mois, période de maintenance de 10 jours, etc.).

Ces deux stratégies sont appliquées, en général, d'une manière complémentaire. Mais notre intérêt est limité, dans cette première partie, seulement à la tolérance aux fautes.

II.2.- La tolérance aux fautes

Les systèmes tolérants aux fautes correspondent à un type de système qui inclut dès sa phase de conception l'idée qu'il ne peut pas être parfaitement réalisé. Il faut donc, appliquer une stratégie pour faire face aux fautes résiduelles et/ou aux fautes qui surgissent en raison du vieillisement du système.

Cette stratégie doit considérer tant les fautes anticipées que les fautes non-anticipées (principalement d'origine logicielle), intégrant ainsi le logiciel et le matériel. L'implantation de cette stratégie exige quatre étapes [AND82]:

a) Détection d'erreur
b) Evaluation des dommages
c) Recouvrement d'erreurs
d) Traitement de la faute

En général ces étapes ne sont pas si clairement définies et leur ordre d'exécution peut être différent. Ainsi, bien que la détection soit toujours la première étape, la reconfiguration éventuelle (traitement de la faute) peut se faire avant le recouvrement. Il est bien reconnu qu'un système bien structuré facilite grandement l'inclusion de la tolérance aux fautes et un système réparti est, en principe, potentiellement bien structuré.

II.3.- Les systèmes répartis

Beaucoup de chercheurs et de fabricants affirment, que les systèmes répartis sont, par nature, capables de fournir plus de fiabilité qu'un système centralisé [ENS78]. Mais cette affirmation peut être vraie seulement dans le cas d'un dessein correct. Cette capacité des systèmes répartis est rendue possible grâce à la redondance incluse, la modularité matérielle qui réduit les dommages (propagation de l'erreur), etc. Les composants du système sont aussi des systèmes autonomes et

indépendants, de manière que la région affectée par la défaillance d'un composant a des limites physiques bien définies. D'autres caractéristiques facilitent aussi cette capacité: la plus grande simplicité des sites par rapport à un site central et unique, la possibilité de modifier et/ou de remplacer des parties du système, etc. On augmente ainsi la robustesse du système et on facilite sa reconfiguration.

Cette fiabilité potentielle a contribué à ce que l'utilisation de ce type de systèmes se soit accrue dans les cas de systèmes temps réel: contrôle de processus, applications spatiales, médicales, etc. Mais leur application a révélé aussi des problèmes qui demandent de nouvelles solutions par rapport à celles existant dans les systèmes centralisés:

- le parallèlisme,
- la synchronisation,
- l'allocation des ressources,
- l'accés à l'information répartie; etc.

Ce dernier problème exige la revue en détail, des mécanismes d'accés concurrent existants, à cause des paramètres ajoutés: délai de transmission, contrôle réparti, détection d'erreur, etc.

III.- Cadre général

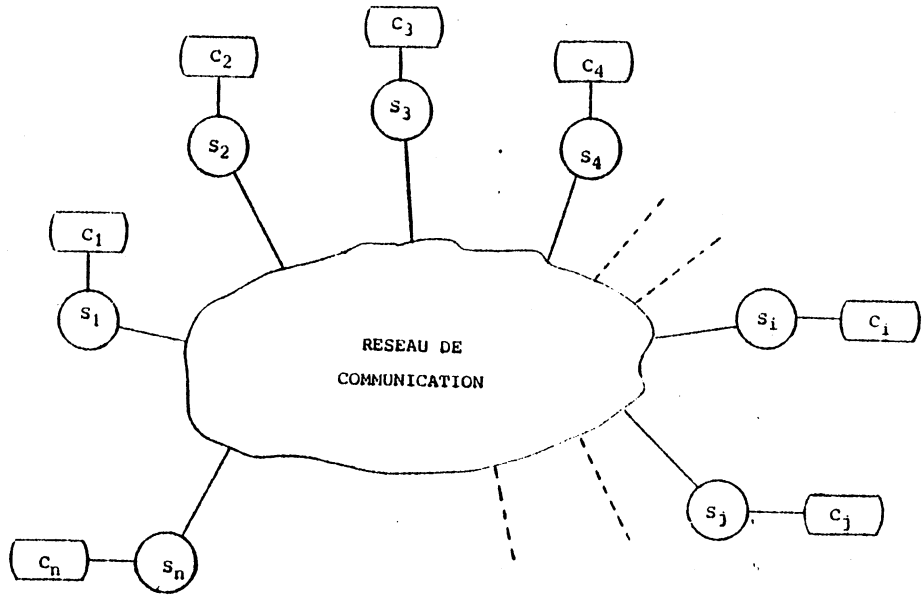
III.1.- Le support

Le cadre général où doit se placer cette première partie, est celui de systèmes géographiquement répartis. Ces systèmes répartis (voir figure 2) sont supposés réalisés par des sites ou stations (au sens large du terme) interconnectés par des moyens de communication quelconque.

On suppose qu'un niveau de base est muni de mécanismes détectant les fautes, matérielles et/ou logicielles, avec un temps de détection qui suit une certaine distribution. Ces mécanismes de détection peuvent être des moyens de test en-ligne de microprocesseurs tels que décrits en [COU81a] (pour les fautes matérielles), ou des moyens de niveau plus élevé tels que l'observateur [AYA79]. Leur objectif est d'éviter les défaillances du système en les prévenant. On ne fera pas de distinction entre fautes matérielles et fautes logicielles.

L'objet de la présente étude ne concerne pas ces mécanismes, que l'on suppose étudiés par ailleurs. Seule leur caractéristique de latence a de l'importance pour les études envisagées. Enfin, on suppose que le niveau de base assure l'envoi d'un message signalant qu'une erreur (faute) a été détectée (par la suite, cette hypothèse sera néanmoins levée pour certaines pannes).

Ainsi, notre objectif de recherche est d'étudier des mécanismes à mettre en oeuvre à un niveau supérieur afin d'atteindre des critères de sûreté de fonctionnement malgré l'existence de temps de latence de détection. La notion fondamentale à appliquer est le confinement d'une information erronée avant son utilisation, empêchant de cette manière la propagation de l'erreur.



C_i : copie i
S_i : terminal intelligent i
$[S_i, C_i]$: station i

Figure 2.- Système géographiquement réparti

III.2.- Application

On considère deux stations, l'une (A) produisant des informations qui sont utilisées par (B). Lorsqu'une faute survient en (A), (B) est prévenue de cette faute, au bout d'un certain temps (x) par les mécanismes du niveau de base. La voie de communication peut être la même que pour les informations, ou une autre avec des caractéristiques différentes de fiabilité (voir figure 3). Si (B) attend un certain temps (au moins égal à x) après réception, avant d'utiliser cette information (en la faisant "vieillir"), la valeur erronée ne sera pas utilisée en cas de faute sur (A). Ce temps est appelé "temps de confinement". Il est donc important d'étudier le confinement possible d'erreurs, à l'intérieur d'un niveau, en fonction des divers paramétres agissant tels que:

- La distribution des temps de détection de fautes.
- La distribution des temps de transfert des informations entre (A) et (B).
- La distribution des temps de transfert des signalisations de faute entre (A) et (B).

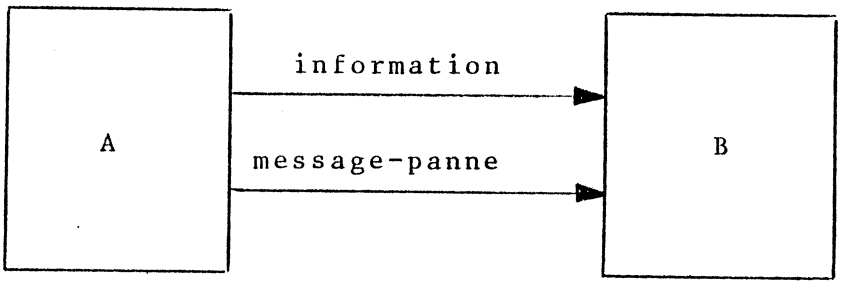


Figure 3.- Communication entre deux stations

Selon les applications, ou selon les tâches d'une application, les exigences peuvent être diverses: reprise de données contenues dans une station, reprise de processus, gestion de données à copies multiples, ou simplement le confinement d'erreur, comme dans l'exemple ci-dessus. Ainsi, par exemple, les contraintes actuelles du domaine téléphonique imposent de reprendre les conversations dejà établies (reprise de données, c'est à dire reprise des contextes téléphoniques). Ici, le problème de confinement est celui à résoudre pour que, en général, une station puisse reprendre des données contenues dans une autre station défaillante. En effet, une station qui maintiendrait une copie des données d'une autre station dans le cas de défaillance, ne saurait pas quelles données sont erronées parmi toutes les données, si elle procédait à une mise à jour immédiate.

Le simple confinement s'avère utile dans les cas de systèmes de commande répartie. Des stations peuvent être, les unes de commande (déroulant des algorithmes de commande), les autres d'interface avec le procédé commandé. Si l'information provenant d'une autre station est destinée à modifier une consigne pour la commande du procédé (qui nécessite des caractéristiques strictes de securité et/ou de disponibilité), la station d'interface peut retarder la modification, empêchant ainsi la génération d'une commande irréversible éventuellemente erronée.

III.3.- Le programme de recherche

Le travail presenté dans cette première partie s'inscrit dans un programme de recherche plus vaste, comprenant trois phases:

1ère. phase: Mesure de confiance (confinement) dans un système multi-niveaux

2ème. phase: Maintien de la cohérence d'informations à copies multiples utilisant une mesure de confiance. Mise en oeuvre et évaluation

3ème. phase: Reprise des processsus

III.3.1.- 1ère. phase

Cette phase concerne l'évaluation de la confiance dans l'information en vue du confinement de données erronées dans les systèmes répartis (cas particulier de système multi-niveaux), ainsi que la modélisation, les méthodes de calcul, etc., et leur application à deux systèmes de communication particuliers ETHERNET [MET76] et LISA [MAR79], munis de certains protocoles.

L'objectif est de répondre à la question: quelle est la confiance qu'un niveau peut accorder à une information lui parvenant? La réponse à cette question dépend des paramètres de distribution du temps de détection et de transferts, et de la "vieillesse" de cette information. En effet, un niveau (station) recevant une information peut cumuler le temps écoulé depuis la réception de

cette information, et la confiance dans l'information est une fonction non-décroissante de ce temps écoulé, tant qu'un message de signalisation de faute n'est pas reçu.

III.3.2.- 2ème. phase

Cette deuxième phase consiste à intégrer la mesure de confiance (répresenté par le temps de confinement), étudiée dans la 1ère. phase, à un mécanisme de gestion de la cohérence d'information à copies multiples. Une propriété des systèmes répartis est en effet de pouvoir maintenir plusieurs copies de certaines informations dans diverses stations. Le bon fonctionnement du système exige que ces copies soient cohérentes (il y a plusieurs algorithmes pour cela). Le problème du confinement laisse entrevoir que si l'on ne prend pas en compte une mesure de confiance, des mises à jour peuvent être faites par plusieurs stations de manière cohérente, mais à partir d'informations erronées. Il est donc nécessaire d'intégrer la mesure de confiance dans les systèmes de maintien de la cohérence d'informations à copies multiples.

Ceci n'a pas été jusqu'à maintenant étudié car les hypothèses faites généralement pour le traitement de fautes sont qu'une station ne produit plus rien en cas de faute, et donc elle ne se trouve jamais dans un état de faute non-détectée, et par suite susceptible de produire des informations erronées [SHA78].

On notera que le mécanisme proposé n'est pas destiné à remplacer les points de reprise qui restent nécessaires et étudiés par ailleurs [BOU79], mais que deux points successifs pourront être plus éloignés (moins de points de reprise).

Enfin, on peut penser qu'en cas de faute logicielle, un phénomène de manifestation de faute semblable à celui consécutif à des fautes d'origine matérielle [COU79], [OSD79], est susceptible de se produire en fonction des "changements d'états" du système.

III.3.3.- 3ème. phase:

La dernière phase envisagée concerne la reprise de processus. Cela concerne des procédés tel que l'on puisse reprendre le programme en cours d'exécution, si une faute survient sur la station sur laquelle est exécuté ledit programme. Il peut s'agir, par exemple, de procédés tels qu'on ne puisse pas imaginer de les laisser sans surveillance ou commande plus d'un certain temps. Les systèmes répartis permettent d'envisager une telle reprise, non plus seulement après une faute transitoire, mais après une faute permanente, si une copie du code du processus existe sur une autre station et si le contexte du processus est sauvegardé de temps en temps (points de reprise). En cas de panne, le processus peut repartir d'un point de reprise.

Plus précisément, en cas de défaillance survenant sur la station (A) (cf. II.2), la station (B) peut reprendre un processus de (A) si (B) possède une copie du programme et si elle dispose d'un contexte correspondant à un point de reprise. La condition sine qua non est que (A) transmette à (B) ces informations de contexte lorsque (A) établit un point de reprise. Ce contexte reçu par (B) peut être erroné, ce que (B) peut arriver à connaître, mais qui ne permet pas la reprise.

L'élimination de l'effet "domino", [RAN75], peut se faire par insertion supplémentaire de points de reprise, ce qui diminue les intervalles. Minimiser la masse de calculs perdus en cas de panne demande également à diminuer les intervalles. Mais le temps de création des points de reprise augmente avec la fréquence, ce qui incite à espacer lesdits points. L'interaction de certains de ces paramètres a déjà été étudiée [GEL78], mais l'ensemble doit être repris. A cela il faut ajouter le fait qu'au moins, deux contextes successifs doivent être maintenus par la station de reprise si l'on veut pouvoir reprendre un processus avec une bonne probabilité, en raison du temps de détection des fautes. Maintenir plusieurs contextes successifs permet naturellement de

se servir du plus ancien, à qui la plus grande confiance peut être accordée. Mais faire la reprise avec un contexte moins ancien peut éventuellement permettre de réduire la longueur de la chaîne de reprise. La probabilité de succès de la reprise dépend de la confiance attachée à la justesse d'un contexte. C'est l'étude de l'interaction de tous ces paramètres qui fait l'objet de cette dernière phase.

Seul le maintien de la cohérence des copies multiples (2ème. phase) est traité dans ce qui suit.

IV.- Terminologie

Ce chapitre est nécessaire étant donné l'ambiguité de termes et concepts existants à l'heure actuelle dans le domaine de la sûreté de fonctionnement de systèmes. Il y a en particulier quatre termes qui doivent être définis et precisés avec exactitude: défaillance, erreur, faute et panne.

Leur définition passe d'abord par la définition du concept de système, le modèle de structure et fonctionnement d'un système et la définition de spécifications d'un système.

Système: tout mécanisme identifiable et capable de maintenir un schéma de fonctionnement donné sur une interface avec son environnement. Structurellement il est constitué d'un ensemble de composantes ou sous-systèmes qui interagissent entre eux sous contrôle d'un composant particulier appelé soit "dessein" par [AND82], "colle" par [LEE82] ou "algorithme" par [RAN78]. Un composant ou sous-système est aussi un système par lui-même.

Interface: "lieu" d'interaction entre deux systèmes. Par conséquence un environnement est aussi un système.

Un système alors, interagit avec son environnement fournissant des réponses aux stimuli sur leur interface. Extérieurement, le fonctionnement du système peut-être décrit par les états externes qu'il peut prendre et par la fonction qui définit les transitions d'état. Mais ce fonctionnement extérieur est la manifestation de l'activité interne du système définit par les états internes. L'état interne d'un système est l'ensemble ordonné des états externes de ses composantes. Ainsi, un état externe n'est que l'abstraction d'un état interne, et par conséquence il y a des états internes qui ne sont pas définis. Les changements d'état interne sont les conséquences des interactions entre les sous-systèmes et entre le système et son environment.

La sûreté de fonctionnement d'un système implique de différencier clairement le fonctionnement correct du système du fonctionnement incorrect. Pour cela on utilise les spécifications du système qu'on suppose une réference autorisée et exacte [LEE82], [AND82], et que l'on peut appliquer comme un test pour déterminer si le fonctionnement du système est ou non acceptable. A partir de ces concepts on peut définir:

Défaillance: c'est l'événement qui arrive lorsque le fonctionnement d'un système dévie, pour la première fois, de celui décrit par les spécifications [AND82].

Lorsqu'un système subit une défaillance il se trouve dans un certain état interne qu'on pourrait appeler e_n. Si l'on considère que le système fonctionne initialement de manière correcte (e_0: état interne initial correct), il doit y avoir une transition d'état interne responsable de la défaillance. On dit alors qu'une transition est erronée si elle est responsable d'une défaillance ultérieure. Les états internes situés entre la transition erronée et la défaillance (e_n) sont appelés: états erronés. Un état valide est un état non erroné [AND81].

Erreur: c'est la partie d'un état erroné qui fait la différence avec un état valide.

Faute: est une erreur dans un composant.

Panne: communément utilisé pour désigner une défaillance d'origine matérielle.

Ainsi, une faute dans un système génère une erreur dans le système qui peut produire, éventuellement, une défaillance du système. On peut dire aussi que la faute d'un composant produit la défaillance du composant qui produit ainsi une erreur dans le système. La panne d'un composant, étant un type de défaillance du

composant, peut produire des erreurs dans le système qui seront précurseurs d'une éventuelle défaillance du système.

V.- Modèles de latence et de confinement

V.1.- Modèle de latence de détection

Pendant le fonctionnement normal (sous l'application) d'un circuit digital, il y a, en général, un délai entre l'occurrence d'une faute permanente et la première erreur à la sortie [SHE75], (voir figure 4). Ce délai est appelé la latence d'erreur (LE) de la faute. C'est un attribut de toute faute. Elle dépend du circuit, de la faute même et de la séquence d'entrées. On suppose que dans la séquence d'entrées il y a au moins une entrée, qui force à la manifestation de la faute: circuit auto-testable. C'est à dire que, en principe, la faute partitionne l'ensemble d'entrées en deux sous-ensembles: un qui donne des sorties incorrectes et l'autre qui donne des sorties correctes.

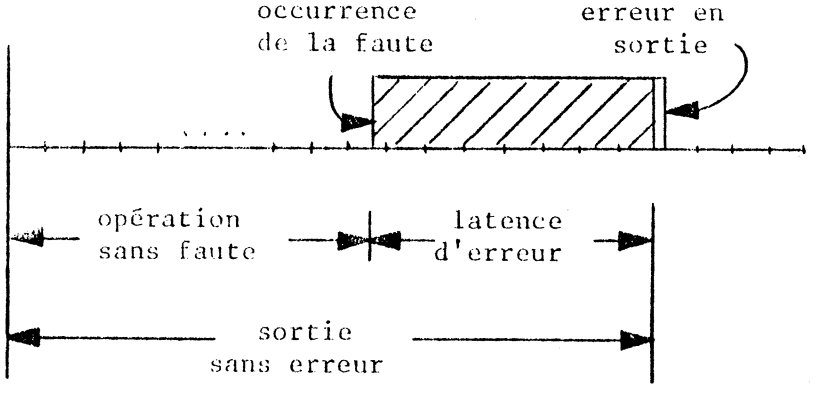


Figure 4.- Latence d'erreur

Pour un système, et tel que l'on confirme notre modèle de système du chapitre IV, ce modèle de latence d'erreur peut être aussi valable. Il y a, en général, un temps qui s'écoule entre l'occurrence d'une faute et la manifestation extérieure de la faute (sortie dehors des spécifications): défaillance.

En revenant aux circuits, une technique d'auto-test peut permettre la détection des fautes rapidement et avant que ne se produise une contamination de données dûe aux erreurs créées par la faute (circuit totalement auto-testable). Mais pour des raisons de coût ou de perfomance tous les circuits ne peuvent pas être totalement auto-testables. Il y a des circuits partiellement auto-testables, c'est à dire des circuits où les fautes ne sont pas détectées immédiatement et où on admet généralement qu'il y a une propagation des erreurs avant que la faute (une erreur produite par elle) soit détectée.

Cela nous ramène à un modèle plus détaillé [SHE78], où une faute dans un circuit partitionne la séquence d'entrée en trois sous-ensembles:

- Un qui donne une sortie correcte.
- Un qui donne une sortie incorrecte mais non-détectée.
- Et un autre qui donne une sortie incorrecte détectée.

Ainsi, une faute ne produit pas immédiatement une erreur, et lorsque celle-là se produit elle n'est pas immédiatement détectée (voir figure 5).

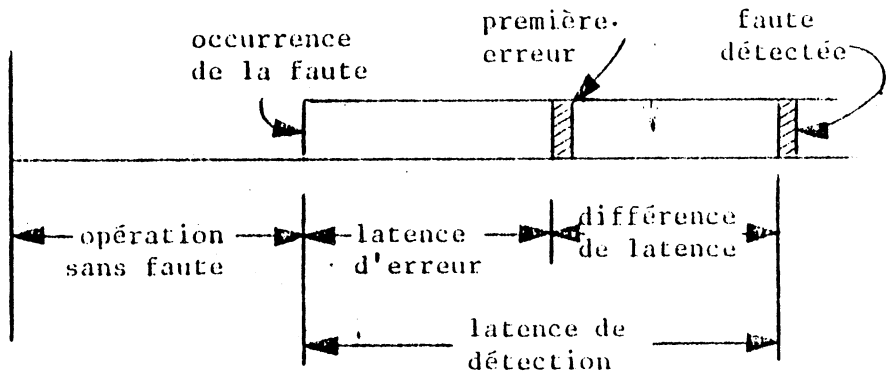


Figure 5.- Latence de détection

Ainsi, au niveau des systèmes la figure 6 est une bonne répresentation de ces concepts.

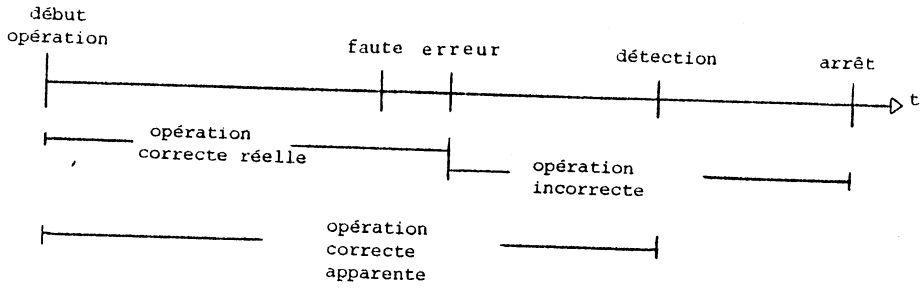


Figure 6.- Panne et détection dans un système

Finalement [SHE78] définit ainsi la latence d'erreur et la latence de détection, dans les circuits digitaux:

> La latence d'erreur (LE) dûe à une faute F est le temps écoulé entre l'occurrence de la faute F et la première sortie incorrecte dûe à F.
>
> La latence de détection (LD) dûe à une faute F est le temps écoulé entre l'occurrence de la faute F et la première détection d'une sortie incorrecte dûe à F.

La technique de test de circuits décrite plus haut peut être assimilée, au niveau de système, à la technique de test en-ligne continu avec détection totale d'erreur. Tel qu'on classe les circuits en: partiellement et totalement auto-testables [SHE78], on peut aussi classer les systèmes ou sous-systèmes (par rapport aux pannes qui sont detectées) . Et les procédures de recouvrement utilisées dependront de la classe à laquelle appartient le système.

Ainsi, un système totalement auto-testable n'a pas besoin des procédures de recouvrement, il suffit de remplacer le système par un autre et de continuer l'exécution. Pour un système

partiellement auto-testable, on sait que la faute sera toujours détectée (dans un temps fini) mais quelques erreurs dûes à la faute seront sorties du système ou sous-système et auront contaminé l'environnement ou le reste du système avant la détection. Dans ce cas-là, la séquence d'entrée comprend tant l'application comme les vecteurs (programmes) de test utilisés. La présence d'une latence de détection dans les systèmes, est expliquée en général par l'utilisation d'une redondance minimale (matérielle ou logicielle).

Mais la proprieté d'être totalement auto-testable étant difficile à obtenir (autant pour les circuits que pour les systèmes), on peut se limiter à celle de partiellement auto-testable. Alors des procédures de recouvrement seront necessaires, et diverses études montrent les caractéristiques de ces procédures et les intervalles entre points de recouvrement. Certaines applications (temps réel) exigent un niveau très bas de contamination en cas de faute présente dans le système (fiabilité concernée par l'intégrité). L'intervalle est alors, transformé en un retard de sortie (temporisation) qui permet un confinement d'erreur.

V.2.- Modèle général de confinement d'erreur

En utilisant un système multi-niveaux, on conçoit que chacun des niveaux peut ne pas utiliser immédiatement une information qui lui parvient. Ainsi, en se plaçant dans le niveau i de la figure 7 et si les notations sont les suivantes:

X[o]: variable aléatoire représentant le temps écoulé entre l'émission de l'information et la détection d'une faute tel qu' il existe une faute

X[1], X[2],
X[3], ...X[i]: variables aléatoires représentant le temps de transfert des signalisations de faute

Y[1], Y[2],
Y[3],....Y[i]: variables représentant le temps de transfert des

informations

t: le temps de résidence de l'information dans le niveau i, c'est à dire le temps écoulé entre la réception de l'information et son utilisation réelle

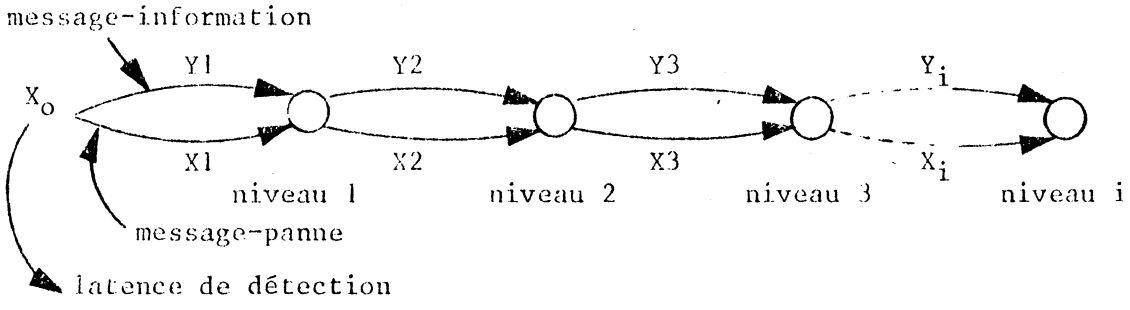


Figure 7.- Modèle général multi-niveaux

On aura:

$$\begin{aligned}&Pr\ (\text{confinement dans le niveau } i \ /\ \text{une faute arrive})\\ &= Pr\ (\text{confinement}) =\\ &= Pr\ (X[o] + X[1] + \ldots < Y[1] + Y[2] + \ldots + t)\\ &= \int f(X[o] + X[1] + \ldots)\ f(Y[1] + Y[2] + \ldots)\ dx\ dy\end{aligned}$$

Pour des distributions discrètes:

$$Pr\ [\text{confinement}] = \sum f(X[0] + X[1] + \ldots)\ f(Y[1] + Y[2] + \ldots)$$

La confiance que le niveau i accorde à une information est évidemment une fonction non-décroissante de t, sauf lorsqu'un message de faute survient. Dans les cas de deux stations (A) et (B), on peut supposer que la station (B) utilise l'information envoyée par (A) uniquement lorsque cette information a atteint un certain niveau de confiance.

VI.- Horloges logiques

Dans les systèmes répartis, le concept de l'ordre d'arrivée des événements a été analysé soigneusement par [LAM78]. La relation "est arrivé avant" définit seulement un ordre partiel dans l'ensemble d'événements du système. [LAM78] a donné un algorithme réparti qui étend cet ordre à un ordre total.

Il considère qu'un système réparti est constitué d'un ensemble de processus, chacun formé d'un ensemble d'événements avec, à priori, un ordre total. L'envoi et la réception d'un message sont aussi considérés comme des événements. A partir de là, [LAM78] définit la relation "est arrivé avant" (-->) comme:

> La plus petite relation qui satisfait les trois conditions suivantes:
>
> (1) Si a et b sont des événements dans le même processus et a vient avant b alors a --> b.
>
> (2) Si a est l'événement envoi d'un message d'un processus, et b est l'événement réception du message par un autre processus alors a --> b.
>
> (3) Si a --> b et b --> c alors a --> c.

C'est à dire que a --> b s'il est possible pour a d'affecter éventuellement b. Deux événements sont concurrents si ni l'un ni l'autre peuvent éventuellement s'affecter. En conséquence, la relation (-->) établit seulement un ordre partiel irréflexif, sur l'ensemble d'événements du système. Ainsi, dans la figure 8, qui utilise la même représentation graphique de [LAM78]:

> P, Q, et R sont des processus.
>
> p_1, p_2, p_3, q_1, q_2,..... sont des événements.

Et p_1 --> r_3 parce que:

p_1 --> q_2; q_2 --> q_4; q_4 --> r_2; et r_2 --> r_3.

Mais p_3 et q_3 sont des événements concurrents parce que:

p_3 --> q_3 et q_3 --> p_3.

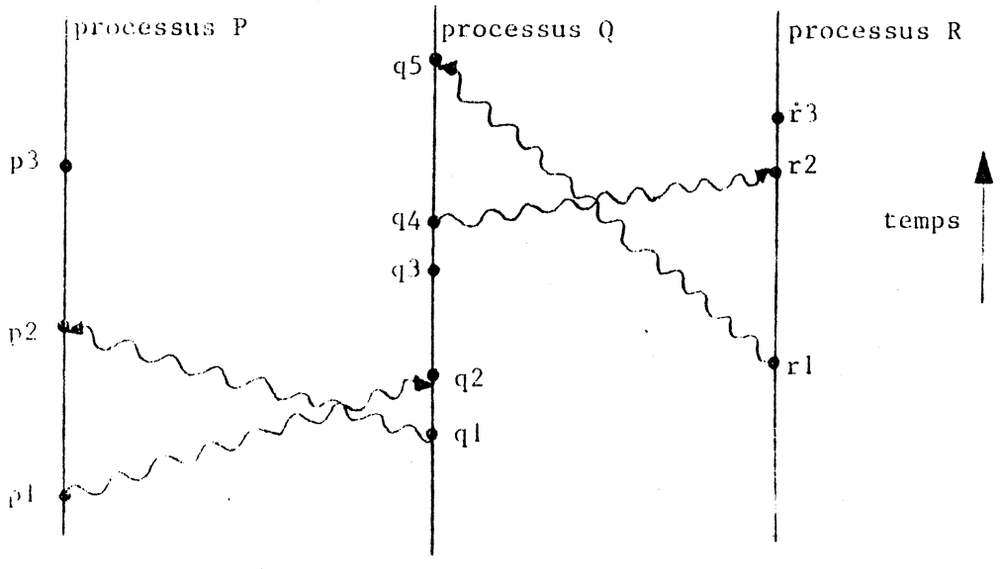


Figure 8.- Ordre des événements

[LAM78] considére un horloge comme étant simplement une manière de donner un numèro à un événement, où le numèro est considèré comme l'instant (le temps) auquel l'événement est arrivé. Ainsi l'horloge C_i pour le processus P_i est une fonction qui donne un numèro $C_i(a)$ à tout événement a dans P_i. Le système global d'horloges est representé par la fonction C qui donne à tout événement a le numèro C(a) où C(a) = $C_j(a)$ si a est un événement du processus P_j. Les horloges C_i sont appelées horloges logiques. Pour établir un système correct d'horloges, on a la condition suivante:

Condition des horloges: Pour deux événements a et b quelconques:

si $\underline{a}$ --> $\underline{b}$ alors $C(a) < C(b)$

Cette condition est satisfaite si les deux conditions suivantes (plus élementaires) sont satisfaites:

C1.- Si $\underline{a}$ et $\underline{b}$ sont des événements dans le processus P_i et $\underline{a}$ vient avant $\underline{b}$ alors $C_i(a) < C_i(b)$

C2.- Si $\underline{a}$ est l'envoi d'un message par P_i et $\underline{b}$ est la reception de ce message par P_j alors $C_i(a) < C_j(b)$

[LAM78] essaie alors d'introduire les horloges dans les processus de manière à satisfaire la condition des horloges (c'est à dire satisfaire C1 et C2). La règle suivante satisfait la condition C1:

IR1: Chaque P_i augmente son C_i entre deux événements successifs.

Pour satisfaire C2 il faut que chaque message $\underline{m}$ porte une estampille T_m avec une valeur égale à la valeur de l'horloge lorsque le message est envoyé. A la réception de $\underline{m}$ le processus récepteur doit avancer son horloge, si nécessaire, à une valeur plus grande que T_m:

IR2: a) Si $\underline{a}$ est l'événement envoi d'un message $\underline{m}$ par le processus P_i, alors le message $\underline{m}$ porte une estampille $T_m = C_i(a)$.

b) A la réception du message $\underline{m}$, le processus P_j met C_j plus grand ou égal à sa valeur actuelle et plus grand que T_m.

Ce système d'horloges est utilisé par [LAM78] afin de créer un ordre total des événements. Pour cela il suffit d'ordonner les

événements selon le temps d'occurrence. Pour les cas d'égalité (forcement entre événements de processus différents) il utilise un ordre total entre processus, défini arbitrairement. Finalement, il définit la relation d'ordre total suivante, sur l'ensemble d'événements du système (==>):

> Si a est un événement du processus P_i et b un de P_j
> alors a ==> b si:
>
> (1) $C_i(a) < C_j(b)$; ou
> (2) $C_i(a) = C_j(b)$ et $P_i < P_j$

Cette relation d'ordre total est utilisée fréquemment pour résoudre des problèmes de synchronisation, particulièrement pour les systèmes répartis.

VII.- La cohérence en copies multiples

Les copies multiples trouvent leur justification dans l'allocation optimale de l'ensemble d'informations: temps d'accès (en lecture) réduit, disponibilité, etc. Ainsi, dans une organisation à copies multiples, l'accès en lecture est amélioré remarquablement et de cette manière l'accès moyen à une information dans un système réparti peut être réduit.

On peut considèrer trois cas d'utilisation de copies multiples [COR81] (voir figure 9):

Cas 1: C'est le cas de systèmes multi-processeurs à mémoire commune. Chaque processeur à une mémoire locale où sont recopiés les objets à utiliser par un processeur. Dans le cas d'utilisation d'un objet (sa représentation) par plus d'un processeur, c'est la représentation se trouvant en mémoire commune qui est utilisée comme référence.

Cas 2: Il n'y a pas de mémoire commune, mais un objet appartient à un seul site. Si un processeur veut utiliser un objet duquel il n'est pas propriétaire, il demande une copie de l'objet au site propriétaire. Dans le cas de multi-utilisation c'est la copie d'origine qui est utilisée comme référence.

Cas 3: Chaque site à une copie et il n'y a pas de représentation privilégiée. On exige que toutes les copies du même objet soient identiques. C'est le cas le plus général et l'objet de notre étude.

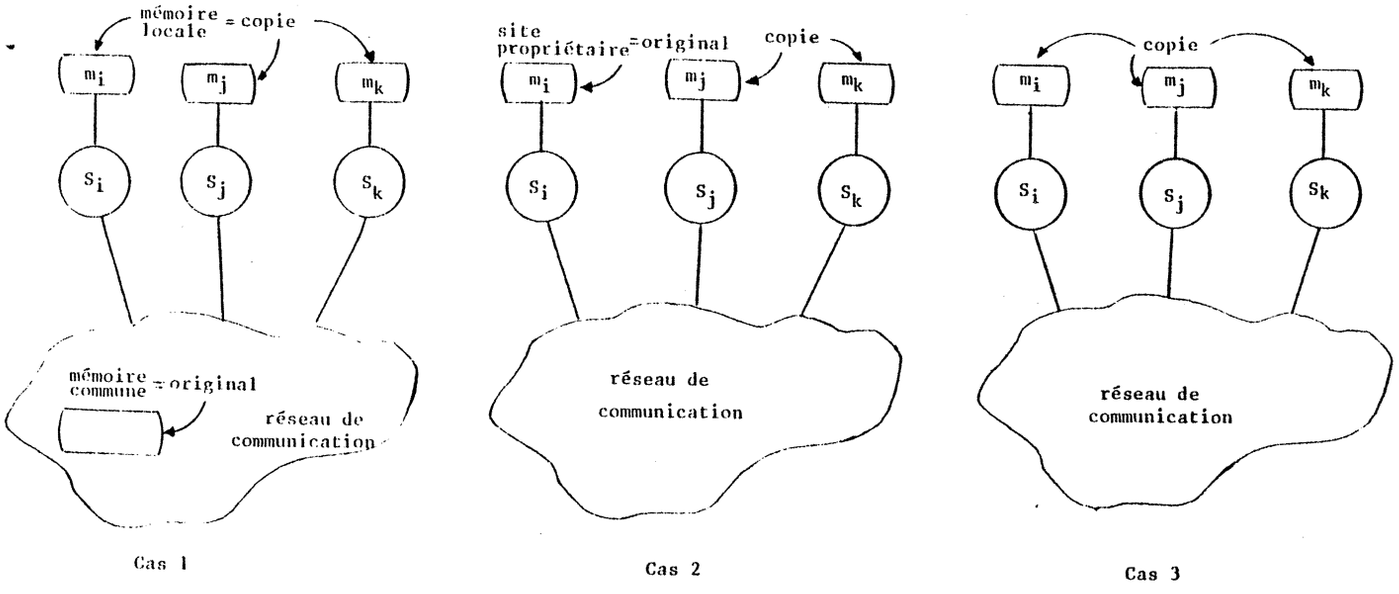


Figure 9.- Utilisation de copies multiples

Mais l'existence de copies multiples de la même information, entraîne deux types de problèmes:

- Toute modification locale implique sa répercussion dans chaque station du système.

- Il faut éviter que l'accès concurrent en modification nous amène à un état non souhaité: perte de la cohérence.

La cohérence est un terme utilisé dans le domaine de systèmes de gestion des Bases de Données. L'aperçu sur cette notion, donné ci-dessous, est extrait de la littérature existant dans ce domaine.

D'abord, il faut remarquer qu'il peut exister une confusion de termes entre cohérence et intégrité. On peut dire que l'intégrité est une mesure de la fidélité ou exactitude avec laquelle une information représente une certaine réalité. Tandis que la

cohérence est la qualité, d'une information, d'être liée ou connectée logiquement et correctement à une autre. La première implique la seconde. Une information intègre est cohérente, une information cohérente n'est pas forcément intègre. Un système, avec la technologie actuelle, peut garantir 100% de la cohérence (dans la limite de ses contraintes). L'intégrité a un caractère subjectif qui empêche un système de la garantir totalement; il peut se limiter seulement à la renforcer et à rendre l'information plausible, vraisemblable.

Ainsi par exemple, si deux données X et Y doivent avoir la même valeur parce qu'elles représentent des entités ou des caractéristiques qui sémantiquement sont ainsi, le système peut au moyen de règles à accomplir, garantir cette qualité. La cohérence est ainsi maintenue. Par contre, s'il existe une donnée représentant par exemple l'âge d'une personne (33 ans), le système ne peut pas garantir la fidélité de cette information. Il peut empêcher que la donnée prenne des valeurs aberrantes (-19, 350, etc.). Mais il n'a pas des moyens pour éviter qu'elle prenne des valeurs erronées mais plausibles (35, 36, etc.). C'est le problème de l'intégrité de données.

La notion de cohérence est réalisée au moyen des règles de cohérence, renforcées soit au niveau du système de gestion, soit au niveau des applications. La figure 10 montre les types de cohérence concernés et les responsables de leur renforcement (système ou application) . Ce choix est permis dans les cas des systèmes mono-utilisateurs. Dans ce type de systèmes le problème de la cohérence est minime: les données doivent représenter correctement les objets, c'est la cohérence sémantique. Dans le cas d'accés concurrent en systèmes multi-utilisateurs, le système garantit, en plus, que l'accés concurrent ne détruise pas l'état cohérent de l'information. Une condition suffisante est la sérialisabilité d'exécution de modifications, c'est à dire l'équivalence de l'exécution concurrente avec une exécution sérielle quelconque. Les mécanismes qui sont chargés de cette tâche sont appelés "mécanismes de contrôle de l'accès

concurrent". Le type de cohérence obtenu est appelé: cohérence interne (cohérence sémantique + sérialisabilité).

	UN ORIGINAL CENTRALISE		COPIES REPARTIES
	MONO-UTILISATEUR	MULTI-UTILISATEUR	MULTI-UTILISATEUR
cohérence sémantique	{ système application }	système	système
cohérence interne	-----	système	système
cohérence mutuelle	-----	-----	système

Figure 10.- Types et responsables de cohérence selon les systèmes

Dans les cas de systèmes répartis, la cohérence concerne deux aspects:

- la cohérence interne; et
- la cohérence mutuelle

La cohérence interne a déjà été définie et elle englobe les deux notions nommées plus haut: la cohérence sémantique et la sérialisabilité.

La cohérence mutuelle est une notion propre aux copies multiples (voir figure 10). C'est la qualité qui fait que toutes les copies de l'information convergent au même état lorsque toute l'activité de modification est arrêtée. Un exemple emprunté à [TH079] doit nous permettre d'éclaircir la différence entre ces deux concepts:

Supposons un ensemble de données: x, y, z qui ont comme valeurs initiales $x_0 = y_0 = z_0 = 1$. La règle de cohérence étant: $x + y + z = 3$. Supposons aussi deux processus: P_1 et P_2 qui font:

P_1: x:= -1; y:= 3

P_2: y:= -1; z:= 3

Chacun garantit indépendamment le maintien de la cohérence interne. Mais si dans un système réparti (avec deux stations S_i et S_j), on exécute ces deux processus concurremment de la manière suivante:

> S_i et S_j exécutent P_1 et P_2 (dans n'importe quel même ordre) alors la cohérence mutuelle est maintenue mais la cohérence interne est détruite.

> S_i seulement exécute P_1 et S_j exécute seulement P_2, alors la cohérence interne est maintenue mais la cohérence mutuelle est détruite.

Pour assurer la cohérence mutuelle, il suffit d'intégrer dans le même ordre sur chaque copie toutes les modifications apportées à l'information. A partir de là, deux degrés de cohérence peuvent être dégagés: la cohérence forte et la cohérence faible [WIL79]. La cohérence forte est celle qui est obtenue par l'exécution totalement séquentielle des modifications. A la fin de chaque modification l'état de toutes les copies est le même. La cohérence faible est obtenue en arrêtant toute nouvelle modification sur le système; alors, après un temps fini les copies deviendront identiques.

VIII.- Algorithmes de cohérence

Les algorithmes existant pour le maintien de la cohérence peuvent être classés en trois groupes [MIL80]:

a)- ceux qui utilisent des verrous globaux;
b)- ceux qui utilisent des estampilles; et
c)- ceux qui utilisent des tickets.

a)- Cela consiste à réaliser une exclusion mutuelle globale sur l'ensemble des copies lors de toute réservation et à rapporter les modifications sur l'ensemble des copies avant de relâcher l'exclusion mutuelle. Inconvénients: difficulté de réalisation, interdiction des exécutions parallèles d'accès (toutes les copies étant bloquées pendant la durée d'une modification).

b)- Chaque site possède un horloge et on associe à chaque modification, le temps de son émission (estampille). Les modifications sont diffusées vers toutes les copies et elles sont prises en compte dans le même ordre dans toutes les stations (l'ordre des estampilles). On trouve la même difficulté que dans le premier cas, mais l'exécution parallèle est permise.

c)- Les algorithmes évitent tous les problèmes, en permettant qu'à une seule station de réaliser les mises à jour: celle qui a le ticket. Celui-ci circule entre les stations suivant un chemin fourni par un anneau virtuel [LEL78].

Dans notre approche on utilisera des algorithmes de classe b)-; c'est à dire des algorithmes utilisant des estampilles, en raison de leur caractéristique (utilisation du temps). Une éventuelle application aux algorithmes de type différent pourra être étudiée ultérieurement. Dans ce contexte-là, quelques algorithmes utilisant cette approche de manière répartie sont ceux de:

- Kaneko et al.
- Herman et Verjus
- Rosenkrantz
- Thomas
- Bernstein

On s'intéresse aux algorithmes répartis, considérés comme plus proches de notre objectif de sûreté de fonctionnement. De cet ensemble on a pris celui de Kaneko et al., un des plus simples et élégants. Un bref rappel des mécanismes de mise à jour employés par les algorithmes de Kaneko et de Herman et Verjus nous permettront de comprendre ce qui suivra. Dans les deux cas l'exécution de modifications s'effectue de manière indépendante dans chaque site, mais dans le même ordre chronologique. Les modifications sont vehiculées d'un site à un autre au moyen de messages.

Pour l'algorithme de Kaneko chaque site posséde cinq queues où sont placées les modifications en fonction de l'estampille qu'elles possèdent. Les horloges des sites avancent de manière synchronisée au moyen de messages-temps, de manière qu'il n'y ait jamais plus d'un intervalle de différence entre les horloges de deux sites quelconques. Cela garantit que tout message émis au temps k est reçu, au plus tard, avant que l'horloge du site récepteur avance à $k+2$. Tout site se trouvant dans l'intervalle k, exécute les modifications placées dans sa queue $Q[k-2]$ après vérification de non-conflit avec d'autres modifications déjà exécutées. Toute modification en conflit est refusée. On considère deux modifications en conflit si l'exécution d'une modifie les données sur lesquelles est basée l'autre. La défaillance d'un site est détectée par l'absence du message-temps.

Pour Herman et Verjus, chaque site posséde N queues, une pour chaque site (inclus lui même) et qui sont destinées à contenir les messages provenant du site correspondant. Lorsqu'un message

arrive à un site, il est placé dans la queue correspondante et il est exécuté lorsque les deux conditions suivantes sont remplies:

- Aucune queue est vide
- Le message a l'estampille la plus ancienne

Lorsqu'une queue reste vide un certain temps, le site envoie un message-enquête au site correspondant. Si celui-ci n'est pas en panne il répond avec un message d'acquittement qui sera placé dans la queue correspondante comme tout autre message. On satisfait ainsi à la première condition.

IX.- Confinement et temporisation

La méthode de confinement est concernée, en principe, par le problème de la protection de l'information [LAM73]. Son objectif est de créer un environnement tel, qu'un programme ("non-sûr") puisse y être exécuté de manière fiable. Le confinement consiste, à ce qu'une information ne sorte du programme confiné que par des chemins contrôlés (paramètres par exemple). [LAM73] fait mention explicitement de la notion de confinement mais restreinte au logiciel: lorsqu'un programme sûr, appelé "client", fait appel à un autre programme non-sûr (co-routine ou sous programme) appelé "service". Le "client" doit prendre des précautions pour éviter que le "service" réalise des actions non-prévues, en lisant ou modifiant des données auxquelles l'accès ne lui est pas garanti. Le fait de contrôler le domaine d'action d'un "service" est appelé le problème du confinement. [LAM73] donne aussi quelques règles de confinement à suivre dans la réalisation d'un programme de "service" pour qu'il soit fiable.

Dans l'application du confinement, l'environnement est donné, très naturellement par l'architecture même du système réparti. Les stations qui forment le système, communiquent seulement par des messages qui sont transmis par des voies de communication établies entre elles.

Mais notre problème n'est pas un problème de protection mais un problème de sûreté de fonctionnement, et plus exactement un problème de tolérance aux fautes. Ainsi, notre objectif principal sera d'empêcher la propagation d'erreurs, en limitant (à une station) les dommages engendrés par une défaillance. Une fois la station défaillante mise hors-service, on pourra garantir le fonctionnement sûr du système.

La notion de confinement utilisée, dans cette étude, est différente de celle de [LAM73] dans la mesure où elle est concernée seulement par la sortie d'erreurs et non par la sortie

de l'information en général. On considère que ces erreurs ont comme origine des fautes detectables (transitoires ou permanentes), de manière que les mécanismes de détection, internes à chaque station, puissent réaliser leur fonction.

Les concepteurs de systèmes tolérants aux fautes sont très soucieux de la propagation d'erreurs dans les systèmes. Parce que même si les mécanismes de détection de fautes sont parfaits, c'est à dire qu'il n'y a pas de faute non-détectée, l'existence d'une latence de détection les oblige à admettre que le fonctionnement continu et sûr d'un système doit passer par une étape de recouvrement d'erreur. Et cela implique une évaluation des dommages commis par la faute. Etant donné la complexité des systèmes informatiques, il est clair que dans certains cas il est impossible de bien localiser ces dommages et on a intérêt à réduire au maximum sa propagation.

Le degré de tolérance dépend toujours de la réussite avec laquelle: les fautes sont détectées, les erreurs sont confinées et les états sont réparés et/ou remplacés. Préparer un système au confinement d'erreurs signifie bien évaluer la portée des fautes possibles.

Dans un système tolérant aux fautes, une fois que les dommages ont été évalués, la phase suivante est le recouvrement des erreurs. Notre notion de temporisation est basée sur le principe de recouvrement. Les techniques utilisées par celui-ci, doivent permettre de placer le système dans un état où le traitement puisse continuer sans problème. D'où l'utilisation de points de reprise dont un paramètre important est l'intervalle entre points. Cet intervalle est critique en applications temps réel où les commandes une fois émises ne peuvent pas être rattrapées. La solution est de transformer l'intervalle calculé en un retard amortisseur à la sortie (temps de confinement) [SHE78]. Cela veut dire, retarder l'exécution des commandes en attendant que la vérification de l'état du système puisse valider la bonne qualité des commandes. En bref, c'est un recouvrement d'erreur "anticipé"

qui va accroître la fiabilité.. Les exigences de l'application peuvent fixer une limite à ce retard en restreignant de cette manière la fiabilité à atteindre.

Tout cela signifie que l'on peut associer à une information (message), une mesure de confiance. Cette mesure dépend des paramètres de distribution du temps de détection et de transfert, et de la "vieillesse" de cette information. En effet, une station recevant une information peut cumuler le temps écoulé depuis sa réception. La confiance de l'information est une fonction non-décroissante de ce temps écoulé, tant qu'un message de signalisation de faute n'est pas reçu. Le temps écoulé depuis la réception de l'information est le seul temps physique pouvant être considéré, une station ne connaissant que son temps physique local. Lorsqu'un tel message est reçu, la confiance de l'information est zéro et l'erreur peut être confinée. Lorsque les temps de détection et transfert sont bornés, cette confiance peut atteindre éventuellement le 100% de sa valeur. Si cette information est une demande de mise à jour, la cohérence de l'information dans le système est ainsi garantie. Cela est représenté par la figure 11.

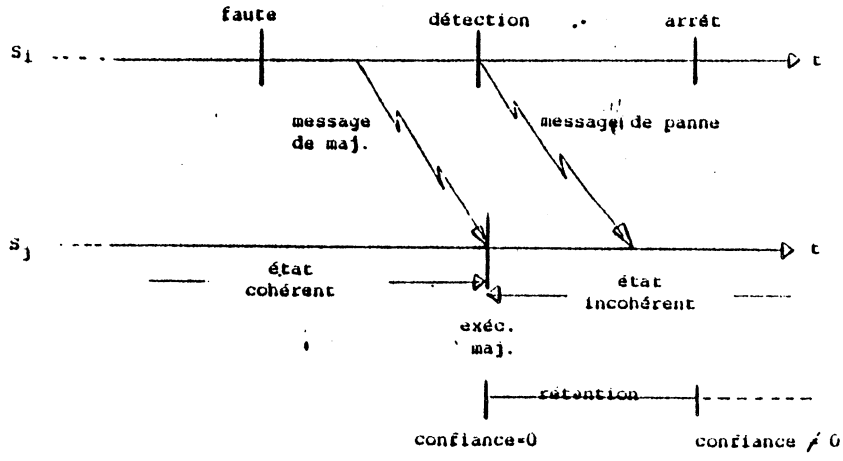


Figure 11.- Mesure de confiance

Lorsque la station S_j reçoit le message de mise à jour (maj) provenant de la station S_i, elle lui assigne une valeur de confiance égale à zero. Si elle exécute immédiatement cette maj, elle risque de créer un état incohérent si un message de panne arrive peu de temps après. Elle doit retarder l'exécution de la maj, en attendant que le message atteingne sa valeur de confiance maximale. Le risque de créer un état incohérent est inversement proportionnelle à la mesure de confiance d'un message.

X.- Hypothèses établies

La complexité des systèmes nous oblige, en général, à les simplifier au moyen de modèles et d'hypothèses qui éliminent les ambiguités et qui nous permettent de bien définir les frontières du problème. Ainsi, dans notre cas on établira des hypothèses de système, c'est à dire, des limitations sur l'architecture du système en général et des hypothèses supplémentaires qui concernent l'application des algorithmes [POS82].

Les algorithmes devront considérer deux cas spécifiques: le cas de panne contrôlée et le cas de panne non-contrôlée (voir figure 12). Dans le premier cas la faute est détectée et un message de faute est émis avant la panne de la station. Dans le second cas, la faute génère une panne intempestive et la station n'a pas le temps pour communiquer avec le reste du système.

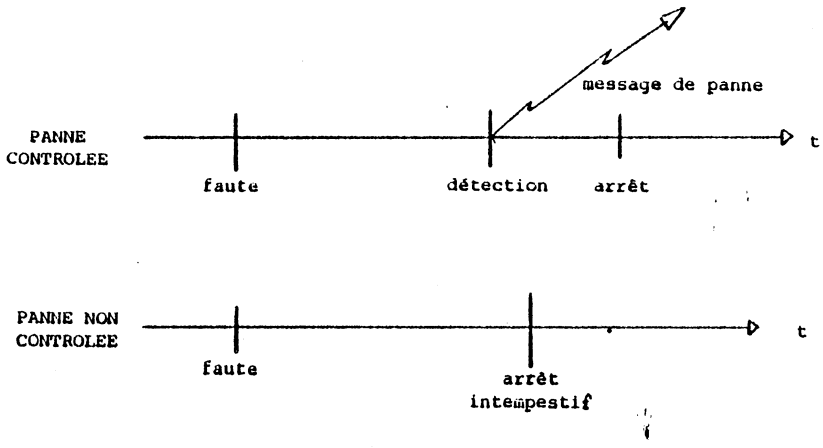


Figure 12.- Types de panne

HYPOTHESE H1

Le système réparti est constitué de N stations, numérotées (arbitrairement) par ordre de priorité de 1 à N, et liées par un réseau de communication sûr et de topologie non définie, mais virtuellement complète. (On appelle réseau de communication sûr,

celui qui sous un fonctionnement sans panne, libère les messages d'une unité à une autre, dans l'ordre dans lequel ils sont envoyés, sans altération et sans perte de messages)

HYPOTHESE H2

Chaque station i ($1 \leq i \leq N$) est fonctionnellement autonome et possède:

a)- Une copie de l'information commune qu'on appellera COPIE (C[i]); un MODULE DE GESTION (MGC[i])est chargé de l'accés à C[i].

b)- Deux types d'horloge nommées, l'une LOGIQUE et l'autre PHYSIQUE:

- l'horloge logique (HL[i]) est contrôlée par un module (MHL[i]); le concept de base correspond à celle de [LAM78];

- l'horloge physique (HP[i]) qui correspond exactement à la notion d'horloge physique de [LAM78], mais avec une seule condition:

$$\frac{d\,HP}{dt} \cong 1$$

c'est à dire, qu'il doit y avoir une fréquence constante dans la station i, mais qui peut être différente de celle de la station j ($j \neq i$).

c)- Un vecteur des valeurs de temps de confinement (TC[i]) pour chaque station j ($1 \leq j \leq N$), incluse elle même, en unités de temps d'HP[i].

d)- Des processus d'application ($P^1[i]$, $P^2[i]$,...$P^r[i]$; on supposera r=1 et un processus par station: $P^1[i] = P[i]$), qui accédent à la base par l'intermediaire du MGC[i]. P[i] passe

ses demandes de mise à jour (MAJ[i]) au MGC[i], lequel est chargé de les envoyer à tous les autres MGC[j] ($j \neq i$) du système.

e)- Eventuellement et dans les cas de certaines pannes de matériel, plusieurs processeurs, dont un sera chargé de communiquer la panne de la station au reste du système (processeur de communication, du genre coupleur de SKALP [COU81]).

HYPOTHESE H3

Chaque MAJ[i] porte une estampille (EST[i]) égale à la valeur de HL[i] à l'instant de l'émission de la demande. A cet EST[i] on ajoutera le numèro de la station et la priorité du processus dans la station, le tout formant l'identification de la MAJ[i] (IDE[i]). La figure 13 montre la structure d'un message de mise à jour. L'erreur éventuellement portée par ces messages est du type "erreur subtile", c'est à dire une erreur qui n'est pas facilement observable parce qu'elle ne détruit pas la sémantique de l'information, et parce que les dommages causés par elle sont intrinsèques à l'information même. Ainsi, l'erreur dans une demande de MAJ ne modifie ni son destinataire ni sa source, mais elle pourra modifier d'autres paramètres d'information sur leur valeur. Tous les codes de détection et/ou correction d'erreur qui pourraient être implantés dans des procédures de transfert de messages sont supposés ne pas être utiles vis à vis de ces erreurs.

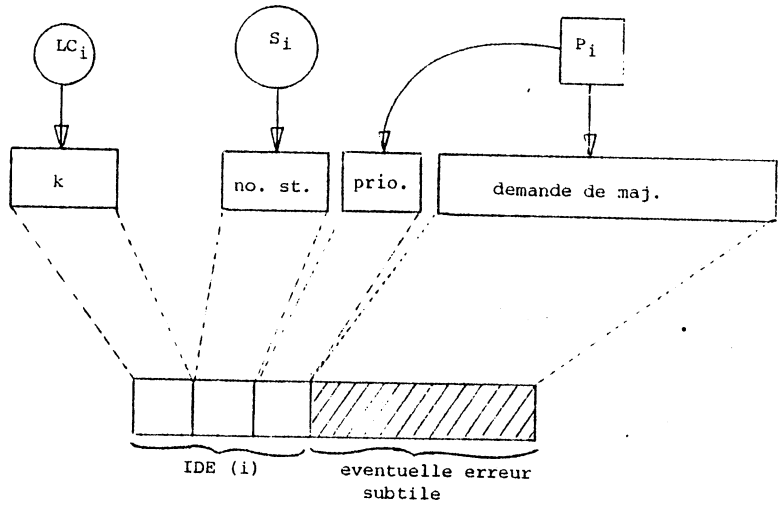


Figure 13.- Structure du message de mise à jour

HYPOTHESE H4

A la réception d'une MAJ[i], chaque station j ($1 \leq j \leq N$) lui adjoint la valeur de son HP[j] (VHP[j]) au moment de la réception.

XI.- Algorithme de Kaneko avec temporisation

[KAN79] donne un algorithme de base pour sa méthode de synchronisation. C'est sur cet algorithme que notre méthode de maintien de la cohérence avec confinement d'erreur est réalisée [POS82]. Des hypothèses supplémentaires seront établies avant de décrire la première modification de l'algorithme. Mais ce nouvel algorithme ne supporte pas tous les cas de panne controlée d'une station. Une phase de synchronisation et quelques hypothèses s'avérent nécessaires pour pouvoir garantir la cohérence dans tous les cas. La panne non-contrôlée d'une station oblige à un traitement différent.

XI.1.- Hypothèses supplémentaires

Aux hypothèses établies au préalable, il faut ajouter les suivantes:

HYPOTHESE H5

Chaque HL[i] (différent de celui de [LAM78]) est défini par les deux règles suivantes:

a)- Chaque MHL[i] envoie un message-temps (Z#k) à tous les autres MHL[j] ($1 \leq j \leq n$, $i \neq j$) lorsque son HL[i] avance de k-1 à k.

b)- Chaque MHL[i] avance son HL[i] de k-1 à k après que:
  (1)- MHL[i] ait reçu Z#k-1 de tout autre MHL[j]
  (2)- MHL[i] ait été notifié par son MGC[i] que toutes les demandes à exécuter au temps k-1 ont été terminées.

HYPOTHESE H6

Chaque station possède plusieurs queues ou files d'attente (Q):

- Cinq files d'attente (lorsque HL[i] = k):

  Q[k-3], Q[k-2], Q[k-1], Q[k], Q[k+1]

- Une file d'attente d'exécution (X[i]) ordonnée par ordre ascendant des priorités (égale à la file Q[k-2] après épuration et tri)

XI.2.- Inclusion de la temporisation (Partie I)

L'algorithme est le suivant (en supposant un processus par station):

1èr. pas: P[i] fait une demande de lecture à MGC[i]

2ème. pas: MGC[i] fournit tout de suite les données demandées (ELEC[i])

3ème. pas: A partir des éventuels calculs faits sur ELEC[i], P[i] fait une demande de MAJ[i] à MGC[i]; HL[i] = k à ce moment-là. L'ensemble des données à modifier est EMAJ[i]

4ème. pas: MGC[i] envoie MAJ[i] aux autres MGC[j]; la valeur de EST[i] est égale à k

5ème. pas: MGC[j] ($1 \leq j \leq N$) place MAJ[i] dans sa file Q[k], (attachée de son VHP) qui correspond aux demandes générées au temps logique k.

6ème. pas: La file Q[k] est traitée lorsque HL[j] change de k+1 à k+2. Ce traitement consiste à déterminer si l'ELEC concernant une MAJ n'a pas été modifié avant l'exécution de la MAJ. S'il a été modifié ou s'il sera modifié, (par des MAJ de la file Q[k] avec une priorité plus importante) MGC[j] refuse la demande et P[i] sera notifié par MGC[i]

7ème. pas: Une demande se trouvant dans X[i] est exécutée si elle remplit les deux conditions suivantes:

- elle est à la tête de la filé X[i]; et
- HP[i] - VHP [j] > TC[j] (j: station origine de la demande);

c'est à dire, qu'aucune demande est exécutée si elle n'a pas atteint sa confiance maximale.

8ème. pas: GBD[i] prévient P[i] que sa demande a été exécutée de manière satisfaisante.

La figure 14 représente le déroulement de cet algorithme.

Etant donné, que chaque station accepte le même ensemble de demandes et les exécute dans le même ordre (si elles sont exécutées concurremment, la sérialisabilité de l'exécution produit un résultat équivalent), la cohérence est maintenue. Il faut remarquer que cette cohérence est du type faible, [WIL79], par rapport au temps physique, mais du type fort par rapport au temps logique.

PROPOSITION P1

La Partie I de l'algorithme modifié de KANEKO et al. garantit la cohérence interne.

Cela est assez évident si on assume que l'algorithme original mantient la cohérence interne (sa démonstration n'est pas du ressort de cette étude). Une séquence de mises à jour dans l'algorithme modifié est identique à celle de l'algorithme original. Si cette dernière est serialisable, la première l'est aussi. En plus aucune information erronée à été introduite dans les copies. Le progrès est clair lorsque l'on compare avec

l'algorithme original où des modifications erronées ont pu être faites sur les copies. Mais tandis que l'algorithme original est capable de maintenir la cohérence mutuelle [KAN79], on ne peut rien dire sur cet aspect pour l'algorithme modifié (voir XII.4).

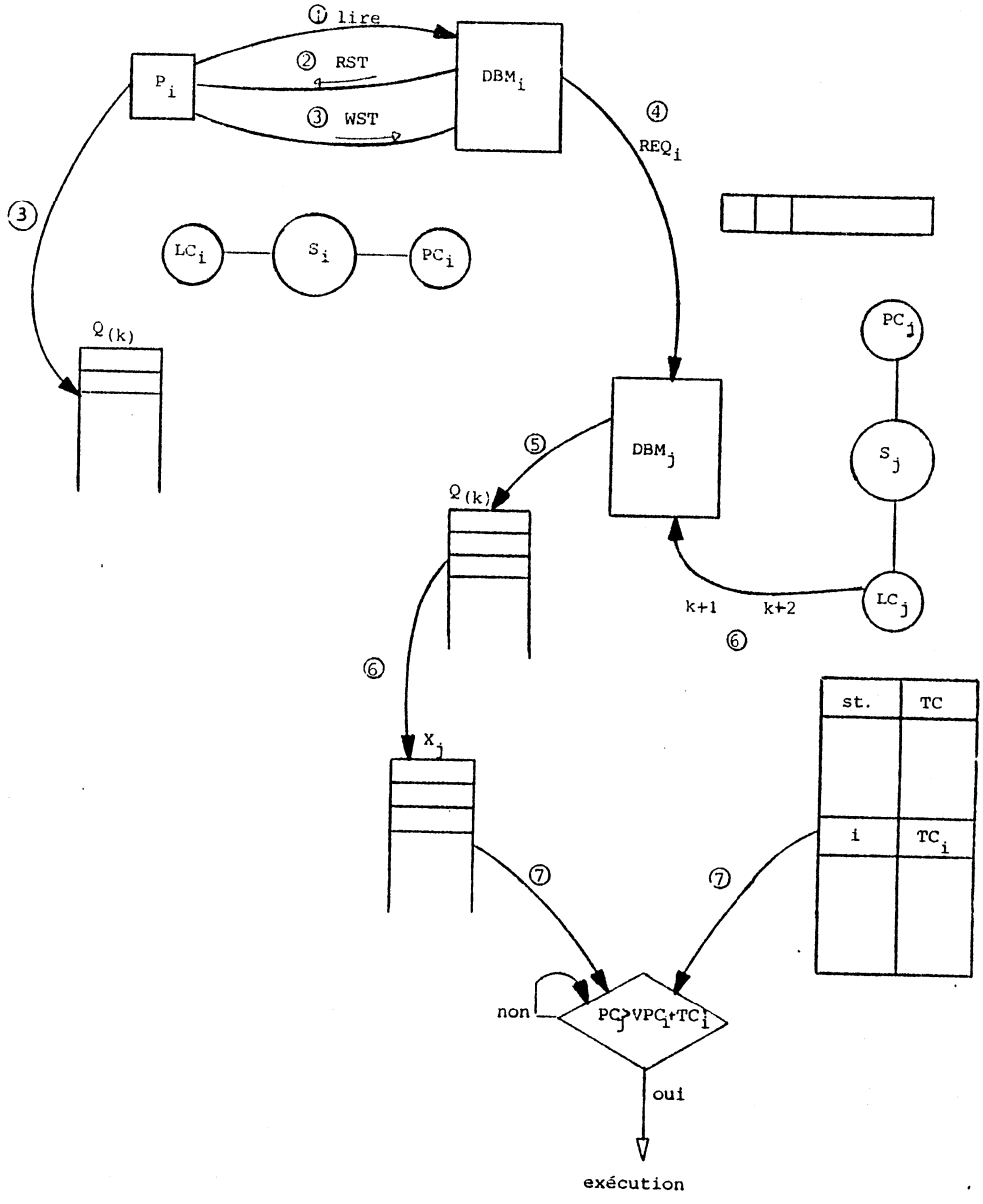


Figure 14.- Algorithme avec temporisation

XI.3.- Exemple de fonctionnement sans panne

Soit deux stations i et j qui se sont échangées des demandes A, B, C, D, E, F et G au temps logique k-2:

A, C et E ont la station i comme origine

B, D, F et G ont la station j comme origine

A l'entrée en k-2, les HP avaient les valeurs:

$$HP[i] = 20; \qquad HP[j] = 84$$

Après réception des demandes, les Q[k-2] ont les contenus suivants (avec les valeurs des HP's à la réception):

STATION i			STATION j		
	Q[k-2]	HP		Q[k-2]	HP
	G	40		G	99
IC	F	39	IC	E	98
	D	37		F	97
Ci 35 ui	E	35	Ci 30uj	C	95
Cj 50 ui	C	30	Cj 15uj	D	93
	B	27		A	91
	A	25		B	90

Les vecteurs de confinement dans chaque station figurent à coté de chaque file d'attente.
Les valeurs des HP's à l'entrée du temps logique k sont:

$$HP[i] = 55; \qquad HP[j] = 110$$

Chaque Q[k-2] avant de se transformer en file X doit être épurée (conflits supprimés) et ordonnée par priorité; si la priorité de la station j est plus grande que la priorité de la station i, les files deviennent:

X[i]	
E	35
C	30
A	25
G	40
F	39
D	37
B	27

X[j]	
E	98
C	95
A	91
G	99
F	97
D	93
B	90

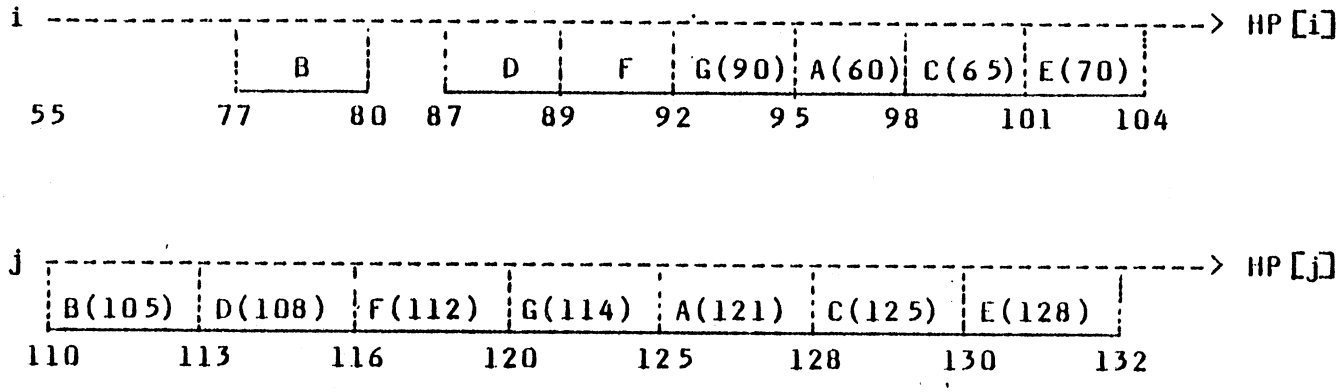
Où on aperçoit que l'ordre d'exécution est B, D, F, G, A, C et E

Si on suppose des temps d'exécution dans chaque station et pour chaque demande:

station i
B = 3 ui
D = 2 ui
F = 3 ui
G = 3 ui
A = 3 ui
C = 3 ui
E = 3 ui

station j
B = 3 uj
D = 3 uj
F = 4 uj
G = 5 uj
A = 3 uj
C = 2 uj
E = 2 uj

On peut établir un chronogramme très simple de leur exécution dans chaque station. Le numèro entre parenthèses représente le temps physique à partir duquel la demande pouvait être satisfaite (message utilisable).



Il faut faire deux remarques sur cet exemple:

1) Le mécanisme de priorités donne la priorité d'exécution statiquement à certaines stations favorisant de cette manière leur temps de réponse; un mécanisme de priorité dynamique pourrait être plus juste.
2) Les TC's dans chaque station sont établis en unités d'accord avec l'HP de la station. Cela veut dire que tous les calculs se font dans des unités standards (us) qui sont alors converties en unités correspondantes (ui, uj). Ainsi, si les calculs standards ont donné:

$$Ci = 7 \text{ us}$$
$$Cj = 10 \text{ us}$$

ces valeurs peuvent être utilisées par la station i, seulement si elles sont converties aux unités de HP[i]. Si us = 5 ui:

$$Ci = 35 \text{ ui}$$
$$Cj = 50 \text{ ui}$$

XI.4.- Panne contrôlée d'une station (Partie II)

Supposons maintenant que les deux stations i et j du sous-chapitre XII.3, aient reçu les demandes B, D, F, G, A, C, et E de la même station origine: r. L'ordre d'exécution est toujours le même et les temps d'exécution correspondent à ceux du chronogramme.

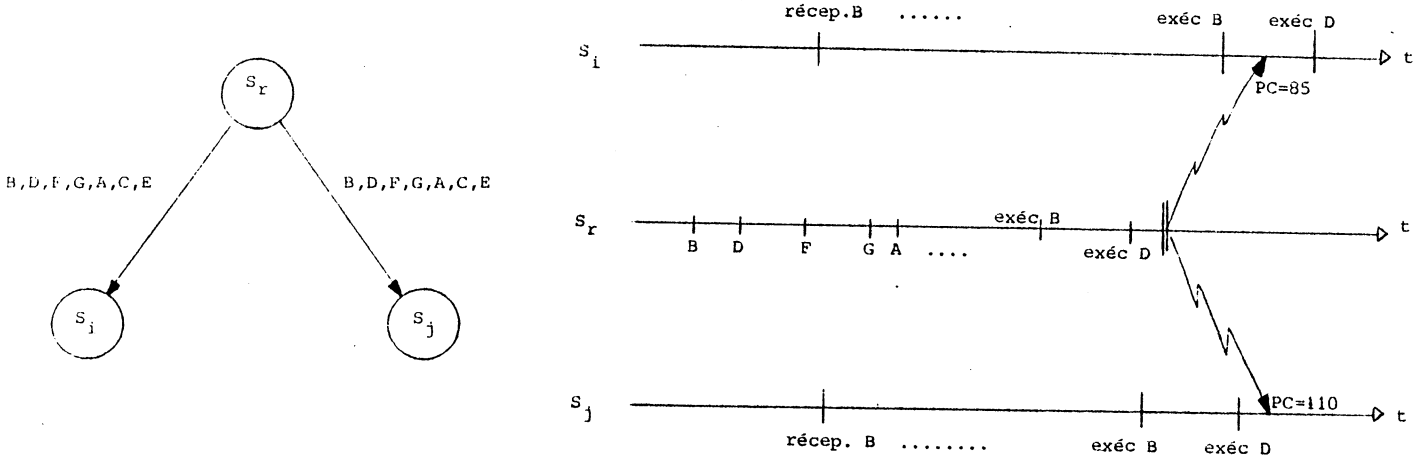


Figure 15.- Cas de panne d'une station

La station r, après avoir exécuté (éventuellement) les MAJ B et D (voir figure 15), diffuse un message de panne (les demandes: F, G,..., etc. sont erronées). Ce message peut arriver à la station i lorsque HP[i] = 85 (il devait arriver avant 89) et à la station j lorsque HP[j] = 110 (il devait arriver avant HP[j] = 112). Dans les deux cas les demandes F, G,..., etc. ne sont pas exécutées, mais pour la station i la demande D est considérée comme erronée!!.

L'algorithme décrit plus haut, doit permettre, dans le cas de la détection d'une panne dans une station, de maintenir la cohérence du système avec le maximum de transparence pour les utilisateurs.

On peut voir que l'algorithme garantit qu'aucune demande erronée soit exécutée, mais il ne peut pas garantir que toute demande non-erronée soit exécutée. Alors, deux stations quelconques peuvent arriver à deux états différents: incohérence mutuelle!!. La station j a exécuté la demande D et la station i non.

Pour surmonter ces difficultés, aux six hypothèses déjà établies, il faut ajouter les trois suivantes:

HYPOTHESE H7

Une station i qui auto-détecte une faute, génére un message du genre:

< moi, la station i, je suis en panne >

qui est diffusé à tout le système, et la station i arrête tout traitement.

HYPOTHESE H8

Le message < moi, la station i, je suis en panne > arrive à tout autre station du système de manière sûre, au moyen d'un anneau virtuel, ou par acquittement multiple, etc., de manière à ce que tous les MGC[j] ($1 \leq j \leq N$, $i \neq j$) reçoivent effectivement ce message.

HYPOTHESE H9

Le message de panne doit permettre à toute station réceptrice, de déterminer exactement et correctement les demandes d'origine i considérées comme erronées. Cela peut être envisagé de diverses manières:

-Soit les délais de transmission des messages de panne sont fixes (plus que bornés)

-Soit le message de panne de la station i, inclut

l'identification de la dernière demande d'origine i exécutée par elle-même.

-Soit le message contient la valeur du temps physique à l'instant de l'émission.

-Soit il existe une phase de synchronisation après réception du message.

D'un point de vue réaliste, la première alternative n'est pas réalisable, et les deux suivantes ne seront pas prises en compte parce que le message de panne doit être "le plus simple posssible". Il nous reste que la solution de la synchronisation, où les stations survivantes interchangent des messages sur le contenu de leurs files d'attente, de manière à les mettre au même état par rapport aux messages provenant de la station en panne.

Finalement, l'algorithme à utiliser par la station j, dans le cas de l'arrivée d'un message de panne provenant de la station i, doit exécuter les pas suivants:

1èr. pas: Suspension de l'exécution des demandes dans la station j

2ème. pas: Effacement des messages provenant de la station en faute et considérés comme erronés et exécution de ceux considérés corrects (synchronisation)

3ème. pas: Ré-organisation du réseau

4ème. pas: Re-démarrage du traitement normal

La figure 16 montre la séquence de cette procédure.

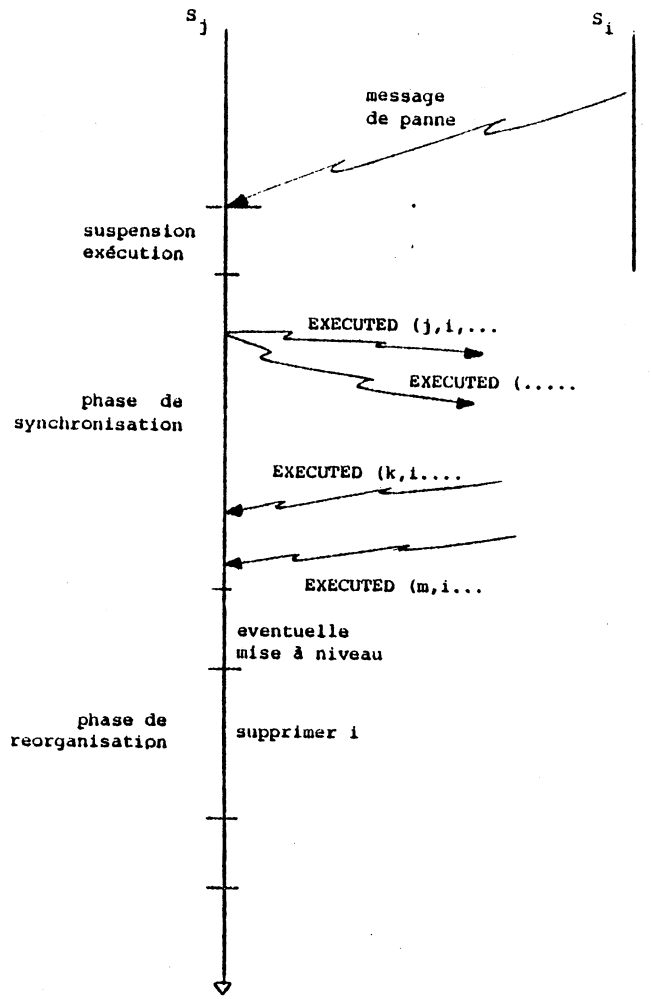


Figure 16.- Traitement du message de panne

Ainsi dans le cas de panne de la station i on trouvera dans les queues de la station j (j ≠ i),des messages erronés qui ne doivent pas être exécutés, mais aussi des messages non-erronés qui doivent être exécutés. On peut , alors, établir le principe suivant:

"Lorsqu'un message de panne de la station i, arrive à la station j aucune demande erronée, de la même source a été exécutée, et ainsi l'état de la copie est intègre".

La suspension de l'exécution de demandes est réalisée indépendamment par chaque station; elle est faite par le blocage du traitement de demandes de n'importe quelle origine dans la

file X; le reste du traitement (réception de messages, traitement des autres files, etc.) doit continuer à se réaliser d'une manière normale.

La synchronisation ou effacement des messages erronés confinés dans la station j se fait de la manière suivante:

Soit m la valeur de HP[j] à la réception du message de panne de la station i. Soit k la valeur du HL[j] au même instant; MGC[j] doit analyser chaque demande en Q[k-2], Q[k-1], Q[k], Q[k+1] et effacer celles qui réunissent les 2 conditions:

- son origine est la station i; et
- (VHP + TC[i]) > m

Il faut remarquer qu'une partie ou toute la file d'attente Q[k-2] a déjà pu être exécutée (correctement).

Cette procédure permet de traiter aussi 2 stations ou plus en mauvais fonctionnement simultanément; pour cela il faut, simplement, admettre que les messages de panne sont reçus l'un après l'autre, chacun générant l'interruption correspondante au démarrage de l'algorithme.

La ré-organisation du réseau consiste à modifier la connaissance qu'a chaque station du reste du réseau, c'est à dire de l'état de chaque station. L'état de la station, origine du message, doit passer d'actif à inactif, et par conséquence des messages ne seront plus envoyés vers cette station jusqu'à une future reconfiguration où l'état de cette station passe d'inactif à actif (re-intégration au système).

Le re-démarrage du traitement normal est réalisé, aussi, indépendamment par chaque station et après la ré-organisation; ce pas concerne le déblocage du traitement de demandes de la file X.

Dans un fonctionnement normal, on a la garantie qu'à l'arrêt de

l'activité de mise à jour, toutes les copies deviennent cohérentes au bout d'un certain temps, c'est à dire qu'à partir d'un état initial cohérent, toutes les demandes de mise à jour du système ont été éffectuées dans la même séquence dans toutes les stations. Mais dans le cas d'anomalie dans une station i, il faut maintenir la cohérence du système par rapport à cette unité i, c'est à dire que toutes les stations survivantes ont dû exécuter la même séquence des demandes provenant de la station i.

Du moment que pour une station i, les messages provenant de la station i (...., M[x-3], M[x-2], M[x-1], M[x], M[x+1], M[x+2],....) vont être exécutés dans l'ordre d'arrivée il est évident que si le message M[x] n'a pas atteint sa confiance maximale à l'arrivée du message de panne, aucun message (M[x+1], M[x+2],..., etc.) arrivés après M[x] aura atteint sa confiance maximale. Alors il suffit de montrer que toutes les stations ne traitent pas le message M[x], mais qu'elles ont traité les messages qui le précèdent dans la séquence (..., M[x-3], M[x-2], M[x-1]). La seule raison pour laquelle une station pourrait arrêter l'exécution des demandes est l'arrivée du message de panne. Et si ce message permet de déterminer quel est le dernier message correct, alors chaque station peut décider jusqu'où elle doive aller dans le traitement des messages provenant de i, et de cette manière toutes arriveront au même état. Mais dans notre cas le message de panne ne nous le permet pas et c'est pour cela qu'il nous faut une phase de synchronisation.

Cette phase de synchronisation donne lieu aux propositions suivantes:

PROPOSITION P2

La Partie II de l'algorithme de KANEKO et al. garanti la cohérence mutuelle du système.

Pour cela il faut que les stations exécutent toutes les

modifications et dans le même ordre. La garantie que toutes les stations en fonctionnement exécutent toutes les modifications, est donnée par H1 (chaque station reçoit tous les messages), H9 et par la phase de synchronisation. L'ordre commun est celui fourni par les estampilles.

Finalement, à partir des propositions 1 et 2, on peut affirmer que l'algorithme modifié de KANEKO et al., qui inclut les Parties I et II, garantit la cohérence du système: interne et mutuelle.

XI.5.- Exemple de synchronisation

Dans l'exemple prévu, les stations survivantes: i et j emettent le message: EXECUTEE (i, r, IDE[B]) et EXECUTEE (j, r, IDE[D]) respectivement.

A la réception du message EXECUTEE (j, r, IDE[D]), la station i sait qu'il y a une station: j qu'a déjà exécutée la demande D. Alors elle doit l'exécuter aussi. La station j comprend qu'elle a exécuté plus de demandes que i et qu'elle doit continuer avec le pas de ré-organisation. Finalement, les deux stations i et j ont exécuté les mêmes demandes provenant de la station r. La cohérence mutuelle est garantie. La figure 17 représente cette exemple.

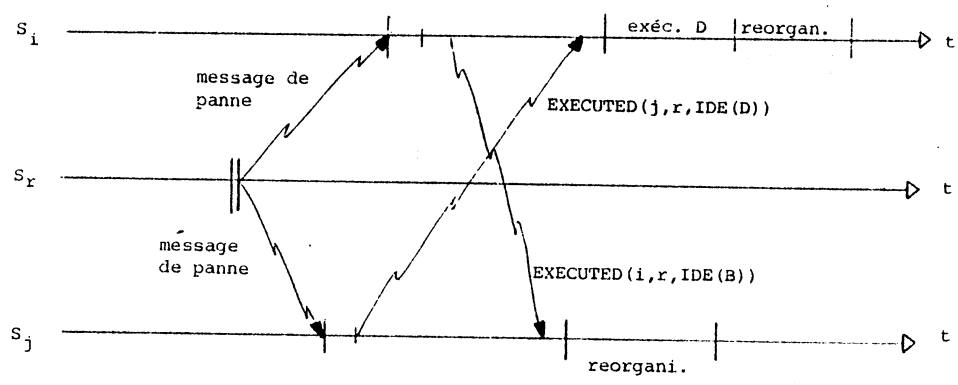


Figure 17.- Synchronisation dans le cas de panne contrôlée

XI.6.- Panne non-contrôlée d'une station

L'arrêt intempestif ou panne non-contrôlée d'une station est le type de défaillance le plus sévère du point de vue contrôle. Elle peut être la conséquence soit d'une coupure d'alimentation, soit d'une faute matérielle ou logicielle lourde, etc. Une station qui subit un arrêt n'a pas le temps de communiquer sa défaillance au reste du système (il n'y a pas de message de faute), elle coupe de manière imprévue toute communication avec le reste du système. Plus grave encore, cette défaillance peut arriver à n'importe quel moment. Par exemple, au bon milieu de la transmission d'une demande ou d'un message-temps. Par conséquence il aura des demandes, provenant de la station en arrêt, qui ne seront pas connues de toutes les stations.

Le seul moyen précis, par lequel le système peut détecter l'arrêt d'une station est l'absence du message-temps correspondant (Z#...). C'est à dire qu'une station qui attend un message-temps d'une autre, qui est à l'arrêt, va être prévenue par un mécanisme de chien de garde qui est déclanché au bout d'un temps d'attente determiné. A partir de là il y a deux alternatives à prendre: la première et la plus simple est de considérer, sans plus, la station suspecte comme défaillante. L'autre alternative est de vérifier l'état de la station suspecte, avant de la considérer défaillante. Cela se fait en lui envoyant un message-enquête qui origine une réponse-état immédiate si elle est en fonctionnement, ou rien si elle est défaillante. En bref, le système a le moyen de savoir si une station est dans un état de panne définitive ou non.

Si une station, se trouvant en HL = $\underline{k}$, détecte la panne avant d'avancer à $\underline{k+1}$, cela veut dire que la station en panne se trouvait en $\underline{k-1}$, les autres stations se trouveront soit en $\underline{k-1}$, soit en $\underline{k}$ [KAN79], mais toutes s'apercevront de la panne lorsqu'elles essayeront d'avancer à $\underline{k+1}$. Il peut arriver, qu'à la detection de la panne par des stations avec HL = $\underline{k}$, d'autres

stations détectent la panne un cycle après, c'est à dire en HL = k+1; cela peut arriver lorsque la panne se produit au milieu de la transmission du message-temps. La complexité de cette situation peut être détournée si on admet l'hypothèse suivante:

HYPOTHESE H10

Chaque MHL avance son HL de k à k+1 et il envoie le message Z#k+1 à toutes les stations comme une action atomique.

Notre seul souci sera le maintien de la cohérence mutuelle; la cohérence interne est couverte par l'hypothèse suivante:

HYPOTHESE H11

Tous les messages émis jusqu'au moment de l'arrêt intempestif sont considérés exempts d'erreur.

Ceci est facile à admettre si on considère que la panne est tellement grave (pour arrêter la station) qu'elle empêche tout fonctionnement (même mauvais) de la station. A ce moment-là, on peut admettre que la procédure à être exécutée n'a pas besoin de synchronisation et peut tourner indépendamment dans chaque station (tous les HL = k):

- Effacer de Q[k-1] tous les messages provenant de la station supposée en panne

- Ré-organiser le réseau

- Continuer le traitement normal

Si l'hypothèse 10 n'est pas applicable, il nous faut un mécanisme de synchronisation qui pourrait être semblable à celui de [CHE80], consistant d'un message envoie à toutes les autres stations: PANNE (k, j, i) lorsque la station j détecte la panne de la station i au temps logique k (avant d'avancer à k+1):

- Lorsqu'une station détecte une panne en i, elle diffuse le message: PANNE (k, j, i); alors, elle attend un message de la même nature de toutes les autres (elle a suspendu tout traitement).

- Lorsqu'elle les reçoit tous, elle vérifie qu'ils soient de la forme PANNE (k, j, i) et non PANNE (k+1, j, i). Si cette condition est satisfaite elle efface tous les messages provenant de i en Q[k-1] et toutes les stations survivantes feront la même chose et deviendront cohérentes. Si on veut être plus exigeant, on pourrait utiliser le message PANNE (k, j, i, IDE[?]) où IDE[?] est l'identificateur de la dernière demande de i exécutée, si elle existe. Si la condition n'est pas satisfaite, il faut utiliser un protocole de synchronisation qui fasse coincider toutes les bases sur la file Q[k-1] et fasse effacer les messages provenant de i seulement à partir de Q[k].

Si H10 est applicable, le traitement de deux ou plus stations en panne simultanément est tout à fait possible avec la même procédure. Si H10 n'est pas applicable, le protocole de synchronisation avec PANNE (k, j, i) doit considèrer cette possibilité lorsqu'une station j attend les messages PANNE des autres stations.

XII.- Algorithme de Herman et Verjus

XII.1.- Hypothèses supplémentaires

Pour résoudre notre problème de confinement en utilisant l'algorithme de [HER79], on se limitera à un des aspects qui couvre cet algorithme, celui qui concerne les mises à jour (il est concerné aussi par la synchronisation des lectures, écritures et estimations). Les hypothèses considérées sont, en général, les mêmes établies pour l'algorithme précédent (H1-H9) avec des différences pour H3, H5 et H6:

HYPOTHESE H3'

L'identification d'une MAJ est réduite à la valeur de l'HL à l'instant de l'émission de la demande, auquel on rajoute le numèro de la station.

HYPOTHESE H5'

La notion d'horloge logique utilisée est celle de [LAM78], où les trois règles suivantes conditionnent sa réalisation:

a)- Chaque MHL avance son HL entre deux événements locaux successifs.

b)- Le MHL[i] de la station réceptrice d'un message provenant de la station j, met la valeur de son HL[i] au:

$$\max (HL[i], EST[j]+1)$$

HYPOTHESE H6'

Chaque station S[i] a une queue Q[j] pour chaque station S[j] ($1 \leq j \leq N$) du sytème (incluse elle même), destinée à contenir tous les messages provenant de la station S[j].

XII.2.- Inclusion de la temporisation

L'algorithme de cohérence avec confinement est le suivant:

1èr. pas: P[i] emet une demande de lecture (LEC[i]) à MGC[i] (HL[i] = k).

2ème. pas: MGC[i] génère une commande DMAJ[i] (début de mise à jour)

3ème. pas: MGC[i] envoie DMAJ[i] à tous les autres MGC[j] ($1 \leq j \leq N$) du système, avec l'estampille correspondante (EST[i] = k).

4ème. pas: MGC[j] ($1 \leq j \leq N$) place DMAJ[i] dans la queue Q[i]. MGC[i] place, en plus, LEC[i] dans la queue Q[i]. L'arrivée de DMAJ[i] donne l'accès exclusif à la queue Q[i] au processus P[i].

5ème. pas: MGC[i] fourni les données demandées (ELEC[i]) à P[i], lorsque LEC[i] et DMAJ[i] sont exécutées (selon "protocole d'exécution" plus loin).

6ème. pas: P[i] emet une demande MAJ[i] à MGC[i], après d'éventuels calculs sur ELEC[i].

7ème. pas: MGC[i] envoie MAJ[i] à tous les autres MGC[j] ($1 \leq j \leq N$).

8ème. pas: MGC[j] place MAJ[i] dans la file Q[i], et lui attache la valeur de HP[j] au moment de la réception (VHP[j]). L'arrivée de MAJ[i] débloque la queue Q[i].

9ème. pas: MGC[j] ($1 \leq j \leq N$) exécute son "protocole d'exécution" qui consiste à analyser toute demande MAJ:

- qui se trouve à la tête d'une queue; et
- qui a comme estampille la plus ancienne de toutes;

L'analyse consiste à vérifier si le temps de confinement: TC[i] pour une demande MAJ[i] est satisfait; si la condition est vraie la demande est exécutée, autrement elle doit attendre que la condition soit vraie.

L'analyse exige que toutes les files soient non-vides; s'il y a des files vides la station i attend que la condition de temps de confinement soit satisfaite par la demande la plus ancienne, avant d'envoyer un message spécial (ENQ) aux stations dont les files sont vides. A la réception de ce message toute station doit répondre immédiatement et par conséquence sa file devient non-vide au bout d'un temps fini.

Il faut remarquer la différence de traitement de LEC[i] par rapport à MAJ[i]; la première n'est pas diffusée et n'a pas besoin d'attendre pour satisfaire la condition de temps de confinement.

XII.3.- Panne contrôlée d'une station

Le traitement est semblable à celui appliqué à l'algorithme modifié de [KAN79] à la différence que la file X[i] n'existe plus dans l'algorithme de Herman et Verjus.

Après la réception du message de détection de panne les stations survivantes doivent se mettre d'accord sur la dernière modification, provenant de la station en panne, à être exécutée. Les autres demandes, en attente d'exécution seront éliminées. Cette phase de synchronisation est identique à celle montrée en XII.5 et elle est également basée sur les hypothèses H7, H8 et H9 du chapitre XII.4.

XII.4.- Panne non-contrôlée d'une station

Dans le cas d'un mécanisme de communication à diffusion sûre la solution est très simple. Chaque station S[i] détecte la panne d'une station S[j] après avoir exécuté la dernière MAJ[j] provenant de la station S[j]. Cette MAJ[i] est la même pour toutes les stations et par conséquence la cohérence est maintenue.

Lorsque le mécanisme de communication n'est pas un mécanisme à diffusion sûre, quelques stations ont pu exécuter ou recevoir des demandes que les autres stations ne connaissent pas. Dans ce cas-là, il nous faut un mécanisme de synchronisation pour pouvoir mettre toutes les stations survivantes dans un état cohérent. La solution pourrait être un protocole de synchronisation semblable à celui de [CHE80] avec des messages PANNE (IDE[?], j, i), tel que celui envisagé pour l'algorithme modifié de [KAN79]:

- Néanmoins, il faut prévoir qu'une station a pu, déjà exécuter une demande qu'une autre station ne posséde pas encore. Dans ce cas-là la synchronisation devient très problématique parce que l'on aura besoin du transfert des demandes entre stations.

- Dans les autres cas la synchronisation est très simple. L'interchange de messages de panne: PANNE(IDE[i], j, i) est suffisant. Chaque station vérifie quelle a été la demande la plus jeune provenant de i pas encore exécutée dans le système et elle exécute toutes les antérieures.

XIII.- Analyse comparée des algorithmes

L'objectif de cette analyse n'est pas d'établir une hiérarchie de qualité entre les deux méthodes modifiées. Cela est une tâche tout à fait impossible étant donné le nombre de critères à prendre en compte [WIL79]. L'analyse se limite à montrer quelques caractéristiques de ces mécanismes qui pourront nous permettre de décider sur la convenance de l'un par rapport à l'autre en ce qui concerne un objectif précis. Cela pourrait nous permettre un choix très relatif pour un objectif global: le maintien de la cohérence dans les systèmes répartis. Une analyse plus détaillé est, peut être, à faire. Cela implique l'utilisation des outils de simulation et/ou d'évaluation mathématique, qui exigent à leur tour une formalisation rigoureuse de la méthode.

De toute la gamme de critères existants, on a choisi ceux qui, à notre avis, semblent les plus importants pour notre problème:

- la correction;
- l'efficacité, qui concerne:
  - le coût (de communication),
  - la simplicité,
  - la rapidité,
  - le parallèlisme;
- la robustesse.

Le but final est de pouvoir répondre aux questions suivantes:

- les deux algorithmes, sont-ils corrects après modification?

- l'inclusion de la temporisation, modifie-t-elle les coûts de communication de chaque méthode?

- est-ce qu'une modification est plus simple que l'autre?

- quel est le rapport des ralentissements introduits par

les deux modifications?

- le parallèlisme est-il modifié?, si oui dans quelle mesure?

- l'amélioration globale de la robustesse, produite par ces deux modifications est-elle substantielle?, y a-t-il une différence entre elles?

On peut avancer que les critères de coût et de parallèlisme n'ont pas été modifiés par rapport aux originaux. Les autres critères ont été modifiés à divers degrés.

Pour ce qui concerne la correction, on sait par [BER81] que beaucoup de mécanismes de maintien de la cohérence sont difficiles à démontrer corrects (et même quelques-uns sont incorrects). Néanmoins on peut dire que les mécanismes doivent, au moins, assurer la sérialisabilité des actions et éviter les blocages ("interblocage", "blocage permanent", etc.). La correction des horloges et l'exécution des mises à jour dans l'ordre chronologique garantissent la sérialisabilité des actions. Pour la plupart, la correction de nos mécanismes s'appuie évidemment sur la correction de leurs mécanismes d'origine. La simplicité de la modification nous permet d'affirmer que la relative correction des algorithmes n'a pas été modifiée. Sauf le décalage inclus, il n'a pas eu de changement dans la séquence d'opérations originales. Il faut remarquer que l'algorithme de Kaneko peut reporter à l'infini l'exécution d'une demande refusée par conflict.

Le coût de communication en état normal (sans panne, ni conflict) peut être évalué en fonction du nombre de messages nécessaires pour traiter une demande de mise à jour. Ce nombre-là n'est pas modifié par l'inclusion de la temporisation. Il est toujours (1+1/r) pour une liaison à diffusion et (N-1)(1+1/r) pour une liaison bipoint dans le cas de Kaneko (r demandes par cycle de temps logique). Pour Herman et Verjus où il n'a qu'une demande

par cycle, le nombre n'a pas changé non plus: deux messages en cas de diffusion et 2(N-1) messages en cas de bipoint.

La simplicité des deux modifications est équivalente, sauf pour le cas de panne non-contrôlée. La seule solution possible pour Herman et Verjus est la synchronisation avec un éventuel transfert de messages. Pour Kaneko on a adopté un schéma plus simple mais qui en étant plus simple est moins performant (perte d'information correcte).

L'inclusion de la temporisation produit un ralentissement dans les deux cas. Mais pour la modification de Kaneko le ralentissement est moins sensible que celui de Herman et Verjus où l'application absorbe tout le retard. Dans un cas idéal (pour Kaneko) le ralentissement de l'application pourrait être nul: le temps de confinement est égal à la période de l'horloge.

Le dernier critère est peut être le plus important dans notre cas. Il concerne la robustesse de mécanismes dans le cas de panne contrôlée et non-contrôlée d'une station. Ce critère dépend du type de liaison de communication utilisé: bipoint ou diffusion. Pour une liaison bipoint:

- Pour la panne contrôlée, le traitement est presque le même pour les deux algorithmes; il leur faut une étape de synchronisation pour garantir la cohérence mutuelle des stations survivantes avec une utilisation maximale de l'information correcte.

- Pour la panne non-contrôlée, les algorithmes doivent pouvoir contrôler trois schémas possibles. Il faut rappeler que l'algorithme de Kaneko et al. détecte cette panne par l'absence du message-temps tandis que l'algorithme de Herman et Verjus le font par un état de queue vide.

    + Panne non-contrôlée propre (celle qui arrive entre deux modifications): est prise en compte par Kaneko

modifié de manière très simple, avec une légère perte d'information supposée correcte. L'algorithme de Herman et Verjus modifié a besoin d'une phase de synchronisation, sinon la cohérence mutuelle est perdue.

+ Panne non-contrôlée impropre 1 (celle qui arrive au beau milieu d'une modification) est traitée par Kaneko de la même manière que la panne propre. L'algorithme d'Herman et Verjus modifié a toujours besoin d'une phase de synchronisation.

+ Panne non-contrôlée impropre 2 (qui arrive au milieu d'un message-temps et seulement pour Kaneko) est traitée de manière simple avec l'implantation des actions atomiques. Autrement il faut une étape de synchronisation.

Pour une liaison à diffusion il y a qu'un seul type de panne non-contrôlée: propre. Dans ce cas-là, l'algorithme d'Herman et Verjus modifié n'a pas besoin d'une étape de synchronisation. Finalement la récupération dans le cas d'une panne non-contrôlée est plus simple en [KAN79] modifé. Il garantit absolument le maintien de la cohérence mutuelle sans avoir recours à la synchronisation des stations (avec une éventuelle perte des messages corrects). Dans [HER79] il faut l'accord des stations par rapport à la dernière demande exécutée.

XIV.- Conclusions

Bien qu'une implantation de ces algorithmes n'ait pas été possible pour des raisons de temps, une partie des objectifs ont été pourtant atteints.

La faisabilité de la prise en compte de la notion de latence de détection est effective. Les algorithmes qui utilisent des horloges logiques admettent sans trop de difficultés l'insertion d'une temporisation. La modification de ces algorithmes est tout à fait transparente à l'utilisateur, qui pourrait seulement percevoir un léger retard (qui n'a pas été évalué) dans le temps d'exécution de son application. Ce retard d'exécution, dans le cas de l'algorithme de Kaneko est absorbé partiellement par le mécanisme même. Il est évident qu'un compromis doit être recherché entre ce retard et le surplus de fiabilité atteinte. Aucun autre symptôme négatif ne sera visible. Au contraire, le concepteur du système peut envisager une diminution des points de reprise, étant donné le contrôle existant sur la panne.

La surcharge sur le réseau de communication, on l'a vu, est différente pour les deux algorithmes, mais n'est pas influencée par la modification, au moins pour le fonctionnement sans panne. Pour le traitement de pannes, l'objectif d'utilisation maximale des informations correctes complique partiellement l'algorithme de Kaneko, et surcharge le réseau avec (N-2)x(N-1) messages pour N stations. Avec un réseau à diffusion cela se réduirait à N-1 messages.

L'efficacité réelle de cette approche ne peut pas être évaluée sans une implantation réelle. Il reste à demontrer et à évaluer le gain que l'on peut obtenir avec cette approche et la possibilité de prise en compte de la notion de latence par d'autres algorithmes.

Un prochain objectif est de mettre en oeuvre une application des

algorithmes et mécanismes conçus précédemment, sur un système convenable. On pourrait ainsi, évaluer réellement les perfomances obtenues après implantation et les répercussions éventuelles sur l'arrangement de données et sur l'organisation de l'application.

A notre avis, l'acceptation de l'inclusion d'une temporisation dans un système multi-niveaux est semblable à l'acceptation de la redondance comme un moyen d'accroître la fiabilité d'un système. Il y a quelques années, le coût était le principal obstacle à l'utilisation de la redondance. L'évolution de la technologie, origine de la baisse des coûts du matériel, a permis à la redondance de devenir un méthode applicable aux systèmes d'aujourd'hui. La temporisation doit suivre le même chemin. Des composants étant de plus en plus rapides, des réalisations (temps réel par exemple) qui fournissaient un niveau de fiabilité pour une vitesse donnée, peuvent accroître leur fiabilité en conservant égaux leur temps d'exécution.

Références

[AND81] ANDERSON T., LEE P. A.
"Fault Tolerance: Principles and Practice"
Prentice-Hall, 1981

[AND82] ANDERSON T., LEE P. A.
"Fault tolerance terminology proposals"
Proceed. of the 12th. Fault Tolerant Computing Symposium, California, USA, Juin 1982

[AVI76] AVIZIENIS A.
"Fault-Tolerant Systems"
IEEE Transactions on Computers, Vol. C-25, No. 12, Decembre 1976

[AYA79] AYACHE J. M., AZEMA P., DIAZ M.
"OBSERVER: A concept for on-line detection of control errors in concurrent systems"
Proceed. of the 9th. Fault Tolerant Computing Symposium, Madison, USA, Juin 1979

[BAN80] BANINO J. S., CARISTAN A., GUILLEMONT M., MORISSET C., ZIMMERMANN H.
"CHORUS: An architecture for distributed systems"
INRIA, R.R. No. 42, Novembre 1980

[BEN82] BEN RONDHANE M., COURTOIS B.
"Error confinement/data recovery in distributed systems"
2nd. Symposium on Reliability ,in Distributed Software and Database Systems, Pittsburgh, USA, Juillet 1982

[BER79] BERNSTEIN P. A., GOODMAN N.
"Approaches to concurrency control in distributed data base systems"
National Computer Conference, 1979

[BER81] BERNSTEIN P. A., GOODMAN N.
"Concurrency control in Distributed Database Systems"
ACM, Computing Surveys, Vol. 13, No. 2, Juin 1981

[BOU79] BOUCHET P., DAMESTOY B., GELENBE E., JOUVE M., PARENT CH., PARFUS P.
"Procédures de reprise"
Monographies d'informatique de l'AFCET, 1979

[CHE80] CHENG W. K., BELFORD G. G.
"A clock synchronisation algorithm for update synchronization on local broadcast networks"
Proceed. of Distributed Computing Systems, 1980

[COR81] CORNAFION
"Systèmes Informatiques Répartis: Concepts et Techniques"
Dunod Informatique, 1981

[COU79] COURTOIS B.
"Some results about the efficiency of simple mechanisms for de detection of microcomputer malfunctions"
Proceed. of the 9th. Fault Tolerant Computing Symposium, Madison, USA, Juin 1979

[COU81] COURTOIS B., MARINESCU M., PONS J. F.
"SKALP: SKeleton Architecture for fault tolerant distributed Processing"
EUROMICRO journal, Vol. 7, No. 5, Mai 1981

[COU81a] COURTOIS B.
"A methodology for on-line testing of microprocessors"
Proceed. of the 11th. Fault Tolerant Computing Symposium Portland, USA, Juin 1981

[COU82] COURTOIS B., EYNARD J., GEAY F., MARCHAL P., POSTIGO C.
"Test en-ligne et hors-ligne de microprocesseurs: classification et applications"
Rapport de Recherche IMAG, No. 276, Nov. 1981

[ENS78] ENSLOW P. H. jr.
"What is a "Distributed" Data Processing System"
Computer, Janvier 1978

[GEL78] GELENBE E., DEROCHETTE D.
"Performance of rollback recovery systems under intermittent failures"
CACM Vol. 21, No. 6, Juin 1978

[HER79] HERMAN D., VERJUS J. P.
"An algorithm for maintaining the consistency of multiple copies"
Proceed. of Distributed Computing Systems, 1979

[HOR73] HORNING J. J., RANDELL B.
"Process Structuring"
ACM Computing Surveys 5, 1, 1973

[HOR74] HORNING J. J., LAUER H. C., MELLIAR-SMITH P. M., RANDELL B.
"A program structure for error detection and recovery"
Operating Systems, Lecture Notes in Computer Science No. 16, Springer-Verlag

[KAN79] KANEKO A., NISHIHARA Y., TSURUOKA K., HATTORI M.
"Logical clock synchronization method for duplicated database control"
Proceed. of Distributed Computing Systems, 1979

[KIM79] KIM K. H.
"Error detection, reconfiguration and recovery in distributed processing systems"
1st. Intern. Confer. on Distributed Computing Systems, Alabama, USA, Octobre 1979

[KIM82] KIM K. H.
"Issues in design of fault-tolerant distributed systems"
2nd. Symposium on reliability in distributed software and database systems, Pittsburgh, USA, Juillet 1982

[JOU79] JOUVE M., PARENT CH.
"Qu'est-ce qu'une base de données cohérente?"
Journées AFCET-IP: Bases de Données Cohérentes, Paris Mai 1979

[LAM73] LAMPSON B. W.
"A note on the confinement problem"
CACM, Octobre 1973

[LAM78] LAMPORT L.
"Time, clocks, and the ordering of events in a distributed system"
ACM Communications, Vol. 21, No. 7, Juillet 1978

[LAM78-1] LAMPORT L.
"The implementation of reliable distributed multiprocess systems"
Computer Networks 2, 1978

[LEE82] LEE P. A., MORGAN D. E.
"Fundamental concepts of fault tolerant computing. Progress Report, Proceed. of 12th. Fault Tolerant Computing Symposium, Juin 1982

[LEL78] LE LANN G.
"Algorithms for distributed data-sharing systems which use tickets"
Proceed. of the 3rd. Berkeley Workshop on Distributed Data Management and Computer Networks, Août 1978

[LOM77] LOMET D. B.
"Process structuring, synchronisation and recovery using atomic actions"
Proceed. ACM Conf. on Language Design for Reliable Software, SIGPLAN Notices 12, 3, Mars 1977

[MAR79] MARINESCU M.
"LISA: A communication mechanism for local networks"
1st. Conf. on Distributed Computing Systems, Alabama, USA, Octobre 1979

[MET76] METCALFE R. M., BOGGS D. R.
"ETHERNET: distributed packet switching for local computer networks"
CACM, Vol. 19, No. 7, July 1976

[MIL80] MILENKOVIC M.
"Synchronization of concurrent updates in redundant distributed databases"
Proceed. of the International Symposium on Distributed Data Bases, Paris-FRANCE, Mars 1980

[OSD79] OSDEN S.
"The DC-9-80 digital fligth guidance system's monitoring techniques"
Proceed. of the AIAA Guidance and Control Conference Boulder, Août 1979

[POS82] POSTIGO G., COURTOIS B.
"Mutual and Internal Consistency of Multiple Copies taking into account a detection latency"
Rapport de Recherche IMAG, No. 311, Septembre 1982

[RAN75] RANDELL B.
"System structure for software fault tolerance"
IEEE Transactions on Software Engineering, Vol. SE-1, No. 2, Juin 1975

[RAN78] RANDELL B.
"Reliable Computing Systems"
Lecture notes in Computer Science No. 60, 1978

[RAN78a] RANDELL B., LEE P. A., TRELEAVEN P. C.
"Reliability Issues in Computing System Design"
ACM Computing Surveys, Vol. 10, No. 2, Juin 1978

[SHA78] SHAPIRO R. M., MILLSTEIN R. E.
"Failure recovery in a distributed data base system"
Proceed. of Distributed Computing Systems, 1978

[SHE75] SHEDLETSKY J. J., McCLUSKEY E. J
"The error latency of a fault in a combinational digital circuit"
Proceed. of the 5th. Fault Tolerant Computing Symposium, Paris, FRANCE, Juin 1975

[SHE78] SHEDLETSKY J. J.
"A rollback interval for networks with an imperfect self-checking property"
IEEE Transaction on Computers Vol. C-27, No. 6, Juin 1978

[THO79] THOMAS R. H.
"A majority consensus approach to concurrency control"
ACM TODS, Vol. 4, No. 2, 1979

[WIL79] WILMS P.
"Etude et comparaison d'algorithmes de maintien de la cohérence dans les bases de données reparties"
Thèse Docteur-Ingenieur INPG, Novembre 1979

SECONDE PARTIE

RESUME

Cette seconde partie concerne la réalisation des algorithmes pour le test fonctionnel hors-ligne de la partie contrôle du microprocesseur M6800. Ces algorithmes ont été déduits à partir de ceux établis pour le test fonctionnel en-ligne.
La conception de ces algorithmes est basée sur un modèle global de pannes de l'unité centrale integrée, qui a permis la définition et la classification des pannes considérées. Pour le test hors-ligne, les différentes procédures pour le test de chaque instruction (phases adressage et opération), dans chaque classe, ont été dégagées. On aborde aussi des considérations d'implantation sur des machines de test (classiques et légères). Les conditions pour le test en-ligne de chaque instruction, un moyen de renforcement des hypothèses de panne et une partie du logiciel dejà réalisé sont donnés en annexe.

NOTATIONS

UN	111....11
ZERO	000....00
V_o	valeur initiale
V_f	valeur finale
R_i	registre quelconque
V·	condensé multiplicatif du vecteur V
DB	bus données
A, B	registres accumulateurs
CO	registre compteur ordinal
X	registre d'index
SP	registre pointeur de pile
T	registre tampon
C	bit retenue (registre d'état)
V	bit dépassement (registre d'état)
Z	bit zéro (registre d'état)
N	bit négatif (registre d'état)
I	bit masque d'interruption (registre d'état)
H	bit demi-retenue (registre d'état)
BUS	valeur contenue dans un bus

Sommaire Seconde Partie

*I.- Introduction 93
II.- Le test de circuits 96
II.1.- Le test 96
II.2.- Types de test pour microprocesseur 97
III.- Le microprocesseur M6800 100
*III.1.- Test du M6800 101
*III.2.- Test de la partie opérative 102
*IV.- Rappels du modèle de base 104
*V.- Algorithmes du test en-ligne 106
*VI.- Algorithmes du test hors-ligne 115
VI.1.- Classes 0, I, II et III 120
VI.1.1.- Phase adressage (classes 0 et II) 120
VI.1.1.1.- Adressage immédiat 120
VI.1.1.2.- Adressage direct 121
VI.1.1.3.- Adressage indexé 121
VI.1.1.4.- Adressage étendu 122
VI.1.2.- Phase adressage (classes 0, I, II et III) 122
VI.1.2.1.- B.1: Non-modification des registres opératifs 122
VI.1.2.1.1.- Adressage immédiat 123
VI.1.2.1.2.- Adressage direct 123
VI.1.2.2.- B.2: Non-modification des registres non-opératifs 123
VI.1.3.- Phase opération 123
VI.1.3.1.- Test des classes 0 et III 123
VI.1.3.2.- Test des classes I et II 126
VI.2.- Classe IV 127
VI.2.1.- Adressage immédiat 127
VI.2.2.- Adressage direct 128
VI.2.3.- Adressage indexé 128
VI.2.4.- Adressage étendu 133
VI.2.5.- Adressage relatif 133
VI.2.6.- Phase opération 133

VI.3.- Classe V 136
*VII.- Implantation 140
VII.1.- Machine de test classique 140
VII.2.- Machine de test légère M6800 140
*VIII.- Conclusions 143
Références 144
Annexes

I.-Introduction

L'utilisation de plus en plus large des circuits LSI a créé l'environnement propice au développement accéléré des techniques de test des systèmes digitaux. Ces techniques sont utilisées pour la vérification en fin de fabrication et/ou pour la validation durant l'utilisation.

Le test logique des circuits, c'est à dire la vérification de la fonction logique réalisée par le circuit, peut être appliqué aux circuits dans leur environnement de travail (test en-ligne) ou en dehors (test hors-ligne), [COU82]. Quelque soit le cas, généralement, on crée et on applique une séquence d'entrées (génération de vecteurs de test) et on observe les sorties résultantes.

Pour les circuits de dessin régulier ou simple, où la structure interne est relativement facile à connaître, le test analytique représente une solution adéquate (mémoires, operateurs, etc.). Pour les circuits complexes, difficiles à connaître, une bonne solution est le test fonctionnel.

La complexité des Unités Centrales Intégrées (UCI) force à l'application systématique de ces méthodes. L'UCI est considérée comme un système de blocs fonctionnellement et structurellement différents, pour lesquels l'application des méthodes est aussi différente.

Une étude rapide de l'état de l'art concernant les méthodes de test fonctionnel pour microprocesseur a été faite dans [COU80-1]. Son analyse révéle les caractéristiques les plus remarquables de ces méthodes et a permis la conception d'une nouvelle méthode, dont la réalisation est l'objet de cette étude:

[BRE80] développe des primitives de test fonctionnel, mais pour des circuits MSI (compteurs, etc.).

[HAC75], [AND76], [SCR78], [BIS78] et [LIP79] ont développé des procédures de test mais sans être trés explicites.
[LEA73] reconnaît que la partie contrôle doit être testée au niveau des instructions.
[BAR76] et [CHI76] décomposent le microprocesseur en unités fonctionnelles (compteur ordinal, mémoire, etc.), mais les hypothèses testées ne sont pas précisées.
[SRI77] et [BAL79] considérent, également, le découpage en unités fonctionnelles, une hiérarchie d'instructions (instructions plus fiables), une hiérarchie d'unités (unités plus complexes) et la notion de "poids" d'un programme; mais ils ne considérent pas le test de séquencement des commandes de la partie contrôle.
[AKE78] utilise les diagrammes de décision binaire comme modèle, mais reconnaît qu'ils sont positifs seulement si l'utilisateur pense qu'ils testent effectivement le circuit considéré.
[THA78] a défini des hypothèses de panne et des procédures de test fonctionnel pour microprocesseur, mais dans sa méthode:

- le fonctionnement des opérateurs doit être testé au préalable,
- il faut connaître les commandes sur l'ALU et la structure de l'ALU,
- il ne considère pas le test de la gestion d'un registre d'état,
- les hypothèses de panne sont particulièrement orientées vers la partie opérative, ce qui est confirmé en [THA79-1] et [THA79-2].

Le test fonctionnel de la partie contrôle des UCI's considère, généralement, cette partie comme un élément passif, sans prendre en compte sa capacité intelligente; en conséquence le coût du test (génération des vecteurs de test, logiciel et matériel extra) atteint un niveau appréciable. Dans la méthode développée en [COU80-1], pour le test en-ligne, on utilise cette caractéristique intelligente pour tester l'élément lui même, ce qui représente une large économie sur le coût du test.

Pour la réalisation d'un test hors-ligne on a choisi le microprocesseur M6800 en raison du niveau d'études sur ce microprocesseur, l'étendue de son utilisation et les résultats disponibles. La validation de la stratégie développée pourrait, sans doute, encourager l'application de ces méthodes pour le test d'autres microprocesseurs (M68000, IAPX-432, etc.).

Cette étude concerne la création et la programmation des algorithmes de test hors-ligne pour la partie contrôle des UCI's, à partir des algorithmes conçus pour le test en-ligne [COU80-1].

Le chapitre II est un rappel de notions de test de circuits et de types de test pour microprocesseurs. Le chapitre suivant montre brièvement le microprocesseur M6800, ses principales caractéristiques et, selon notre méthode, le cadre général du test du M6800. Dans le chapitre suivant on fait un rappel des fondements du modèle de base, support de cette méthode. Le chapitre V décrit les algorithmes qui ont été déduits pour le test en-ligne. Le chapitre suivant justifie la démarche suivie pour l'obtention des algorithmes du test hors-ligne, il présente une modification de la classification des pannes, l'organisation du logiciel, ses limites et restrictions, avec quelques exemples d'algorithmes. L'implantation sur une machine de test légère est ensuite abordée. L'annexe A montre les conditions nécessaires pour le test en-ligne, établies en [COU80-1]. L'annexe B montre la possibilité d'extension des hypothèses de panne établies. L'annexe C montre, en ASSEMBLEUR M6800, une partie du logiciel concernant le test de la classe IV.

II.- Le test de circuits

Tel qu'on l'a vu dans la première partie, chapitre II, il y a deux stratégies complémentaires utilisables pour atteindre un certain degré de fiabilité: la prévention de fautes et la tolérance aux fautes [AVI76]. Cette seconde partie concerne la première stratégie, où le niveau souhaité de fiabilité est obtenu par la suppression de toutes les causes possibles de non-fiabilité. Mais dans cettte étude on cherche seulement, à s'assurer que le système (le microprocesseur) n'a pas des fautes d'origine matérielle.

La prévention des fautes est réalisée par des techniques qui peuvent être classées en deux groupes [AND81]:

- Techniques pour éviter les fautes
- Techniques pour supprimer les fautes

Le premier groupe est utilisé pour éviter d'introduire des fautes dans le système au moment de sa réalisation. Il est constitué par les méthodologies de conception, l'utilisation de composantes fiables, le contrôle de la qualité, la formalisation de programmes, etc. Le second est utilisé pour trouver et supprimer les fautes qui ont été introduites dans le système par inadvertance. Il comprend la vérification du système et la suppression des fautes que l'on a rendue observables. Les techniques considérées sont: le test et la validation.

II.1.- Le test

Dans un sens très général, on peut dire que le test est le processus qui consiste à appliquer une séquence d'entrées à un circuit, à observer la séquence de sortie et à la comparer avec une séquence de sortie pré-calculée. Toute différence est considérée comme la manifestation d'une faute. Les fautes peuvent être introduites soit à la conception, soit à la fabrication,

soit pendant le stockage, soit pendant l'utilisation. Par conséquent, le test doit être applicable pendant toute la vie du circuit, et sa conception dépendra des fautes recherchées.

Les fautes d'origine matérielle étant de deux types: logiques et paramètriques, on peut classer le test en fonction de celles-ci: le test paramètrique ou dynamique et le test logique ou statique.

- Test paramètrique: on mesure les niveaux de voltage et de courant, en vérifiant également le fonctionnement temporel du circuit.

- Test logique: consiste en l'application de séquences d'entrées binaires en analysant les sorties pour vérifier le fonctionnement logique correct du circuit.

En général le test est réalisé au moyen d'un équipement de test automatique (ATE). Dans certains cas on utilise un circuit supposé correct, comme référence, pour faire la comparaison. Dans d'autres cas, l'équipement contrôlé par un ordinateur, gère des programmes de test qui génèrent des séquences d'entrées, interprétant et comparant les sorties. Une des phases la plus ardue et coûteuse du test est la génération de séquences de test à être appliquées. Pour certains systèmes (temps réel), les programmes de test font partie du système d'application. Ils sont appliqués périodiquement en tenant compte de la possibilité du matériel ou du logiciel, utilisé pour le test, d'avoir aussi des fautes.

II.2.- Types de test pour microprocesseur

En plus du critère qui permet de classer les types de test en paramètriques et logiques, on peut trouver d'autres critères de classification. Ainsi, une classification spécifique pour le test d'unités centrales intégrées, et dont certains résultats peuvent être utiles au test des cartes ou de systèmes, répartit les types de test en deux groupes: en-ligne et hors-ligne.

- Test en-ligne: c'est le test qui est appliqué plus ou moins simultanément avec l'application et dans son environnement.

- Test hors-ligne: c'est le test qui se déroule en-dehors de toute application.

[ARL79] différencie deux types de test en-ligne: continu et discontinu

- Test en-ligne continu: la simultanéité entre le test et l'application est parfaite.

- Test en-ligne discontinu: il y a une alternance de phases de test et de phases d'application

[COU82] applique un critère de localisation (l'unité reste ou non dans son environnement d'application) pour différencier deux types de test dans le test hors-ligne: in-situ et ex-situ.

Test hors-ligne in-situ: l'unité centrale intégrée ne peut qu'exécuter des programmes, néanmoins spécifiques en vue du test. Bien que des mécanismes matériels puissent être actifs durant le déroulement de ces programmes, aucun moyen n'est supposé tel que l'on puisse à tout instant observer les valeurs des signaux transitant sur les bus.

Test hors-ligne ex-situ: les valeurs des signaux transitant sur les bus sont supposées observables, on applique des vecteurs de test et on observe si les sorties correspondent ou non aux valeurs sans panne.

Ainsi, selon ces critères il existe quatre classes de test:

- Test en-ligne continu

- Test en-ligne discontinu
- Test hors-ligne in-situ
- Test hors-ligne ex-situ

Une autre classification du test: test analytique et test fonctionnel est décrite au sous-chapitre III.1.

III.- Le microprocesseur M6800

Il est le composant principal de la famille Motorola 6800. Après l'Intel 8080, il est le second microprocesseur 8-bits le plus utilisé. Ses éléments fondamentaux sont montrés dans la figure 18.

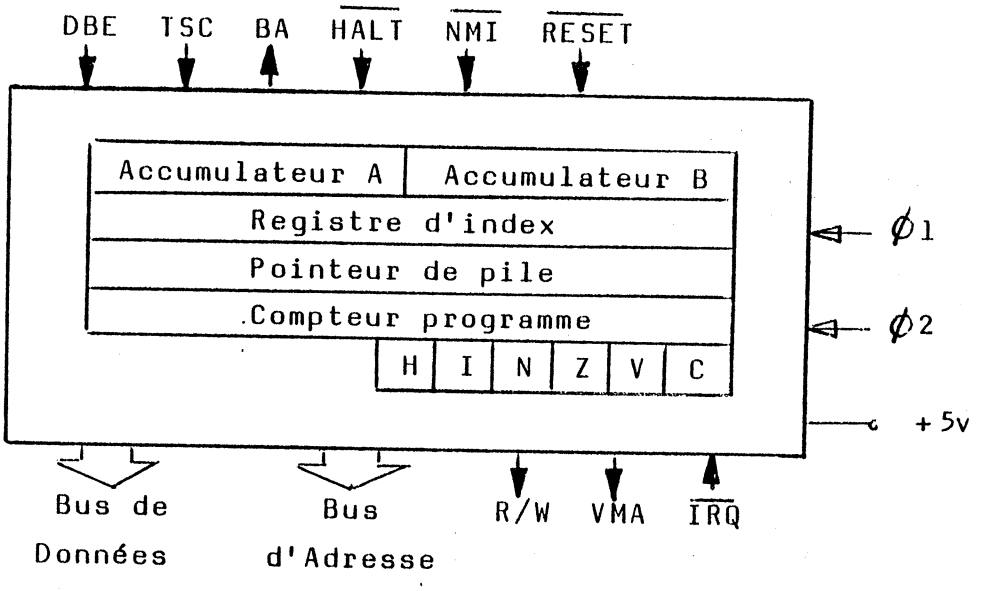


Figure 18.- Le microprocesseur M6800

Le jeu d'instructions comprend 72 instructions qui réalisent différentes fonctions, telles que: arithmétique décimale et binaire, décalages simples et circulaires, chargements de registres, etc. Il y a six modes d'adressage: immédiat, direct, étendu, indexé, relatif et implicite. Ainsi, le jeu d'instructions donne 197 codes d'opération nécessaires à l'identification de toutes les instructions. Les codes invalides (non-utilisés) sont au nombre de 59, diffusés de manière arbitraire parmi les codes existants.

Le M6800 a 13 registres internes, mais seulement 6 sont accessibles à l'utilisateur, tel qu'il est montré par le modèle

de programmation de la figure 19; les 7 restants sont transparents au programmeur.

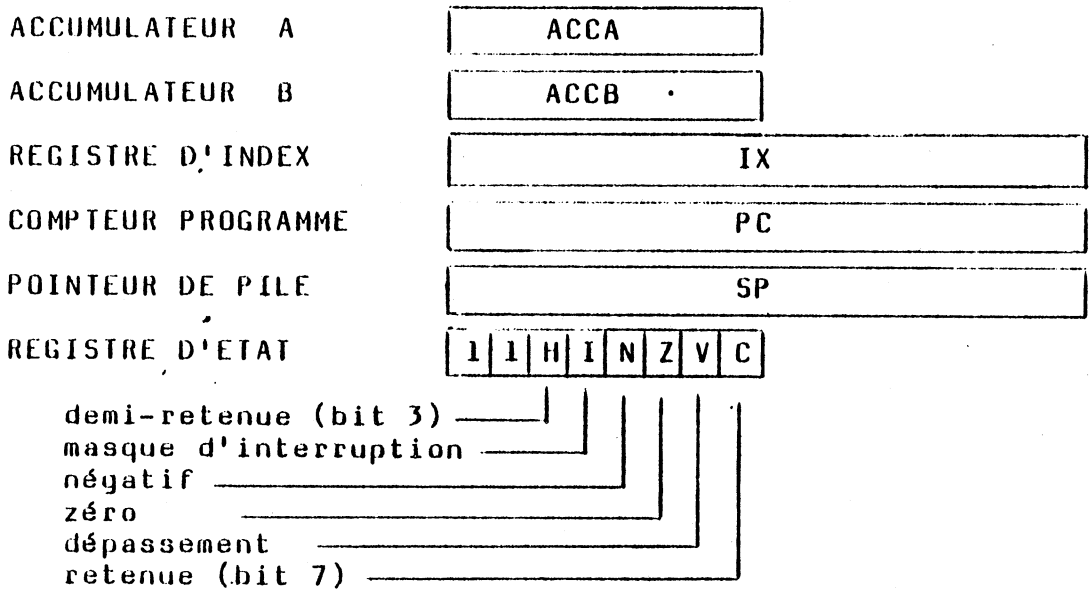


Figure 19.- Modèle de programmation du M6800

L'exécution de chaque instruction peut être décomposée en trois phases:

- acquisition de l'instruction
- adressage des opérandes (phase éventuelle)
- opération

III.1.- Test du M6800

Le test des microprocesseurs peut être envisagé comme le test d'un composant LSI à grande complexité. Notre étude est concernée par le test logique des microprocesseurs à partir de séquences de test prédéterminées (contrairement au test de type aléatoire). Dans cette optique on considère deux types de test:

- test analytique: où la génération de vecteurs est basée

sur la structure physique du microprocesseur.

- test fonctionnel: où la génération de vecteurs est indépendante de la structure physique du microprocesseur en prenant en compte seulement l'aspect fonctionnel.

Une pratique courante est de considérer l'unité centrale comme formée par des blocs fonctionnels et de tester chaque bloc presque indépendamment à partir des hypothèses établies sur le bon fonctionnement des autres blocs. On a envisagé le test du M6800 comme un test en deux phases:

- le test de la partie opérative

- le test de la partie contrôle

III.2.- Test de la partie opérative

Des programmes de test, qui matérialisent des vecteurs de test, sont exécutés par l'unité centrale pour vérifier le bon fonctionnement de la partie opérative. Ces programmes sont déterminés de manière classique à partir d'une analyse de la structure physique de cette partie. Cela a motivé une étude des mécanismes de fautes des circuits intégrés qui a donné origine à la définition des hypothèses de panne montrées dans la table de la figure 20 [COU80].

Classe 0	Classe I	Classe II
un défaut physique simple: 1 contact, 1 pré-contact, 1 MOS s-on ou s-open, 1 Al coupé ... grille flottante ⇒ MOS s-	classe 0 + : 1 court-circuit entre 1 Al et l'autre Al le plus proche géographiquement Idem pour diffusions	classe I + : courts-circuits entre Al quelconques idem diffusions défauts multiples

Figure 20.- Hypothèses de panne pour le test analytique

Les algorithmes qui testent cette partie ont déjà été conçus par [COU80]. Ils ont été réalisés à partir du même modèle de base utilisé pour le test de la partie contrôle.

IV.-Rappels du modèle de base

Le modèle décrit en [COU80-1] permet l'application d'un test fonctionnel pour la Partie Contrôle (PC) des UCI's. Comme beaucoup de modèles il considère l'UCI comme un système partitionné fonctionnellement, chaque bloc fonctionnel pouvant être testé avec une relative indépendance vis-à-vis des autres. Dans le cas de notre modèle il y a deux blocs principaux:

- la partie contrôle; et
- la partie opérative (PO)

Et, entre les deux, une interface constituée par les signaux de commandes d'E/S sur les bus et d'opérations pour les opérateurs; il faut en plus ajouter les interfaces de ces deux parties avec un registre d'état qui n'est pas localisé ni dans la PC ni dans la PO, sa place est un cas d'espèce (figure 21).

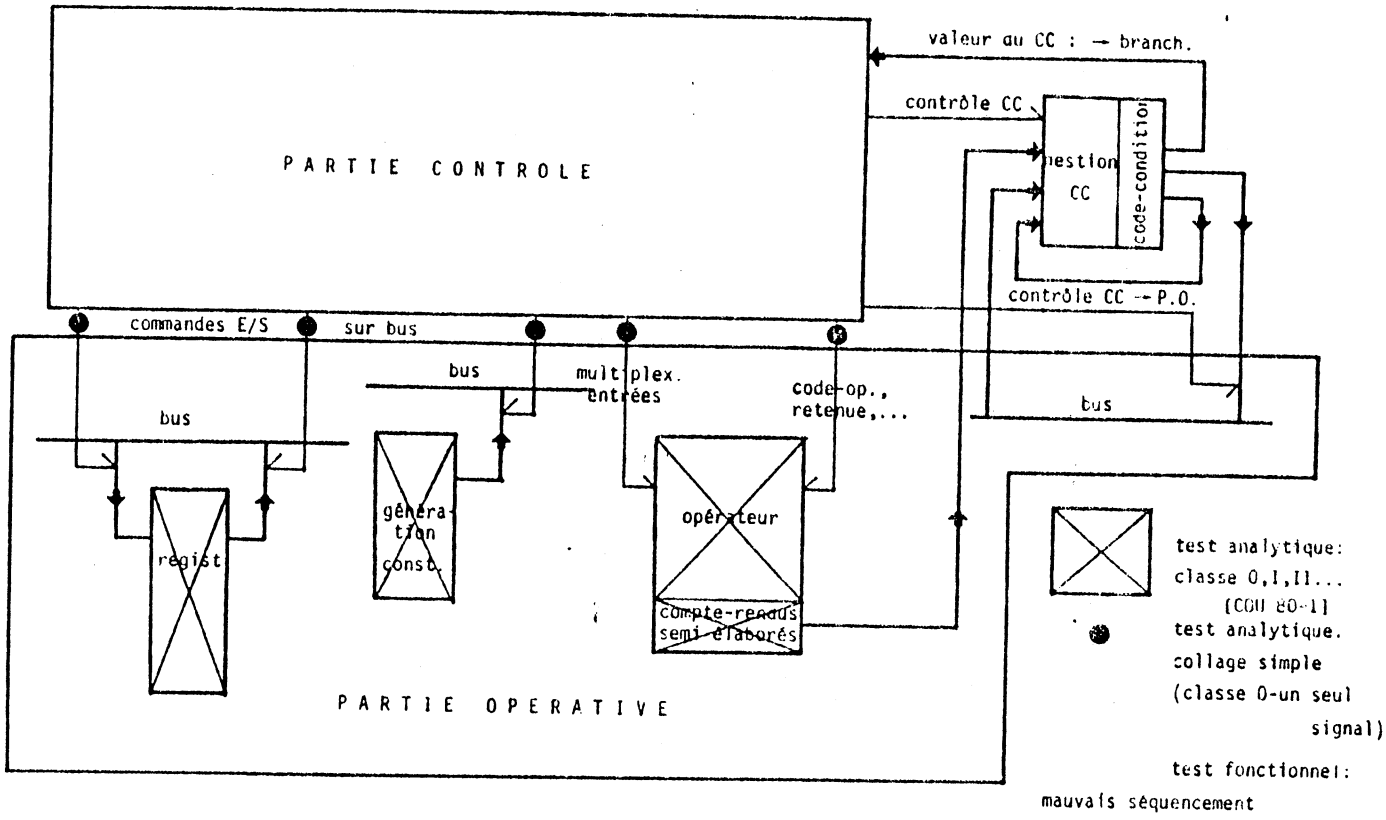


Figure 21.- Modèle et hypothèses de panne pour le M6800

Associés à ce registre d'état on trouve les circuits nécessaires à sa gestion; ainsi, l'interface entre la PO et ces circuits est constituée par les valeurs des comptes-rendus des opérateurs et les valeurs du registre d'état. L'interface PC-circuits de gestion est formée des valeurs du registre d'état et des signaux de contrôle de ces circuits.

Pour le test de la PC il faut vérifier (cas du test fonctionnel) le bon séquencement des commandes, en observant l'interface PC-PO et PC-circuit de gestion du registre d'état; mais l'étendue du problème nécessite l'établissement d'hypothèses (particulièrement pour les commandes des opérateurs), en vue de réduire le nombre des cas pris en compte. Cette réduction est dépendante de l'UCI considérée et de chacun des opérateurs.

Il faut remarquer que le modèle a été conçu initialement pour le test en-ligne, mais sa généralisation au test hors-ligne, ne retire rien à sa validité.

V.-Algorithmes du test en-ligne(Partie Contrôle)

Quelques définitions concernant des éléments du modèle aideront à comprendre les explications ultérieures:

Les registres et les instructions sont considérés comme étant opératifs ou non-opératifs.

- Registre opératif: registre non destiné, en général, à contenir une valeur devant servir à l'élaboration d'une adresse-instruction (reg. A et B).

- Registre non-opératif: registre destiné à contenir, en général, une valeur devant servir à l'élaboration d'une adresse-instruction (reg. X, SP et CO (compteur ordinal)).

- Instruction opérative: instruction manipulant des données destinées, en général, à être utilisées pour l'élaboration d'une adresse-instruction (instr. ADDA, LDAB, etc).

- Instruction non-opérative: instruction manipulant des données destinées, en général, à être utilisées pour l'élaboration d'une adresse-instruction, ou bien ne manipulant aucune donnée (instr. CPX, STS, etc.).

A l'aide de cette classification on distingue, pour le test en-ligne, six classes de pannes:

CLASSE 0: Une instruction opérative ne "calcule" pas correctement: le résultat n'est pas le résultat juste. Elle modifie la valeur des registres opératifs qu'elle ne devrait pas modifier.

CLASSE I: Une instruction non-opérative modifie de manière intempestive des registres opératifs.

CLASSE II: Une instruction non-opérative ne "calcule" pas correctement, lorsqu'elle doit calculer. Elle modifie de manière intempestive des registres non-opératifs.

CLASSE III: Une instruction opérative modifie de manière intempestive des registres non-opératifs.

CLASSE IV: Une instruction, opérative ou non, ne gère pas correctement le registre d'état.

CLASSE V: Une instruction (non-opérative) de branchement ne "branche" pas à bon escient.

Il faut remarquer que ces classes de pannes sont de types différents, sans aucune hiérarchie entre elles. Le test de ces classes de pannes nécessite des tests plus élémentaires, tels que le test d'un transfert élémentaire:

Ainsi, si on nomme OR et DEST les registres connectés en entrée et en sortie respectivement, sur un bus, neuf cas de figure sont possibles:

OR	DEST	RESULTAT
NC	NC	---
E	NC	---
S	NC	OR <-- UN
NC	E	---
NC	S	DEST <-- UN
E	E	OR <-- ET(OR,DEST) DEST <-- ET (OR, DEST)
E	S	DEST <-- OR
S	S	OR <-- UN DEST <-- UN
S	E	OR <-- DEST

NC: registre non - connecté
E: registre connecté en entrée
S: registre connecté en sortie

Les erreurs possibles nous conduisent à déterminer les procédures de test nécessaires pour les différentes classes de panne.

Pour le test des classes 0 et II, sur un cycle:

-Test de la bonne sélection d'une origine, en manifestant les erreurs:

a- Aucune origine n'est sélectionnée,
b- une autre origine que la normale est sélectionnée,
c- en plus de la bonne origine, une ou plusieurs autres sont sélectionnées,
d- la bonne origine n'est pas sélectionnée, mais plusieurs autres le sont;

Les conditions de test sont:

(A: valeur de l'origine normale,
K_i: les valeurs des autres origines possibles)

$$A \neq UN \qquad (1')$$

$$A = UN \qquad (1)$$
$$K_i = 0;$$

Etant donné que les conditions (1) et (1') sont incompatibles, deux séquences de test sont nécessaires.

-Test de bonne sélection d'une destination, basé sur les conditions suivantes:

$$V_f \neq UN \qquad (2)$$
$$V_f \neq ET\ (V_o, V_f)$$

Pour le test des classes I et III, sur un cycle (non-modification des registres), les conditions suivantes suffisent:

$$R_i = UN \qquad (3)$$
$$BUS = 0$$

Finalement, il faut considérer la multiplicité spatiale et temporelle de l'exécution d'une instruction. Pour la multiplicité temporelle, il faut établir les conditions de test pour des erreurs se produisant durant plusieurs cycles, pendant l'exécution d'une instruction. Il faut bien préciser que des valeurs sont commandables et observables seulement au début et à la fin d'une instruction, qui comprend en général plusieurs cycles.

Durant chacun des cycles, de mauvais séquencements peuvent

apparaittre, tels que l'on a vu antérieurement. Pour détecter des fautes de la PC qui se produisent dans un cycle donné, il faut déterminer les opérandes nécessaires pour détecter le mauvais séquencement durant le cycle, mais aussi celui avant et après ce cycle. Ainsi par exemple, pour un registre utilisé par une instruction selon la séquence suivante:

```
------
------
 OR
------
 DEST
------
```

----: registre non-utilisé,
OR: registre utilisé en origine,
DEST: registre utilisé en destinataire.

La valeur finale qui doit être chargée dans le registre (s'il n' a pas de faute) doit être non seulement différente de ET (V_o, V_f), mais aussi différente de ce que peut devenir V_o perturbée durant le premier et deuxième cycle (pour le cas où le registre ne soit pas chargé). Elle doit être également différente de ce qu'elle peut devenir en cas de perturbation durent le cinquième cycle.

[COU80-1] définit des séquences de base d'utilisation des registres auxquelles sont liées de conditions de test spécifiques. La figure 22 montre un échantillon de séquences de base d'utilisation d'un registre. Ainsi par exemple, pour la séquence No. 5, deux étapes sont nécessaires:

1- V_o = UN

$BUS^*_j = 0$	pour tous les cycles avant utilisation comme origine
$K^*_i = 0$	pendant le cycle d'utilisation comme origine

2- Il y deux positions binaires telles que:

dans la position k: $V_0[k] = BUS\ [k] = 0$

$V_f[k] = BUS\ [k] = 1$

dans la position k': $V_0[k'] = BUS\ [k'] = 1$

$V_f[k'] = BUS\ [k'] = 0$

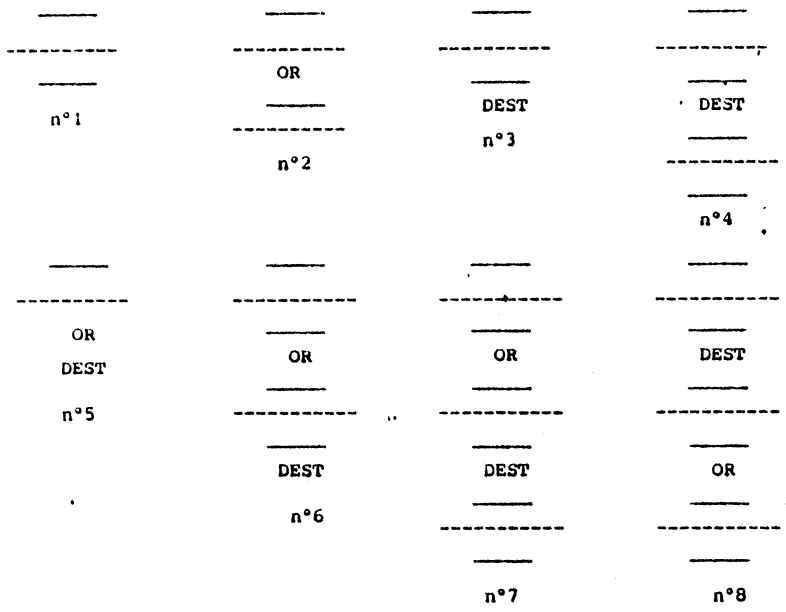


Figure 22.- Séquences de base d'utilisation d'un registre

Pour ce qui concerne à la multiplicité spatiale, il faut reconnaître qu'à une instruction correspond un ensemble de séquences, une pour chaque registre utilisé. Des conditions de test sont à mettre en oeuvre en fonction de la classe de test considéré. La table de la figure 23 résume les classes concernées en fonction du type de registre et des différentes séquences, en supposant qu'une instruction opérative ne concerne, pour la classe 0, que des registres opératifs (ce qui ne sera pas le cas pour des phases d'adressage). L'interaction entre les différentes conditions (différentes séquences, différentes conditions de test pour les oérateurs) peut amener à plusieurs étapes, i. e. commander et observer des valeurs plusieurs fois.

type de registre \ séquence	1	2	3	4	5	6	7	8
opératif	0,I	0	0	0	0	0	0	0,II
non opératif	II,III	II	II	II	II	II	II	

Figure 23.- Classes de test concernés en fonction du type de registre et des différentes séquences

Dans le test en-ligne, les classes II et III ne sont pas testées, mais les algorithmes pour toutes les classes ont eté établis en [COU80-1]. Après la définition du modèle, la question suivante était: comment réaliser le test?; il fallait établir certaines hypothèses sur le mauvais séquencement des commandes (classes 0, I, II et III):

-un seul type d'instruction est affecté en cas de panne, mais toutes les instructions de ce type utilisant des modes d'adressage différents sont alors affectées;

-un seul mode d'adressage est affecté en cas de panne, mais tous les types d'instructions utilisant ce mode sont alors affectés;

-il n'y a pas simultanément de mauvais séquencement pendant la phase d'adressage et pendant la phase opération.

Cela nous amène à respecter les règles suivantes dans la réalisation des algorithmes (classes 0, I, II et III):

-test des modes d'adressage, par exécution d'une instruction de type I et de mode M, en commandant/observant à l'aide d'instructions de type quelconque et de mode différent de M;

-test des types, par exécution d'une instruction de type I et de mode M, en commandant/observant à l'aide d'instructions de type différent de I et de mode quelconque.

L'hypothèse de non-simultanéité des pannes de type et de mode, simplifie beaucoup la réalisation des algorithmes; la suppression de cette hypothèse forcera à une attention particulière et très restrictive dans le choix des instructions de commande/observation, tel qu'il est montré dans l'annexe A.

Pour la classe IV, la démarche est semblable à ce qui a été vu précédemment, mais cette fois-ci l'objectif est de vérifier la bonne manipulation du registre d'état; les hypothèses de non-simultanéité réduisent, aussi, le nombre de tests à faire.

Pour la classe V le test est exhaustif; il n'y a pas d'hypothèse restrictive; on teste instruction par instruction (de branchement) et on vérifie la bonne exécution pour chaque état des indicateurs.

Finalement, les algorithmes du test fonctionnel pour la PC du M6800 sont organisés de la manière suivante:

1.- Algorithmes de test des classes 0, I, II, III.
2.- Algorithmes de test de la classe IV.
3.- Algorithmes de test de la classe V.

Chacun de ces groupes est concerné par:

-le test de la phase d'adressage; et
-le test de la phase opération;

exception faite de la classe V (instr. de branchement), qui,

formée des instructions non-opératives, est testée en partie par les classes I et II.
Le test de la phase d'adressage des classes 0, I, II et III concerne:

A: le bon "calcul", c'est à dire l'obtention des bons opérandes mémoire.
B: la non-modification des registres:
B.1 Opératifs
B.2 Non-opératifs.

Le test de la phase opération des classes 0, I, II et III considère chacune de ces classes séparément.

Le test de la phase d'adressage de la classe IV considère le test de chaque indicateur (non-modification) pour chaque mode d'adressage.

Le test de la phase opération de la classe IV considère les différentes possibilités de mise à jour des indicateurs par les différents types d'instruction.

Le test de la classe V est réduit au test de la phase opération seulement.

VI.-Algorithmes du test hors-ligne (Partie contrôle)

Tel qu'il a été dit en [COUBO-1], les différentes classes de panne établies par le modèle, pour le test en-ligne, sont aussi applicables au test hors-ligne. Ce type de test demande un test complet du circuit, c'est à dire le test de toutes les classes de panne. Un réaménagement des classes est possible, dans le but de diminuer l'encombrement mémoire et le temps de test.

La classification des pannes et les conditions de commande et d'observabilité décrites dans la section antérieure, font que la complexité d'analyse et de conception de chaque test d'instruction se trouve largement simplifiée. Alors, avec cet objectif de simplification, on maintient l'organisation établie pour le test en-ligne et on essaie de combiner les classes là où cela est possible; c'est à dire là où les conditions de commandabilité et d'observabilité se rapprochent. La figure 24 montre toutes les combinaisons possibles.

Pour le test de la phase d'adressage des classes 0, I, II et III, c'est à dire, l'ensemble complet des instructions, les conditions sont telles que la combinaison des classes n'est pas possible; par conséquence les algorithmes de test hors-ligne pour la phase d'adressage sont identiques aux algorithmes du test en-ligne pour la même phase.

Le caractère particuliér de la classe V (circuit isolé dans la pastille) force à la même conclusion, le test de cette classe est fait de la même manière que pour le test en-ligne. De cette façon, on ne considère plus les classes numérotées 11, 12, 13 et 14. Les classes numérotées 3, 4, 5, 6, 7 et 8 font apparaître des actions différentes sur des registres opératifs et non-opératifs, ce qui n'a pas lieu d'être pour le test hors-ligne. Finalement, le test des pannes de classe IV implique des conditions de test particulières qui ne vont pas avec celles des autres classes, alors on élimine les groupes 9 et 10.

1.- CLASSES 0 et III	une instruction opérative ne "calcule" pas correctement, le résultat n'est pas le résultat juste. Elle modifie la valeur des registres opératifs et/ou non-opératifs qu'elle ne devrait pas modifier.
2.- CLASSES I et II	une instruction non-opérative ne "calcule" pas correctement, lorsqu'elle doit calculer. Elle modifie intempestivement des registres opératifs et/ou non-opératifs.
3.- CLASSES 0 et IV	une instruction opérative ne "calcule" pas correctement: le résultat n'est pas le résultat juste. Elle modifie la valeur des registres opératifs qu'elle ne devrait pas modifier. Elle ne gère pas correctement le registre d'état.
4.- CLASSES I et IV	une instruction non-opérative modifie de manière intempestive des registres opératifs. Elle ne gère pas correctement le registre d'état.
5.- CLASSES II et IV	une instruction non-opérative ne "calcule" pas correctement, si elle doit calculer. Elle modifie de manière intempestive des registres non-opératifs. Elle ne gère pas correctement le registre d'état.
6.- CLASSES III et IV	une instruction opérative modifie de manière intempestive des registres non-opératifs. Elle ne gère pas correctement le registre d'état.
7.- CLASSES I et V	une instruction de branchement ne "branche" pas à bon escient. Elle modifie de manière intempestive des registres opératifs.

Figure 24.- Groupes de classes de pannes (lère. partie)

8.- CLASSES II et V	une instruction de branchement ne "branche" pas à bon escient. Elle modifie de manière intempestive des registres non-opératifs.
9.- CLASSES 0, III et IV	une instruction opérative ne "calcule" pas correctement, le résultat n'est pas le réultat juste. Elle modifie la valeur des registres opératifs et/ou non opératifs qu'elle ne devrait pas modifier. Elle ne gère pas correctement le registre d'état.
10.- CLASSES I, II et IV	une instruction non-opérative ne "calcule" pas correctement, si elle doit calculer. Elle modifie de manière intempestive des registres opératifs et/ou non-opératifs. Elle ne gère pas correctement le registre d'état.
11.- CLASSES I, II et V	une instruction de branchement ne "branche" pas à bon escient. Elle modifie de manière intempestive des registres opératifs et/ou non-opératifs.
12.- CLASSES I, IV et V	une instruction de branchement ne "branche" pas à bon escient. Elle modifie de manière intempestive des registres opératifs. Elle ne gère pas correctement le registre d'état.
13.- CLASSES II, IV et V	une instruction de branchement ne "branche" pas à bon escient. Elle modifie de manière intempestive des registres non-opératifs. Elle ne gère pas correctement le registre d'état.
14.- CLASSES I, II, IV et V	une instruction de branchement ne "branche" pas à bon escient. Elle modifie de manière intempestive des registres opératifs et/ou non-opératifs. Elle ne gère pas correctement le registre d'état.

Figure 24.- Groupes de classes de pannes (2ème. partie)

Le test de la phase opération des classes 0, I, II et III (groupes 1 et 2) est le seul où une combinaison de classes est possible, avec une éventuelle économie dans l'encombrement mémoire et dans le temps de test. Ainsi les classes 0 et III se trouvent combinées (instr. opératives) pour être traitées comme une seule classe pendant le test de la phase opération:

Classe 0 et III: une instruction opérative ne "calcule" pas correctement, le résultat n'est pas le résultat juste. Elle modifie la valeur des registres qu'elle ne devrait pas modifier.

De la même manière les classes I et II donnent:

Classe I et II: une instruction non-opérative ne "calcule" pas correctement, si elle doit calculer. Elle modifie la valeur des registres qu'elle ne devrait pas modifier.

On remarquera l'impossibilité de combiner, à cause des conditions, certaines classes, 0 et I par exemple.

Globalement les programmes sont structurés comme suit (Fig. 25):

-D'abord le test de la phase adressage concernant toutes les instructions et couvrant les hypothèses dejà décrites. On remarquera qu'on n'y considère pas les problèmes de gestion du registre d'état. Ici sont testés les modes d'adressage immédiat, direct, indexé et étendu. L'adressage relatif est testé seulement pour la non-modification des registres (classes 0, I, II et III). Cette partie suit l'organisation logique de [COU80-1]:

A: Phase adressage: classes 0 et II (bon "calcul")
B: Phase adressage: classes 0, I, II et III (non-modification):

B.1: Non-modification des registres opératifs
B.2: Non-modification des regitres non-opératifs

-Après, on teste la phase opération, où on doit vérifier toutes les instructions, une par une. On considère séparément les instructions opératives (classes 0 et III) et les instructions non-opératives (classes I et II).

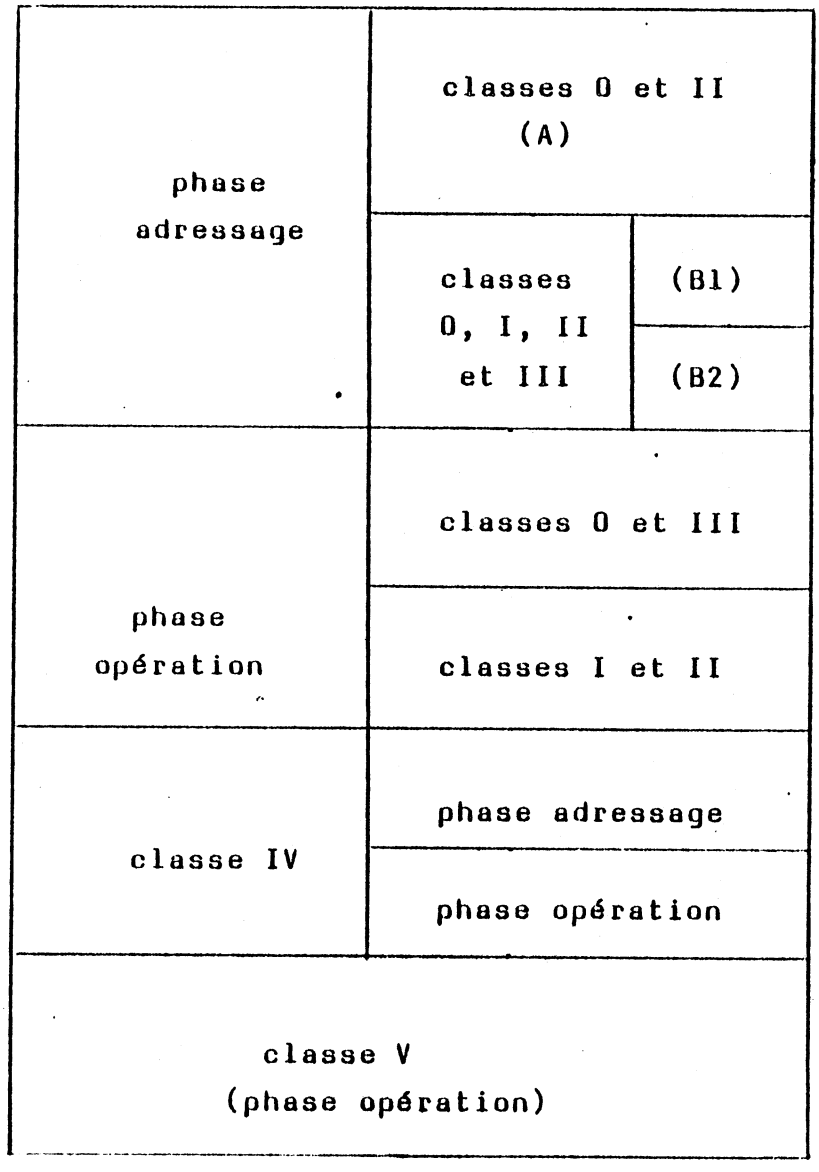


Figure 25.- Organisation du logiciel

-Ensuite, on teste la bonne gestion du registre d'état par les instructions. Cette fois-ci, encore, on considère séparément les phases d'adressage et d'opération. Pour la phase d'adressage on teste les modes immédiat, direct, indexé, étendu et relatif. Pour la phase opération, on doit tester toutes les instructions, une par une.

-Finalement, on teste le branchement à bon escient de manière exhaustive, c'est à dire tous les cas possibles de branchement conditionnel et non-conditionnel.

Le programme a plusieurs points de sortie dans le cas de panne detectée (EADIO2, EAINO2, etc.), ce qui laisse la possibilité d'un traitement spécial pour chaque cas de panne, mais ces points de sortie peuvent être réduits à un seul. FINTST est la sortie pour une UCI sans panne selon nos hypothèses).

VI.1.-Classes 0, I, II et III

VI.1.1.- Phase adressage. Test des classes 0 et II (bon "calcul")

Pour ces classes, on teste les modes d'adressage immédiat, direct, indexé et étendu (l'adressage relatif ne détermine pas une adresse opérande). Les algorithmes sont, en général, les mêmes que pour le test en-ligne. Ainsi:

VI.1.1.1.- Adressage immédiat

Bien qu'il n'y ait pas de phase d'adressage proprement dit, ni d'utilisation d'opérateur, l'opérande est adressé par le compteur ordinal. Il faut tester que l'adresse obtenue ne fasse pas partie de toutes celles qui pourront prendre la place de l'adresse correcte. Ainsi le test débute à l'adresse $A190 et l'adresse correcte doit être $A1A1, adresse de la position de mémoire

contenant la valeur $AA. Le module de programme peut être enregistré sur une ROM. Après le positionnement des registres (A, B, SP, X et Y) à la valeur ZERO, on exécute l'instruction-test:

LDAA #$AA

Toutes les positions de mémoire correspondant aux possibles adresses erronées (8) sont chargées avec une valeur différente de $AA ($FF). Il ne reste plus qu'à vérifier si le reg. A a été bien chargé.

VI.1.1.2.- Adressage direct

Dans ce cas-là, il n'y a pas d'utilisation d'opérateur; le test consiste à écrire en mémoire une valeur, $C3, en mettant les éventuelles valeurs prises par erreur à $F0. Ce test est placé à l'adresse $F0DC, la mémoire utilisée peut être une ROM, les adresses $0F, $F0 correspondent à une mémoire RAM. L'instruction-test est:

STAA $0F

L'adresse $000F est chargée initialement à la valeur $AA; le reg. A contient la valeur $C3; les registres qui n'interviennent pas sont chargés à la valeur $F0 (chaque octet). Il ne reste plus qu'à vérifier si la position de mémoire à l'adresse $000F a bien la valeur $C3.

VI.1.1.3.- Adressage indexé

Dans ce mode d'adressage le calcul de l'adresse utilise l'opérateur de calcul; il faut vérifier que ce calcul utilise les bons opérandes. La méthode est la même que précédemment, il faut écrire une valeur dans une position de mémoire et vérifier la bonne réalisation. Ce test exige d'être placé à une adresse telle que $A200, on utilise une ROM (les variables ADR1 et ADRIN devront faire partie d'une RAM). Après le positionnement des

registres (A, B, SP et T à ZERO, X à $0FF0), de l'indicateur C à UN et de la mémoire d'adresse $1008 à $AA, on teste avec l'instruction-test:

COM $18,X

Finalement, on vérifie qu'à l'adresse $1008 on a bien $55.

VI.1.1.4.- Adressage étendu

Ce test est localisé à l'adresse $A233 et il utilise une ROM (sauf la variable ADRET, d'adresse $100F). L'instruction-test est:

COM $100F

Utilisée après avoir mis tous les registres à ZERO et la mémoire d'adresse $100F à la valeur $AA. Il faut vérifier simplement si la mémoire d'adresse $100F contient bien $55 à la fin de l'exécution.

VI.1.2.- Phase adressage. Test des classes 0, I, II et III (non-modification).

VI.1.2.1.- B.1: Non-modification des registres opératifs

Il faut tester que les registres A et B ne sont pas modifiés par la phase adressage des instructions; on utilise des instructions non-opératives ce qui permet de tester A et B d'un seul coup. Dans tous les cas, les reg. A et B sont initialisés à UN et toutes les valeurs transitant sur les bus, pendant l'exécution de l'instruction, ont au moins un bit à zéro dans chaque octet.

VI.1.2.1.1.- Adressage immédiat

Placé à l'adresse $A09C ce test est très simple. L'instruction-test étant:

LDX #$0000

il faut vérifier que A et B restent bien à UN.

VI.1.2.1.2.- Adressage direct

Placé à l'addresse $A0B2, il faut rajouter, aux conditions antérieures (adressage immédiat), que l'adresse directe contient au moins un zéro.

Les autres modes sont réalisés d'une manière similaire, en prenant en compte pour les adressages indexé et relatif qu'il y a un calcul à faire et que les résultats devront, aussi, avoir au moins un zéro dans chaque octet.

VI.1.2.2.- B.2: Non-modification des registres non-opératifs

La non-modification des registres non-opératifs X et SP, est testée en utilisant des instructions opératives. L'analyse est identique au test B. 1

VI.1.3.- Phase opération

VI.1.3.1.- Test des classes 0 et III

Ce test concerne la phase opération des instructions opératives, et il doit vérifier que les instructions "calculent" bien et ne modifient pas des registres qui ne doivent pas être modifiés. Le problème se limite à trouver l'ensemble minimum des conditions de test, à partir des conditions du test de:

a.- bon "calcul" et non-modification des registres opératifs; et

b.- non-modification des registres non-opératifs.

a.- La table I en [COU80-1] donne toutes les conditions à remplir pour faire le test correspondant; cela peut se faire en un ou plusieurs pas.

b.- La table IV en [COU80-1] donne les conditions de test de la non-modification des registres non-opératifs X et SP. Ces conditions peuvent être combinées éventuellement avec celles de (a).

Il y a 61 instructions (types) classées comme instructions opératives. Le test de chacune étant similaire dans leur démarche, on a choisi deux d'entre elles pour exemple: ADDA et ANDB.

Instruction ADDA: A <---- A + M

Le test doit vérifier le bon déroulement de l'opération A + M; c'est à dire que la commande appliquée à l'opérateur de calcul soit bien + et non un autre qui donne le même résultat:

A + M = A * M (où * représente une autre opération du genre A , M ---> A)
(pour ce cas, * représente la soustraction)

Aussi, le bon "calcul" de A + M exige que ce soit exactement A et M qui soient pris comme opérandes et non un autre (sélection de la bonne origine):

-Pour A il faut 2 conditions disjointes:

A = UN
$BUS^{\bullet}_{j}$ = 0 ; pour tous les cycles, sauf celui d'utilisation

$K'_i = 0$; pour le cycle d'utilisation

et $A \neq UN$

-Pour M, il faut aussi 2 conditions disjointes:

$M' = UN$
$K_i = 0$

et $M \neq UN$

Cela peut finalement être fait de la manière suivante:

ler. pas: $A = UN$; $M = UN$
$BUS'_j = 0$
$K'_i = 0$ (pendant l'exécution); et

2ème. pas: $A \neq UN$; $M \neq UN$

La sélection du bon destinataire exige que:

$V_f \neq UN$
$V_f \neq ET(V_o, V_f)$

ce qui peut être combiné avec le 2ème. pas et donner:

ler. pas: $A = \$FF$; $M = \$FF$
$B', T', \ldots\ldots = 0$

2ème. pas: $A = \$01$; $M = \$0F$

Pour la non-modification du registre opératif B, les conditions de test sont:

$B = UN$
$BUS'_j = 0$

Ce qui peut être inclus dans le 2ème. pas qui devient:

2ème. pas: $A = \$01$; $M = \$0F$; $B = UN$; $BUS^{c}_{j} = 0$

Pour la non-modification des registres non-opératifs X et SP il y a deux alternatives:

-1ere. alternative: $X = SP = UN$; $BUS^{c}_{j} = 0$
-2ème. alternative: $X^{\cdot} = SP^{\cdot} = 0$ (avec la position k = 1)
$BUS^{c}_{j} = 0$ (avec la position k = 0)

Et c'est ainsi, que les conditions de test du 2ème. pas deviennent:

$A = \$01$; $M = \$0F$; $X = SP = \$FF$; $B = UN$; $BUS^{c}_{j} = 0$

La démarche pour les autres instructions opératives est semblable.

VI.1.3.2.- Phase opération. Test classes I et II

L'obtention des algorithmes de test pour la phase opération de ces classes est faite de la même manière que pour les instructions opératives. On utilise cette fois-ci les tables II et III de [COU80-1]. Il y a 42 instructions (types) non-opératives (incluses les instructions de branchement). Les exemples choisis sont les instructions CPX et STS.

Pour l'instruction CPX l'algorithme de base est celui de [COU80.1] pour le bon "calcul" (3 pas). La non-modification du registre X est incluse dans le ler. pas; la non-modification des registres A, B et SP est incluse dans le 2ème. pas. Finalement, il faut remarquer que pour le test de toute la phase opération l'emplacement des programmes peut être quelconque.

VI.2.- Classe IV

Le test de cette classe est le plus complexe de tous; il doit comprendre le test de la phase d'adresssage et le test de la phase opération.

On remarquera que tous les indicateurs ne peuvent pas être testés pour certains modes d'adressage, étant donné que la phase opération modifie ces indicateurs qui peuvent ne pas être observables (en plus, il ne sert à rien de tester la bonne gestion des indicateurs pendant l'adressage s'ils doivent être modifiés par la phase opération). Ainsi, pour l'adressage immédiat on peut tester seulement la bonne gestion des indicateurs H, I, et C, étant donné qu'il n'existe pas d'instruction avec adressage immédiat ne modifiant pas les indicateurs N, Z, et V pendant la phase opération.

Les indicateurs doivent être testés une fois initialisés à ZERO (première étape) et une autre fois, initialisés à UN (seconde étape). Il faut remarquer que certains indicateurs sont totalement testés (dans le cas de non-modification) avec une seule initialisation (ZERO ou UN), étant donné que les signaux de contrôle sont totalement indépendants de la valeur initiale.

Dans quelques cas (adressage indexé: indicateurs I, Z, V, C) le test de la bonne gestion du circuit doit se faire en plus d'un pas par étape pour pouvoir couvrir toutes les conditions qui ne sont pas compatibles.

VI.2.1.- Adressage immédiat

Pour ce mode-là, seulement les indicateurs H, I, et C sont testables; il n'y a pas de transfert sur le bus DB et l'opérateur de calcul n'est pas utilisé (pas de compte-rendu). L'instruction-test utilisée est LDAA, qui peut être placée à n'importe quelle adresse. Le test se fait en deux étapes: indicateurs initialisés

à ZERO et indicateurs initialisés à UN; les conditions sont si simples qu'un pas suffit pour chaque étape.

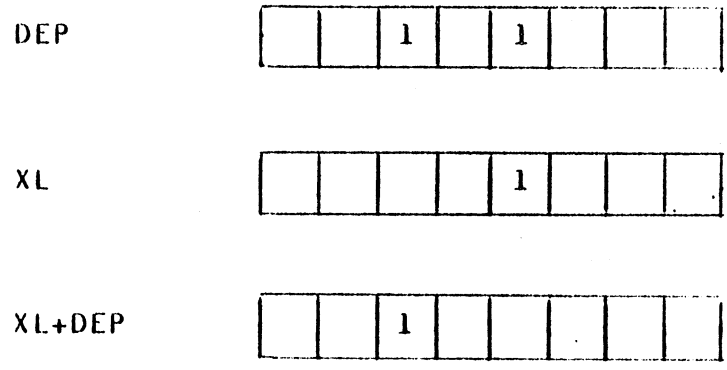
VI.2.2.- Adressage direct

Pour ce mode, aussi, les seuls indicateurs testables sont H, I et C; il y a transfert sur le bus DB (adresse directe) mais l'opérateur de calcul n'est pas utilisé, en conséquence il n'y a pas de comptes-rendus. L'adresse de l'instruction-test (LDAA) n'est pas fixe. Le test se fait, aussi, en deux étapes: indicateurs initialisés à ZERO et indicateurs initialisés à UN. Les problèmes sont semblables au mode immédiat, mais cette fois-çi l'opérande (adresse directe) doit être pris en compte: $FF lorsque les indicateurs sont initialisés à ZERO et $CO lorsque les indicateurs sont initialisés à UN.

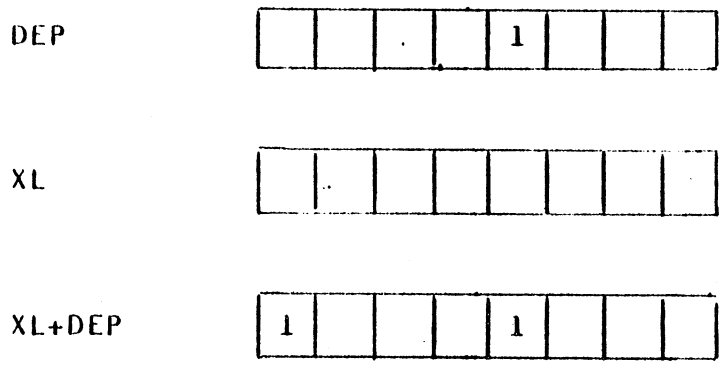
IV.2.3.- Adressage indexé

Dans ce cas-là, tous les indicateurs sont concernés; la seule instruction qui utilise ce mode d'adressage et qui ne modifie pas les indicateurs pendant la phase execution, est JMP. Une analyse superficielle du circuit (en vue d'une optimisation) peut nous montrer que, pour les indicateurs H et N, il suffit d'une seule étape du test: indicateurs initialisés à ZERO; pour les autres les deux étapes sont nécessaires. Pendant cette phase il y a des transferts sur le bus DB et on utilise l'opérateur de calcul. Dans ce qui suit, on montre l'analyse faite pour tester la gestion que l'on utilise dans cette phase sur les indicateurs H et N (l'analyse faite pour les autres indicateurs est semblable):

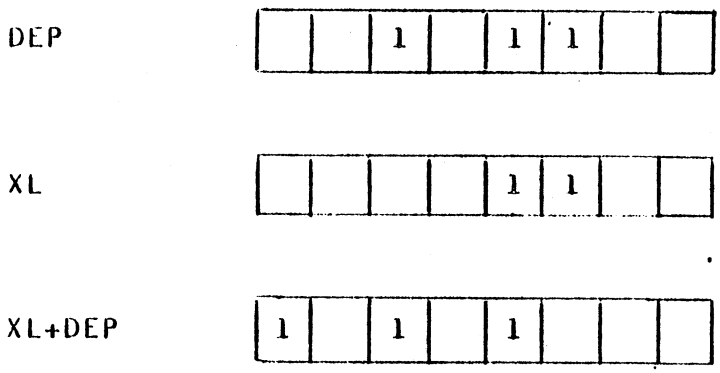
L'opérateur de calcul est utilisé pour réaliser la somme XL+DEP (XL: octet poids faibles du registre index, DEP: opérande déplacement) et c'est d'après ce résultat que les comptes-rendus sont générés. Les transferts sur le bus DB sont DEP et XL+DEP, dans cet ordre. Ainsi pour l'indicateur H, initialisé à ZERO, la configuration minimale des octets qui correspondent doit être:



Pour l'indicateur N, initialisé à ZERO:



Par conséquence, il est possible de tester H et N dans un seul pas avec la configuration suivante:



Ainsi, les bits correspondants du transfert sur le bus DB et les comptes-rendus sont mis à UN, permettant le test des

indicateurs H et N. De la même manière, on ajoute les conditions de test pour les indicateurs I, Z, V et C, mais pour ces cas-là, les conditions ne sont plus totalement compatibles et il nous faut deux pas pour tester l'indicateur Z et deux pas pour l'indicateur C. Finalement on test les indicateurs (initialisés à ZERO) en trois pas, avec les configurations suivantes:

ler. pas:

DEP	0	1	1	1	1	1	1	1
XL	0	1	1	1	1	1	1	1
XL+DEP	1	1	1	1	1	1	1	0

2éme. pas:

DEP	0	0	0	0	0	0	0	0
XL	0	0	0	0	0	0	0	0
XL+DEP	0	0	0	0	0	0	0	0

3ème pas:

DEP	1	0	0	0	0	0	0	1
XL	1	0	0	0	0	0	0	0
XL+DEP	0	0	0	0	0	0	0	1

Pour des raisons d'utilisation de mémoire le troisième pas devient:

DEP	1	0	0	0	0	0	0	1
XL	1	0	1	1	0	0	0	0
XL+DEP	0	0	1	1	0	0	0	1

Ce qui suit explique comment chaque indicateur trouve ses conditions de test satisfaites dans les trois pas décrits plus haut:

II : a- Transfert sur le bus DB (pas 1): bit 5 du DEP = 1
bit 5 du XL+DEP = 1

b- Compte-rendu (pas 1): bit 3 du DEP = 1
bit 3 du XL = 1

N : c- Transfert sur le bus DB (pas 1): bit 3 du DEP = 1
bit 3 du XL+DEP = 1

d- Compte-rendu (pas 1): bit 7 du XL+DEP = 1

Z : e- Transfert sur le bus DB (pas 1): bit 2 du DEP = 1
bit 2 du XL+DEP = 1

f- Compte-rendu (pas 1): bits 0-7 du DEP = 0
bits 0-7 du XL = 0
bits 0-7 du XL+DEP = 0

I : g- Trasnfert sur le bus DB (pas 1): bit 4 du DEP = 1
bit 4 du XL+DEP = 1

V : h- Transfert sur le bus DB (pas 1): bit 1 du DEP = 1
bit 1 du XL+DEP = 1

i- Compte-rendu (pas 1): bit 6 du DEP = 1
bit 6 du XL = 1
bit 7 du DEP = 0
bit 7 du XL = 0
bit 7 du XL+DEP = 1

C : j- Transfert sur le bus DB (pas 1): bit 0 du DEP = 1
bit 0 du XL+DEP = 1

k- Compte-rendu (pas 1): bit 7 du DEP = 1
bit 7 du XL = 1
bit 7 du XL+DEP = 0

Pour l'initialisation des indicateurs à UN, H et N ne sont pas pris en compte. Une analyse du circuit montrera que le test fait pour I (initialisé à ZERO) a couvert presque tous les cas de fautes probables de la gestion de cet indicateur,

exception faite de deux cas qui sont testables pour I initialisé à UN sans autre condition. Pour l'indicateur Z les configurations sont:

DEP	0 0 0 0 0 0 0 0
XL+DEP	(il existe un bit à 1)

Pour des raisons d'utilisation de la mémoire on utilise XL = $41 et XL+DEP ≠ $41. Les indicateurs V et C se trouvent dans le même cas que I.

VI.2.4.- Adressage étendu

Pour ce mode d'adressage, aussi, la seule instruction utilisable est JMP; mais le problème est moins complexe parce qu'il n'y a pas de comptes-rendus de l'opérateur de calcul. Pour le transfert sur le bus DB, il faut prendre en compte l'octet des poids forts et l'octet des poids faibles de l'adresse. En limitant l'utilisation des adresses, on teste en deux pas: d'abord pour les poids faibles ($3F) et après pour les poids forts ($FF).

VI.2.5.- Adressage relatif

Les conditions sont semblables au mode indexé, mais l'octet poids faible du PC remplace l'octet poids faible du registre d'index. L'instruction de test est BRA.

VI.2.6.- Phase opération

Pour tester les pannes de classe IV dans la phase opération des instructions, il faut tester tous les types d'instruction de l'ensemble (103) et celà pour chaque indicateur selon les fonctions à réaliser:

(↕) : mettre l'indicateur soit à zéro, soit à un, selon le résultat de l'opération
(.) : ne pas modifier
(S) : mettre l'indicateur à un
(R) : mettre l'indicateur à zéro

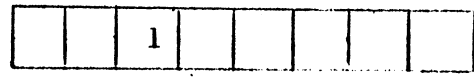
Un cas à part sont les instructions TPA et TAP.

Pour montrer la démarche, on a choisi quelques instructions répresentatives de ces actions; le test pour les autres instructions est semblable.

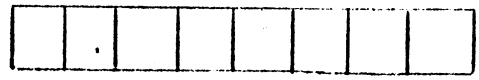
	H	I	N	Z	V	C
ADDB	↕	.	↕	↕	↕	↕
TSX	.	.	.	.	.	.
COMB	.	.	↕	↕	R	S
SEI	.	S	.	.	.	.
CLRB	.	.	R	S	R	R

Les conditions de test pour chaque instruction et chaque indicateur (selon l'action à réaliser sur lui) sont données en [COU80-1]. Le problème est réduit, simplement, à essayer de mettre ensemble les conditions pour le test des indicateurs dans le cas d'une instruction. Ainsi, pour ADDB le test de H, N, Z, V et C doit être fait en deux étapes: lere. étape: les indicateurs initialisés à UN et les compte-rendus à ZERO, et 2ème. étape: les indicateurs initialisés à ZERO et les compte-rendus mis à UN; pour cette instruction la valeur sur le bus DB est la valeur de l'opérande. Ainsi pour la lère. étape les conditions de test sont:

Indicateur H: opérande

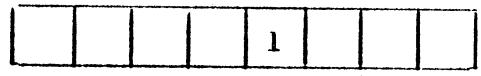


registre B

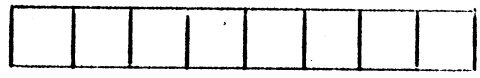


B + opérande

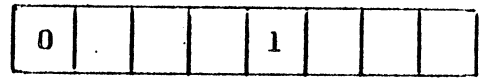
Indicateur N: opérande



registre B

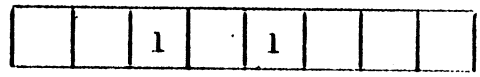


B + opérande

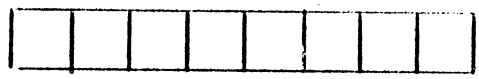


Il est très simple de combiner ces conditions dans une seule:

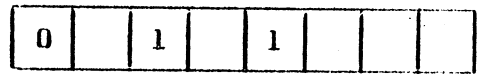
Indicateurs H, N: opérande



registre B

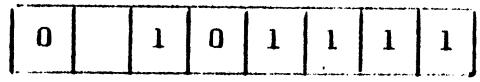


B + opérande

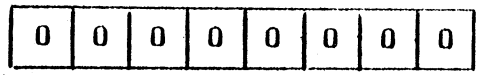


Si on continue avec le même raisonnement on aboutit aux conditions suivantes pour le test de l'instruction ADDB (lère. étape):

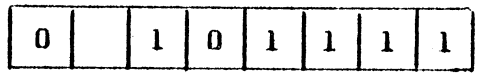
ler. pas: opèrande



registre B



B + opérande



2ème. pas: opérande

1	1	1	1	1	1	1	1

registre B

0	0	0	0	0	0	0	1

B + opérande

0	0	0	0	0	0	0	0

Pour le cas des instructions telles que TSX, COMB, SEI, CLRB, l'analyse est encore plus simple.

VI.3.- Classe V

Pour cette classe, les algorithmes du test hors-ligne ont été totalement programmés. Le test est exhaustif et identique au test en-ligne; chaque valeur du registre d'état (combinaison des valeurs différentes pour les indicateurs N, Z, V et C) est utilisée pour tester chacune des instructions de branchement. La localisation des programmes de test dans la mémoire n'est fixe pour aucune instruction; celles-ci peuvent être placées à n'importe quelle adresse, en respectant le séquencement du programme. Aucune des instructions utilisées (sauf celles d'initialisation du registre d'état) modifie les indicateurs. C'est un module réentrable, en conséquence enregistrable sur ROM; il occupe 971 octets, ce qui n'inclut pas les procédures de traitement des points de sortie FINTST et PCLV:

FINTST: est la sortie du module pour une unité sans panne de classe V. Dans notre cas c'est aussi la sortie pour une unité sans panne.

PCLV: est la sortie du module pour une unité avec panne de classe V.

Quelques modifications légères au module pourront nous permettre de préciser la configuration du registre d'état qui détecte la

panne; il est suffisant d'observer le registre d'état à la sortie PCLV. Des modifications moins simples pourront permettre de préciser l'instruction de branchement qui détecte la panne. Ainsi pour ce dernier cas et pour la configuration N = Z = V = C = 0, il faudrait changer les instructions:

```
EOCO    BCC    EOZO
        JMP    PCLV     ; sortie erreur
```

par l'ensemble des instructions suivantes:

```
EOCO    BCC    EOZO
        TPA
        LDAB   #0       ; code d'instruction
        JMP    PCLV
```

Et cela pour chaque instruction de branchement et pour chaque configuration. La structure du programme de test de la classe V est la suivante:

```
debut
     CC0000:  traitement de la configuration N=Z=V=C=0;
              ------------------------
              --------------------
     CC0001:  traitement de la config. N=V=Z=0; C=1;
              ----------------------
              --------------------
     CC0011:  traitement de la config. N=V=0; Z=C=1;
     -----    ----------------------
     -----    --------------------
      ----    ----------------
     CC1111:  traitement de la config. N=Z=V=C=1;
     -----    ------------------
      ----    ----------------
      ----    ----------
     CC0111:  traitement de la config. N=0; Z=V=C=1
fin.
```

Pour chaque traitement de test CCxxxx, après le positionnement des indicateurs, on teste les instructions applicables si la condition est vraie, puis si la condition est fausse. Ainsi, après le test de la configuration N=C=1; Z=V=0, le programme teste la configuration N=1, Z=V=C=0 de la manière suivante:

```
CC1000    EQU    *
          CLC           ;faire CC = $C8
* branchement de type 1: condition vraie
*
B8T1      EQU    *
E8C0      BCC    E8Z0
          JMP PCLV      ; sortie erreur
E8Z0      BNE E8NV1
          JMP PCLV      ; sortie erreur
             -
             -
             -
             -
E8V0      BVC E8
          JMP PCLV      ; sortie erreur
E8        BRA B8T2
          JMP PCLV      ; sortie erreur
* branchement de type 2: condition fausse
*
B8T2      EQU   *
          BCS  PCLV8    ; erreur
          BEQ  PCLV8    ; erreur
           -    -
           -    -
           -    -
           -    -
          BVS  PCLV8    ; erreur
          BRA  CC1010   ; test suivant
PCLV8     EQU   *
          JMP PCLV      ; sortie erreur
```

En conclusion, la figure 26 montre l'organisation du logiciel et l'encombrement de mémoire de chaque module. Pour certains la valeur représente exactement l'encombrement (*), tandis que pour d'autres, c'est une estimation faite à partir des tests programmés pour quelques instructions.

<table>
<tr><td rowspan="2">phase
adressage</td><td colspan="2">(*)classes 0 et II
(A)
(150 octets)</td></tr>
<tr><td>(*)classes
0, I, II
et III
(250 octets)</td><td>(B1)
(B2)</td></tr>
<tr><td rowspan="2">phase
opération</td><td colspan="2">classes 0 et III
(4K)</td></tr>
<tr><td colspan="2">classes I et II
(2,5 K)</td></tr>
<tr><td rowspan="2">classe IV</td><td colspan="2">(*)phase adressage
(350 octets)</td></tr>
<tr><td colspan="2">phase opération
(2,5 K)</td></tr>
<tr><td colspan="3">(*)classe V
(phase opération)
(980 octets)</td></tr>
</table>

Figure 26.- Organisation et taille du logiciel

VII.- Implantation

Le logiciel de test hors-ligne de la partie contrôle, completé par le logiciel de test de la partie opérative, est destiné à être implanté à la fois sur une machine de test classique (TEKTRONIX 3270) et sur un équipement de test automatique du type "tout ou rien". Cette machine de test nommée machine de test légère est orientée exclusivement vers le test du microprocesseur M6800; elle est différente des machines de test classiques par son fondement, conception et coût.

VII.1.- Machine de test classique

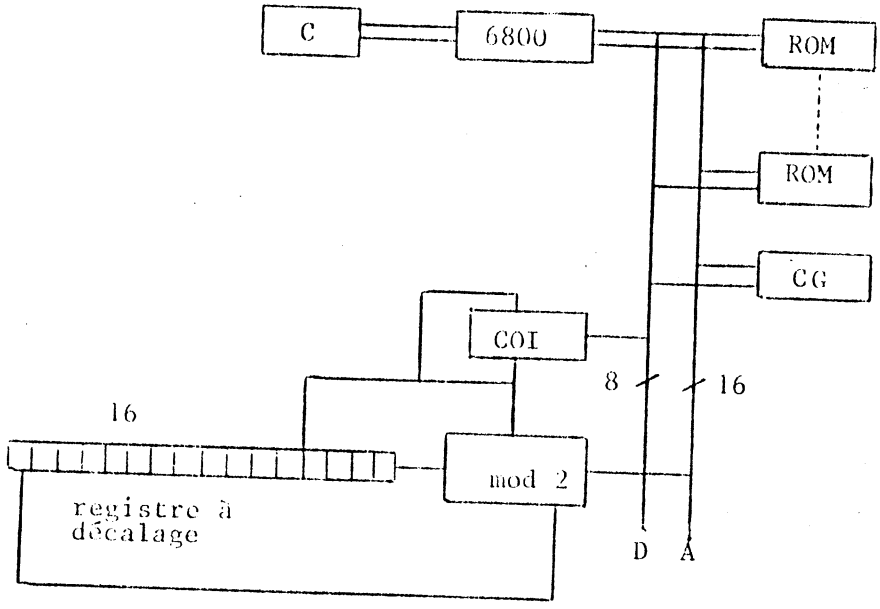
Une machine de test classique est basée sur le principe consistant à vérifier les réponses à des vecteurs de test; ces vecteurs de test (générées ou stockées quelque part) sont envoyés sur le circuit, puis on observe et on compare les résultats avec des valeurs justes, qui sont soit enregistrées par avance, soit déterminées en temps réel à l'aide d'une unité de référence. C'est à dire qu'on utilise la machine pour le test d'un circuit non-intelligent. Le logiciel décrit plus haut sera la base de la génération de vecteurs de test classiques destinés à être implantés sur une machine de test classique TEKTRONIX 3270 où l'on vérifiera les réponses du circuit. Cela doit faciliter l'analyse de défaillance par localisation des parties défaillantes.

VII.2.- Machine de test légère orientée M6800

Le logiciel obtenu ne peut pas être suffisant en lui-même, puisqu'il n'est apte à détecter les hypothèses qu'il teste que s'il n'y a pas de déséquencement des programmes dû à l'occurrence des erreurs (non de pannes) sur l'adresse de la prochaine instruction (un programme ne peut jamais être suffisant en lui-même pour le test, quel qu'il soit).

Pour détecter ces occurrences de déséquencement, une approche originale a été considérée. De la même manière que les propriétés inhérentes à une unité centrale sont utilisées pour le test en-ligne, ces mêmes propriétés peuvent être utilisées pour un test hors-ligne; c'est à dire qu'on utilise la capacité intelligente du circuit testé, en ajoutant des mécanismes matériels pour la détection des déséquencements. Mais les mécanismes utilisés par le test en-ligne: chien de garde (CG) et détecteur de code-opération invalide (COI) ne sont pas très adaptés pour la détection des pannes du coeur, qui ne créent pas très souvent de déséquencement (le mécanisme COI doit permettre d'abord de repérer chaque code-opération, puis de reconnaître s'il fait partie ou non du jeu d'instructions; le chien de garde est relancé périodiquement s'il n'y a pas de panne). Un autre mécanisme est très bien adapté pour détecter ces ocurrences: l'analyse de signature. En effet, l'analyse de signature est très bien adaptée à la détection de très peu de discordances dans une série de données qui peut être très longue; étant donné que le logiciel est déterminé de manière à manifester les pannes, il n'y a pas de problème dans le choix de la séquence d'entrée. L'analyse de signature sera active sur le bus adresse, seulement lors de l'acquisition du code-opération. Il y a diverses signatures possibles, utilisant des polynômes différents; dans notre cas, on utilise le schéma de [DAV80] qui n'utilise qu'une boucle de réaction.

La figure 27 montre la structure de la machine de test légère orientée M6800; on peut considérer qu'elle est plus conforme à l'esprit de test d'unité centrale, que ne l'est une machine de test classique.



CG: chïen de garde
COI: détecteur de code opération invalide
mod 2: analyseur de signature utilisant un registre à décalage avec réaction

Figure 27.- Machine de test légère orientée M6800

VIII.- Conclusions

Cette implantation a satisfait notre objectif principal: minimiser l'encombrement mémoire et le temps d'exécution du programme de test. Un encombrement de 10K pour la partie contrôle, qui ajouté aux 2.5K ou 3K nécessaires pour le test de la partie opérative, constitue un résultat satisfaisant de ce point de vue. Le temps d'exécution est facilement mesurable étant donné que le programme est presque totalement séquentiel.

La qualité du programme de test n'est pas à montrer étant donné que les hypothèses prises en compte sont plus fortes que celles généralement utilisées. L'efficacité du test sera définitivement démontrée avec la réalisation de la machine légère, qui permettra une évaluation réelle. On peut noter que l'application de cette méthodologie pour le test de la partie contrôle du M68000 menerait à une machine de test légère encore plus simple que celle réalisée pour le M6800.

Une autre perspective pour cette méthode serait la possibilité de son application à d'autres microprocesseurs, moyennant l'automatisation de la génération de vecteurs de test et une disponibilité plus grande de l'information sur l'architecture, fournie par les fabricants.

REFERENCES

[AND81] ANDERSON T., LEE P. A.
"Fault tolerance: Principles and Practice"
Prentice-Hall 1981

[ARL79] ARLAT J.
"Conception d'un microcalculateur tolérant aux fautes par diversification fonctionnelle"
Thèse de Docteur-Ingénieur, Toulouse, 1979

[AVI76] AVIZIENIS A.
"Fault-Tolerant Systems"
IEEE Transactions on Computers, Vol. C-25, No. 12, Decembre 1976

[AKE78] AKERS S.B.
"Functional testing with binary decision diagrams"
8th. Fault-tolerant computing symposium, Toulouse-FRANCE, Juin 1978

[AND76] ANDERSON R.E.
"Test methodes change to meet complex demandes"
Electronics, Avr. 15, 1976

[BAL79] BALLARD D.R.
"Designing fail safe microprocessor systems"
Electronics, Jan. 4, 1979

[BAR76] BARRACLOUGH W., CHIANG A.C.L., SOHL W.
"Techniques for testing the microcomputer family"
Proc. of the IEEE, Vol. 64, No. 6, Juin 1976

[BIS78] BISSET S.
"LSI tester gets microprocessors to generate their own test patterns"
Electronics, Mai 25, 1978

[BRE80] BREUER M. A., FRIEDMAN A. D.
"Functional level primitives in test generation"
IEEE Transactions on Computers Vol. C-29, No. 3, Mars 1980

[CHI76] CHIANG A.C.L., McKASKILL R.
"Two new approaches simplify testing of microprocessors"

Electronics, Jan. 22, 1976

[COU80] COURTOIS B.
"Mecanismes de panne, hypothèses panne et test analytique de circuits LSI-NMOS (HMOS)"
Rapport de Recherche IMAG, No. 196, Avril 1980

[COU80-1] COURTOIS B.
"Test foctionnel de la partie contrôle d'unités centrales integrées"
Rapport de Recherche IMAG, No. 203, Mai 1980

[COU82] COURTOIS B., EYNARD J., GEAY F., MARCHAL P., POSTIGO C.
"Test en-ligne et hors-ligne de microprocesseurs: classification et applications"
Rapport de Recherche IMAG, No. 276, Nov. 1981

[CHI76] CHIANG A.C.L., McKASKILL R.
"Two new approaches simplify testing of microprocessors"
Electronics, Jan. 22, 1976

[DAV80] DAVID R.
"Testing by feedback shift registers"
IEEE Transactions on Computers, Vol. C.29, No.7, Mai 1977

[HAC75] HACKMEISTER D., CHIANG A.C.L.
"Microprocessor test techniques reveals instruction pattern sensitivity"
Computer Design, Dec. 1975

[LEA75] LEAMAN R.J., LLOYD M.H., REPTON C.S.
"The development and testing of a processor self-test program"
The Computer Journal, Vol. 16, No. 4, Avr. 1973

[LIP79] LIPPMAN M.D., DONN E.S.
"Design forethought promotes easier testing of microprocessor boards"
Electronics, Jan. 18, 1979

[SCR78] SCRUPSKI S.E.
"Why and how users test microprocessors"
Electronics, Mar. 2, 1978

[SRI77] SRINI V.P.
"Fault diagnosis of microprocessor systems"
Computer Jan. 1977

[THA78] THATTE S.M., ABRAHAM J.A.
"A methodology for functional level testing of microprocessors"
Proceed. of the 8th. Fault Tolerant Computing Symposium, Toulouse, FRANCE, Juin 1978.

[THA79-1] THATTE S.M.
"Test generation for microprocessors"
thèse Ph.D., University of Illinois, May 1979

[THA79-2] THATTE S.M., ABRAHAM J.A.
"Test generation for general microprocessor architectures"
Proceed. of the 9th. Fault Tolerant Computing Symposium, Madison-USA, Juin 1979.

ANNEXE A

			cycle				opérateur			de	calcul		
instructions opératives et opération	registres opératifs type de séq. A	B	origine DB normale	conditions pour test origine DB	origine ADLI normale	conditions pour test origine ADLI	commande d'entrée DB normale v. III-1	conditions pour test commande DB	carry-in normal	conditions pour test carry-in	code operation opérateur v. III-1	conditions pour test code operation	conditions pour test transf. instr.
A+DA A+M → A	5	1	M	(1)-(1')	A		DB	R≠A+M ≠A+SR(M) ≠A+DAA(A)	0	1	ADD	M≠1F résultats donnés par les 1F code faux.	R≠A-M Rem. 1
A+DB B+M → A	1	5	M	(1)-(1')	B		DB	R≠B+M ≠B+SR(M) ≠B+DAA(A)	0	1	ADD	R≠1F résultats donnés par les 1F code faux	R≠B-M
ABA A+B → A	5	2	B	2	A		DB	R≠A+B ≠A+SR(B) ≠A+DAA(A)	0	1	ADD	idem ci-dessus	R≠A-b
SBCA A+M+C → A	5	1	M	(1)-(1')	A	5	DB	≠A+M+C ≠A+SR(M)+C ≠A+DAA(A)+C	C	C=0,1	ADD	idem ci-dessus	R≠A-M ≠A-M-1
ABCB B+M+C → B	1	5	M	(1)-(1')	B	5	DB	R≠B+M+C ≠B+SR(M)+C ≠B+DAA(A)+C	C	C=0,1	ADD	idem ci-dessus	R≠B-M ≠B-M-1
ANDA A.M → A	5	1	M	(1)-(1') A=FF	A	5 M=FF	DB	R≠A.M ≠A.SR(M) ≠A.DAA(A)	1	testé par le code opération	AND	idem ci-dessus	R≠A+M ≠A-M ≠A+M+1 ≠A-M-1
ANDB B.M → B	1	5	M	(1)-(1') B=FF	B	5 M=FF	DB	R≠B.M ≠B.SR(M) ≠B.DAA(A)	1	testé par le code opération	AND	idem ci-dessus	R≠B+M ≠B-M ≠B+M+1 ≠B-M-1

Table I - (suite)

instructions opératives et opération	registres opératifs type de séq. A	B	cycle : origine DB normale	conditions pour test origine DB	origine ABLI normale	conditions pour test origine ABLI	opérateur : commande d'entrée DB normale v. III-1	conditions pour test commande DB	de calcul : carry-in normal	conditions pour test carry-in	code opération opérateur v. III-1	conditions pour test code opération	conditions pour test transf. instr.
BIT A A.M	2	1	M	(1)-(1') A=FF	A	2 M=FF	DB	INDIC(A.M) ≠INDIC(A.M) ≠INDIC(A.SR(M)) ≠INDIC(A.DAA(A))	1	testé par le code opération	AND	INDIC(A.M) ≠INDIC(15 résultats donnés par les 15 codes faux)	reste par le code opération
BIT B B.M	1	2	M	(1)-(1') B=FF	B	2 M = FF	DB	INDIC(B.M.) ≠INDIC(B.M) ≠INDIC(B.SR(M)) ≠INDIC(B.DAA(A))	1	teste par le code opération	AND	INDIC(B.M) ≠INDIC(15 résultats donnés par les 15 codes faux)	teste par le code opération
CLR 00 → M	1	1	M	(2)	00	K_{ABLI}=FF. M=FF	DB	testé par l'origine ABLI	1	testé par le code opération	AND	testé par l'origine ABLI	testé par l'origine ABLI
CLRA 00 → A	5	1	A	5	00	K_{ABLI}=FF A=FF	DB	testé par l'origine ABLI	1	testé par le code opération	AND	testé par l'origine ABLI	testé par l'origine ABLI
CLRB 00 → B	1	5	B	5	00	K_{ABLI}=FF B=FF	DB	testé par l'origine ABLI	1	testé par le code opération	AND	testé par l'origine ABLI	testé par l'origine ABLI
CMPA A-M	2	1	M	(1)-(1') A=FF	A	2 M=FF	$\overline{DB}$	INDIC(A-M) ≠INDIC(A+M+1) ≠INDIC(A+SR(M)+1) ≠INDIC(A+DAA(A)+1)	1	0 INDIC(A-M) ≠INDIC(A-M-1)	ADD	INDIC(A-M) ≠INDIC(15 résultats donnés par les 15 codes faux)	INDIC(A-M) ≠INDIC(A+M)

Table I - (suite)

instructions opératives et operation	registres opératifs type de séq. A	B	cycle: origine DB normale	condition pour test origine DB	origine ABLI normale	conditions pour test origine ABLI	commande d'entrée DB normale v. III-1	opérateur: conditions pour test commande DB	carry-in normal	de calcul: conditions pour test carry-in	code operation opérateur v. III-1	conditions pour test code opération	conditions pour test transf. instr.
CMPB B-M	1	1	M	(1)-(1') B=FF	B	A M=FF	$\overline{DB}$	INDIC(B-M) ≠INDIC (B+M+1) ≠INDIC (B+SR(B)+1) ≠INDIC (B+DAA(A) +1)	1	0 INDIC(B-M) ≠INDIC (B-M-1)	ADD	INDIC(B-M) ≠INDIC (15 résultats ...)	INDIC(B-M) ≠INDIC(B+M) ≠INDIC (B+M+1)
COMA A-B	2	2	B	2 A=FF	A	2 B=FF	$\overline{DB}$	INDIC(A-B) ≠INDIC (A+B+1) ≠INDIC (A+SR(B)+1) ≠INDIC (A+DAA(A)+1)	1	0 INDIC(A-B) ≠INDIC (A-B-1)	ADD	INDIC(A-B) ≠INDIC (15 resultats ...)	INDIC(A-B) ≠INDIC(A+B) ≠INDIC (A+B+1)
COM M → M	1	1	M	(2)	FF	$K_{ABLI} \neq FF$	$\overline{DB}$	$\overline{M} \neq FF$ +SR(M)+1 $\overline{M} \neq FF$+ DAA(A)+1	1	0 testé par l'origine	ADD	$\overline{M} \neq 15$ resultats ...)	$\overline{M} \neq FF+M$ $\neq FF+M+1$
COMA $\overline{A}$ → A	5	1	A	5	FF	$K_{ABLI} \neq FF$	$\overline{DB}$	$\overline{A} \neq FF$+ SR(A)+1 $\overline{A} \neq FF$ +DAA(A)+1	1	0 testé par l'origine ABLI	ADD	$\overline{A} \neq 15$ resultats ...)	$\overline{A} \neq FF+A$ $\neq FF+A+1$
COMB $\overline{B}$ → B	1	5	B	5	FF	$K_{ABLI} \neq FF$	$\overline{DB}$	$\overline{B} \neq FF$ +SR(B)+1 ≠FF+ DAA(A)+1	1	0 testé par l'origine ABLI	ADD	$\overline{B} \neq 15$ résultats ...)	$\overline{B} \neq FF+B$ $\neq FF+B+1$
NLG 00-M → M	1	1	M	(2)	00	K_{ABLI}=FF	$\overline{DB}$	R≠00 +SR(M)+1 ≠00+DAA(A)+1	1	0 testé par l'origine	ADD	R≠ 15 resultats ...)	R≠00+M ≠00+M+1

Table I - (suite)

instructions opératives et opération	registres opératifs type de séq. A	registres opératifs type de séq. B	cycle opérateur de calcul : origine DB normale	conditions pour test origine DB	origine ABLI normale	conditions pour test origine ABLI	commande d'entrée DB normale v. III-1	conditions pour test commande DB	carry-in normal	conditions pour test carry-in	code opération opérateur v. III-1	conditions pour test code opération	conditions pour test transf. instr.
NEGA 00-A → A	5	1	A	5	00	$K_{ABLI}=FF$	$\overline{DB}$	R≠00 +SR(A)+1 ≠00+ DAA(A)+1	1	0 testé par l'origine ABLI	ADD	R≠ 15 résultats ...	R≠00+A ≠00+A+1
NEGB 00-B → B	1	5	B	5	00	$K_{ABLI}=FF$	$\overline{DB}$	R≠00 +SR(B)+1 ≠00+ DAA(A)+1	1	0 teste par l'origine ABLI	ADD	R≠ 15 résultats ...	R≠00+B ≠00+B+1
DAA	5	1	FF	$K_{DB}\neq FF$	A	5	DAA	R≠FF+A ≠00+A	0	1 testé par l'origine ABLI	ADD	R≠ 15 résultats ...	R≠A-DAA(A) ≠A-DAA(A)-1
DEC M-1 → M	1	1	M	(1)-(1') (2)	FF	$K_{ABLI}\neq FF$	DB	R≠FF+$\overline{M}$ ≠FF+SR(M) ≠FF+DAA(A)	0	1 testé par l'origine ABLI	ADD	R≠ 15 résultats ...	R≠FF-M ≠FF-M-1
DECA A-1 → A	5	1	A	5	FF	$K_{ABLI}\neq FF$	DB	R≠FF+$\overline{A}$ ≠FF+SR(A) ≠FF+DAA(A)	0	1 testé par l'origine ABLI	ADD	R≠ 15 résultats ...	R≠FF-A ≠FF-A-1
DECB B-1 → B	1	5	B	5	FF	$K_{ABLI}\neq FF$	DB	R≠FF+$\overline{B}$ ≠FF+SR(B) ≠FF+DAA(A)	0	1 testé par l'origine ABLI	ADD	R≠ 15 résultats ...	R≠FF-B ≠FF-B-1
EORA A ⊕ M → M	5	1	M	(1)-(1')	A	5	DB	R≠A ⊕ SR(M) ≠A ⊕ DAA(A)	0	1 teste par le code opération	X-OR	R≠ 15 résultats	R≠A+M ≠A+M+1 ≠A-M ≠A-M-1

Table I - (suite)

instructions opératives et opération	registres opératifs type de séq. A	B	cycle : origine DB normale	conditions pour test origine DB	origine ABLI normale	conditions pour test origine ABLI	opérateur : commande d'entrée DB normale v. III-1	conditions pour test commande DB	carry-in normal	de calcul : conditions pour test carry-in	code opération opérateur v. III-1	conditions pour test code opération	conditions pour test transf. instr.
EORB B⊕M → I	1	5	M	(1)-(1')	B	5	DB	R≠S ⊕ SR(M) ≠AB|DAA(A)	0	1 testé par le code opération	X-OR	R≠ 15 résultats ...	R≠B+M ≠B+M+1 ≠B-M ≠B-M-1
INC M+1 → M	1	1	M	(1)-(1') (2)	00	K_{ABLI}=FF	DB	R≠M+1 ≠SR(M)+1 ≠DAA(A)+1	1	0 testé par l'origine ABLI	ADD	R≠ 15 résultats ...	R≠-M-1
INCA A+1 → A	5	1	A	5	00	K_{ABLI}=FF	DB	R=$\bar{A}$+1 ≠SR(A)+1 ≠DAA(A)+1	1	0 testé par l'origine ABLI	ADD	R≠ 15 résultats ...	R≠-A-1
INCB B+1 → B	1	5	B	5	00	K_{ABLI}=FF	DB	R/$\bar{B}$+1 ≠SR(B)+1 ≠DAA(A)+1	1	0 testé par l'origine ABLI	ADD	R≠ 15 résultats ...	R≠-B-1
LDAA M → A	5	1	M	(1)-(1')	FF	K_{ABLI}≠FF M=FF	DB	M≠SR(M) ≠DAA(A)	1	0	AND	M≠ 15 résulats ...	M≠FF+M ≠FF-M ≠FF+M+1 ≠FF-M-1
LDAB M → B	1	5	M	(1)-(1')	FF	K_{ABLI}≠FF M=FF	DB	M≠SR(M) ≠DAA(A)	1	0	AND	M≠15 résulats ...	M≠FF+M ≠FF-M ≠FF+M+1 ≠FF-M-1
ORAA A|M → A	5	1	M	(1)-(1') A=00	A	5 M=00	DB	R≠A|SR(M) ≠A|DAA(A)	0	1	OR	R≠ 15 résultats ...	R≠A+M ≠A+M+1 ≠A-M ≠A-M-1

Table I - (suite)

instructions opératives et opération	registres opératifs type de seq. A	B	cycle : origine DB normale	conditions pour test origine DB	origine ABLI normale	conditions pour test origine ABLI	opérateur : commande d'entrée DB normale v. III-1	conditions pour test commande DB	de : carry-in normal	conditions pour test carry-in	calcul : code opération opérateur v. III-1	conditions pour test code opération	conditions pour test transf. instr.
ORAB B ∨ M → B	1	5	M	(1)-(1') B=0()	B	5 M=0()	DB	R≠B ∣SR(M) ≠B ∣DAA(A) 1	0	1	OR	R≠ 15 résultats ...	R≠B+M ≠B+M+1 ≠B-1 ≠B M-1
PUSA A → M_{SP} SP-1 → SP	2	1	A Rem. 2	2	-	-	-		-	-	-	-	-
PUSB B → M_{SP} SP-1 → SP	1	2	B Rem. 2	2		-	-	-	-	-	-	-	-
PULA SP+1 → SP M_{SP} → A	3	1	M Rem. 2	(1)-(1')	-	-	-	-	-	-	-	-	-
PULB SP+1 → SP M_{SP} → B	1	3	M Rem. 2	(1)-(1')	-	-	-	-	-	-	-	-	-
ROL ROL(M) → M	1	1	M	(1)-(1')	FF	$K_{ABLI} \neq FF$	DB	B ≠ROL(M) ≠ROL(SR(M), ≠ROL(DAA(A))	C	C=0,1	SL	R≠ 15 résultats ...	R≠-M ≠M-1 ≠M+1 ≠ROR(M) ≠ASL(M) ≠ASR(M) ≠LSR(M)
ROLA ROL(A) → A	5	1	A	5	FF	$K_{ABLI} \neq FF$	DB	B≠ROL(A) ≠ROL(SR(A)) ≠ROL(DAA (A))	C	C=0,1	SL	R≠ 15 résultats ...	R≠-A ≠A-1 ≠A+1 ≠ROR(A) ≠ASL(A) ≠ASR(A) ≠LSR(A)

Table I - (suite)

Instruction opératives et opération	registres opératifs type de seq. A	registres opératifs type de seq. B	cycle origine DB normale	cycle conditions pour test origine DB	cycle origine ABLI normale	cycle conditions pour test origine ABLI	opérateur commande d'entrée de normale v. III-1	opérateur conditions pour test commande DB	opérateur carry-in normal	DB conditions pour test carry-in	calcul code opération opérateur v. III-1	calcul conditions pour test code operation	conditions pour test transf. instr.
ROLB ROL(B) → B	1	5	B	5	FF	K_{ABLI}≠FF	DB	R=ROR(DAA (A)) R≠ROR(DAA (A))	0	1 testé avec origine ABLI	SI	R≠ 15 résultats ...	R≠=L ≠B-1 ≠B+1 ≠ROR(B) ≠ASL(B) ≠ASR(B) ≠LSR(B)
ROR ROR(M) → M	1	1	M	(1) = (1')	00	K_{ABLI}=FF	SR	R=ROR(DAA (A)) R≠ROR(DAA (A)) R/M R≠M	0	1 testé avec origine ABLI	ADD	R≠ 15 résultats ...	R≠=M ≠M-1 ≠M+1 ≠ROL(M) ≠ASL(M) ≠ASR(M) ≠LSR(M)
RORA ROR(A) → A	5	1	A	5	00	K_{ABLI}=FF	SR	R=ROR(DAA (A)) R≠ROR(DAA (A)) ≠A ≠Ā	0	1 testé avec origine ABLI	ADD	R≠ 15 résultats ...	R≠=A ≠A-1 ≠A+1 ≠ROL(A) ≠ASL(A) ≠ASR(A) ≠LSR(A)
RORB ROR(B) → B	1	5	B	5	00	K_{ABLI}=FF	SR	R=ROR(DAA (A)) R≠ROR(DAA (A)) ≠B ≠B̄	0	1 testé avec origine ABLI	ADD	R≠ 15 résultats ...	R≠=B ≠B-1 ≠B+1 ≠ROL(B) ≠ASL(B) ≠ASR(B) ≠LSR(B)

Table 1 - (suite)

instructions opératives et opération	registres opératifs type de séq. A	registres opératifs type de séq. B	cycle opérateur de calcul : origine DB normale	conditions pour test origine DB	origine ABLI normale	conditions pour test origine ABLI	commande d'entrée DB normale v. III-1	conditions pour test commande DB	carry-in normal	conditions pour test carry-in	code opération opérateur v. III-1	conditions pour test code opération	conditions pour test transf. instr.
ASL ASL(M) → M	1	1	M	(1)-(1')	FF	$K_{ABLI} \neq FF$	DB	R≠ASL($\bar{M}$) ≠ASL(SR(M)) ≠ASL(DAA(M))	0	1 testé avec origine ABLI	SL	R≠ 15 résultats ...	R≠-M ≠M-1 ≠M+1 ≠ROL(M) ≠ROR(M) ≠ASR(M) ≠LSR(M)
ASLA ASL(A) → A	5	1	A	5	FF	$K_{ABLI} \neq FF$	DB	R≠ASL($\bar{A}$) ≠ASL(SR(A)) ≠ASL(DAA(A))	0	1 testé avec origine ABLI	SL	R≠ 15 résultats ...	R≠-A ≠A-1 ≠A+1 ≠ROL(A) ≠ROR(A) ≠ASR(A) ≠LSR(A)
ASLB ASL(B) → B	1	5	B	5	FF	$K_{ABLI} \neq FF$	DB	R≠ASL($\bar{B}$) ≠ASL(SR(B)) ≠ASL(DAA(A))	0	1 testé avec origine ABLI	SL	R≠ 15 résultats ...	R≠-B ≠B-1 ≠B+1 ≠ROL(B) ≠ROR(B) ≠ASR(B) ≠LSR(B)
ASR ASR(M) → M	1	1	M	(1)-(1')	00	$K_{ABLI} = FF$	SR	R≠M ≠$\bar{M}$ ≠ASR(DAA(A))	0	1 testé avec origine ABLI	ADD	R≠ 15 résultats ...	R≠-M ≠M-1 ≠M+1 ≠ROL(M) ≠ROR(M) ≠ASL(M) ≠LSR(M)

instructions opératives et opération	registres opératifs type de séq. A	registres opératifs type de séq. B	cycle — origine DB normale	cycle — conditions pour test origine DB	cycle — origine ABLI normale	cycle — conditions pour test origine ABLI	opérateur — commande d'entrée DB normale v. III-1	opérateur — conditions pour test commande DB	opérateur — carry-in normal	calcul — conditions pour test carry-in	calcul — code opération opérateur v. III-1	calcul — conditions pour test code opération	calcul — conditions pour test transf. instr.
ASRA ASR(A) → A	5	1	A	5	00	K_{ABLI}=FF	SR	R≠A ≠A ≠ASR(DAA(A))	0	1 testé avec origine ABLI	ADD	R≠ 15 résultat ...	R≠-A ≠A-1 ≠A+1 ≠ROL(A) ≠ROR(A) ≠ASL(A) ≠LSR(A)
ASRB ASR(B) → B	1	5	B	5	00	K_{ABLI}=FF	SR	R≠B ≠B ≠ASR(DAA(A))	0	1 testé avec origine ABLI	ADD	R≠ 15 résultats ...	R≠-B ≠B-1 ≠B+1 ≠ROL(B) ≠ROR(B) ≠ASL(B) ≠LSR(B)
LSR LSR(M) → M	1	1	M	(1)-(1')	00	K_{ABLI}=FF	SR	R≠M ≠M ≠LSR(DAA(A))	0	1 testé avec origine ABLI	ADD	R≠ 15 résultats ...	R≠-M ≠M+1 ≠M-1 ≠ROL(M) ≠ROR(M) ≠ASL(M) ≠ASR(M)
LSRA LSR(A) → A	5	1	A	5	00	K_{ABLI}=FF	SR	R≠A ≠A ≠LSR(DAA(A))	0	1 testé avec origine ABLI	ADD	R≠ 15 résultats ...	R≠-A ≠A+1 ≠A-1 ≠ROL(A) ≠ROR(A) ≠ASL(A) ≠ASR(A)

Table 1 - (suite)

instructions opératives et opération	registres opératifs type de séq. A	registres opératifs type de séq. B	cycle opérateur de calcul : origine DB normale	conditions pour test origine DB	origine ABLI normale	conditions pour test origine ABLI	commande d'entrée DB normale v. III-1	conditions pour test commande DB	carry-in normal	conditions pour test carry-in	code opération opérateur v. III-1	conditions pour test code opération	conditions pour test transf. instr.
LSRB LSR(B) → B	1	2	B	2	FF	$K_{ABLI} \neq FF$	SR	R≠B ≠$\overline{B}$ ≠LSR(DAA(A))	0	1 testé avec origine ABLI	ADD	R≠ 15 résultats ...	R≠-B ≠B-1 ≠B+1 ≠ROL(B) ≠ROR(B) ≠ASL(B) ≠LSR(B)
ATAA A → M	2	1	A	2	FF	$K_{ABLI} \neq FF$ A≠FF	DB	A≠DAA(A) A≠SR(A)	1	0 testé avec code opération	AND	R≠ 15 résultats ...	A≠FF-A ≠FF+A+1 ≠FF-A-1
BTAB B → M	1	2	B	2	FF	$K_{ABLI} \neq FF$ B=FF	$\overline{DB}$	B≠DAA(A) B≠SR(B)	1	0 testé avec code opération	AND	R≠ 15 résultats ...	B≠FF-B ≠FF+B+1 ≠FF-B-1
SUBA A-M → A	2	1	M	(1)-(1*)	A	5	$\overline{DB}$	R≠A+M+1 ≠A+SR(M)+1 ≠A+DAA(A)+1	1	0 testé avec origine ABLI	ADD	R≠ 15 résultats ...	R≠A+M
SUBB B-M → B	1	2	M	(1)-(1*)	B	5	$\overline{DB}$	R≠B+M+1 ≠B+SR(M)+1 ≠B+DAA(A)+1	1	0 testé avec origine ABLI	ADD	R≠ 15 résultats ...	R≠B+M
SBA A-B → A	2	1	B	2	A	5	$\overline{DB}$	R≠A+B+1 ≠A+SR(B)+1 ≠A+DAA(A)+1	1	0 testé avec origine ABLI	ADD	R≠ 15 résultats ...	R≠A+B

Table 1 - (suite)

instructions opératives et opération	registres opératifs type de séq. A	registres opératifs type de séq. C	cycle de calcul : origine DB normale	conditions pour test origine DB	origine ABLI normale	conditions pour test origine ABLI	opérateur : commande d'entrée DE normale v. III-1	conditions pour test commande DB	carry-in normal	conditions pour test carry-in	code opération opérateur v. III-1	conditions pour test code opération	conditions pour test transf. instr.
SBCA A-M-C → A	3	1	M	(1)-(1')	A	5	$\overline{DB}$	R≠A+M ≠A+M+1 ≠A+SR(M) ≠A+SR(M)+1 ≠A+DAA(A) ≠A+DAA(A)+1	C	C = 0,1	ADD	R≠ 15 résultats ...	testé par commande DB
SBCB B-M-C → B	1	0	M	(1)-(1')	B	5	$\overline{DB}$	R≠B+M ≠B+M+1 ≠B+SR(M) ≠B+SR(M)+1 ≠B+DAA(A) ≠B+DAA(A)+1	C	C = 0,1	ADD	R≠ 15 résultats ...	testé par commande DB
TAB A → B	2	3	A	2	FF	$K_{ABLI} \neq FF$ A=FF	DB	A≠SR(A) A≠DAA(A)	1	0	AND	A≠ 15 résultats ...	A≠FF+A ≠FF-A ≠FF+A+1 ≠FF-A-1
TBA B → A	3	2	B	2	FF	$K_{ABLI} \neq FF$ B=FF	DB	B≠SR(B) B≠DAA(A)	1	0	AND	B≠ 15 résultats ...	B≠FF+B ≠FF-B ≠FF+B+1 ≠FF-B-1
TST M=00	1	1	M	(1)-(1') M=UN $K_{DB} = 00$	00	$K_{ABLI} = FF$ M=00	DB	INDIC(M) ≠INDIC($\overline{M}$) ≠INDIC(SR(M)) ≠INDIC(DAA(M))	0	1 INDIC(M) ≠INDIC(M+1)	ADD	INDIC(M) ≠INDIC (15 résultats ...)	
TSTA A=00	-	1	A	2 A=UN $K_{DB-A} = 00$	00	$K_{ABLI} = FF$ A=00	DB	INDIC(A) ≠INDIC($\overline{A}$) ≠INDIC(SR(A)) ≠INDIC(DAA(A))	0	1 INDIC(A) ≠INDIC(A+1)	ADD	INDIC(A) ≠INDIC (15 résultats ...)	

Table 1 - (suite)

instructions opératives et opération	registres opératifs type de seq. A	B	cycle opérateur de calcul: origine DB normale	conditions pour test origine DB	origine ABLT normale	conditions pour test origine ABLT	commande d'entrée DB normale v. III-1	conditions pour test commande DB	carry-in normal	conditions pour test carry-in	code opération opérateur v. III-1	conditions pour test code opération	conditions pour test transf. instr.
TSTB B-00	1	-	B	2 B-UN $K_{DB-B}=00$	00	$K_{ABLT}=FF$ B=0	DB	INDIC(B) ≠INDIC(B) ≠INDIC(SR(B)) ≠INDIC(DAA(N))		INDIC(B) ≠INDIC(B+1)	ADD	INDIC(B) ≠INDIC (15 résultats ...)	

instructions non opératives et opération	registres opératifs type de séq.		transferts sur bus DB	transferts sur bus ABLI (rem. 1)
	A	B	$BUS_{DB} = 0$	$BUS_{ABLI} = 0$
CPX XH/XL → M, M + 1	1	1	M M + 1	XH XL
DEX X - 1 → X	1	1	XH après décrémentation	XL avant décrément XL après décrément
DES SP - 1 → SP	1	1	-	SPL avant décrém. SPL après décrém.
INX X + 1 → X	1	1	XH après incrémentation	XL avant incrément XL après incrément
INS SP + 1 → SP	1	1	-	SPL avant incrém. SPL après incrém.
LDX M → XH M + 1 → XL	1	1	M M + 1	FF
LDS M → SPH M + 1 → SPL	1	1	M M + 1	FF
STX XH → M XL → M + 1	1	1	XH XL	FF
TXS SPH → M SPL → M+1	1	1	SPH SPL	FF
TXS X - 1 → SP	1	1	-	XL avant décrément XL après décrément
TSX SP + 1 → X	1	1	-	SPL avant incrém. SPL après incrém.
Branchements Rem. 2				
JUMP Rem. 2	1	1	PCL de PC+1 PCH de PC+2	SPL de SP SPL de SP - 1 SPL de SP - 2
RTS Rem. 2	1	1	PCL de PC+2 si code PCL de PC+3 si code PCH de PC+2 si code PCH de PC+3 si code	SP de SP SPL de SP - 1 SPL de SP - 2

Table II - Conditions de test pour la classe I séquences des registres opératifs des instructions non opératives

instructions non opératives et opération	registres opératifs type de séq		transferts sur bus DB	transferts sur bus ABLI (rem. 1)
	A	B	$BUS_{DB} = 0$	$BUS_{ABLI} = 0$
RTI	4	1	CC B A XH XL PCH PCL	SPL de SP SPL de SP+1 SPL de SP+2 SPL de SP+3 SPL de SP+4 SPL de SP+5 SPL de SP+6 SPL de SP+7
RTS	1	1	PCH de retour PCL de retour	SPL de SP SPL de SP+1 SPL de SP+2
β:SWI	2	2	PCL de β+1 PCH de β+1 XL XH A B CC (FFFA) (FFFB)	SPL de SP SPL de SP-1 SPL de SP-2 SPL de SP-3 SPL de SP-4 SPL de SP-5 SPL de SP-6 SPL de SP-7 FA FB
β:WAI	2	2	PCL de β+1 PCH de β+1 XL XH A B CC PCL, X = C si NMI X = B si IRQ (FFFA) (FFFB)	SPL de SP SPL de SP-1 SPL de SP-2 SPL de SP-3 SPL de SP-4 SPL de SP-5 SPL de SP-6 SPL de SP-7 FA FB β = C si NMI β = B si IRQ
CLC	1	1	-	-
CLI	1	1	-	-
CLV	1	1	-	-
SEC	1	1	-	-
SEI	1	1	-	-
SEV	1	1	-	-
TAP A → CC	2	1	A	-
TPA CC → A	3	1	CC	-

Table II - (suite)

Table III - Conditions du test pour la classe II:types de séquence des registres non opératifs et cycles opérateur de calcul et incrémenteur

	registres non opératifs type de seq.		cycles opérateur												opérateur	incrémenteur		
instruct non opératives et opération	X	SP	origine DB normale	conditions pour test origine DB	origine ABLI normale	conditions pour test origine ABLI	commande DB normale	conditions pour test commande DB	carry-in normal	conditions pour test carry-in	code opération opérateur	conditions pour test code opération	conditions pour test transf. instr	origines incrément. ABH / ABL	code opération incrément v. III-2	conditions pour test origines	conditions pour test code opération
(X XL/XH M/M+1		1	M M + 1	(1) = 1* XH = FF (1) = 1* XL = FF	XL XH	M FF 2 M+1 = FF	$\overline{DB}$ $\overline{DB}$	INDIC(R) ≠ INDIC[XH+(M),1,XL-(M+1)] ≠INDIC[XH-(M),XL+(M+1)-1] ≠ INDIC[XH+(M)+1,XL+(M+1)+1] ≠INDIC[XH+SR(M)+1,XL-(M+1)] ≠INDIC[XH-M,XL+SR(M+1)+1] ≠INDIC[XH+SR(M)+1,XL+SR(M+1)+1] ≠INDIC[XH+DAA(A)+1, XL-(M+1)] ≠INDIC[XH-(M),XL+DAA(A)+1] ≠INDIC[XH+DAA(A)+1,XL+DAA(A)+1]	1	v INDIC(R)≠ INDIC[XH-(M),-1,XL-(M+1)] ≠INDIC[XH-(M),XL-(M+1)-1] ≠INDIC[XH-(M)+1,XL-(M+1)-1]	ADD ADD	INDIC(R)≠ 1% result. ...,XL-(M+1)] +[XH-(M), 1% result. ...] +[1% result ...,15 résultats ...]	INDIC(R)≠ INDIC[XH+(M),XL-(M+1)] ≠INDIC[XH-(M),XL+(M+1)] ≠INDIC[XH+(M),XL+(M+1)]	-	-	-	-
DFX X - 1 - X	5	1	XH après décrément.	5 XL après décrément. = FF	XL après décrément.	5 XH après décrément. = FF	DB	INDIC(R) ≠ INDIC[XL+SR(XH)]≠IN. [XL+DAA(A)]	0	1 INDIC(R)≠1 [XH après+ XL après+1]	ADD	INDIC(R)≠ INDIC[1% résultats ...]	INDIC(R) ≠ INDIC(XL-1)	X avant décrément.	DEC	5	≠ résult données par les 7 codes fam.

Table III - (suite

instruct. non opératives et opération	registres non opératifs type de séq. X	registres non opératifs type de séq. SP	cycles opérateur de calcul : origine DB normale	conditions pour test origine DB	origine ABLI normale	conditions pour test origine ABLI	commande DB normale	conditions pour test commande DB	carry-in normal	conditions pour test carry-in	code opération opérateur	conditions pour test code opération	conditions pour test transf. instr.	opérateur incrémenteur : origines incrément. ABH / ABL	code opération incrément. v. III-2	conditions pour test origines	conditions pour test code opération
DPS SP - 1 → SP	1	5	-	-	-	-	-	-	-	-	-	-	-	SP avant décrément.	DEC	5	R ≠ 7 résultats...
INX X + 1 → X	5	1	XH après incrément.	5 XL après incr. = FF	XL après incrément.	5 XH après incr. = FF	DB	INDIC(R) ≠INDIC⌈XL+SR(XH)⌉ ≠INDIC⌈XL+DAA(A)⌉	0	1 INDIC(R)≠ INDIC(XH après+XL après+1)	ADD	INDIC(R)≠ INDIC(15 résultats ...)	INDIC(R)≠ INDIC(XL--XH-1)	X avant incrément.	INCR	5	R ≠ 7 résultats ...
INS SP + 1 → SP	1	5	-	-	-	-	-	-	-	-	-	-	-	SP avant incrément.	INCR	5	R ≠ 7 résultats
LDX M → XH M+1 → XL	3	1	M M + 1	(1) - (1' (1) - (1'	FF FF	$K_{ABLI} \neq$ FF M = FF $K_{ABLI} \neq$ FF M+1 = FF	DB DB	M≠SR(M) M≠DAA(A) M+1≠SR(M+1) M+1≠DAA(A)	1	0	AND	M≠15 rés. .... M+1≠ 15 résult....	M≠ FF+M ≠ FF-M ≠ FF+M+1 ≠ FF-M-1 M+1≠FF+(M+1) ≠FF-(M+1) ≠FF+(M+1)+1 ≠FF-(M+1)-1	-	-	-	-
LDS M → SPH M+1 → SPL	1	5	M M + 1	(1) - (1') (1) - (1')	FF FF	$K_{ABLI} \neq$ FF M = FF $K_{ABLI} \neq$ FF M+1 = FF	DB DB	M≠SR(M) M≠ DAA(A) M+1≠SR(M+1) M+1≠DAA(A)	1	0	AND	M≠ 15 rés. ... M+1 ≠ 15 résult ...	M≠ FF+M ≠ FF-M ≠ FF+M+1 ≠ FF-M-1 M+1≠FF+(M+1) ≠FF-(M+1) ≠FF+(M+1)+1 ≠FF-(M+1)-1		-	-	-

Table III - (suite)

Instruction non opératives et opération	registres non opératifs type de séq. X	registres non opératifs type de séq. SP	cycles opérateur de calcul : origine DD normale	conditions pour test origine DD	origine ABLI normale	conditions pour test origine ABLI	commande DD normale	conditions pour test commande DD	carry-in normal	conditions pour test carry-in	code opération opérateur	conditions pour test code opération	conditions pour test transf. instr.	opérateur incrementeur : origines incrément. ABH / ABL	code operation increment. v. III-2	conditions pour test origines	conditions pour test code opération
(PC) XH → M; XL → M+1	3	1	XH XL	2 /	FF FF	$F_{ABLI} \neq FF$ XH = FF $F_{ABLI} \neq FF$ XL = FF	DD	XH≠SR(XH) XH≠DAA(A) XL≠SR(XL) XL≠DAA(A)	1	0	ADD	XH ≠ 15 résult. XL ≠ 15 result...	XH≠FF-XH ≠FF+XH+1 ≠FF-XH-1 XL≠FF-XL ≠FF+XL+1 ≠FF-XL-1	-	-	-	-
SPH → M SPL → M+1	1	1	SPH SPL	/ 1	FF FF	$F_{ABLI} \neq FF$ SPH = FF $F_{ABLI} \neq FF$ SPL = FF	DE	SPH≠SR(SPH) SPH≠DAA(A) SPL≠SR(SPL) SPL≠DAA(A)	1	0	AND	SPH ≠ 15 result... SPL ≠ 15 résult...	SPH≠FF-SPH ≠FF+SPH+1 ≠FF-SPH-1 SPL≠FF-SPL ≠FF+SPL+1 ≠FF-SPL-1	-	-	-	-
XH-1 → XH XH-1 → SP	2	3	-	-	-	-	-	-	-	-	-	-	-	XH-XL av. décrement.	DLC	≠	N ≠ ? result...
(PC) + X → PC	3	2	-	-	-	-	-	-	-	-	-	-	-	SPH-SPL av. incrément.	(PC)	≠	N ≠ ? result...
Branch. Rem. ?																	

Table III.8 - Conditions de test pour la classe II types de séquence des registres non opératifs

instructions non opératives et opération	registres non opératifs type de séq.		transferts sur bus DB	transferts sur bus ABH	transferts sur bus ABL/ABLI	opérateur		incrémenteur	
	X	SP	$BUS_{DB} = 0$	$BUS_{ABH} = 0$	$BUS_{ABL/ABLI} = 0$	origines incrément. ABH / ABL	code opération III-2	conditions pour test origines	condi pour : code c
branchements Rem. 1									
↓:BSR	1	7	PCL de ⓖ + 2 PCH de ⓖ + 2	SPH de SP SPH de SP-1 SPH de SP-2	SPL de SP SPL de SP-1 SPL de SP-2	SPH - SPL Rem. 2	DEC	7	R fina de ce q serait y avait erreur -dant u cycle
JMP Rem. 2									
↓:ISR	1	7	PCL de ⓖ+2 si ind. PCL de ⓖ+3 si éten PCH de ⓖ+2 si ind. PCH de ⓖ+3 si éten.	SPH de SP SPH de SP-1 SPH de SP-2	SPL de SP SPL de SP-1 SPL de SP-2	SPH - SPL Rem. 2	DEC	7	R ≠ de ....
RTI	4	7	CC B A XH XL PCH PCL	SPH de SP SPH de SP+1 SPH de SP+2 SPH de SP+3 SPH de SP+4 SPH de SP+5 SPH de SP+6 SPH de SP+7	SPL de SP SPL de SP+1 SPL de SP+2 SPL de SP+3 SPL de SP+4 SPL de SP+5 SPL de SP+6 SPL de SP+7	SPH - SPL Rem. 2	INCR	7	R ≠ de ....
RTS	1	7	PCH de retour PCL de retour	SPH de SP SPH de SP+1 SPH de SP+2	SPL de SP SPL de SP+1 SPL de SP+2	SPH - SPL Rem. 2	INCR	7	R ≠ de ....
↓:SWI	2	7	PCL de ⓖ + 1 PCH de ⓖ + 1 XL XH A B CC (FFFA) FFFB)	SPH de SP SPH de SP-1 SPH de SP-2 SPH de SP-3 SPH de SP-4 SPH de SP-5 SPH de SP-6 SPH de SP-7	SPL de SP SPL de SP-1 SPL de SP-2 SPL de SP-3 SPL de SP-4 SPL de SP-5 SPL de SP-6 SPL de SP-7 FA FB	SPH - SPL Rem. 2	DEC	7	R ≠ de ....
↓:WAI	2	7	PCL de ⓖ + 1 PCH de ⓖ + 1 XL XH A B CC FA,A = C si NMI	SPH de SP SPH de SP-1 SPH de SP-2 SPH de SP-3 SPH de SP-4 SPH de SP-5 SPH de SP-6 SPH de SP-7	SPL de SP SPL de SP-1 SPL de SP-2 SPL de SP-3 SPL de SP-4 SPL de SP-5 SPL de SP-6 SPL de SP-7	SPH - SPL Rem. 2	DEC	7	R ≠ de ....

suite p. suivante

Table III bis - (suite)

instructions non opératives et opération	registres non opératifs type de seq.		transferts sur bus DB	transferts sur bus ABH	transferts sur bus ABL/ABLI	opérateur		incrémenteur	
	X	SP	$BUS_{DB} = 0$	$BUS_{ABH} = 0$	$BUS_{ABL/ABLI} = 0$	origines incrément. ABH / ABL	code opération III-2	conditions pour test origines	conditions pour test code opér.
WAI suite			1 = 0 si IRQ (FFF4) (FFF5)		F9 F8,3 = 0 si NMI 5 = 0 si IRQ				
CLC	1	1	-	-	-	-	-	-	-
CLI	1	1	-	-	-	-	-	-	-
CLV	1	1	-	-	-	-	-	-	-
SEC	1	1	-	-	-	-	-	-	-
SEI	1	1	-	-	-	-	-	-	-
SEV	1	1	-	-	-	-	-	-	-
TAP	-	1	A	-	-	-	-	-	-
TPA	1	-	CC	-	-	-	-	-	-

Table IV - Conditions de test pour la classe III:types de séquence des registres non opératifs

instructions opératives	registres non opératifs		transferts sur bus DB	transferts sur bus ABL/ABLI
	X	SP	BUS_{DB} =0	$BUS_{ABL/ABLI}$=0
ADDA	1	1	M R	A
ADDB	1	1	M R	B
ABA	1	1	B R	A
ADCA	1	1	M R	A
ADCB	1	1	M R	B
ANDA	1	1	M R	A
ANDB	1	1	M R	B
BITA	1	1	M	A
BITB	1	1	M	B
CLR	1	1	M 00	00
CLRA	1	1	M 00	00
CLRB	1	1	M 00	00
CMPA	1	1	M	A
CMPB	1	1	M	B
CBA	1	1	B	A
COM	1	1	M $\overline{M}$	-
COMA	1	1	A $\overline{A}$	-
COMB	1	1	B $\overline{B}$	
NEG	1	1	M R	0

Table IV - (suite)

instructions opératives	registres non opératifs		transferts sur bus DB	transferts sur bus ABL/ABLI
	X	SP	BUS_{DB} =0	$BUS_{ABL/ABLI}$=0
[illegible]NHA	1	1	A R	CC
[illegible]NHB	1	1	B R	CO
DAA	1	1	-	A
DEC	1	1	M R	-
DECA	1	1	A R	-
DECB	1	1	B R	-
[illegible]RA	1	1	M R	A
[illegible]RB	1	1	M R	B
INC	1	1	M R	CO
[illegible]NCA	1	1	A R	CO
[illegible]NCB	1	1	B R	CO
[illegible]DA LDAB	1	1	M	-
[illegible]RAA	1	1	M R	A
[illegible]RAB	1	1	M R	B
PSHA	1	S	A	-
PSHB	1	S	B	-
PULA PULB	1	S	M	-
ROL	1	1	M R	-
ROLA	1	1	A R	-

Table IV - (suite)

instructions opératives	registres non opératifs		transferts sur bus DB	transferts sur bus ABL/ABLI
	X	SP	BUS_{DB} =0	$BUS_{ABL/ABLI}$ =0
ROLB	1	1	B R	-
ROR	1	1	M R	00
RORA	1	1	A M	00
RORB	1	1	B M	00
ASL	1	1	M R	-
ASLA	1	1	A R	-
ASLB	1	1	B R	-
ASR	1	1	M R	00
ASRA	1	1	A R	00
ASRB	1	1	B R	00
LSR	1	1	M R	00
LSRA	1	1	A R	00
LSRB	1	1	B R	00
STAA	1	1	A	-
STAB	1	1	B	-
STBA	1	1	M P	A
SUBB	1	1	M R	B
SBA	1	1	B R	A

Table IV.- (suite)

instructions opératives	registres non opératifs		transferts sur bus DB	transferts sur bus ABL/ABLI
	X	SP	BUS^{*}_{DB} =0	$BUS^{*}_{ABL/ABLI}$=0
SBCA	1	1	M R	A
SBCB	1	1	M R	B
TAB	1	1	A	-
TBA	1	1	B	-
TMT	1	1	M	00
TSTA	1	1	A	CC
TSTB	1	1	B	DD

ANNEXE B

A partir de l'hypothèse de l'existence de pannes seulement dans un type (de l'ensemble de types) ou dans un mode on peut considérer que pour le test d'une instruction de type I et de mode M, en utilisant des instructions de type différent de I (I') et de mode différent de M (M') pour la commande/observation, il y a plusieurs résultats possibles (on nomme I' tout type différent de I):

1.- Tout est correct (il n'y a pas de panne)
2.- Seulement le mode M est en panne
3.- Seulement le type I est en panne
4.- Le type I et le mode M sont corrects, mais M' est en panne
5.- Le type I et le mode M' sont en panne (I' et M sont corrects par hypothèse)
6.- Le type I' et le mode M sont en panne
7.- Le type Iet le mode M sont en panne
8.- Le type I et le mode M sont corrects, mais I' est en panne
9.- le type I' et le mode M' sont en panne

Les conséquences de ces résultats possibles sont:

1.- Le programme se déroule normalement
2.- La panne est détectée
3.- Idem à 2
4.- Il y a deux alternatives dépendantes du fait que l'exécution des instructions de mode M' génèrent une erreur observable ou non pendant l'exécution du module; si c'est observable (effet catastrophique) elle est détectée, sinon elle le sera pendant le test de M'
5.- Il y a aussi, deux alternatives dépendantes du fait que la panne de M' cache (complémente) ou non la panne de I pendant l'exécution du module; si c'est non, alors la panne est observable; autrement la panne de M' sera observable seulement si le type I n'est pas utilisé pour le test du mode M' ou si étant utilisé, il ne crée pas une erreur complémentée

6.- Idem à 5

7.- Il y a deux alternatives: les pannes se complémentent ou non; si c'est non, alors la panne est détectée, autrement la panne sera detectée dans le test du mode M'

8.- Idem à 4

9.- Idem à 7, mais la panne sera forcément observable dans le test de M' et/ou I'

Finalement on observe que la compléxité est accrue lorsqu'on élimine l'hypothèse qui restreint la simultanéité des pannes pour le type et le mode d'une instruction.

ANNEXE C

```
00620 00993                 *
00630 00994                 *******************************************************
00640 00995                 *                                                     *
00650 00996                 *   TEST CLASSE IV: MAUVAISE GESTION DU CC            *
00660 00997                 *                                                     *
00670 00998                 *******************************************************
00680 00999                 *
00690 01000                 * TEST EN DEUX PHASES:
00100 01001                 *                     - PHASE ADRESSAGE
00110 01002                 *                     - PHASE EXECUTION
00120 01003                 *
00130 01004A AA00                   ORG    $AA00
00140 01005                 *
00150 01006                 *********************************
00160 01007                 *
00170 01008                 *     PHASE  D'ADRESSAGE
00180 01009                 *
00190 01010                 *   TEST DE MODES D'ADRESSAGE:
00200 01011                 *                               - IMMEDIAT
00210 01012                 *                               - DIRECT
00220 01013                 *                               - INDEXE
00230 01014                 *                               - ETENDU
00240 01015                 *                               - RELATIF
```

```
00260 01017                    ****************************
00270 01018                    *
00280 01019                    *  ADRESSAGE IMMEDIAT
00290 01020                    *
00300 01021                    *
00310 01022                    * SEULEMENT H, I, ET C SONT TESTEABLES
00320 01023                    * LES AUTRES SONT MODIFIES PAR LA PHASE EXECUTION
00330 01024                    *
00340 01025                    * - IL N'Y A PAS DE TRANSFERT SUR DB
00350 01026                    * - IL N'Y A PAS DE COMPTE-RENDU (C.R.) DE L'OPERATEUR
00360 01027                    *
00370 01028                    * ** ADRESSE INSTRUCTION-TEST QUELCONQUE
00380 01029                    * ** OPERAND QUELCONQUE
00390 01030                    *
00400 01031                    * PAIM: PHASE D'ADRESSAGE IMMEDIAT
00410 01032                    * EAIM03: PANNE DE H, I ET/OU C AVEC
00420 01033                    *           ADR. IMMEDIAT
00430 01034                    * TEST INDICATEURS  H, I, ET C.
00440 01035                    *
00450 01036          AA00  A PAIM0  EQU    *
00460 01037                    *
00470 01038                    * INDICATEURS INITIALISES A ZERO
00480 01039                    *
00490 01040A AA00 4F                 CLRA
00500 01041A AA01 06                 TAP
00510 01042                    *
00520 01043                    *
00530 01044A AA02 86 FF    A         LDAA   #$FF     ; INSTRUCTION-TEST ****
00540 01045                    *
00550 01046                    *
00560 01047A AA04 24 03 AA09         BCC    BOUC01
00570 01048A AA06 7E FFA2  A         JMP    EAIM03   ; C EST A 1
00580 01049          AA09  A BOUC01 EQU    *
00590 01050A AA09 07                 TPA
00600 01051A AA0A 48                 ASLA
00610 01052A AA0B 48                 ASLA
00620 01053A AA0C 48                 ASLA
00630 01054A AA0D 24 03 AA12         BCC    BOUC02
00640 01055A AA0F 7E FFA2  A         JMP    EAIM03   ; H EST A 1
00650 01056          AA12  A BOUC02 EQU    *
00660 01057A AA12 48                 ASLA
00670 01058A AA13 24 03 AA18         BCC    PAIM1
00680 01059A AA15 7E FFA2  A         JMP    EAIM03   ; I EST A 1
```

```
00690 01060                    *
00700 01061          AA18  A PAIM1  EQU    *
00710 01062                    *
00720 01063                    * INDICATEURS INITIALISES A 1
00730 01064                    *
00740 01065A AA18 96 23     A         LDAA   CONST1
00750 01066A AA1A 06                  TAP
00760 01067                    *
00770 01068                    *
00780 01069A AA1B 86 00     A         LDAA   #$00     ; INSTRUCTION-TEST *****
00790 01070                    *
00800 01071                    *
00810 01072A AA1D 25 03 AA22          BCS    BOUC03
00820 01073A AA1F 7E FFA2   A         JMP    EAIM03   ; C EST A ZERO
00830 01074          AA22   A BOUC03 EQU    *
00840 01075A AA22 07                  TPA
00850 01076A AA23 48                  ASLA
00860 01077A AA24 48                  ASLA
00870 01078A AA25 48                  ASLA
00880 01079A AA26 25 03 AA2B          BCS    BOUC04
00890 01080A AA28 7E FFA2   A         JMP    EAIM03   ; H EST A ZERO
00900 01081          AA2B   A BOUC04 EQU    *
00910 01082A AA2B 48                  ASLA
00920 01083A AA2C 25 03 AA31          BCS    PADI0
00930 01084A AA2E 7E FFA2   A         JMP    EAIM03   ; I EST A ZERO
00940 01085                    *
```

```
00960 01087                    **********************
00970 01088                    *
00980 01089                    * ADRESSAGE DIRECT
00990 01090                    *
01000 01091                    * PADI: PHASE D'ADRESSAGE DIRECT
01010 01092                    * EADI03: PANNE EN H, I OU C AVEC AD. DIRECT
01020 01093                    *
01030 01094                    * SEULEMENT LES INDICATEURS H, I, ET C SONT TESTABLES;
01040 01095                    * IL N'Y A PAS DE COMPTE-RENDU DE L'OPERATEUR
01050 01096                    *
01060 01097                    * ADRESSE INSTRUCTION-TEST QUELCONQUE
01070 01098                    * OPERANDE: $FF
01080 01099                    *
01090 01100          AA31  A PADI0  EQU    *
01100 01101                    * INDICATEURS INITIALISES A ZERO
01110 01102A AA31 4F                 CLRA
01120 01103A AA32 06                 TAP
01130 01104                    *
01140 01105                    *
01150 01106A AA33 96 FF    A         LDAA   $FF      ; INSTRUCTION-TEST ****
01160 01107                    *
01170 01108                    *
01180 01109A AA35 24 03 AA3A         BCC    BOUC05
01190 01110A AA37 7E FFA2  A         JMP    EADI03   ; C EST A 1
01200 01111          AA3A  A BOUC05 EQU    *
01210 01112A AA3A 07                 TPA
01220 01113A AA3B 48                 ASLA
01230 01114A AA3C 48                 ASLA
01240 01115A AA3D 48                 ASLA
01250 01116A AA3E 24 03 AA43         BCC    BOUC06
01260 01117A AA40 7E FFA2  A         JMP    EADI03   ; H EST A 1
01270 01118          AA43  A BOUC06 EQU    *
01280 01119A AA43 48                 ASLA
01290 01120A AA44 24 03 AA49         BCC    PADI1
01300 01121A AA46 7E FFA2  A         JMP    EADI03   ; I EST A 1
01310 01122                    *
01320 01123          AA49  A PADI1  EQU    *
01330 01124                    * INDICATEURS INITIALISES A UN
01340 01125                    *
01350 01126                    * OPERANDE: $C0
01360 01127                    *
01370 01128A AA49 4F                 CLRA
01380 01129A AA4A 43                 COMA
```

```
01390 01130A AA4B 06                TAP
01400 01131                   *
01410 01132                   *
01420 01133A AA4C 96 C0    A        LDAA   $C0      ; INSTRUCTION-TEST ****
01430 01134                   *
01440 01135                   *
01450 01136A AA4E 25 03 AA53        BCS    BOUC07
01460 01137A AA50 7E FFA2  A        JMP    EADI03   ; C EST A 0
01470 01138          AA53  A BOUC07 EQU    *
01480 01139A AA53 07                TPA
01490 01140A AA54 48                ASLA
01500 01141A AA55 48                ASLA
01510 01142A AA56 48                ASLA
01520 01143A AA57 25 03 AA5C        BCS    BOUC08
01530 01144A AA59 7E FFA2  A        JMP    EADI03   ; H EST A 0
01540 01145          AA5C  A BOUC08 EQU    *
01550 01146A AA5C 48                ASLA
01560 01147A AA5D 25 03 AA62        BCS    BRIN01
01570 01148A AA5F 7E FFA2  A        JMP    EADI03   ; I EST A 0
01580 01149                   *
01590 01150          AA62  A BRIN01 EQU    *
01600 01151A AA62 7E B1F7  A        JMP    PAIN01
01610 01152                   *
```

```
01630 01154                     **************************
01640 01155                     *
01650 01156                     * ADRESSAGE INDEXE
01660 01157                     *
01670 01158                     * LES SIGNAUX DE CONTROLE POUR CERTAINS INDICATEURS
01680 01159                     * (H, N) PEUVENT ETRE TOTALEMENT TESTES EN UNE SEULE
01690 01160                     * ETAPE: INDICATEUR INITIALISE A ZERO. LES AUTRES
01700 01161                     * INDICATEURS (I, Z, V, C) ONT BESOIN DE DEUX ETAPES
01710 01162                     * UNE INITIALISE A ZERO ET L'AUTRE INITIALISE A UN.
01720 01163                     *
01730 01164                     * LA PREMIERE ETAPE A TROIS PARTIES (CC=0)
01740 01165                     *
01750 01166                     * PREMIERE ETAPE, PREMIERE PARTIE
01760 01167                     *
01770 01168                     * ADRESSE INSTRUCTION-TEST: $B1FC
01780 01169                     *       (ADRESSE DE SAUT: $..FE)
01790 01170                     * DEP: $7F
01800 01171                     * XL:  $7F
01810 01172                     * XL + DEP: $FE
01820 01173                     *
01830 01174A B1F7                       ORG     $B1F7
01840 01175                     *
01850 01176           B1F7  A PAIN01 EQU     *
01860 01177A B1F7 CE B17F  A           LDX     #$B17F
01870 01178A B1FA 4F                   CLRA
01880 01179A B1FB 06                   TAP
01890 01180                     *
01900 01181                     *
01910 01182A B1FC 6E 7F    A           JMP     $7F,X     ; INSTRUCTION-TEST ****
01920 01183                     *
01930 01184                     *
01940 01185A B1FE 07                   TPA               ; ADRESSE: $B1FE = $7F+ (X)
01950 01186A B1FF 81 C0    A           CMPA    #$C0
01960 01187A B201 27 03 B206           BEQ     BRIN02
01970 01188A B203 7E FFA2  A           JMP     EAIN0     ; INDICATEURS SONT A UN
01980 01189                     *
01990 01190           B206  A BRIN02 EQU     *
02000 01191A B206 7E B2F9  A           JMP     PAIN02
02010 01192                     *
02020 01193                     * PREMIERE ETAPE, DEUXIEME PARTIE
02030 01194                     *
02040 01195                     * ADRESSE INSTRUCTION-TEST: $B2FE
02050 01196                     *     ( ADRESSE DE SAUT: $..00)
```

```
02060 01197                    * DEP: $00
02070 01198                    * XL:  $00
02080 01199                    * XL + DEP: 00
02090 01200                    *
02100 01201A B2F9                      ORG    $B2F9
02110 01202                    *
02120 01203           B2F9  A PAIN02 EQU    *
02130 01204A B2F9 CE B300  A           LDX    #$B300
02140 01205A B2FC 4F                   CLRA
02150 01206A B2FD 06                   TAP
02160 01207                    *
02170 01208                    *
02180 01209A B2FE 6E 00    A           JMP    0,X        ; INSTRUCTION-TEST *****
02190 01210                    *
02200 01211                    *
02210 01212A B300 07                   TPA               ; ADRESSE: $B300 = 0 +(X)
02220 01213A B301 81 C0    A           CMPA   #$C0
02230 01214A B303 27 03 B308           BEQ    BRIN03
02240 01215A B305 7E FFA2  A           JMP    EAIN0      ; INDICATEURS SONT A UN
02250 01216                    *
02260 01217           B308  A BRIN03 EQU    *
02270 01218A B308 7E B22A  A           JMP    PAIN03
02280 01219                    *
02290 01220                    * PREMIERE ETAPE, TROISIEME PARTIE
02300 01221                    *
02310 01222                    * ADRESSE INSTRUCTION-TEST: $B22F
02320 01223                    *    (ADRESSE SAUT: $..31)
02330 01224                    * DEP: $81
02340 01225                    * XL:  $B0
02350 01226                    * XL + DEP: $31
02360 01227                    *
02370 01228A B22A                      ORG    $B22A
02380 01229                    *
02390 01230           B22A  A PAIN03 EQU    *
02400 01231A B22A CE B1B0  A           LDX    #$B1B0
02410 01232A B22D 4F                   CLRA
02420 01233A B22E 06                   TAP
02430 01234                    *
02440 01235                    *
02450 01236A B22F 6E 81    A           JMP    $81,X
02460 01237                    *
02470 01238                    *
02480 01239A B231 07                   TPA
```

```
02490 01240A B232 81 C0    A        CMPA   #$C0
02500 01241A B234 27 03 B239        BEQ    PAIN1
02510 01242A B236 7E FFA2  A        JMP    EAIN0    ; INDICATEURS SONT A UN
02520 01243                *
02530 01244                *
02540 01245                * DEUXIEME ETAPE (CC=1)
02550 01246                *
02560 01247                * ADRESSE INSTRUCTION-TEST: $B23F
02570 01248                * DEP: $00
02580 01249                * XL: $41
02590 01250                * XL + DEP: $41  ; IL Y A AU MOINS UN 1
02600 01251                *
02610 01252          B239  A PAIN1  EQU    *
02620 01253A B239 CE B241  A        LDX    #$B241
02630 01254A B23C 96 23    A        LDAA   CONST1
02640 01255A B23E 06                TAP
02650 01256                *
02660 01257                *
02670 01258A B23F 6E 00    A        JMP    0,X      ; INSTRUCTION-TEST ******
02680 01259                *
02690 01260                *
02700 01261A B241 07                TPA
02710 01262A B242 91 23    A        CMPA   CONST1
02720 01263A B244 27 03 B249        BEQ    BRIN04
02730 01264A B246 7E FFA2  A        JMP    EAIN1    ; INDICATEURS SONT A ZERO
02740 01265                *
02750 01266          B249  A BRIN04 EQU    *
02760 01267A B249 7E A637  A        JMP    PAET1
02770 01268                *
```

```
02790 01270                     **********************
02800 01271                     *
02810 01272                     * ADRESSAGE ETENDU
02820 01273                     *
02830 01274                     * IL N'Y A PAS DE COMPTE-RENDUES
02840 01275                     *
02850 01276                     * PREMIERE PARTIE
02860 01277                     *
02870 01278                     * ADRESSE INSTRUCTION-TEST: $A63C
02880 01279                     *    (ADRESSE SAUT: $..3F)
02890 01280                     * POIDS FAIBLE ADRESSE: $3F
02900 01281                     *
02910 01282A A637                       ORG     $A637
02920 01283                     *
02930 01284                     * INDICATEURS INITIALISES A ZERO
02940 01285                     *
02950 01286           A637  A PAET1   EQU     *
02960 01287A A637 CE FFA2  A            LDX     #EAET0
02970 01288A A63A 4F                    CLRA
02980 01289A A63B 06                    TAP
02990 01290                     *
03000 01291                     *
03010 01292A A63C 7E A63F  A            JMP     *+3       ; INSTRUCTION-TEST ****
03020 01293                     *
03030 01294                     *
03040 01295A A63F 07                    TPA
03050 01296A A640 81 C0    A            CMPA    #$C0
03060 01297A A642 27 02 A646            BEQ     BRIN05
03070 01298A A644 6E 00    A            JMP     0,X       ; INDICATEURS SONT A UN
03080 01299                     *
03090 01300           A646  A BRIN05  EQU     *
03100 01301A A646 CE FF50  A            LDX     #PAET2
03110 01302A A649 6E 00    A            JMP     0,X
03120 01303                     *
03130 01304                     * DEUXIEME PARTIE
03140 01305                     *
03150 01306                     * ADRESSE INSTRUCTION-TEST: $FF55
03160 01307                     *  (ADRESSE SAUT: $FF..)
03170 01308                     * POIDS FORTS ADRESSE: $FF (OU $3F)
03180 01309                     *
03190 01310A FF50                       ORG     $FF50
03200 01311                     *
03210 01312                     * INDICATEURS INITIALISES A ZERO
```

```
03220 01313                    *
03230 01314           FF50  A PAET2  EQU    *
03240 01315A FF50 CE FFA2  A         LDX    #EAET1
03250 01316A FF53 4F                 CLRA
03260 01317A FF54 06                 TAP
03270 01318                    *
03280 01319                    *
03290 01320A FF55 7E FF58  A         JMP    *+3       ; INSTRUCTION-TEST ****
03300 01321                    *
03310 01322                    *
03320 01323A FF58 07                 TPA
03330 01324A FF59 81 C0    A         CMPA   #$C0
03340 01325A FF5B 27 02 FF5F         BEQ    BRIN06
03350 01326A FF5D 6E 00    A         JMP    0,X       ; INDICATEURS SONT A 1
03360 01327                    *
03370 01328           FF5F  A BRIN06 EQU    *
03380 01329A FF5F CE B376  A         LDX    #PARE01
03390 01330A FF62 6E 00    A         JMP    0,X
03400 01331                    *
```

```
03420 01333                     ********************
03430 01334                     *
03440 01335                     * ADRESSAGE RELATIF
03450 01336                     *
03460 01337                     * LES OPERANDES ONT LES MEMES CARACTERISTIQUES
03470 01338                     * QUE POUR L'ADRESSAGE INDEXE.
03480 01339                     *
03490 01340                     * PREMIERE ETAPE: PREMIERE PARTIE
03500 01341                     *
03510 01342                     * INDICATEURS INITIALISES A ZERO
03520 01343                     *
03530 01344                     * ADRESSE INSTRUCTION-TEST: $B37D
03540 01345                     *    (ADRESSE DEVIATION: $..FE)
03550 01346                     * DEP: $7F
03560 01347                     * PCL: $7F
03570 01348                     * PCL + DEP: $FE
03580 01349                     *
03590 01350A B376                       ORG     $B376
03600 01351                     *
03610 01352           B376  A PARE01 EQU     *
03620 01353A B376 C6 B7     A         LDAB    #$B7
03630 01354A B378 D7 28     A         STAB    RSEX
03640 01355A B37A 4F                  CLRA
03650 01356A B37B 06                  TAP
03660 01357                     *
03670 01358                     *
03680 01359A B37C 20 7F B3FD          BRA     BOUC12   ; INSTRUCTION-TEST ****
03690 01360                     *
03700 01361                     *
03710 01362A B3FD                     ORG     *+$7F
03720 01363                     *
03730 01364           B3FD  A BOUC12 EQU     *
03740 01365A B3FD 07                  TPA
03750 01366A B3FE 81 C0     A         CMPA    #$C0
03760 01367A B400 0A                  CLV
03770 01368A B401 0C                  CLC
03780 01369A B402 07                  TPA
03790 01370A B403 48                  ASLA
03800 01371A B404 48                  ASLA
03810 01372A B405 48                  ASLA
03820 01373A B406 48                  ASLA
03830 01374A B407 48                  ASLA
03840 01375A B408 97 29     A         STAA    RSEX+1
```

```
03850 01376A B40A DE 28    A        LDX    RSEX
03860 01377A B40C 6E 00    A        JMP    0,X
03870 01378                 *
03880 01379A B700                   ORG    $B700
03890 01380A B700 7E FFA2  A        JMP    EARE0
03900 01381                 *
03910 01382A B780                   ORG    $B780
03920 01383A B780 7E B4F7  A        JMP    PARE02
03930 01384                 *
03940 01385                 * PREMIERE ETAPE: DEUXIEME PARTIE
03950 01386                 *
03960 01387                 * ADRESSE INSTRUCTION-TEST: $B4FE
03970 01388                 *    (ADRESSE DEVIATION: $..00)
03980 01389                 * DEP: $00
03990 01390                 * PCL: $00
04000 01391                 * PCL + DEP: $00
04010 01392                 *
04020 01393A B4F7                   ORG    $B4F7
04030 01394                 *
04040 01395                 *
04050 01396          B4F7   A PARE02 EQU    *
04060 01397A B4F7 C6 B8    A        LDAB   #$B8
04070 01398A B4F9 D7 28    A        STAB   RSEX
04080 01399A B4FB 4F                CLRA
04090 01400A B4FC 06                TAP
04100 01401                 *
04110 01402                 *
04120 01403A B4FD 20 00 B4FF        BRA    BOUC09   ; INSTRUCTION-TEST ****
04130 01404                 *
04140 01405                 *
04150 01406          B4FF   A BOUC09 EQU    *
04160 01407A B4FF 07                TPA
04170 01408A B500 81 C0    A        CMPA   #$C0
04180 01409A B502 0A                CLV
04190 01410A B503 0C                CLC
04200 01411A B504 07                TPA
04210 01412A B505 48                ASLA
04220 01413A B506 48                ASLA
04230 01414A B507 48                ASLA
04240 01415A B508 48                ASLA
04250 01416A B509 48                ASLA
04260 01417A B50A 97 29    A        STAA   RSEX+1
04270 01418A B50C DE 28    A        LDX    RSEX
```

```
0428@ 01419A P50E 6E 00    A        JMP    0,X
0429@ 01420
0430@ 01421A B800                   ORG    $B800
0431@ 01422A B800 7E FFA2  A        JMP    EARE0
0432@ 01423                *
0433@ 01424A B880                   ORG    $B886
0434@ 01425A B880 7E B5B7  A        JMP    PARE03
0435@ 01426                *
0436@ 01427                * PREMIERE ETAPE: TROISIEME PARTIE
0437@ 01428                *
0438@ 01429                * ADRESSE INSTRUCTION-TEST: $B5BE
0439@ 01430                *   (ADRESSE DEVIATION: $..01)
0440@ 01431                * DEP: $41
0441@ 01432                * PCL: $C9
0442@ 01433                * PCL + DEP: $01
0443@ 01434                *
0444@ 01435A B5B7                   ORG    $B5B7
0445@ 01436                *
0446@ 01437          B5B7  A PARE03 EQU    *
0447@ 01438A B5B7 C6 B9    A        LDAB   #$B9
0448@ 01439A B5B9 D7 28    A        STAB   RSEX
0449@ 01440A B5BB 4F                CLRA
0450@ 01441A B5BC 06                TAP
0451@ 01442                *
0452@ 01443                *
0453@ 01444A B5BD 20 42 B601        BRA    BOUC1@   ; INSTRUCTION-TEST ****
0454@ 01445                *
0455@ 01446                *
0456@ 01447A B601                   ORG    $B601
0457@ 01448                *
0458@ 01449          B601  A BOUC1@ EQU    *
0459@ 01450A B601 07                TPA
0460@ 01451A B602 81 C@    A        CMPA   #$C@
0461@ 01452A B604 0A                CLV
0462@ 01453A B605 0C                CLC
0463@ 01454A B606 07                TPA
0464@ 01455A B607 48                ASLA
0465@ 01456A B608 48                ASLA
0466@ 01457A B609 48                ASLA
0467@ 01458A B60A 48                ASLA
0468@ 01459A B60B 48                ASLA
0469@ 01460A B60C 97 29    A        STAA   RSEX+1
0470@ 01461A B60E DE 28    A        LDX    RSEX
```

```
04710 01462A B610 6E 00    A        JMP    0,X
04720 01463                 *
04730 01464A B900                    ORG    $B900
04740 01465A B900 7E FFA2  A        JMP    EARE0
04750 01466                 *
04760 01467A B980                    ORG    $B980
04770 01468A B980 7E B665  A        JMP    PARE1
04780 01469                 *
04790 01470                 * DEUXIEME ETAPE
04800 01471                 *
04810 01472                 * INDICATEURS INITIALISES A UN
04820 01473                 *
04830 01474                 * ADRESSE INSTRUCTION-TEST: $B66D
04840 01475                 *    (ADRESSE DEVIATION: $..6F)
04850 01476                 * DEP: $00
04860 01477                 * PCL: $6F
04870 01478                 * PCL + DEP: $6F
04880 01479                 *
04890 01480A B665                    ORG    $B665
04900 01481                 *
04910 01482          B665  A PARE1  EQU    *
04920 01483A B665 C6 BA    A        LDAB   #$BA
04930 01484A B667 D7 28    A        STAB   RSEX
04940 01485A B669 96 23    A        LDAA   CONST1
04950 01486A B66B 06                TAP
04960 01487                 *
04970 01488                 *
04980 01489A B66C 20 00 B66E        BRA    BOUC11   ; INSTRUCTION-TEST ****
04990 01490                 *
05000 01491                 *
05010 01492          B66E  A BOUC11 EQU    *
05020 01493A B66E 07                TPA
05030 01494A B66F 91 23    A        CMPA   CONST1
05040 01495A B671 0A                CLV
05050 01496A B672 0C                CLC
05060 01497A B673 07                TPA
05070 01498A B674 48                ASLA
05080 01499A B675 48                ASLA
05090 01500A B676 48                ASLA
05100 01501A B677 48                ASLA
05110 01502A B678 48                ASLA
05120 01503A B679 97 29    A        STAA   RSEX+1
05130 01504A B67B DE 28    A        LDX    RSEX
```

```
05140 01505A B67D 6E 00    A        JMP     0,X
05150 01506                 *
05160 01507A BA00                   ORG     $BA00
05170 01508A BA00 7E FFA2  A        JMP     EARE1
05180 01509                 *
05190 01510A BA80                   ORG     $BA80
05200 01511A BA80 01                NOP
```

```
05220 01513                    ************************************
05230 01514                    *                                  *
05240 01515                    *  PHASE   EXECUTION               *
05250 01516                    *                                  *
05260 01517                    ************************************
05270 01518                    *
05280 01519                    * INSTRUCTION ADDB
05290 01520                    *             ====
05300 01521                    * PREMIERE ETAPE
05310 01522                    *
05320 01523A BA81 C6 00     A          LDAB   #0
05330 01524A BA83 96 23     A          LDAA   CONST1
05340 01525A BA85 06                   TAP
05350 01526                      *
05360 01527                      *
05370 01528A BA86 CB 2F     A          ADDB   #$2F      ; INSTRUCTION-TEST *****
05380 01529                      *
05390 01530                      *
05400 01531A BA88 07                   TPA
05410 01532A BA89 81 D0     A          CMPA   #$D0
05420 01533A BA8B 27 03 BA90           BEQ    BOUC31    ; INDICATEURS A ZERO
05430 01534                      *                       ; ( SAUF  I )
05440 01535A BA8D 7E FFA2   A          JMP    E4ADDB    ; MAUVAISE GESTION
05450 01536                      *
05460 01537           BA90  A BOUC31 EQU    *
05470 01538A BA90 C6 01     A          LDAB   #$01
05480 01539A BA92 96 23     A          LDAA   CONST1
05490 01540A BA94 06                   TAP
05500 01541                      *
05510 01542                      *
05520 01543A BA95 CB FF     A          ADDB   #$FF      ; INSTRUCTION-TEST ****
05530 01544                      *
05540 01545                      *
05550 01546A BA97 07                   TPA
05560 01547A BA98 81 F5     A          CMPA   #$F5
05570 01548A BA9A 27 03 BA9F           BEQ    BOUC32    ; V A ZERO
05580 01549A BA9C 7E FFA2   A          JMP    E4ADDB    ; MAUVAISE GESTION
05590 01550                      *
05600 01551                      * DEUXIEME ETAPE
05610 01552                      *
05620 01553           BA9F  A BOUC32 EQU    *
05630 01554A BA9F C6 7C     A          LDAB   #$7C
05640 01555A BAA1 4F                   CLRA
```

```
05650 01556A BAA2 06                TAP
05660 01557                 *
05670 01558                 *
05680 01559A BAA3 CB 54     A       ADDB    #$54    ; INSTRUCTION-TEST ****
05690 01560                 *
05700 01561                 *
05710 01562A BAA5 07                TPA
05720 01563A BAA6 81 EA     A       CMPA    #$EA
05730 01564A BAA8 27 03 BAAD        BEQ     BOUC33   ; N, V A UN
05740 01565A BAAA 7E FFA2   A       JMP     E4ADDB   ; MAUVAISE GESTION
05750 01566                 *
05760 01567         BAAD    A BOUC33 EQU    *
05770 01568A BAAD 4F                CLRA
05780 01569A BAAE 16                TAB
05790 01570A BAAF 06                TAP
05800 01571                 *
05810 01572                 *
05820 01573A BAB0 CB 00     A       ADDB    #00      ; INSTRUCTION-TEST ****
05830 01574                 *
05840 01575                 *
05850 01576A BAB2 27 03 BAB7        BEQ     BOUC34   ; Z A UN
05860 01577A BAB4 7E FFA2   A       JMP     E4ADDB   ; MAUVAISE GESTION
05870 01578                 *
05880 01579         BAB7    A BOUC34 EQU    *
05890 01580A BAB7 C6 80     A       LDAB    #$80
05900 01581A BAB9 4F                CLRA
05910 01582A BABA 06                TAP
05920 01583                 *
05930 01584                 *
05940 01585A BABB CB 80     A       ADDB    #$80     ; INSTRUCTION-TEST ****
05950 01586                 *
05960 01587                 *
05970 01588A BABD 25 03 BAC2        BCS     BOUC35   ; C A UN
05980 01589A BABF 7E FFA2   A       JMP     E4ADDB   ; MAUVAISE GESTION
05990 01590                 *
```

```
06810 01592                     **********************
06820 01593                     *
06830 01594                     * INSTRUCTION TSX
06840 01595                     *             ===
06850 01596                     *
06860 01597                     * IL N'Y A PAS DE COMPTE-RENDU
06870 01598                     * BUS PRECHARGE A UN
06880 01599                     *
06890 01600           BAC2  A BOUC35 EQU    *
06900 01601A BAC2 4F                  CLRA
06910 01602A BAC3 06                  TAP
06920 01603                     *
06930 01604                     *
06940 01605A BAC4 30                  TSX             ; INSTRUCTION-TEST ****
06950 01606                     *
06960 01607                     *
06970 01608A BAC5 07                  TPA
06980 01609A BAC6 81 C0    A          CMPA   #$C0
06990 01610A BAC8 27 03 BACD          BEQ    BOUC36   ; INDICATEURS NON-MODIFIES
07000 01611A BACA 7E FFA2  A          JMP    E4TSX    ; MAUVAISE GESTION
07010 01612                     *
07020 01613           BACD  A BOUC36 EQU    *
07030 01614A BACD 96 23    A          LDAA   CONST1
07040 01615A BACF 06                  TAP
07050 01616                     *
07060 01617                     *
07070 01618A BAD0 30                  TSX             ; INSTRUCTION-TEST ****
07080 01619                     *
07090 01620                     *
07100 01621A BAD1 07                  TPA
07110 01622A BAD2 91 23    A          CMPA   CONST1
07120 01623A BAD4 27 03 BAD9          BEQ    BOUC37   ; INDICATEURS NON-MODIFIES
07130 01624A BAD6 7E FFA2  A          JMP    E4TSX
07140 01625                     *
```

```
06360 01627                    ************************
06370 01628                    *
06380 01629                    * INSTRUCTION COMB
06390 01630                    *             ====
06400 01631                    *
06410 01632           BAD9  A BOUC37 EQU    *
06420 01633A BAD9 D6 23     A         LDAB   CONST1
06430 01634A BADB 86 FA     A         LDAA   #$FA
06440 01635A BADD 06                  TAP
06450 01636                    *
06460 01637                    *
06470 01638A BADE 53                  COMB             ; INSTRUCTION-TEST ****
06480 01639                    *
06490 01640                    *
06500 01641A BADF 07                  TPA
06510 01642A BAE0 81 F5     A         CMPA   #$F5
06520 01643A BAE2 27 03 BAE7          BEQ    BOUC38    ; TEST TOTALE DE V ET C
06530 01644                    *                       ; TEST PARTIEL DE H, I, N, Z
06540 01645A BAE4 7E FFA2   A         JMP    E4COMB    ; MAUVAISE GESTION
06550 01646                    *
06560 01647           BAE7  A BOUC38 EQU    *
06570 01648A BAE7 5F                  CLRB
06580 01649A BAEB 86 06     A         LDAA   #$06
06590 01650A BAEA 06                  TAP
06600 01651                    *
06610 01652                    *
06620 01653A BAEB 53                  COMB             ; INSTRUCTION-TEST ****
06630 01654                    *
06640 01655                    *
06650 01656A BAEC 07                  TPA
06660 01657A BAED 81 C9     A         CMPA   #$C9
06670 01658A BAEF 27 03 BAF4          BEQ    BOUC39    ; TEST FINAL DE H, I, N, Z
06680 01659A BAF1 7E FFA2   A         JMP    E4COMB    ; MAUVAISE GESTION
06690 01660                    *
```

```
06710 01662                     ****************************
06720 01663                     *
06730 01664                     * INSTRUCTION SEI
06740 01665                     *             ===
06750 01666                     *
06760 01667           BAF4  A BOUC39 EQU    *
06770 01668A BAF4 4F                  CLRA
06780 01669A BAF5 06                  TAP
06790 01670                     *
06800 01671                     *
06810 01672A BAF6 0F                  SEI             ; INSTRUCTION-TEST ****
06820 01673                     *
06830 01674                     *
06840 01675A BAF7 07                  TPA
06850 01676A BAF8 81 D0    A          CMPA   #$D0
06860 01677A BAFA 27 03 BAFF          BEQ    BOUC40   ; I A UN
06870 01678A BAFC 7E FFA2  A          JMP    E4SEI    ; MAUVAISE GESTION
06880 01679                     *
06890 01680           BAFF  A BOUC40 EQU    *
06900 01681A BAFF 96 23    A          LDAA   CONST1
06910 01682A BB01 06                  TAP
06920 01683                     *
06930 01684                     *
06940 01685A BB02 0F                  SEI             ; INSTRUCTION-TEST ****
06950 01686                     *
06960 01687                     *
06970 01688A BB03 07                  TPA
06980 01689A BB04 81 FF    A          CMPA   #$FF
06990 01690A BB06 27 03 BB0B          BEQ    BOUC41   ; INDIC. NON-MODIFIES
07000 01691A BB08 7E FFA2  A          JMP    E4SEI
07010 01692                     *
```

```
07030 01c9A                  **************************
07040 01695                  *
07050 01696                  * INSTRUCTION CLRB
07060 01697                  *             ====
07070 01698                  *
07080 01699          BB0B  A BOUC41 EQU    *
07090 01700A BB0B 86 00    A        LDAA   #$00
07100 01701A BB0D 06                TAP
07110 01702                  *
07120 01703                  *
07130 01704A BB0E 5F                CLRB            ; INSTRUCTION-TEST ****
07140 01705                  *
07150 01706                  *
07160 01707A BB0F 07                TPA
07170 01708A BB10 81 C4    A        CMPA   #$C4
07180 01709A BB12 27 03 BB17        BEQ    BOUC42   ; Z A UN
07190 01710A BB14 7E FFA2  A        JMP    E4CLRB
07200 01711                  *
07210 01712          BB17  A BOUC42 EQU    *
07220 01713A BB17 96 23    A        LDAA   CONST1
07230 01714A BB19 06                TAP
07240 01715                  *
07250 01716                  *
07260 01717A BB1A 5F                CLRB            ; INSTRUCTION-TEST ****
07270 01718                  *
07280 01719                  *
07290 01720A BB1B 07                TPA
07300 01721A BB1C 81 F4    A        CMPA   #$F4
07310 01722A BB1E 27 03 BB23        BEQ    BOUC43   ; N, V, C A ZERO
07320 01723                  *                       ; H, I NON-MODIFIES
07330 01724A BB20 7E FFA2  A        JMP    E4CLRB
07340 01725          BB23  A BOUC43 EQU    *
07350 01726A BB23 7E C000  A        JMP    CC0000   ; TEST CLASSE V
07360 01727                  *
```

AUTORISATION DE SOUTENANCE

VU les dispositions de l'article 3 de l'arrêté du 16 avril 1974,

VU le rapport de présentation de

. Monsieur B. COURTOIS, Chargé de recherche

Monsieur POSTIGO Carlos

est autorisé à présenter une thèse en soutenance pour l'obtention du titre de DOCTEUR de TROISIEME CYCLE, spécialité "Informatique".

Fait à Grenoble, le 13 mai 1983

Le Président de l'I.N.P.-G.

D. BLOCH
Président
de l'Institut National Polytechnique
de Grenoble

P.O. le Vice-Président,