Test et autotest de circuits complexes

José-Maria Gobbi

THESE

présentée à

l' Institut National Polytechnique de Grenoble

pour obtenir le grade de

DOCTEUR INGÉNIEUR

GENIE INFORMATIQUE

par

José-Maria GOBBI

TEST ET AUTOTEST DE CIRCUITS COMPLEXES

Thèse soutenue le 9 décembre 1981 devant la Commission d'Examen :

L. BOLLIET Président

J. CLAUSS
R. GERBER
C. KUBIAK
C. OUANNES
G. SAUCIER
A. VERDILLON

} Examinateurs

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Année universitaire 1979-1980

Président : M. Philippe TRAYNARD

Vice-Présidents : M. Georges LESPINARD
M. René PAUTHENET

PROFESSEURS DES UNIVERSITES

MM.	ANCEAU François	Informatique fondamentale et appliquée
	BENOIT Jean	Radioélectricité
	BESSON Jean	Chimie Minérale
	BLIMAN Samuel	Electronique
	BLOCH Daniel	Physique du Solide - Cristallographie
	BOIS Philippe	Mécanique
	BONNETAIN Lucien	Génie Chimique
	BONNIER Etienne	Métallurgie
	BOUVARD Maurice	Génie Mécanique
	BRISSONNEAU Pierre	Physique des Matériaux
	BUYLE-BODIN Maurice	Electronique
	CHARTIER Germain	Electronique
	CHERADAME Hervé	Chimie Physique Macromoléculaires
Mme	CHERUY Arlette	Automatique
MM.	CHIAVERINA Jean	Biologie, Biochimie, Agronomie
	COHEN Joseph	Electronique
	COUMES André	Electronique
	DURAND Francis	Métallurgie
	DURAND Jean-Louis	Physique Nucléaire et Corpusculaire
	FELICI Noël	Electrotechnique
	FOULARD Claude	Automatique
	GUYOT Pierre	Métallurgie Physique
	IVANES Marcel	Electrotechnique
	JOUBERT Jean-Claude	Physique du Solide - Cristallographie
	LACOUME Jean-Louis	Géographie - Traitement du Signal
	LANCIA Roland	Electronique - Automatique
	LESIEUR Marcel	Mécanique
	LESPINARD Georges	Mécanique
	LONGEQUEUE Jean-Pierre	Physique Nucléaire Corpusculaire
	MOREAU René	Mécanique
	MORET Roger	Physique Nucléaire Corpusculaire
	PARIAUD Jean-Charles	Chimie - Physique
	PAUTHENET René	Physique du Solide - Cristallographie
	PERRET René	Automatique

.../...

MM.	PERRET Robert	Electrotechnique
	PIAU Jean-Michel	Mécanique
	PIERRARD Jean-Marie	Mécanique
	POLOUJADOFF Michel	Electrotechnique
	POUPOT Christian	Electronique - Automatique
	RAMEAU Jean-Jacques	Chimie
	ROBERT André	Chimie Appliquée et des matériaux
	ROBERT François	Analyse numérique
	SABONNADIERE Jean-Claude	Electrotechnique
Mme	SAUCIER Gabrielle	Informatique fondamentale et appliquée
M.	SOHM Jean-Claude	Chimie - Physique
Mme	SCHLENKER Claire	Physique du Solide - Cristallographie
MM.	TRAYNARD Philippe	Chimie - Physique
	VEILLON Gérard	Informatique fondamentale et appliquée
	ZADWORNY François	Electronique

CHERCHEURS DU C.N.R.S. (Directeur et Maître de Recherche)

M.	FRUCHART Robert	Directeur de Recherche
MM.	ANSARA Ibrahïm	Maître de Recherche
	BRONOEL Guy	Maître de Recherche
	CARRE René	Maître de Recherche
	DAVID René	Maître de Recherche
	DRIOLE Jean	Maître de Recherche
	KAMARINOS Georges	Maître de Recherche
	KLEITZ Michel	Maître de Recherche
	LANDAU Ioan-Doré	Maître de Recherche
	MERMET Jean	Maître de Recherche
	MUNIER Jacques	Maître de Recherche

Personnalités habilitées à diriger des travaux de recherche (décision du Conseil Scientifique)

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

MM.	ALLIBERT Michel	
	BERNARD Claude	
	CAILLET Marcel	
Mme	CHATILLON Catherine	
MM.	COULON Michel	
	HAMMOU Abdelkader	
	JOUD Jean-Charles	
	RAVAINE Denis	
	SAINFORT	C.E.N.G.

.../...

MM. SARRAZIN Pierre
SOUQUET Jean-Louis
TOUZAIN Philippe
URBAIN Georges — Laboratoire des Ultra-Réfractaires ODEILLO

E.N.S.M.E.E.

MM. BISCONDI Michel
BOOS Jean-Yves
GUILHOT Bernard
KOBILANSKI André
LALAUZE René
LANCELOT Françis
LE COZE Jean
LESBATS Pierre
SOUSTELLE Michel
THEVENOT François
THOMAS Gérard
TRAN MINH Canh
DRIVER Julian
RIEU Jean

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

MM. BOREL Joseph
CHEHIKIAN Alain
VIKTOROVITCH Pierre

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

MM. BORNARD Guy
DESCHIZEAUX Pierre
GLANGEAUD François
JAUSSAUD Pierre
Mme JOURDAIN Geneviève
MM. LEJEUNE Gérard
PERARD Jacques

E.N.S.H.G.

M. DELHAYE Jean-Marc

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

MM. COURTIN Jacques
LATOMBE Jean-Claude
LUCAS Michel
VERDILLON André

Je tiens à remercier Monsieur L. BOLLIET, Professeur à l'Institut Universitaire de Technologie, pour l'honneur qu'il me fait de présider mon jury.

Je suis très honoré de la présence

de Monsieur GERBER, Ingénieur du Centre National des Télécommunications de Grenoble,

de Monsieur C. KUBIAK, Ingénieur du Centre National d'Etudes des Télécommunications de Lannion,

de Monsieur J. CLAUSS, Ingénieur de la Compagnie Téléphonie Industrielle et Commerciale de Strasbourg,

de Monsieur C. OUANNES, Maître de recherche du CNRS, chargé de mission à l'Agence Nationale de l'Informatique.

de Monsieur A. VERDILLON, Maître Assistant de l'Université Scientifique et Médicale de Grenoble.

Que Madame G. SAUCIER, Professeur à l'Institut National Polytechnique de Grenoble trouve ici l'expression de ma gratitude pour sa participation à ce jury, pour ses nombreux conseils, pour sa direction constante de cette thèse et les discussions critiques qui m'ont permis de l'améliorer.

Je voudrais exprimer ma reconnaissance à Monsieur J.L. RAINARD, Ingénieur au Centre National d'Etudes des Télécommunications de Grenoble pour sa grande contribution à l'élaboration de ce travail.

J'exprime ma reconnaissance aux camarades de l'équipe "Conception et Sécurité des Systèmes Logiques" du Laboratoire de Mathématiques Appliquées de Grenoble. Que soient en particulier remerciés Madame C. BELLON et Monsieur P. CASPI pour leurs critiques et leur collaboration.

Je remercie aussi Monsieur François LANG du Centre Norbert Segard, pour sa collaboration à la réalisation du circuit intégré étudié dans cette thèse.

Je voudrais enfin remercier Madame Simone NOUAILHAS qui a assuré avec dévouement et compétence la frappe de cette thèse.

INTRODUCTION

L'objet de ce travail est l'étude du test et de la testabilité de circuits intégrés complexes. L'avènement de tels circuits a remis en cause les résultats obtenus sur le test dans les années 60-70 ; en effet les recherches sur le test à cette époque se situaient au niveau analytique (circuits décrits comme un ensemble de portes logiques) (I-3) (I-4) (I-6).

A l'apparition de circuits très complexes, un ensemble de méthodes de test à haut niveau (test comportemental, test fonctionnel) ont été proposées (I-7) (I-8) (I-19). De telles méthodes de test sont relativement simples à mettre en oeuvre, mais leur efficacité est difficile à estimer.

Il s'agit donc ici de réétudier les méthodes de test analytiques en vue du test de circuits complexes. Deux points importants se dégagent :

- Pour pouvoir maîtriser la complexité de circuits VLSI, il est primordial de disposer d'une description du circuit qui soit à la fois claire et précise. Ce problème de description en vue du test est l'objet du chapitre I et de l'annexe A-I, et est évoqué tout au long de ce travail.

- Le problème du test doit être pris en compte à toutes les étapes de la conception ; cette prise en compte peut soit mener à la modification du circuit pour faciliter le test (Chapitre II), soit être un des critères fondamentaux de la conception (conception d'un circuit facilement testable et partiellement autotestable (Chapitre III).

PLAN DETAILLE

CHAPITRE I : NIVEAUX DE DESCRIPTION ET METHODES DE TEST ASSOCIEES.

Le chapitre présente les différents types de description possibles d'un circuit, et les classifie en niveaux ; à chaque niveau de description peut être associé un ensemble de méthodes de test utilisables.

Nous nous intéressons particulièrement à une description mixte : description fonctionnelle de la partie contrôle, description structurelle de la partie opérative. Une telle description reste manipulable pour un circuit complexe et permet l'utilisation soit d'une méthode de test de type analytique ascendant, soit d'une méthode de type fonctionnel descendant ; pour cela on transpose sur cette description des méthodes classiques.

CHAPITRE II : TESTABILITE ET POSSIBILITE DE DIAGNOSTIC DE CIRCUITS VLSI.

La testabilité est une des conditions critiques qu'un circuit VLSI doit remplir et elle doit être prise en compte pendant l'étape de conception. La validation du prototype du circuit à la fin du processus de conception, le test de plaquettes après fabrication et le test d'entrée par l'utilisateur peuvent être faciles et efficaces ou bien pratiquement impossibles selon l'organisation interne du circuit et de ses points d'entrée/sortie.

Les contraintes de test peuvent amener à définir des points d'accès supplémentaires : insertion de points de test. Il est donc d'une grande importance :

- de spécifier avant la conception les contraintes de testabilité imposés au circuit à concevoir.

- d'évaluer très vite, à toutes les étapes de la conception la testabilité et les possibilités de diagnostic de la structure proposée, et l'intérêt de l'insertion de points d'accès de test.

Le concepteur doit donc disposer :

- d'une description claire du circuit.

- d'une méthode d'évaluation de la testabilité qui soit utilisable tout au long de la conception.

Nous proposons dans ce chapitre une telle méthode d'évaluation de la testabilité, permettant de comparer, par rapport au test les différentes stratégies d'insertion de points de test.

CHAPITRE III : UN CIRCUIT FACILEMENT TESTABLE ET PARTIELLEMENT AUTOTESTABLE :

Ce chapitre concerne ma participation à la conception d'un circuit correcteur d'erreur pour mémoires dynamiques, conception faite au Centre National d'Etudes des Télécommunications de Grenoble.

ANNEXE I : LES OUTILS CAO ET LE TEST :

Un ensemble d'outils CAO utilisés pour la description de circuits et pour le test sont ici présentés et discutés par rapport aux méthodes de test proposées au chapitre I.

ANNEXE II : TEST DE PROTOTYPE :

Une expérience quasi-industrielle du test de protoype et la mise au point de la méthode est décrite en cette annexe.

C H A P I T R E I

NIVEAUX DE DESCRIPTION D'UN CIRCUIT ET METHODES DE TEST

I-1- ETAT DE L'ART

Un circuit peut être décrit à trois niveaux :

- niveau structurel
- niveau fonctionnel
- niveau comportemental.

a) A un niveau structurel, le circuit est décrit comme une interconnexion de blocs élémentaires. Ces blocs sont généralement des portes logiques mais ils peuvent être aussi soit des entités de niveau plus haut tels que les registres, décodeurs, blocs de mémoire, soit des entités de niveau plus bas tels que composants électroniques, etc...

Les informations données par la description sont essentiellement des informations topologiques de connexions entre blocs élémentaires et peuvent être mises en langages formels (langages au niveau portes ou transferts de registres).

b) Par description au niveau fonctionnel, on entend une description donnant les états du circuit et les transitions entre ces états.

Dans le passé, les circuits séquentiels ont été décrits en tant qu'automates d'états finis ; la description fonctionnelle étant la table de transitions ("flow table") ou le graphe d'états (1).

L'intérêt actuel pour les unités de contrôle ou micro-contrôleurs a conduit à des descriptions fonctionnelles basées sur le graphe d'états renseigné (2) . A ce niveau, l'état d'un circuit séquentiel est défini par l'ensemble de commandes envoyées vers son environnement appelé la partie opérative (P.O.) ou chemin de données : cette description est bien adaptée à des circuits spécialisés ou à des contrôleurs/séquenceurs et permet une description "multiniveaux" très efficace de circuits complexes (voir Fig. I-2).

c) Au niveau algorithmique ou niveau comportemental ("behavioral"), un circuit est décrit par l'algorithme qu'il exécute ou par ses fonctions "usager" (indépendamment du choix des éléments de mémorisation). Par exemple, un microprocesseur sera décrit à travers ses instructions sans prendre en compte le détail de sa structure interne.

Généralement, la séparation des différents types de description n'est pas toujours claire. Quelques descriptions sont un "mélange" de deux ou même de trois niveaux.

Exemple : Considérons un circuit complexe, un microprocesseur ; nous avons les types de descriptions suivants : (tableau 1)

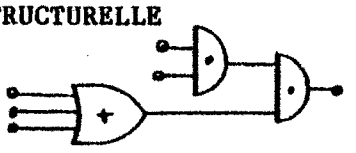
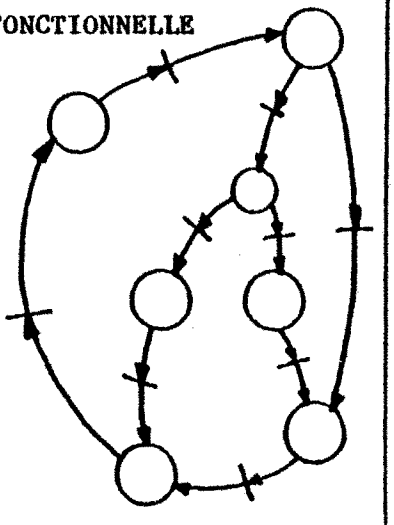
DESCRIPTION STRUCTURELLE 	INTERCONNEXION DE BLOCS ELEMENTAIRES
DESCRIPTION FONCTIONNELLE 	Graphe d'états de la P.C. où : - un chemin correspond à l'exécution d'une instruction. - un état est renseigné par l'ensemble de microopérations exécutées par la partie opérative.
DESCRIPTION COMPORTEMENTALE	Jeu d'instructions.
DESCRIPTION MIXTE (FONCTIONNELLE + COMPORTEMENTALE)	Jeu d'instructions + la description des éléments de mémoire internes.

TABLEAU 1 : NIVEAUX DE DESCRIPTION

I-2- METHODES DE TEST CORRESPONDANTS AUX NIVEAUX DE DESCRIPTION

Pour les descriptions structurelles l'hypothèse de panne est normalement le collage à un ou à zéro ("stuck at fault"). La méthode de test habituellement utilisée est basée sur la propagation de la panne (ou plutôt de l'erreur) sur les chemins du circuit, à partir du point de la panne jusqu'à une sortie observable du circuit.

Ce type de méthodes (D-Algorithme (3), différence Booléenne (4)) a été défini au niveau des portes et est limité par la complexité de l'algorithme de recherche de vecteurs de test.

L'utilisation d'une description structurelle basée sur des entités de niveau supérieur (registres,...) ne modifie pas fondamentalement ce problème de complexité.

Pour les descriptions fonctionnelles, le test consiste à balayer tous les états et à traverser toutes les transitions du graphe en utilisant des séquences d'entrées prédéterminées ("checking experiments") (6).

Les états et les transitions sont identifiés par des séquences d'identification ("distinguishing sequences"). Le problème critique est l'initialisation ("homing or synchronizing séquences").

Les erreurs considérées dans ces méthodes d'identification sont très générales, mais elles sont limitées par un certain nombre d'hypothèses (cette méthode différencie l'automate correct de tout autre automate qui possède au plus le même nombre d'états).

Une extension de ces méthodes pour les parties contrôle a été présentée (7). L'observation des sorties booléennes directes de la machine séquentielle est remplacée par l'observation à travers le chemin de données (Partie Opérative) ; cette observation peut être "différée" dans l'espace et le temps, et des problèmes particuliers de masquage se posent.

Au niveau comportemental, les méthodes de test sont essentiellement des méthodes de couverture : Couverture des activations possibles ou des fonctions usager. Cette approche est largement utilisée pour le test de microprocesseur. Une méthode de test, à ce niveau de description, mettrait en jeu des opérandes aléatoires ou des ensembles d'opérandes définis par les propriétés des différentes fonctions, indépendamment des éléments de mémoire internes au circuit à tester.

Une attention spéciale doit être accordée aux méthodes de test qui sont définies à un certain niveau mais qui utilisent des hypothèses de pannes d'un niveau inférieur.

Entre le niveau fonctionnel et le niveau comportemental, les méthodes de test de microprocesseurs présentées en (8, 9) sont définis au niveau comportemental mais des informations sur la structure interne sont ajoutées, en particulier les éléments de mémoire internes. Cela nous conduit à des hypothèses d'erreurs fonctionnelles et la couverture des instructions est étendue pour inclure la couverture des erreurs fonctionnelles.

Entre les niveaux fonctionnel et structurel nous trouvons la méthode de Poage (10, 11) qui utilise une approche fonctionnelle (distinction des machines bonne et mauvaise). La mauvaise machine est définie par des hypothèses de panne données au niveau structurel.

Le test aléatoire est un cas spécial (12). Des entrées aléatoires sont appliquées au circuit ; la longueur de la séquence de test doit être déterminée par une étude faite au niveau structurelle.

Les différentes méthodes de test sont synthétisées dans le tableau 2 :

DESCRIPTION STRUCTURELLE	. D-Algorithme . Différence Booléenne . Test aléatoire
	. Identification de la machine + hypothèses de pannes structurelles
DESCRIPTION FONCTIONNELLE	. Identification de la machine . Test des unités de contrôle
	. Test d'"activation" + hypothèses de pannes fonctionnelles
DESCRIPTION COMPORTEMENTALE	. Test d'"activation"

TABLEAU 2 : METHODES DE TEST

I-3- VERS UNE DESCRIPTION MULTINIVEAUX DE CIRCUITS COMPLEXES

I-3-1 Idée générale :

Nous considérons ici des "circuits complexes" (LSI), en excluant les circuits combinatoires, où une grande complexité est normalement associée avec une forte répétitivité. Ce dernier type de circuits a été largement étudié (13), (14) et nous ne le prendrons pas en compte.

Nous nous intéressons ici à des outils de description permettant de décrire un circuit complexe comme un tout (La description partielle ne nous intéresse pas), mais à différents niveaux de détail.

Nous allons commencer au niveau fonctionnel et chercher des descriptions qui seraient compatibles ou pourraient être étendues au niveau comportemental ou bien au niveau structurel.

L'idée principale consiste à diviser un circuit complexe en deux parties : la partie contrôle (P.C.) et la partie opérative (P.O.). Ce type de partition, largement utilisé en informatique (15) et en contrôle des procédés (16), sépare le système en : (Fig. I-1).

* La P.C. qui exécute un algorithme, reçoit les commandes du monde extérieur ou d'un niveau supérieur de contrôle.

* La P.O., ou chemin de données, fait le traitement des données selon les commandes de la P.C.

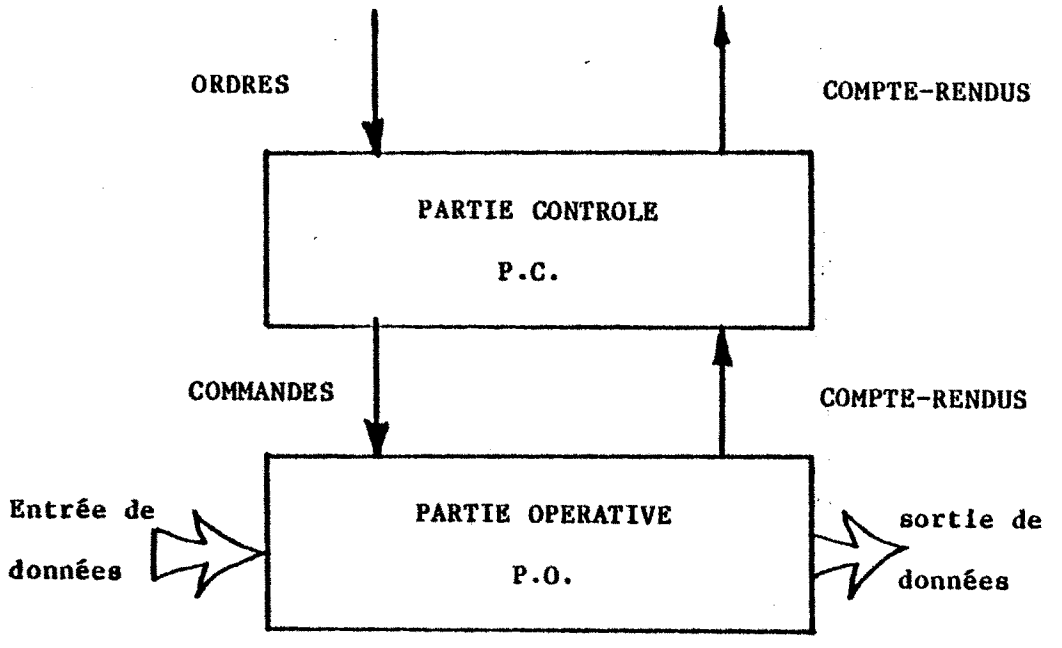


FIGURE I-1 : PARTITION D'UN CIRCUIT COMPLEXE

PROPRIETES D'UNE TELLE DESCRIPTION :

Cette description est récursive (17). La P.O. peut être vide (si elle est considérée directement comme le monde extérieur) ou bien elle-même peut être divisée à nouveau en P.C. et P.O. ...

A chacune de ces deux parties est associé un type différent de description (voir section I-3-2) : l'une montrant le séquencement effectué par la P.C. (graphe de contrôle) et l'autre description étant adaptée à la P.O.

I-3-2 Principes de la description

La P.O. est décrite de façon structurelle au niveau de transfert de registres.

La P.C. est décrite de façon fonctionnelle, cette description est un graphe de contrôle "interprété". Il s'agit d'un graphe d'états classique. Un noeud représente un état de la P.C. L'"interprétation" consiste à associer à un noeud la description des micro-opérations exécutées par la P.O. pendant que le circuit se trouve dans l'état correspondant de la P.C.

Les arcs représentent les transitions possibles entre états ; ils sont renseignés par les conditions de transition (prédicats)

I-3-3 Exemple :

Nous considérons un circuit simple, multiplieur 4 x 4 bits. Ce circuit acquiert en entrée le multiplicateur (4 bits) puis le multiplicande (4 bits). La fin de cette deuxième opération déclenche le processus de calcul ; une fois le calcul terminé, le circuit affiche en sortie les 4 bits poids faible puis les 4 bits poids fort du résultat.

Une description comportementale consisterait, par exemple, en la description en langage pseudo-ALGOL, donnée dans la figure I-2 :

Notre description consiste en :

* La description structurelle de la P.O. (Fig. 3)
* La description fonctionnelle de la P.C. (Fig. 5)

Maintenant il est possible de mettre en évidence la synchronisation du multiplieur avec les autres circuits qui envoient les opérandes sur le BUS et qui lisent les résultats sur le même BUS, à l'aide des signaux de synchronisation MULTI, ADR, FINMULTI.

Les registres nécessaires sont introduits : ACCU, MQ , D, B.

B = Registre destiné à recevoir le multiplicande si MULTI = 1 et ADR = 0.

MQ = Registre destiné à recevoir le multiplicateur si MULTI = 1 et ADR = 1; MQo = bit de poids faible de MQ .

ADDI = Circuit combinatoire additionneur parallèle de 4 bits (4 additionneurs complets de 1 e.b.) (4 "FULL-ADDERS")
Le résultat ACCU + B est chargé dans ACCU complété à gauche par D, la bascule de débordement.

ACCU = Registre accumulateur qui envoie en permanence son contenu sur ADDI. Et simultanément le registre ACCU reçoit en permanence les sorties d'ADDI mais ne les échantillonne que sur le front montant de la commande ADDITION.

D = Bascule de débordement du registre ACCU.

CPTR = Compteur du nombre de pas de l'algorithme de multiplication.

TRIS1, TRIS2 = Amplificateurs trois états ("BUFFERS - TRISTATE")

```
PROCEDURE MULTIPLIER  (entier A, B, C) ·

  Début :
    Lire A ;
    Lire B ;
    C : = A x B ;
    écrire C ;
  Fin.
```

FIGURE I-2 : DESCRIPTION COMPORTEMENTALE DU MULTIPLIEUR.

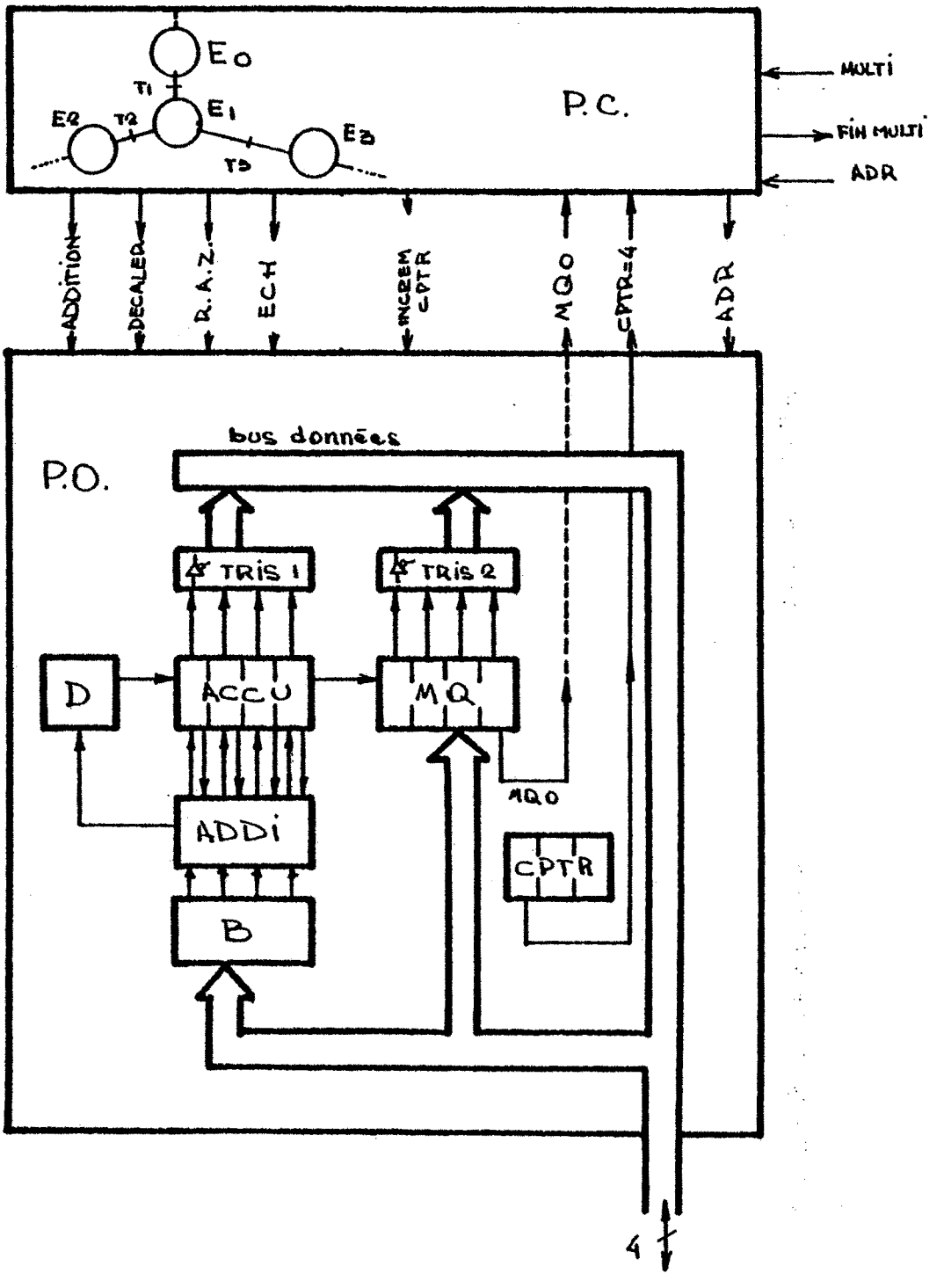


FIGURE I-3a : P.O. DU MULTIPLIEUR.

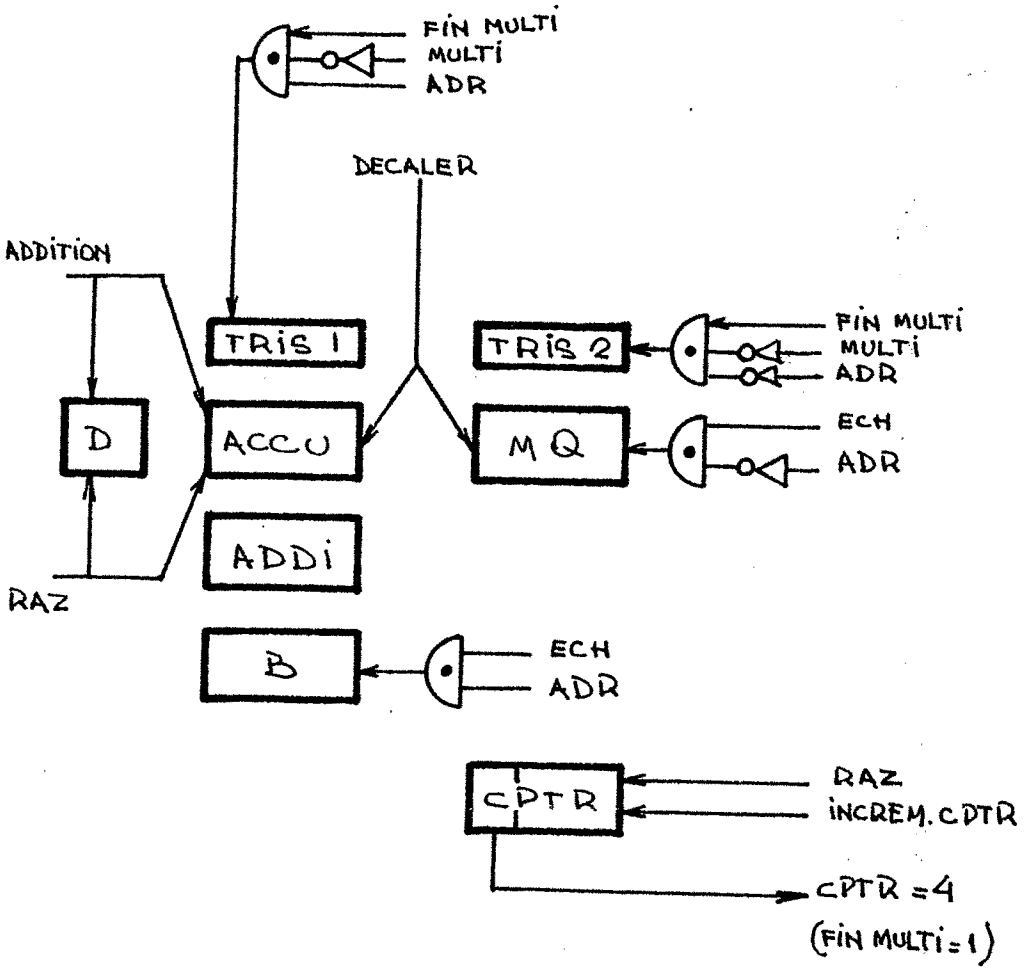


FIGURE I-3b : DETAIL DES COMMANDES SUR LA P.O.

De façon très schématisée voici l'algorithme de multiplication :

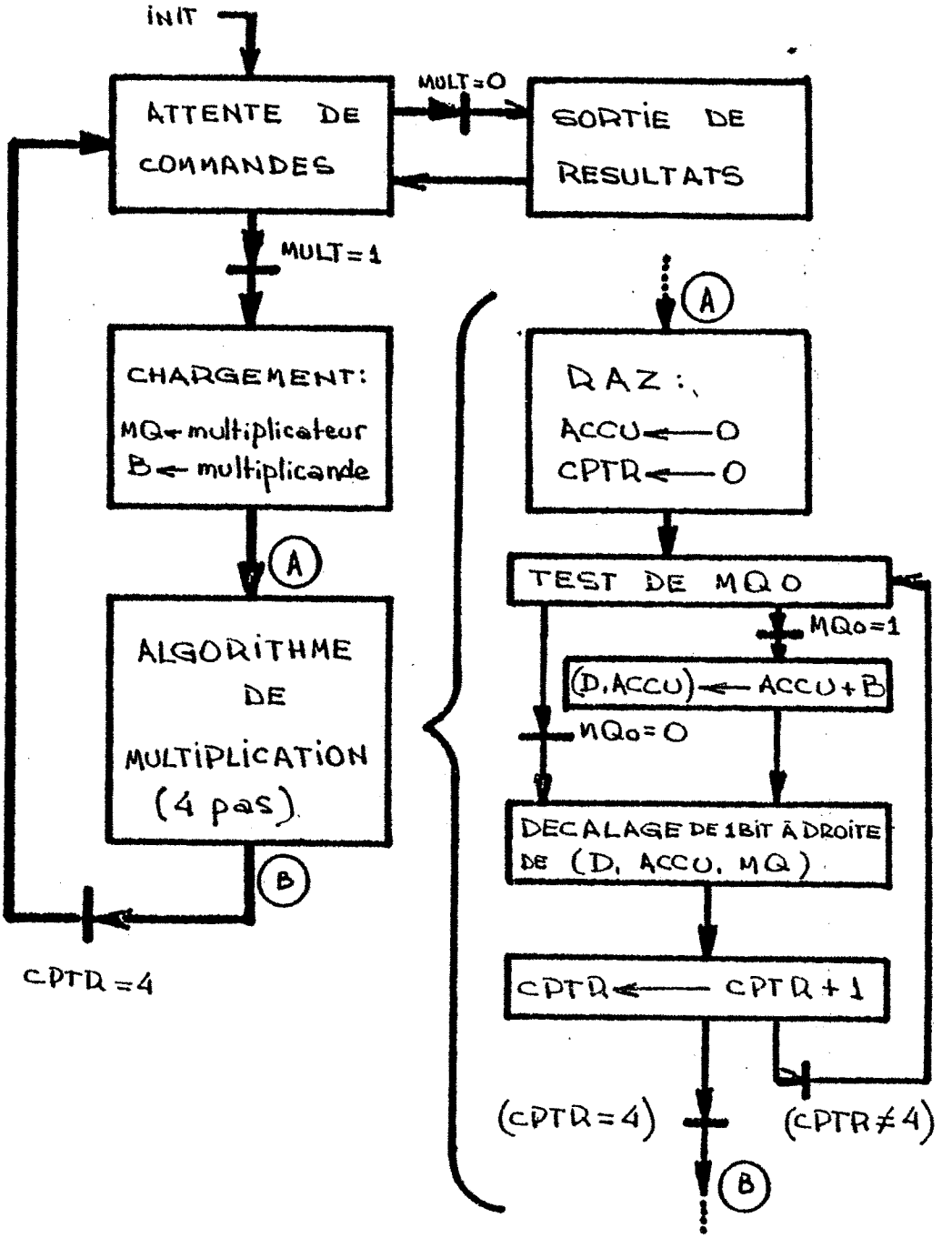


FIGURE I-4 : L'ALGORITHME DE MULTIPLICATION DANS LE SCHEMA GENERAL DE FONCTIONNEMENT

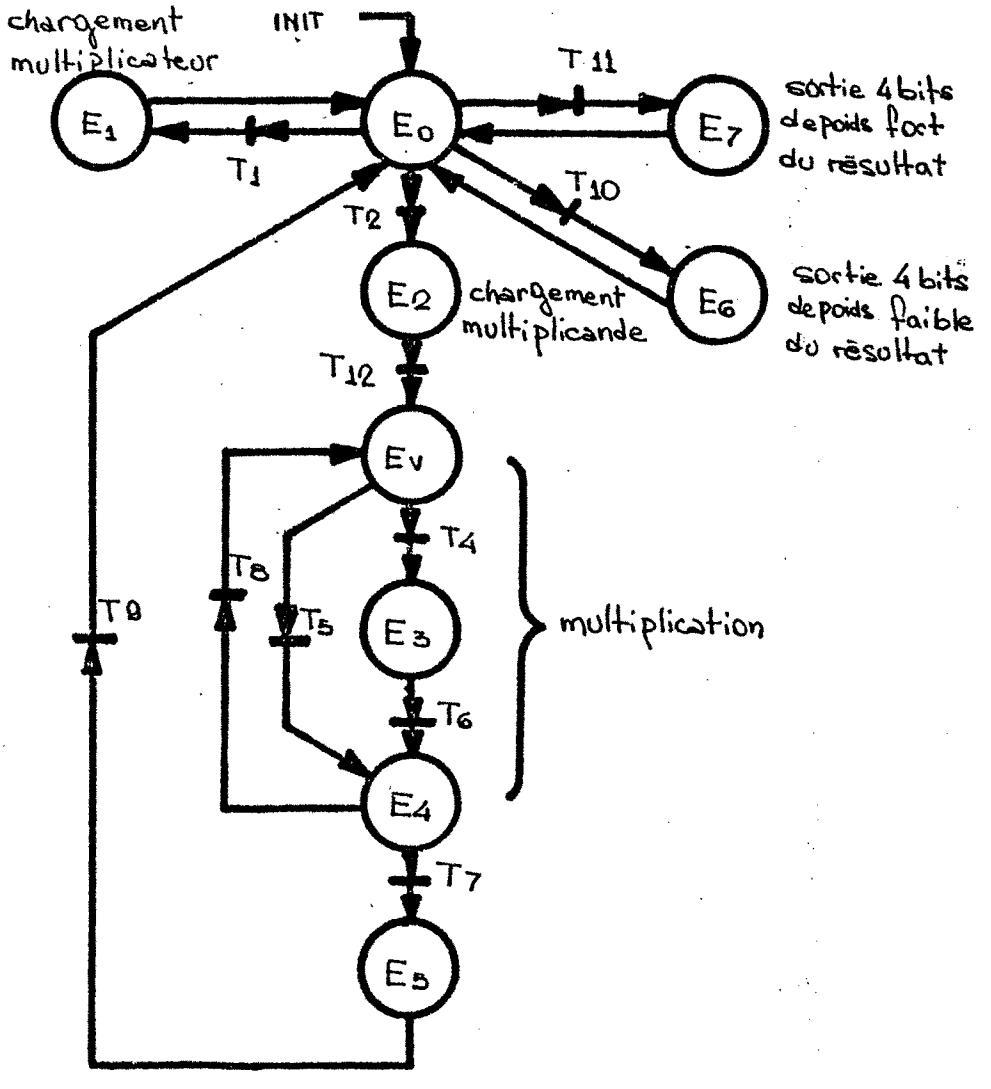


FIGURE I-5a) : GRAPHE DE CONTROLE DU MULTIPLIEUR

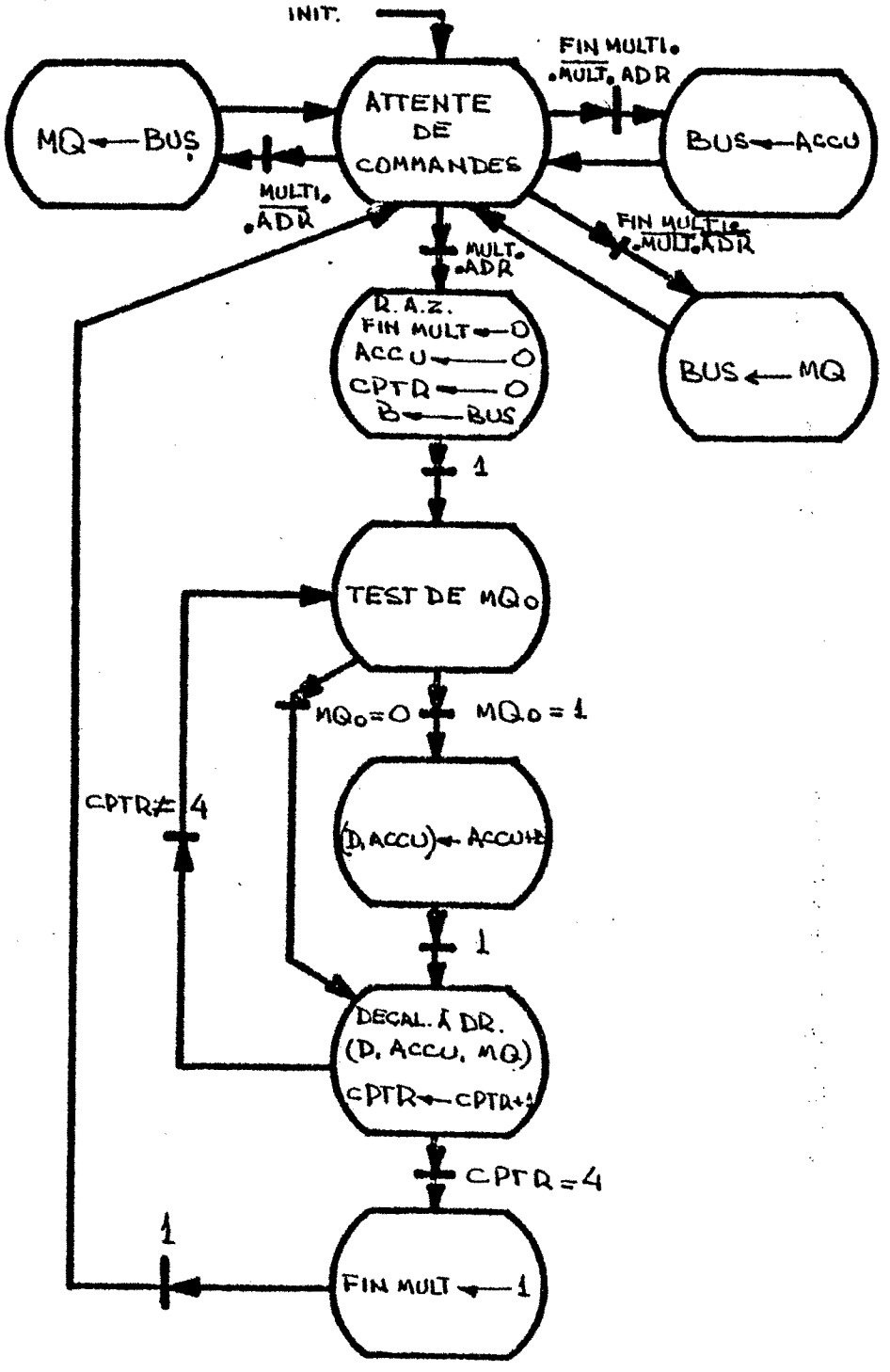


FIGURE I-5b) : GRAPHE "INTERPRETE" DU MULTIPLIEUR

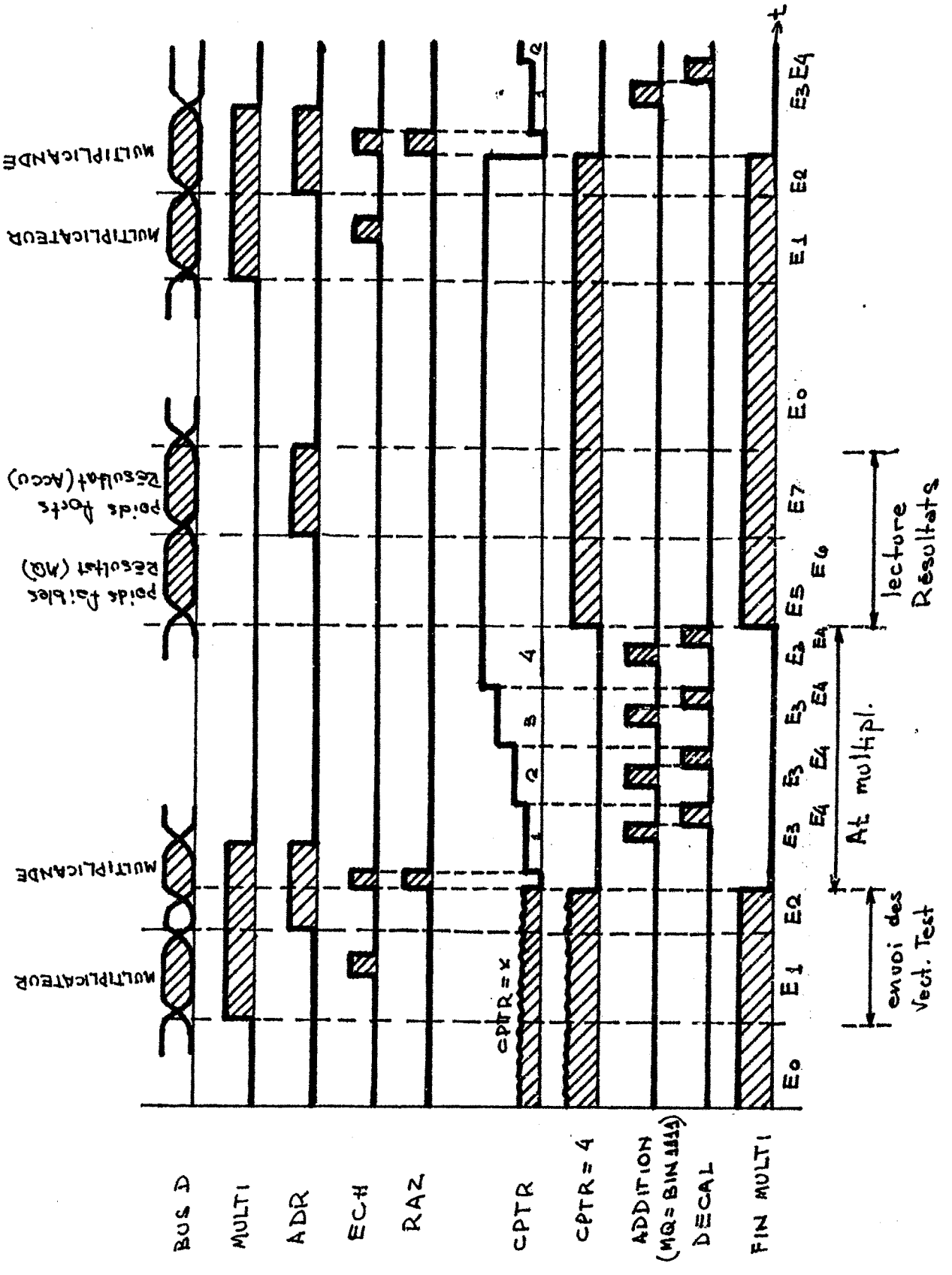


FIGURE I-5-c) : CHRONOGRAMME DU MULTIPLIEUR

I-3-4 Conséquences du type de description sur le problème de test :

La description proposée ici se caractérise par le fait d'être de types différents pour la P.O. et la P.C. :

- La P.C. est décrite au niveau fonctionnel ;
- La P.O. est décrite au niveau structurel.

Cette description permet une approche récursive parce que certaines parties de la P.O. peuvent être divisées à nouveau en P.C. et P.O.

L'avantage de notre description est de permettre l'utilisation de méthodes diversifiées de test (méthodes structurelles, fonctionnelles ou bien comportementales).

En partant d'une description fonctionnelle de la P.C. nous pourrons faire deux extensions :

a) extension vers le niveau structurel en prennant en compte les hypothèses de panne du niveau structurel;

b) extension vers le niveau comportemental en utilisant les méthodes de test par couverture du graphe d'états.

Nous allons regarder ces deux aspects.

I-4 UTILISATION DE LA DESCRIPTION MULTI-NIVEAU POUR METHODES DE TEST ANALYTIQUES

Les méthodes de test analytiques sont en rapport avec les hypothèses de panne au niveau structurel. Une méthode analytique consiste à considérer une panne donnée et dériver la situation de test à partir de cette panne, par exemple, l'ensemble des vecteurs d'entrée du circuit qui permettent de détecter cette panne.

Nous considérons les deux types de pannes suivants :

- Les pannes qui ne changent pas le graphe de contrôle de la P.C.

- Les pannes qui modifient le graphe de contrôle.

I-4-1 Pannes qui ne modifient pas le graphe de contrôle :

Ces pannes sont pour la plupart des pannes localisées sur la P.O. ou sur les fils de commande entre la P.C. et la P.O.

Les opérations pour tester un circuit avec ce type de panne se décomposent en trois étapes : la MANIFESTATION de la panne, et la génération de vecteurs de test pour manifester cette panne ("CONSISTANCE") puis la PROPAGATION du vecteur erroné. Ces étapes sont similaires aux étapes suivies pour une méthode structurelle par chemin sensible (3) ; mais la différence dans notre cas est que la "consistance" et la propagation sont faites au niveau fonctionnel, en utilisatn le graphe de contrôle de la P.C.

I-4-2 Manifestation :

Une panne dans la P.O. est considérée d'abord au niveau structurel ; un collage à "0" ou à "1" peut être considéré de manière classique.

Le test est étudié d'abord au niveau structurel, dans le bloc de transfert de registres dans lequel la panne est considérée (en utilisant le D-algorithme par exemple) ; cette étude nous conduit à déterminer les données d'entrée du bloc de transfert de registres qui mettront en évidence la panne.

Ce qui veut dire trouver des données d'entrée pour lesquelles la sortie du bloc de transfert de registres serait erronée si la panne existait.

Nous devons appliquer cela au graphe de contrôle de la P.C. ; à une panne donnée, nous pouvons associer des noeuds de manifestation (de mise en évidence) c'est-à-dire, des noeuds qui activent une microopération, laquelle met en évidence la panne.(*)

(*) La NON DETECTABILITE d'une panne se traduit par l'absence de toute manifestation de la panne. Elle est liée à la redondance des circuits. Il faut ajouter les pannes qui rendent IMPREVISIBLE le comportement d'un circuit pour l'apparition de certaines configurations interdites sur les entrées ou sur des variables internes.

EXEMPLE :

En regardant l'exemple précédent (Fig. I-3 et I-5) nous supposons que sur la ligne transfert entre le registre ACCU et l'amplif. TRIS1, le fil de bit de poids le plus fort est collé à "1".

La panne sera mise en évidence si deux conditions sont réunies :

a) La valeur d'ACCU est 'OXXX' (avec "X" = valeur indéterminée de ce bit).

b) Le contenu du registre ACCU est transféré sur le bus de données à travers TRIS1, cela signifie que dans le graphe de contrôle (Fig. I-5), le noeud de manifestation sera E7.

I-4-3 Génération de l'ensemble de conditions de test ("CONSISTANCE") :

Le problème est de trouver l'ensemble des entrées du système qui génèrent les conditions adéquates de test.

Comme le graphe de contrôle n'est pas modifié, il est possible d'utiliser une méthode similaire à celle utilisée pour la génération des conditions de test pour la vérification de programmes, par exemple la méthode déductive (18).

Cette méthode consiste à :

a) Définir les noeuds d'entrée du graphe de contrôle (noeuds où l'on peut commander l'entrée de données) ;

b) Propager sur le graphe de contrôle les conditions de test à partir du premier noeud de manifestation de la panne, pour déterminer la séquence nécessaire de vecteurs d'entrée.

La propagation est faite en utilisant le graphe inverse du graphe de contrôle (les arcs sont inversés et chaque fonction est remplacée par la fonction inverse) ; cette propagation inverse est arrêtée quand on arrive au noeud initial (le noeud où l'on initialise le circuit).

Exemple : En utilisant le même exemple qu'en I-3-3 (Fig. I-3 et I-5), nous considérons la panne qui colle à "1" le bit de poids le plus fort sur les lignes de transfert entre ACCU et TRIS1.

Le noeud de manifestation est E7 et nous avons besoin d'une valeur du registre ACCU = "OXXX" juste avant l'exécution du E7.

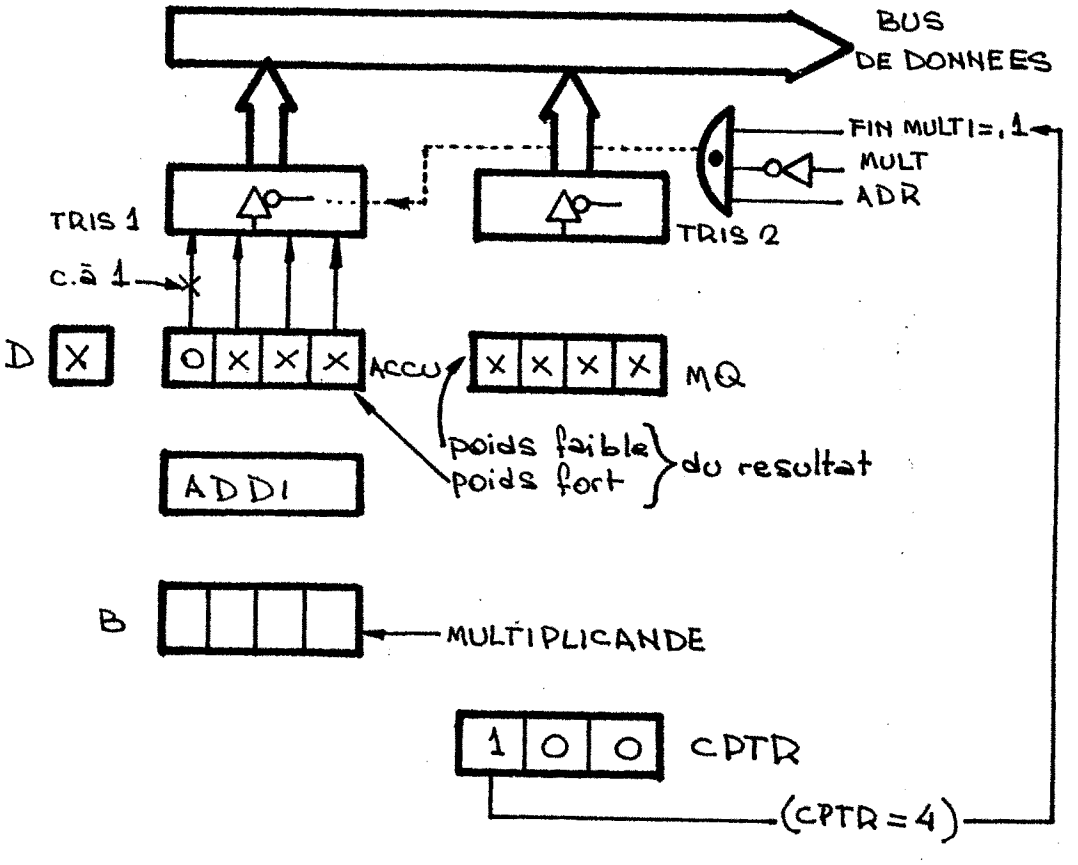


FIGURE I-6 : SITUATION NECESSAIRE AVANT L'EXECUTION

Pour arriver à un résultat qui remplisse cette condition il y a plusieurs paires de valeurs de multiplicande et multiplieur : tous ceux qui donnent comme résultat des valeurs inférieures à "1 000 000" ($<$ 128). Il nous suffit comme exemple de prendre le multiplicande unitaire et un multiplieur arbitraire, "1111" par exemple.

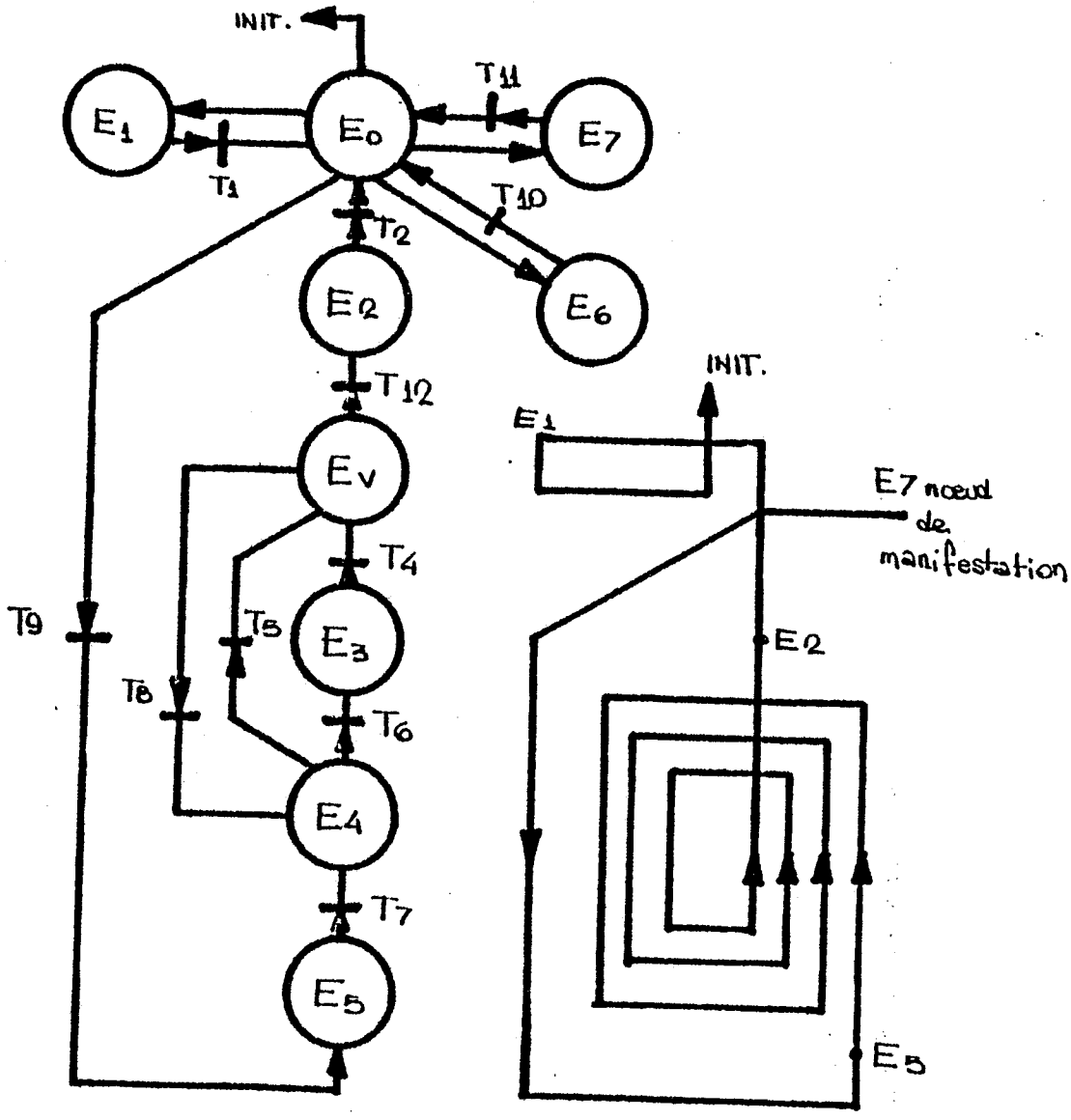


FIGURE I-7 : GRAPHE INVERSE ET CHEMIN SUIVI POUR LA "CONSISTANCE"

Nous allons montrer avec le tableau 3 comment la "consistance" est faite :

- La première colonne nous montre le chemin suivi sur le graphe (en inversant les arcs) ;

- La seconde colonne nous donne les valeurs à prendre par les registres internes avant l'exécution du noeud ou la traversée de la transition ;

- La troisième colonne nous montre les entrées du monde extérieur déduites des valeurs des registres internes.

Supposons un MULTIPLICANDE = "0001" et un MULTIPLIEUR = "1111".

La séquence d'entrées indispensable recherchée pour générer les conditions nécessaires de test peut être lue directement sur la dernière colonne (en partant de la dernière ligne vers la première).

NOEUDS	VALEURS INTERNES (avant d'entrer dans le noeud ou de passer la transition)					ENTREE DE L'EXTERIEUR		
TRANSITIONS	ACCU	MQ	B	CPTR	FIN MULTI	BUS	MULTI	ADR
E7	0XXX	XXXX	0001	XXX	X	0XXX		
T11	0XXX	"	"	XXX	1	XXXX	0	1
E0	0XXX	"	"	XXX	1	"	X	X
T9	0XXX	"	"	XXX	1	"	"	"
E5	0XXX	"	"	XXX	1	"	"	"
T7	0XXX	"	"	100	1	"	"	"
E4	0000	"	"	011	0	"	"	"
T6	0000	"	"	011	0	"	"	"
E3	0001	"	"	011	0	"	"	"
T4	"	XXX1	"	011	0	"	"	"
EV	"	"	"	011	0	"	"	"
T8	"	"	"	011	0	"	"	"
E4	0000	"	"	010	0	"	"	"
T6	"	"	"	010	0	"	"	"
E3	0001	XXX1	"	010	0	"	"	"
T4	"	XX11	"	010	0	"	"	"
EV	"	"	"	010	0	"	"	"
T8	"	"	"	010	0	"	"	"
E4	0000	"	"	001	0	"	"	"
T6	"	"	"	001	0	"	"	"
E3	0001	XX11	"	001	0	"	"	"
T4	"	X111	"	001	0	"	"	"
EV	"	"	"	001	0	"	"	"
T8	"	"	"	001	0	"	"	"
E4	0000	"	"	000	0	"	"	"
T6	0001	"	"	000	0	"	"	"
E3	0001	X111	"	000	0	"	"	"
T4	0000	1111	"	000	0	"	"	"
EV	0000	"	"	000	0	"	"	"
T12	0000	"	0001	000	0	XXXX	"	"
E2	0000	"	XXXX	000	X	0001	"	"
T2	XXXX	"	"	XXX	X	XXXX	1	1
E0	"	"	"	XXX	X	XXXX	X	X
E1	"	1111	"	XXX	X	1111	X	X
T1	"	XXXX	"	XXX	X	XXXX	1	0
E0	"	"	"	XXX	X	XXXX	X	X
Mise en route	XXXX	XXXX	XXXX	XXX	X	XXXX	X	X

TABLEAU 3

I-4-4 Propagation :

Nous devons considérer ici deux cas, selon la conséquence de la panne considérée sur le séquencement.

Si la panne ne modifie pas le séquencement de la P.C., on peut toujours arriver à un noeud de sortie (un NOEUD DE SORTIE est un noeud où les données arrivent à une sortie observable par l'extérieur).

Dans ce cas, le problème de la propagation est alors de trouver le chemin le plus court sur le graphe entre le noeud de manifestation et un noeud de sortie.

La seule difficulté sera d'étudier le masquage de l'erreur : a) par une autre utilisation de la même partie en panne (par exemple, le passage par un autre noeud de manifestation) ;

b) par l'algorithme lui même.

Une panne qui ne modifie pas le graphe de contrôle de la P.C. peut encore modifier le séquencement (en forçant toujours un mauvais choix de chemin sur le graphe). Dans certains cas, aucun noeud de sortie ne peut être atteint. Les deux circuits (l'un avec ce type de panne, l'autre avec panne dans la P.C.) sont équivalents du point de vue extérieur. Alors le test est fait avec les méthodes de I-3-4. Dans les autres cas, on peut utiliser des méthodes de propagation.

EXEMPLE : En utilisant l'exemple de la Fig. I-3, supposons que le troisième bit du registre MQ (MQ2) soit collé à "0".

Au moment de dérouler l'algorithme de multiplication et pour une valeur de CPTR = 2 ce bit arrivera à la position MQo. Il sera testé et alors fera traverser la transition T5 deux fois de suite, quelles que soient les valeurs chargées dans MQ2 et MQ3 initialement.

Cette panne de la P.O. n'a pas modifié le graphe de contrôle, mais elle a faussé le séquencement dépendant des données.

I-4-5 Pannes qui modifient le graphe de contrôle :

Une méthode de test analytique consisterait ici en une approche similaire à la méthode de Poage (10), c'est-à-dire une étude des pannes au niveau structurel et de leurs conséquences au niveau fonctionnel (graphe de contrôle).

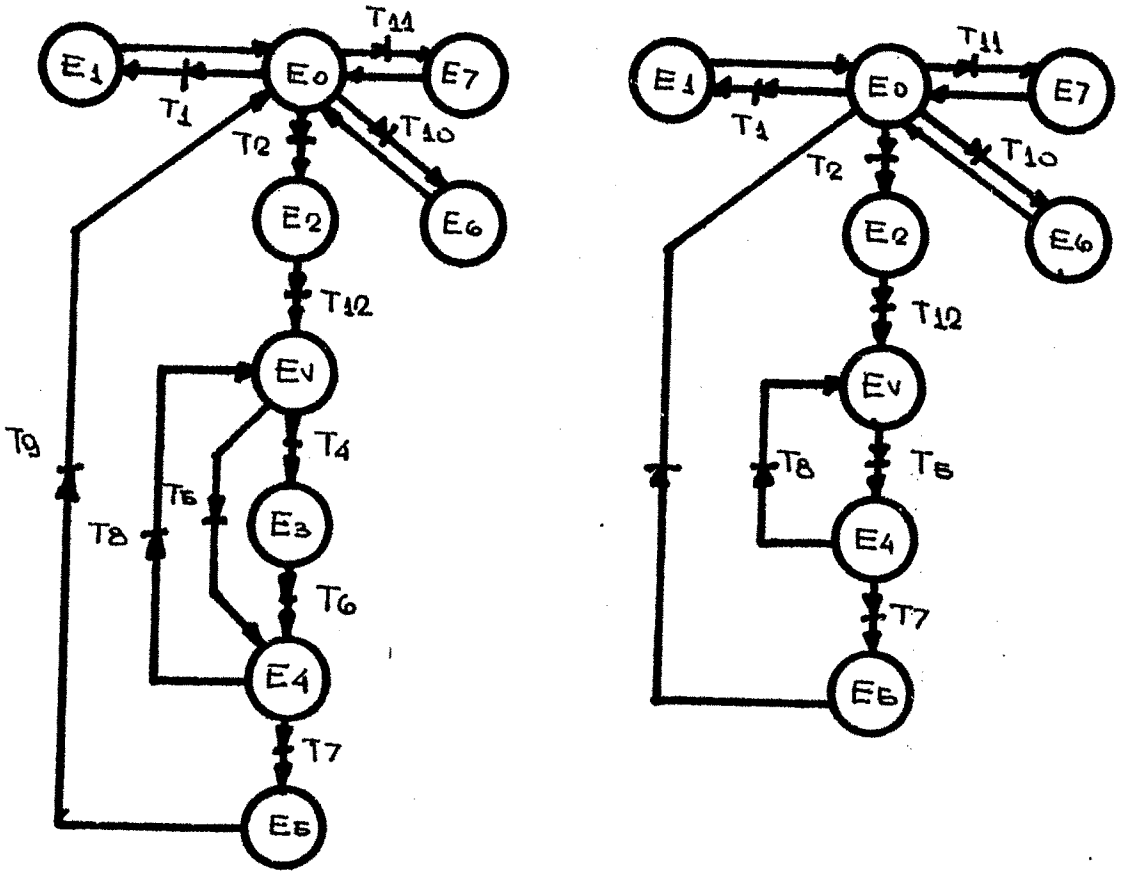
En partant d'une analyse structurelle d'une panne, on établit le graphe de contrôle modifié. La méthode de test consiste alors à considérer le produit (cartésien) du graphe correct et du graphe erroné et à déterminer la séquence la plus courte pour différencier le bon graphe du mauvais.

Le problème critique est celui de l'initialisation. Cette méthode est utilisable seulement dans les cas où un RAZ du coeur du circuit ("hardcore") met le circuit (bon ou mauvais) dans un état initial donné.

EXEMPLE : Supposons qu'une analyse structurelle d'une panne du circuit de la Fig. I-3 nous amène au graphe modifié de la Fig. I-8b.

L'état E3 a été éliminé du graphe de contrôle, par exemple par un collage à "0" de la sortie MQo vers la P.C.

Une séquence distingue les graphes bons et mauvais si une sortie observable est différente suivant l'état (correct ou faux) de la P.C.



a) GRAPHE CORRECT

b) GRAPHE ERRONE

FIGURE I-8

Les sorties observables sont FINMULTI et le BUS de DONNEES.

CHEMIN PARCOURU SUR LE GRAPHE		ENTREES AU GRAPHE			SORTIE DU GRAPHE	
CORRECT	MAUVAIS	BUS	MULTI	ADR	CORRECT	MAUVAIS
Eo	Eo					
E1	E1	1111	1	0		
Eo	Eo					
E2	E2	0001	1	1		
Ev	Ev					
E3	E4					
E4	Ev					
Ev	E4					
E3	Ev					
E4	E4					
Ev	Ev					
E3	E4					
E4	E5					FINMULTI=1
Ev	Eo					"
E3	Eo					"
E4	Eo					"
E5	Eo				FINMULTI=1	"
Eo	Eo				"	"
E6	E6		0	0	BUS=1111	BUS=0000

TABLEAU 4

Le tableau 4 nous donne la séquence de test nécessaire pour la panne donnée.

Les deux graphes sont différenciés par le signal "FINMULTI" qui est d'abord envoyé par le mauvais graphe (13 pas après l'initialisation) pendant que le graphe correct continue à faire la multiplication (jusqu'à 17 pas après l'initialisation).

La panne sera manifestée à l'extérieur du multiplieur:

- soit par l'avance du signal FINMULTI, si on dispose d'une référence temporelle précise,
- soit par la valeur émise sur le bus de données à la fin de l'opération.

En tous cas, la complexité de la méthode est fortement liée au nombre de pannes : en général, chaque panne définit un graphe faux. On doit donc trouver un ensemble de séquences d'entrée capables de différencier le graphe correct de la famille entière des graphes faux (11).

Le nombre de pannes possibles augmente avec la complexité des P.C. et avec elles augmente aussi la complexité de la méthode de test. Malgré cela, le critère pour l'utilisation de cette méthode est la "fréquence" des sorties observables : si le temps écoulé entre un état et la sortie observable, conséquence de cet état, est un temps court, les séquences de test seront courtes, et la méthode sera utilisable.

I-5- APPROCHE COMPORTEMENTALE OU FONCTIONNELLE DESCENDANTE

Dans la section I-4, nous sommes partis d'une panne donnée et nous avons essayé de déduire une situation de test. Maintenant, nous allons considérer une approche descendante. C'est-à-dire nous partons de la description du circuit et nous cherchons un ensemble efficace d'activations de test.

Ici nous allons différencier deux approches importantes : l'approche comportementale et l'approche fonctionnelle.

I-5-1 Approche comportementale :

Cette approche consiste à activer les "fonctions usagers" du circuit ; la description de ces fonctions est indépendante de la structure interne. Pour un microprocesseur, ces fonctions sont l'ensemble des instructions.

Un tel test, appelé aussi test de conformité, consiste à exécuter la séquence suivante pour chaque fonction "usager" ;

- initialiser l'état du circuit ;
- envoyer les entrées de test ;
- exécuter la fonction ;
- observer l'état du circuit.

L'état du circuit est défini ici par la valeur de l'ensemble d'éléments de mémoire contrôlés et observés par l'utilisateur. Les données de test sont soit des données aléatoires soit des données déduites à partir des considérations fonctionnelles telles que la symétrie ou la complexité des fonctions.

De tels tests peuvent être générés automatiquement (19) à partir d'une description "usager".

I-5-2 Approche mixte multi-niveau :

Nous proposons de décrire un circuit complexe avec une description telle que celle donnée en section I-2 : description basée sur les états pour la P.C. (graphe de contrôle), une description structurelle pour la P.O.

Des ensembles de vecteurs de test peuvent être associés avec chaque bloc de la P.O. (on peut faire une bibliothèque de vecteurs de test "déterministes" pour ces blocs). L'approche pour le test serait fonctionnelle et descendante pour la P.C., complétée par l'insertion de vecteurs de test pour les blocs de la P.O.

Le test consistera en l'activation de chaque chemin du graphe de contrôle tout en envoyant leurs vecteurs de test à chaque bloc de la P.O.

Cette approche évite les méthodes analytiques pour la P.C. décrite en section I-4.

Le problème de couverture du graphe est le même que pour le test de programmes mais un peu plus facile ici. Cette solution paraît un compromis acceptable pour des circuits intégrés complexes et amène à des programmes de test efficaces et courts.

I-6- CONCLUSION DU CHAPITRE I

Après cette présentation des différents niveaux de description et des méthodes de test associées, nous avons montré qu'une description multiniveau simplifie le problème de test, spécialement dans le cas de méthodes de test analytique.

L'analyse de pannes et de leurs manifestations sont étudiés au niveau structurel, pendant que la séquence de test est déterminée au niveau fonctionnel. C'est pour cela que la complexité du test se voit réduite.

La même description peut aussi être utilisée pour une méthode de test fonctionnel.

I-7- BIBLIOGRAPHIE DU CHAPITRE I

(1) Z. KOHAVI

"Switching and finite automata theory"

Mc. Grawhill, Computer Science Series, 1970.

(2) T.R. BLAKESLEE

"Digital design with standard MSI and LSI"

John Wiley & Sons, New York, 1975.

(3) J.P. ROTH, W.C. BOURICIUS, P.R. SCHNEIDER

"Programmed algorithms to compute test to detect and distinguish between failures in logic circuits".

IEEE.Trans. on Computers, October 1967.

(4) S.S. YAN, Y.S. TANG

"An efficient algorithm for generating complete tests sets for combinational logic circuits".

IEEE Trans. on Computers, November 1971.

(5) G. SAUCIER, J. LEBRUN, C. ROBACH

"Processor testability and design consequences".

IEEE Trans. on Computers, June 1976.

(6) A.D. FRIEDMAN, P.R. MENON

"Fault detection in digital circuits".

Prentice-Hall Inc, Englewood Cliffs, N.J. 1971.

(7) G. SAUCIER, C. ROBACH
"Dynamic testing of control units".
IEEE Trans. on Computers, July 1978.

(8) S.M. THATTE, J.A. ABRAHAM
" A methodology for functional level testing of microprocessors", Fault-tolerant Computing Symp., Toulouse, June 1978.

(9) G. SAUCIER, C. ALEONARD, C. ROBACH
"Microprocessor systems testing : a review and future prospects".
EUROMICRO Journal, January 1979.

(10) J.P. POAGE
"The derivation of optimum tests for logic circuits".
Ph.D, Princeton University, 1963.

(11) G. SAUCIER
"Recherche d'une séquence de test d'une machine séquentielle"
Revue RAIRO, AFCET (France), 1972.

(12) J.J. SHEDLETSKY
"Random testing : practicality vs. Verified effectiveness".
Fault Tolerant Computin Symp., Los Angeles, June 1977.

(13) D.C. BOSSEN, S.J. HONG

"Cause-effect analysis for multiple fault detection in combinational networks".

IEEE Trans. on Computers, November 1971.

(14) C. TURCAT, A. VERDILLON

"Symetry, automorphism, and test"

IEEE Trans., on Computers, April 1979.

(15) F.J. HILL, G.R. PETERSON

"Digital systems : hardware organization an design".

2nd Ed. John Wiley Ed. New York, 1978.

(16) GRAFCET Group

"Pour une représentation normalisée du cahier des charges d'un automatisme logique".

Revue Automatique & Informatique industrielles, Nov-Dec. 1977.

(17) W. GRASS, H.M. LIPP

"LOGE : a highly effective system for logic design automation"

ACM SIG D.A. News letters, June 1979.

(18) C.A.R. HOARE

"An axiomatic approach to computer programming".

Com. ACM. October 1969.

(19) G. SAUCIER, C. ROBACH

"Microprocessor functional testing"

1980 Test conference, Philadelphia, November 1980.

C H A P I T R E II

TESTABILITE ET POSSIBILITE DE DIAGNOTIC

DE CIRCUITS COMPLEXES (VLSI)

II-1 INTRODUCTION

Nous allons essayer de "mesurer" les améliorations introduites pour la testabilité PC-PO avec l'insetion des points de test. Cette insertion doit être envisagée dès l'étape de conception du circuit.

La testabilité peut être définie comme une valeur qui mesure la difficulté à vaincre pour tester un circuit ; elle peut être utilisée comme une mesure "prédictive".

Notre but est de donner les moyens d'évaluer à priori la testabilité d'un VLSI en étape de conception.

II-2 PROBLEME GENERAL DE TEST

En partant d'un type de DESCRIPTION de circuit donnée en chapitre I :

- description structurelle de la P.O. et

- description fonctionnelle de la P.C., nous allons étudier la façon de tester PC-PO.

a) Testabilité et diagnosticabilité :

Le problème du test n'est pas toujours le même, son but peut varier selon le moment d'application :

- Le TEST EN FIN DE CONCEPTION du circuit est appliqué en étape de "déverminage", probablement sur les premiers échantillons. A ce moment-là, on s'intéresse à la DETECTION et à la LOCALISATION de pannes pour améliorer la conception (ou même, le procédé de fabrication). On connaît la structure interne du circuit et on profite au maximum des points de test pour la mise au point du circuit.

- Le TEST DE FIN DE PRODUCTION ou TRI INDUSTRIEL est un paramètre influençant fortement le coût final du circuit intégré.

En réalité, les circuits sont triés à deux niveaux de leur fabrication :

TEST SUR PLAQUETTE : C'est-à-dire avant les opérations de découpage et de mise en boîtier qui sont très onéreuses.

Tous les plots sont accessibles avant la soudure, et c'est le moment où on peut profiter de certains points de test spéciaux : les plots exclusivement de test qui permettent d'améliorer la finesse du diagnostic.

TEST SUR BOITIER : C'est-à-dire avant stockage ou livraison du produit fini pour le fabricant ou bien test d'entrée (et seul possible dans la plupart des cas) pour l'utilisateur.

On veut détecter seulement les pannes. Une fois le circuit mis en boîtier, on ne dispose plus des points supplémentaires de test qu'on pouvait utiliser sur plaquette.

On voit donc que le test peut avoir deux buts : détection ou localisation des pannes, et que les facilités de test dont on dispose varient suivant l'étape où est appliqué le test.

La testabilité est une mesure de la facilité de détection des pannes d'un circuit.

La diagnosticabilité est une mesure de la facilité de localisation des pannes dans un circuit (ou facilité de diagnostic).

Les mesures doivent être prédictives, pour pouvoir être évaluées au cours de la conception ; elles reposent sur une description au niveau des blocs fonctionnels.

La testabilité et la diagnosticabilité d'un circuit dépendent évidemment des facilités de test dont on dispose (Cf. Chapitre II-4), donc varient suivant le type de test effectué (fin de conception ou de fabrication).

b) Test fonctionnel de la P.C. :

Le test de la P.C. sera fait par COUVERTURE DU GRAPHE D'ETATS avec deux possibilités différentes pour l'observation :

- L'OBSERVATION DIRECTE de la P.C. implique la sortie de signaux de commandes vers le monde extérieur.

- L'OBSERVATION INDIRECTE de la P.C., qui est le cas général, s'effectue via la P.O.

Considérons un circuit ou une partie d'un circuit décrit par un GRAPHE DIRECT du type de celui de la Fig. I-5. Les noeuds correspondant aux états du circuit et les arcs indiquent des transitions entre ces états.

Un état est caractérisé par les valeurs de la mémoire et par les sorties envoyées vers le monde extérieur ou vers la P.O.

Le test consistera à mettre en jeu cette partie du circuit en passant par toutes les transitions du graphe de contrôle.

La testabilité et la diagnosticabilité sont alors assimilées à la notion d'observabilité de l'état. Le circuit appartiendra à une des quatre classes suivantes :

Observabilité de : \ Classe	I	II	III	IV
Eléments de mémoire	d	i	d	i
Signal de commande	d	d	i	i

d = observabilité directe

i = observabilité indirecte.

L'observabilité directe signifie que les valeurs des éléments de la mémoire ou des signaux de commande peuvent être prises directement aux points de test ou sont émises vers le monde extérieur.

Observabilité indirecte des éléments de mémoire. Pendant le test, la traversée d'un état est vérifiée à travers les valeurs des signaux de sortie. L'identification d'un état est complète si on trouve une correspondance biunivoque entre chaque état et les valeurs de l'ensemble des signaux de commande.

Plus généralement, on peut dire que la mesure de diagnosticabilité est inversement proportionnelle au nombre le plus grand d'états qui envoient les mêmes valeurs de signaux de commande. Cela est en rapport avec la possibilité d'identification d'un état à la vue des signaux de commande.

Observation indirecte de signaux de commande : La valeur des signaux doit être observée à travers une opération exécutée par un bloc de la P.O. Cette observabilité est caractérisée par le nombre de couches traversées (après la partition pour la testabilité de la P.O.) et par le délai d'observation.

c) Le test de la P.O. peut se faire :

- avec les commandes normales émises par la P.C. du circuit (contrôle interne) ou bien
- sous contrôle externe spécial de test.

Avec le CONTROLE INTERNE du circuit, il y a deux possibilités de séquencement, normal et spécial :

Le séquencement normal nous permet un test global de PC-PO avec un parcours du graphe de contrôle. Il s'agit du test typique fait par l'usager, avec ses inconvénients classiques de testabilité faible et d'absence de diagnostic fin.

Le séquencement spécial peut nous permettre d'affiner le test et le diagnostic avec un graphe de contrôle particulier de test prévu à la conception. Cette solution a comme inconvénient la surface supplémentaire de Silicium qui est nécessaire pour faire le séquenceur de test.

Avec le CONTROLE SPECIAL EXTERNE du test il nous faut forcement ajouter de points de test sur le circuit. Ces points doivent nous permettre d'amener les commandes de l'extérieur vers la P.O. (en faisant la déconnection de la P.C. propre du circuit).

L'inconvénient de cette solution est le fait de tester exclusivement la P.O. Par contre son avantage est la facilité de mise en oeuvre en fin de conception ou pour le test sur plaquette.

Nous nous intéressons dans les paragraphes suivants au test de P.O., et plus particulièrement au contrôle externe du test (II-4 et II-5).

II-3 TESTABILITE DE LA P.O.

La testabilité et la diagnosticabilité de la P.O. sont étudiées à l'aide d'une modélisation graphique, utilisant un graphe biparti (5).

a) Modélisation :

Un GRAPHE est BIPARTI si l'ensemble X de ses sommets est partitionné en deux classes :

- l'ensemble P des places et
- l'ensemble T des transitions.

Les places sont représentées graphiquement par des cercles et les transitions par des barres.

Un circuit logique à tester est modélisé par un graphe biparti dans lequel :

- Les places $z_i \in Z$ sont des parties matérielles du système (par exemple : additionneur, registre,...) c'est-à-dire des ensembles de circuits logiques ayant une fonction bien définie.

- Les sources $s_j \in S$ (S est contenu dans Z) sont des modules d'accès à partir du monde extérieur du circuit ; les puits $p_k \in P$ (P est contenu dans Z) sont des modules d'observation par l'extérieur du système (interface avec l'extérieur). Autrement dit, toute information (en particulier les données de test) en provenance du monde extérieur ENTRE DANS LES SOURCES $s \in S$ du graphe; toute information (en particulier les résultats de test) est recueillie à l'extérieur du système via les puits $p \in P$ du graphe.

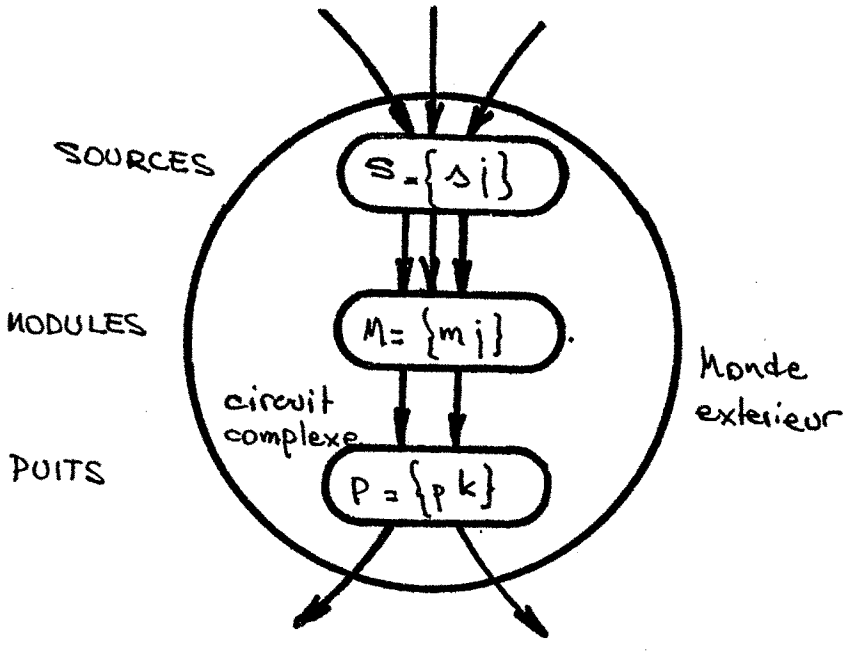


FIGURE II-1 : MODELE DU CIRCUIT LOGIQUE A TESTER

- Les transitions $t_i \in T$ représentent les conditions (fonctions, opérations, commandes) autorisant le transfert d'information d'un module m_i vers un module m_j. Ces conditions sont issues de l'organe de contrôle ou de gestion.

- Les arcs $u_j \in U$, connectant places et transitions, représentent les supports de l'information à travers le circuit.

Cette modélisation est une représentation topologique et fonctionnelle de la P.O. Le graphe peut être obtenu à partir de la description structurelle de la P.O. et du graphe de la P.C.

b) Ecoulement (5) :

Un écoulement est un ensemble de sources, modules, puits et transitions.

Il s'agit d'un sous-graphe biparti tel que :

- Si un module m appartient à E, l'une des transitions prédécesseurs et l'une des transitions successeurs appartiennent à E.

- Si une transition appartient à E, toute place prédécesseur et toute place successeur appartiennent à E.

- Pour tout module de l'écoulement, il existe un chemin (au sens théorie de graphes) appartenant à l'écoulement, allant d'une source $s \in E$ à un puits $p \in E$ via ce module m.

Un ECOULEMENT correspond à la notion intuitive d'ensemble minimal de matériel pouvant fonctionner COMPLETEMENT et ISOLEMENT du reste du matériel :

- Il peut FONCTIONNER COMPLETEMENT : il doit contenir une source et un puits pour l'introduction de l'information initiale et la sortie de l'information transformée.

- C'est un ENSEMBLE ISOLABLE : pour toute transition appartenant à l'écoulement et que définit une opération, les modules d'où provient et où va l'information nécessaire à cette opération, doivent tous appartenir à l'écoulement.

Si on inhibe les transitions provenantes des modules extérieurs à l'écoulement, l'ensemble est effectivement isolé.

Un ECOULEMENT K-COMPATIBLE est un écoulement qui peut être activé par la P.C. normale du circuit.

L'étude de testabilité sous séquencement normal se fera donc à partir des écoulements K-compatibles uniquement.

c) Etude des écoulements par rapport au test :

Le problème de vérification de la P.O. consiste en la détermination d'un ensemble d'écoulements permettant de tester (et éventuellement, localiser) les blocs défectueux d'un circuit complexe.

Hypothèse :

H1) Tout arc arrivant sur une place permet d'acheminer toutes les données nécessaires au test de cette place.

H2) Tout arc partant d'une place permet d'observer tous les résultats de test issus de cette place.

Il s'ensuit que tout module m_i appartenant à un écoulement E_j peut être potentiellement testé par E_j. On dit que m_i est COUVERT par E_j.

Définitions :

- Pour un circuit donné et un ensemble de points de commande et d'observation, donc pour un ensemble d'écoulements, deux modules sont indiscernables intrinsèquement si ils sont couverts par le même sous-ensemble d'écoulements.

- Pour un sous-ensemble d'écoulements, c'est-à-dire pour une stratégie de test, deux modules sont indiscernables pour ce sous-ensemble d'écoulements si ils sont couverts par les même écoulements de ce sous-ensemble.

- Deux modules indiscernables appartiennent au même bloc d'indiscernabilité B_i.

Etant donné une P.O. munie d'un certain ensemble de points de test, ce qui permet de déterminer les sources et les puits, on détermine l'ensemble des écoulements.

On détermine les blocs de modules indiscernables par l'ensemble des écoulements, ou blocs d'indiscernabilité intrinsèque $B_1, B_2, \ldots B_k$; soit N_{Bi} le nombre de modules composant le bloc B_i.

On évalue la diagnosticabilité D_c de ce circuit par :

$$D_c = \frac{1}{\max (N_{Bi})}$$

A chaque module m_i de la description du circuit, on associe un poids p_i pour le test. Ce poids p_i indique la difficulté que ce module présente pour le test, et plus particulièrement pour la consistance et la propagation. Ce poids dépend de la fonction réalisée par le module et non de sa réalisation pour permettre une évaluation prédictive. Nous proposons :

$p_i = 1$ pour un module réalisant la fonction identité

$p_i = 2$ pour un module réalisant une fonction unaire (décalage...)

$p_i = 3$ pour un module réalisant une fonction binaire (additionneur...)

$p_i = 4$ pour une fonction ternaire (UAL...) etc...

Si l'on pouvait accéder à chacun des n modules indépendamment des autres, la testabilité du circut serait :

$$\frac{1}{p_1 + p_2 + \ldots + p_n}$$

En fait, un écoulement E_j comporte plusieurs modules m_{1j}, $m_{2j}, \ldots m_{ij}$ et la testabilité T_{E_j} pour cet écoulement est mesurée par :

$$T_{Ej} = \frac{1}{p_{1j} \times p_{2j} \times \ldots \times p_{ij}}$$

Etant donnée une stratégie de test déterminée, donc un ensemble $\mathcal{E}$ d'écoulements choisi pour effectuer le test d'une P.O., on peut déterminer :

- La diagnosticabilité pour cet ensemble d'écoulements:

$$D_{C,\mathcal{E}} = \frac{1}{\max (N_{Bi})}$$

où N_{Bi} est le nombre de modules des blocs d'indiscernabilité pour l'ensemble $\mathcal{E}$ d'écoulements.

- La testabilité pour cet ensemble d'écoulements :

$$T_{C,\mathcal{E}} = \frac{1}{\sum_{\substack{\text{ensemble des} \\ \text{écoulements}}} \prod_{\substack{\text{modules d'un} \\ \text{écoulements}}} p_{ij}}$$

d) Points de test supplémentaires :

L'application du concept d'ECOULEMENT au TEST d'un CIRCUIT COMPLEXE est faite en deux étapes :

- D'abord l'analyse du modèle du circuit en termes d'écoulements. Cela nous permet d'explorer les possibilités de faire un test complet et la localisation de défauts sans modifications du circuit original.

- La deuxième étape permet de définir une politique de test optimale avec l'introduction des points d'accès supplémentaires. Ces points de test ajoutés permettent :

* d'améliorer la commande et l'observation des différents modules.

* d'accroître la testabilité du circuit avec une réduction du coût du test et un résolution plus fine pour le diagnostic.

Dans l'exemple que nous allons traiter avec l'application des écoulements nous allons étudier successivement :

- pas de points de test supplémentaires ;

- points de test rajoutés pour la commande spéciale de l'extérieur mais pas de points supplémentaires pour l'observation ;

- points de test rajoutés pour la commande et pour l'observation.

II-4 ETUDE D'UN TEST DE P.O. SOUS CONTROLE SPECIAL DE TEST

Considérons l'exemple du multiplieur du chapitre I, dont la description structurelle est répétée par commodité (Fig. II-2 a et b)

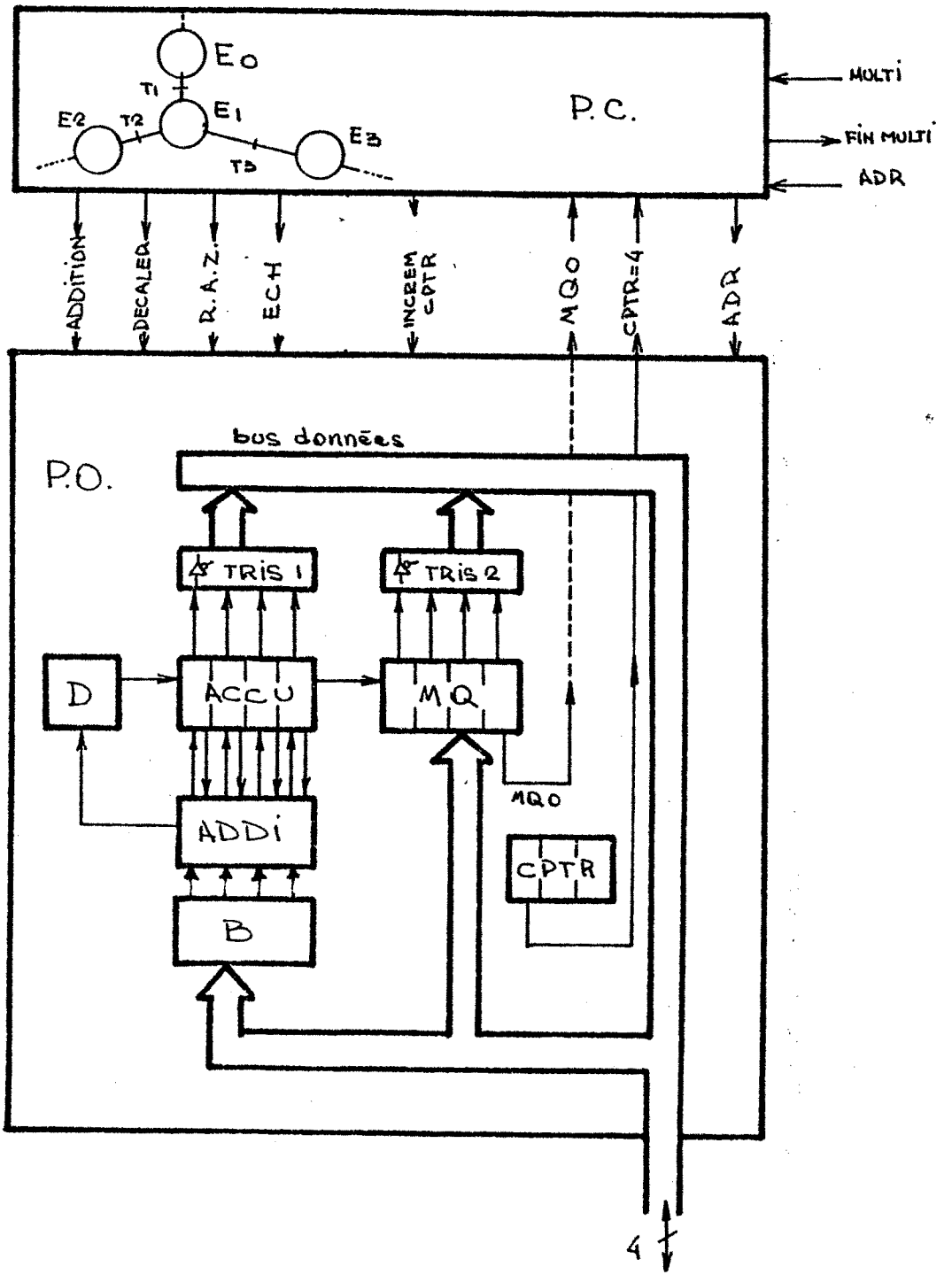


FIGURE II-2 a : P.O. DU MULTIPLIEUR

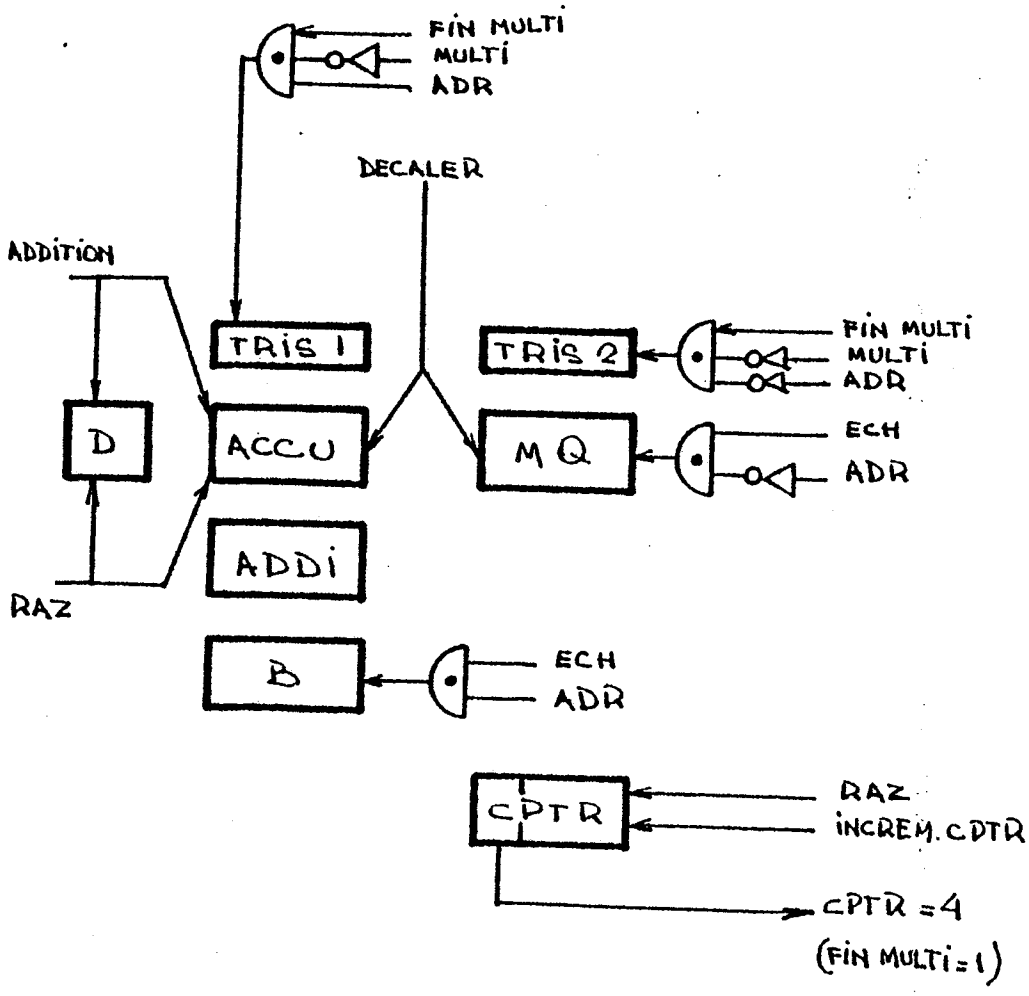


FIGURE II-2 b : DETAIL DES COMMANDES SUR LA P.O.

La figure II-3 donne la modélisation par un graphe biparti de la P.O. du multiplieur. Le compteur CPTR en a été exclu, comme il s'agit en fait de la partie opérative de la P.C., et qu'elle sera testé en même temps que la P.C. Nous allons étudier le test de cette P.O. pour différentes insertions de points de test.

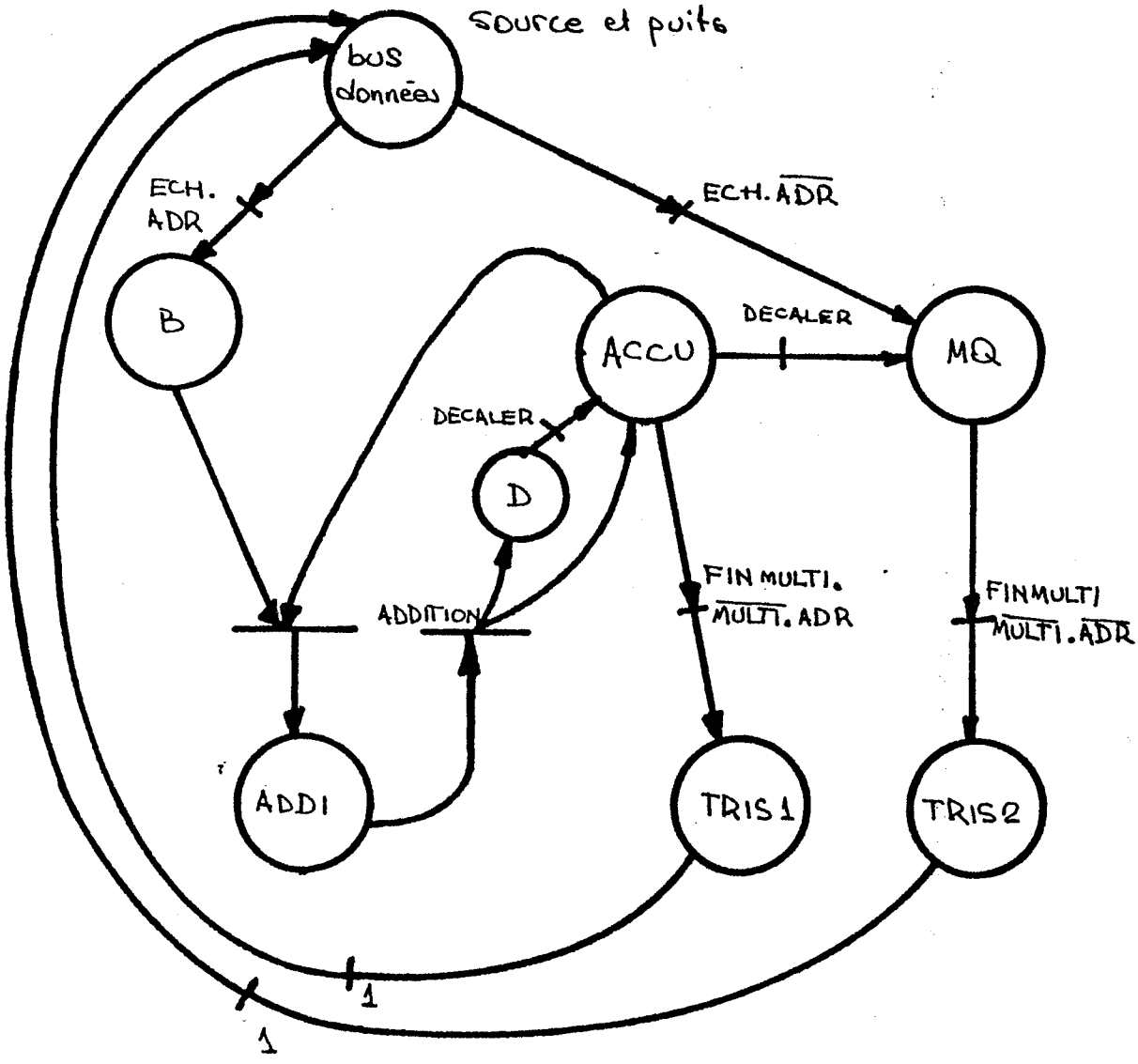


FIGURE II-3 : MODELISATION DE LA P.O. DU MULTIPLIEUR PAR UN GRAPHE BIPARTI

II-4-1 Mise de la P.O. sous contrôle spécial de test :

On ajoute un point de test pour chaque commande sortant de la P.C. vers la P.O. (voir Fig. II-4). Les commandes extérieures MULTI et ADR rentrent par les plots normaux du circuit intégré.

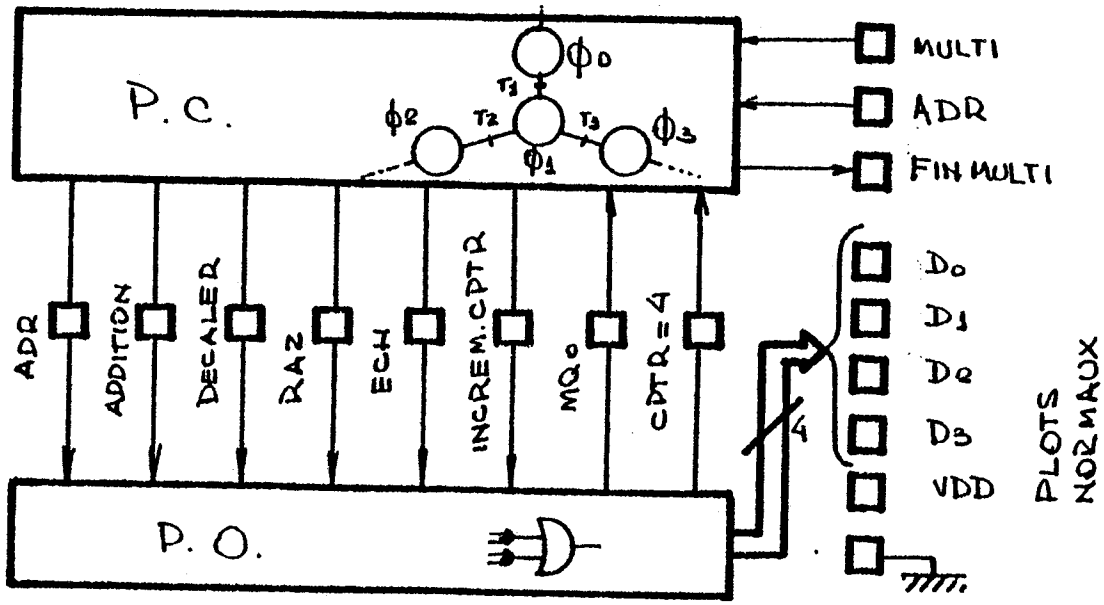


FIGURE II-4 a : POINTS DE TEST RAJOUTES ENTRE P.C. ET P.O.

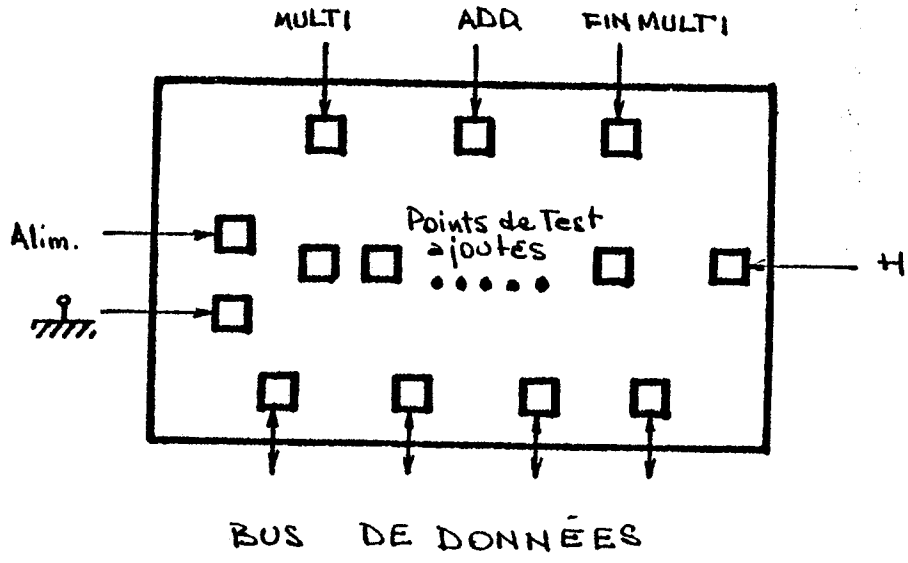


FIGURE II-4 b : LA PUCE ET SES PLOTS DE TEST

Le graphe modélisant ce circuit est toujours celui de la figure II-3.

Les écoulements sont :

$E_1 = \{$Bus données, B, D&ACCU, ADDI, TRIS1$\}$ $\quad T_{E1}=1/2$

$E_2 = \{$Bus données, MQ, TRIS2$\}$ $\quad T_{E2}=1$

$E_3 = \{$Bus données, B, ADDI, D&ACCU, MQ, TRIS2$\}$ $\quad T_{E3}=1/2$

$E_4 = \{$Tous les modules$\}$ $\quad T_{E4}=1/2.$

Indépendamment de toute stratégie de test, les blocs d'indiscernabilité sont :

$B_1 = \{$Bus données$\}$

$B_2 = \{$B, D&ACCU, ADDI$\}$

$B_3 = \{$MQ, TRIS2$\}$

$B_4 = \{$TRIS1$\}$.

La diagnosticabilité de ce circuit est donc 1/3.

On peut choisir de tester le circuit uniquement à l'aide des écoulements E_1 et E_2, qui couvrent tous les modules.

En utilisant uniquement E_1 et E_2 pour le test, la testabilité est :

$$T_{E1,E2} = \frac{1}{2+1} = \frac{1}{3}$$

Les blocs d'indiscernabilité sont :

$B_1 = \{$Bus données$\}$

$B_2 = \{$B, D&ACCU, ADDI, TRIS1$\}$

$B_3 = \{$MQ, TRIS2$\}$.

La diagnosticabilité est : 1/4.

Si de plus, on décide d'arrêter le test dès la première erreur détectée, les blocs d'indiscernabilité sont :

$B_1 = \{$Bus données, B, D&ACCU, ADDI, TRIS1$\}$

$B_2 = \{$Bus données, MQ, TRIS2$\}$.

La diagnosticabilité est : 1/5, la testabilité est toujours 1/3.

II-4-2 Adjonction de points de test pour l'observation:

On ajoute 5 points de test pour observer les sorties de l'additionneur ADDI (Fig. II-5).

Les écoulements sont :

$E'_1 = \{$ Bus données, MQ, TRIS2 $\}$ $\quad T_{E'1}=1$

$E'_2 = \{$ Bus données, B, D&ACCU, ADDI $\}$ $\quad T_{E'2}=1/2$

$E'_3 = \{$ Bus données, B, D&ACCU, ADDI, TRIS1 $\}$ $\quad T_{E'3}=1/2$

$E'_4 = \{$ Bus données, B, ADDI, D&ACCU, MQ, TRIS2 $\}$ $\quad T_{E'4}=1/2$

$E'_5 = \{$ Tous les modules $\}$ $\quad T_{E'5}=1/2.$

Indépendamment de toute stratégie de test, les blocs d'indiscernabilité sont :

$B'_1 = \{$ Bus données $\}$

$B'_2 = \{$ ADDI, B, D&ACCU $\}$

$B'_3 = \{$ MQ, TRIS2 $\}$

$B'_4 = \{$ TRIS1 $\}$.

L'adjonction des points d'observation n'a pas modifié la diagnosticabilité du circuit, ni sa testabilité.

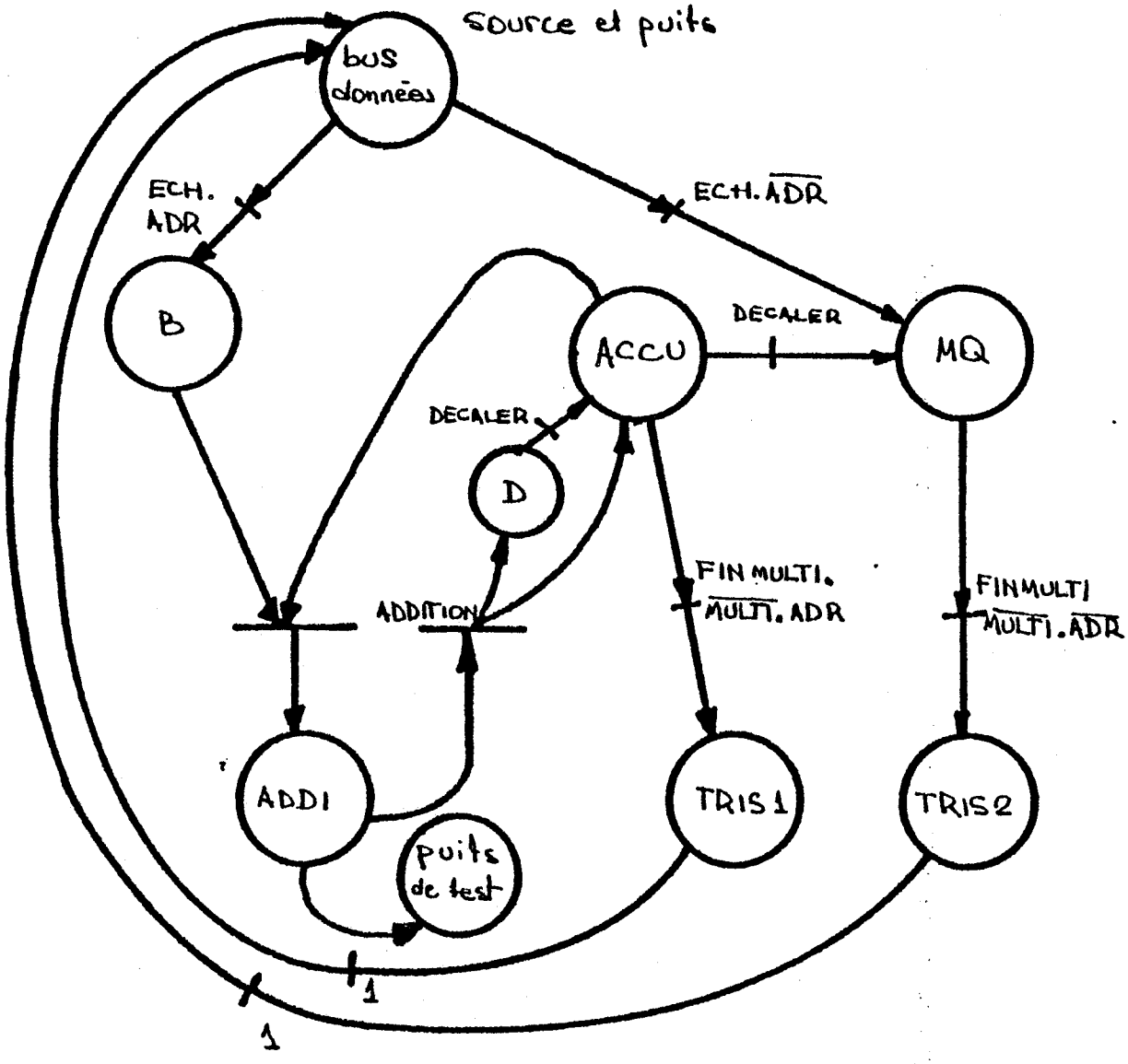


FIGURE II-5 : OBSERVATION DE LA SORTIE DE L'ADDITIONNEUR

II-4-3 Adjonction de points de test pour la commande et l'observation :

Les 5 points de test entre l'additionneur et l'accumulateur seront des plots d'E/S (FiG. II-6).

Les écoulements sont :

$E"_1 = \{$Bus données, MQ, TRIS2$\}$ $T_{E"1}=1$

$E"_2 = \{$Bus données, B, D&ACCU, ADDI$\}$ $T_{E"2}=1/2$

$E"_3 = \{$ACCU, TRIS1, bus données$\}$ $T_{E"3}=1$

$E"_4 = \{$ACCU, MQ, TRIS2, bus données$\}$ $T_{E"4}=1$

$E"_5 = \{$Bus données, B, ADDI, D&ACCU, MQ, TRIS2$\}$ $T_{E"5}=1/2$

$E"_6 = \{$Bus données, B, ADDI, D&ACCU, TRIS1$\}$ $T_{E"6}=1/2$

$E"_7 = \{$Tous les modules$\}$.

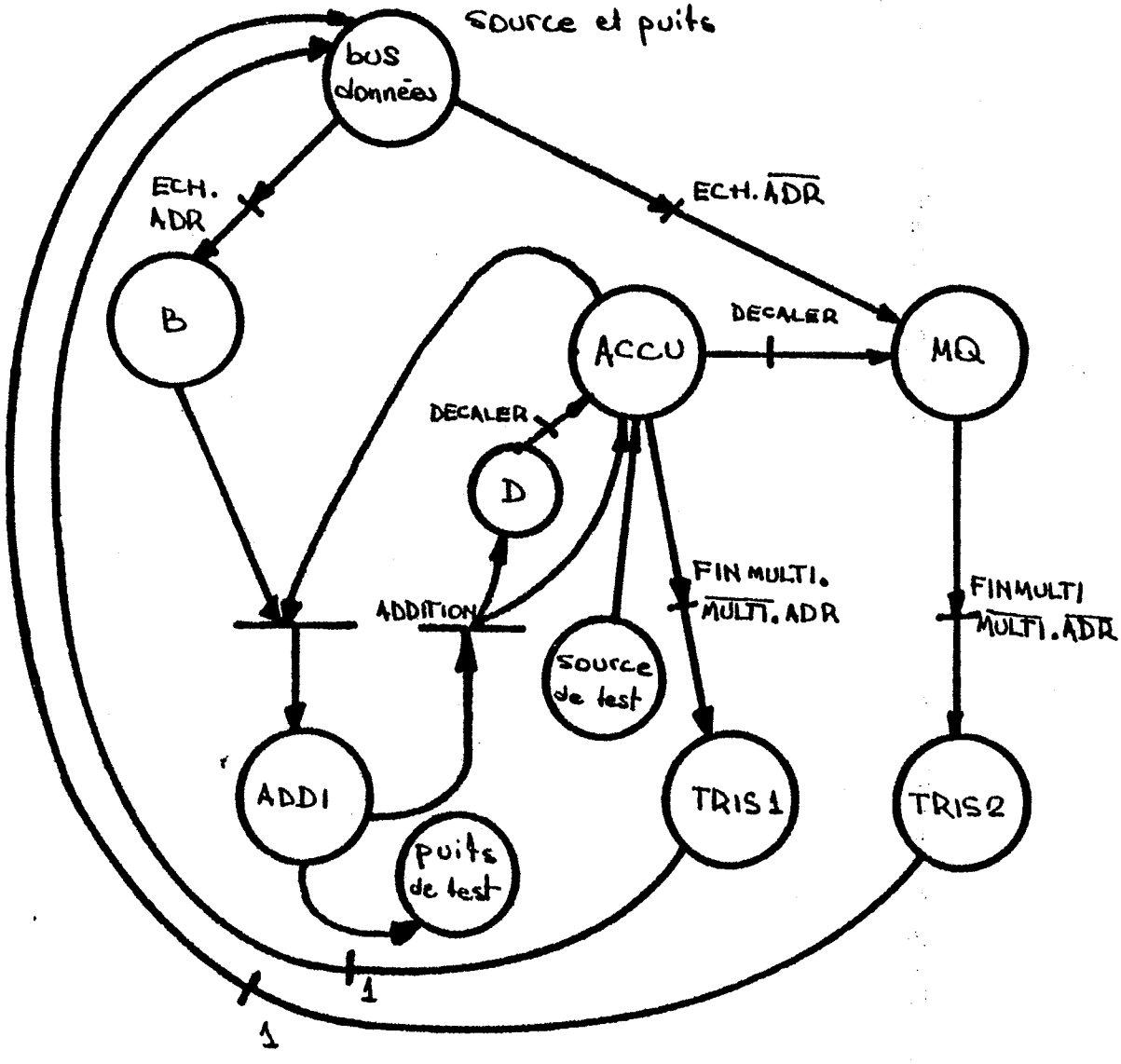


FIGURE II-6 : ADJONCTION DE POINTS DE COMMANDE ET D'OBSERVATION

Les blocs d'indiscernabilité intrinsèque sont :

$$B"_1 = \{\text{Bus données}\}$$

$$B"_2 = \{\text{ADDI, B}\}$$

$$B"_3 = \{\text{ACCU}\}$$

$$B"_4 = \{\text{TRIS1}\}$$

$$B"_5 = \{\text{MQ, TRIS2}\}.$$

La diagnosticabilité intrinsèque du circuit est 1/2.

La testabilité est :

- 1/3 en utilisant les écoulements $E"_1$ et $E"_6$, (diagnosticabilité 1/4).

- 1/5 en utilisant les écoulements $E"_1$, $E"_4$, $E"_2$ et $E"_3$, (diagnosticabilité 1/2).

II-5 REALISATION ET COUT DE POINTS DE TEST

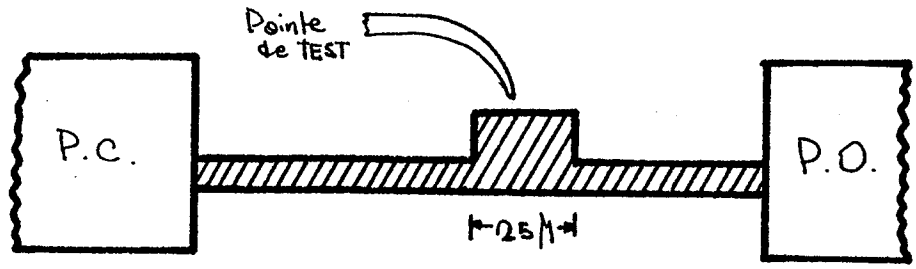
Le point de test ajouté le plus simple, consiste à grossir une métalisation jusqu'aux dimensions qui la feront accessible par les pointes de la machine de test en laboratoire (Fig. II-7 a). Dans certains cas on s'intéresse au forçage d'un "1" logique sur un certain fil de contrôle et on utilise le montage de la Fig. II-7 b.

Dans le cas d'un circuit programmable à travers son bus de données (on a un cas similaire dans la réalisation du CNET Chapitre III) il est possible de disposer d'un bit sur le registre de mode d'opération (*) qui met le circuit en "mode TEST(i)" et qui rend indépendante la P.O. des contrôles émis par la P.C. en mettant la P.O. sur contrôle spécial de test (Fig. II-7 c).

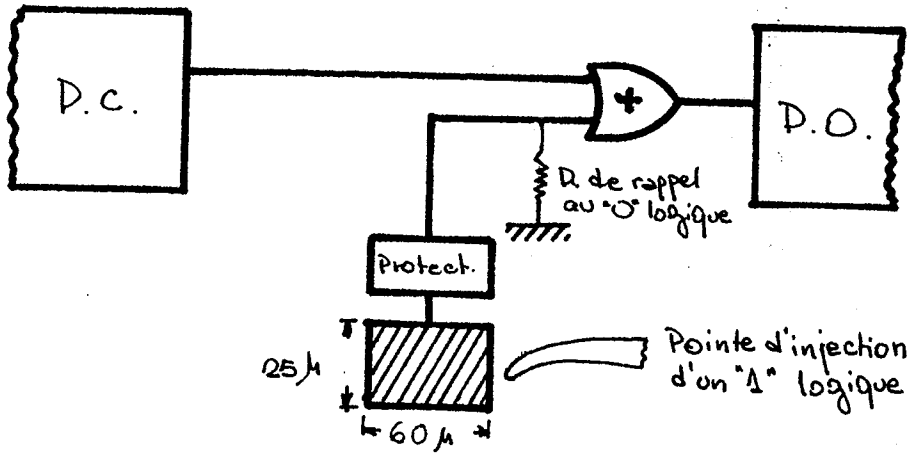
Certains points de test ajoutés peuvent devenir plots définitifs pour le test sur le circuit. La première solution serait d'ajouter une broche pour ce point, pour faire le test sur boftier, mais elle est trop chère.

On peut alors penser à utiliser ce point pour le test sur plaquette avec la machine à pointes pour le test automatique en fin de production. On a éliminé la broche, mais la surface du plot dépassivé de test et son environnement, reste la même que pour les plots normaux.

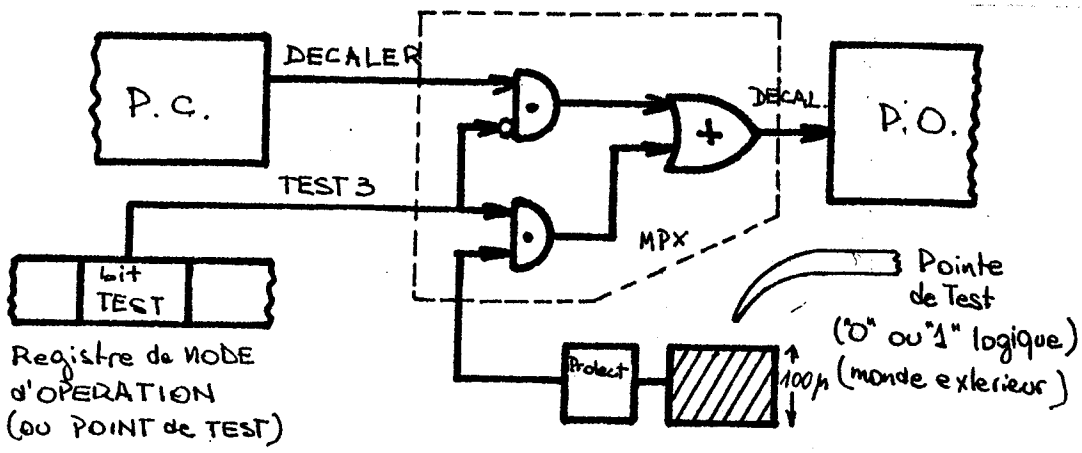
(*) NOTE : On pourrait faire de même avec une broche de test.



a) POINT DE TEST POUR LA LECTURE D'UN SIGNAL



b) FORCAGE D'UN "1" LOGIQUE



c) POINT POUR CONTROLE SPECIAL DE TEST

FIGURE II-7 : POINTS DE TEST POSSIBLES

Finallement on peut réduire la surface du plot de test si on pense l'utiliser pour <u>la mise au point en laboratoire</u> seulement : mise au point sur des machines de test avec micro-mouvements des pointes.

<u>Adjonction des plots de test sur le bus interne de données</u> :

Supposons qu'il soit possible de placer un plot de test à chaque fil du bus interne de données sur le point éloigné géographiquement le plus possible des plots normaux d'E/S (Fig. II-8). On peut la faire avec de simples grossissements des métalisations du bus, la surface restant très limitée.

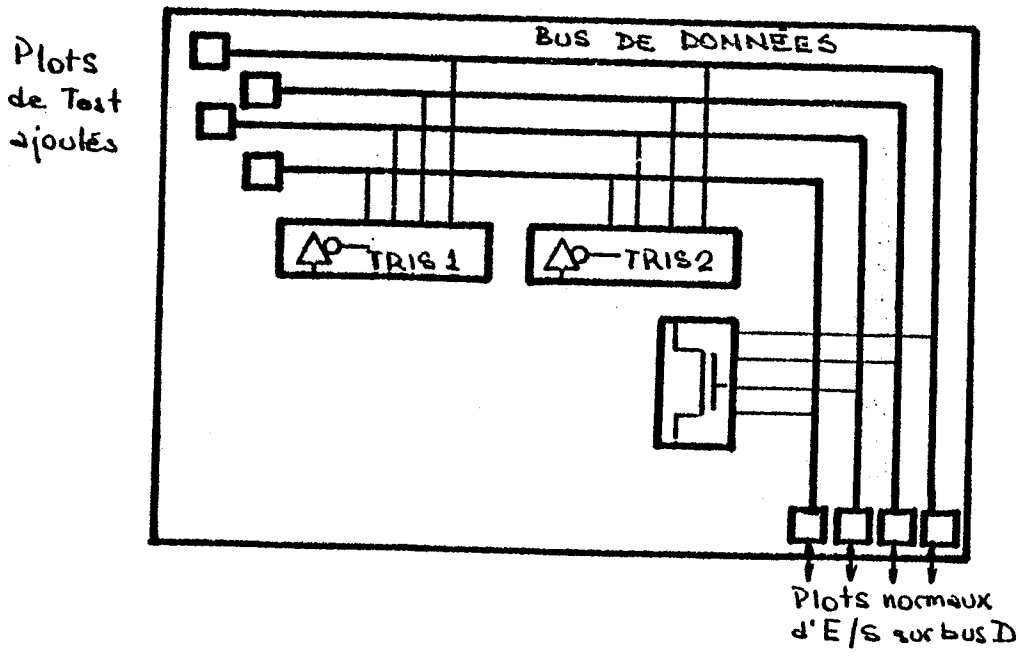


FIGURE II-8 : PLOTS DE TEST SUR LE CHEMIN DE DONNEES

Ces plots nous permettront de vérifier directement le bus de données et ainsi modifier les graphes de testabilité (Fig. II-9). On suppose qu'avec le contrôle spécial de test, il est possible de mettre les sorties sur le bus en 3ème état, en laissant ainsi de hautes impédances sur le bus de données.

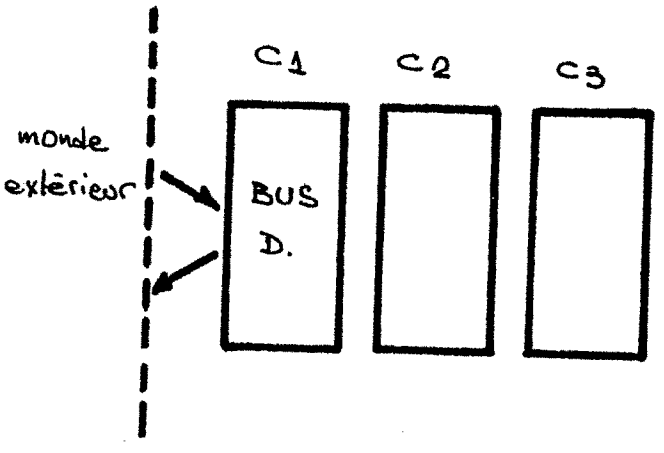


FIGURE II-9 : ECOULEMENTS MODIFIES PAR L'ADJONCTION DE PLOTS SUR BUS DE DONNEES

Le placement de points de test près de la périphérie de la puce dépend fortement du type d'implantation faite (par tranches de bits, par blocs de fonctions, etc) et de la méthode utilisée (dessin symbolique ou dessin au micron, etc). De toutes façons il y aura des contraintes géométriques à respecter :

- l'écartement minimal entre plots de test et plots normaux ;
- l'écartement entre plot de test et l'élément actif le plus proche ;
- le nombre de points de test, dépendant de l'angle disponible pour chaque pointe sur la carte à pointes de la machine de test en fin de production (ou de l'angle pour pointes avec micro-mouvements pour la machine de test de laboratoire). Mais surtout le nombre de points qu'on peut ajouter dépend de la possibilité d'augmentation de la surface du circuit intégré.

Pour l'exemple traité dans ce chapitre, nous avons :

Nombre de plots "normaux" du circuit = 10

Surface occupée par un plot "normal" = 125 x 125 = 0,015 mm^2.

Il faut ajouter la surface occupée par l'environnement du plot :

- les protections si le plot est une entrée;
- la surface occupée par les amplificateurs s'il s'agit d'une sortie.

Supposons qu'on ait choisi la solution qui consiste à tester le circuit en laboratoire (machines de test dotées de pointes avec micro-mouvements), ce qui nous amène à des plots plus petits.

Solution :	Nbre de plots ajoutés	Surface Si ajoutée :
II-4-1	6	$6x(60 \times 60 + \Delta)$
II-4-2	6 + 5	$11x(60 \times 60 + \Delta)$
II-4-3	6 + 5	$11x(60 \times 60 + \Delta) + 5\Delta$

Δ = Surface associée au plot de test (écartement, protection, MPX, bit spécial test...).

II-6 BIBLIOGRAPHIE DU CHAPITRE II

(1) "Hardware description Levels and test for complex circuits"
C. BELLON, J.M. GOBBI, G. SAUCIER - DAC 81 - Nashville TENESSEE USA - 1981.

(2) "Pannes dans les réseaux acycliques" - Thèse de 3ème cycle.
A. VERDILLON - Université de Grenoble - 23.12.1972.

(3) "Conception de séquenceurs locaux"
P. AMBLARD, G. SAUCIER - L'ONDE ELECTRIQUE - 1981.

(4) "Conception de séquenceur central de microprocesseurs"
C. BELLON, G. SAUCIER - AFCET RAIRO - 1981.

(5) "Test et testabilité de systèmes informatiques" Thèse d'Etat
C. ROBACH - Institut National Polytechnique de GRENOBLE - 1979

(6) "An information measure on nets. Application to the testability of digital systems" -) IFAC Workshop.
P. CASPI, A. MILI, C. ROBACH - 1977.

(7) "Contribution au test des circuits intégrés logiques"
J. CAILLAT - Thèse de 3ème cycle
Institut National Polytechnique de Grenoble - 1976.

(8) "Test du multi-enregistreur du centre de commutation E10"
J.L. RAINARD, A. VERDILLON -
Rapport de fin de contrat ENSIMAG - Juin 1975.

C H A P I T R E III

UN CIRCUIT FACILEMENT TESTABLE

ET PARTIELLEMENT AUTOTESTABLE

III-1 OBJECTIF DU CIRCUIT

III-1-1 Présentation générale :

Nous présentons ici l'étude et la réalisation que nous avons faite au Centre National d'Etudes des Télécommunications CNET-GRENOBLE d'un circuit intégré réalisant la détection et la correction d'erreurs dans une mémoire dynamique. Ce circuit est conçu pour être inséré dans un système à microprocesseur de 16 bits.

Il réalise le codage de mots de 16 bits, à l'aide d'un code de Hamming modifié, capable de corriger une erreur affectant un seul bit, et de détecter des erreurs portant sur 2 bits. Les données corrigées sont automatiquement réécrites en mémoire.

Pour assurer une meilleure protection des mots mémoire inutilisés durant des périodes assez longues, le circuit peut profiter des cycles de rafraîchissement pour effectuer des vérifications périodiques de toute la mémoire.

La distinction entre les pannes franches et les erreurs fugitives ou "pannes douces" est effectuée par détection d'erreurs répétitives dans une même zone de la mémoire. Une telle erreur est alors signalée au processeur, accompagnée de sa localisation, afin de permettre une reconfiguration de la mémoire.

De plus, afin d'éviter qu'une panne du circuit de correction ne produise l'indisponibilité, voire la contamination de la mémoire, le circuit a été rendu sûr en présence de panne : Il détecte lui-même ses mauvais fonctionnements et envoie un message d'alarme au C P U (Processeur Central = "CPU"-) qui peut alors le placer dans un état déconnecté, permettant l'utilisation normale de la mémoire (sans correction des erreurs éventuelles). Un effort particulier a été fait pour le test de ce Circuit Intégré de 48 broches d'E/S. (Technologie NMOS, canal 6 microns et grille Si).

III-1-2 Motivations de cette étude :

- Pannes "douces" de la mémoire.

Du fait de leur structure interne, les mémoires dynamiques, de haute densité d'intégration souffrent d'un type de panne particulier : L'information est mémorisée dans des capacités de valeur très petite (typique : 0,04 pF) qui peuvent être déchargées accidentellement par particules α par exemple (1).

Des études récentes ont démontré que la plus grande partie de ces particules sont émises par des éléments radioactifs qui existent dans le matériel d'encapsulation. Les rayonnements cosmiques peuvent aussi provoquer certaines perturbations, particulièrement dans les systèmes embarqués, qui travaillent à grande altitude (2).

Les fabricants de circuits intégrés suggèrent quelques solutions technologiques pour améliorer la situation : utilisation de boftiers peu radioactifs, blindage de la puce et étude de technologies ou de cellules moins sensibles. Mais le problème n'a pas été éliminé en appliquant ces techniques et de nouveaux composants pour contrôler les erreurs de mémoire viennent d'être annoncés (3), (4). Ces composants peuvent utiliser des codes détecteurs et correcteurs d'erreur type Hamming classique (5), capables de corriger les erreurs qui affectent seulement un bit du mot gardé en mémoire.

Dans le cas de 2 bits erronés, ces codes ne peuvent qu'indiquer une erreur irrécupérable.

Cependant, vu que la plus grande partie des mémoires de haute densité sont fabriquées avec une organisation de boftiers par mots de un bit, la possibilité de corriger une erreur simple paraft suffisante pour offrir une bonne protection contre ces mécanismes d'erreurs fugitives : une particule α peut affecter seulement un boftier, donc un bit de mot codé.

- Pannes franches de la mémoire.

Il n'y a pas que des erreurs fugitives. Les mémoires, comme n'importe quel autre composant électronique, peuvent avoir des pannes franches. Une telle panne qui affecte un seul boftier de la mémoire, peut seulement produire des erreurs corrigibles.

Il est possible d'ajouter certaines fonctions intéressantes au circuit classique nécessaire pour mettre en oeuvre le codage de Hamming : par exemple, la discrimination entre pannes "douces" et pannes franches, et la localisation du boîtier de mémoire endommagé peuvent s'avérer très utiles. Il est aussi possible de profiter des cycles de rafraîchissement pour dérouler une vérification périodique de la mémoire complète.

- Pannes du circuit de contrôle.

Remarquons que les pannes qui affectent le circuit de contrôle d'erreurs peuvent avoir un effet beaucoup plus sérieux : Les pannes internes au circuit de détection et de correction peuvent induire des corrections impropres de mots exacts.

La mémoire complète peut alors être contaminée par le circuit dont le but est de protéger l'information stockée. C'est pour cela que le circuit de contrôle d'erreurs est un des points critiques pour la sûreté de fonctionnement de systèmes de mémoires, les autres points sont le multiplexeur d'adresses, le générateur d'horloges et le contrôleur de rafraîchissement.

Le problème de sécurité dû au module de contrôle d'erreurs peut être résolu en rendant ce circuit sûr en présence de panne : en cas de faute interne, le circuit peut détecter son propre mauvais fonctionnement et il peut se déconnecter des bus, en permettant alors un fonctionnement normal mais non protégé de la mémoire.

III-2 FONCTIONS REALISEES PAR LE CIRCUIT

III-2-1 Détection et correction des erreurs mémoire :

Ce circuit est destiné à être inséré entre une mémoire dynamique et un microprocesseur (μP) 16 bits. Sa fonction est de gérer un code correcteur type Hamming capable de corriger les erreurs portant sur un bit seulement, et de signaler la présence des erreurs affectant 2 bits.

Ce code nécessite l'adjonction de 6 bits par mot de 16 bits à protéger. La mémoire devra donc être organisée en mots de 22 bits (Cf. Fig. III-1). On suppose que chaque bit appartenant à un mot codé (22 bits) est stocké dans un boîtier différent (organisation par tranches de bits).

Trois bus sont utilisés pour connecter le système de mémoire, le circuit de contrôle d'erreurs (CCE) et le microprocesseur : Les bus normaux de données et d'adresse plus un bus supplémentaire, dit bus de "Hamming", qui connecte seulement la mémoire et le CCE.

Tous les signaux de contrôle (*) sont envoyés par le microprocesseur vers le CCE, qui renvoie de nouveaux signaux de contrôle vers la mémoire.

(*) Demande mémoire = "mémoire request" = $\overline{MR}_{\mu P}$

Lire = "read" = $\overline{RD}_{\mu P}$

Ecrire = "Write" = $\overline{WR}_{\mu P}$

Rafraîchissement = "Refresh" = $\overline{RFSH}_{\mu P}$

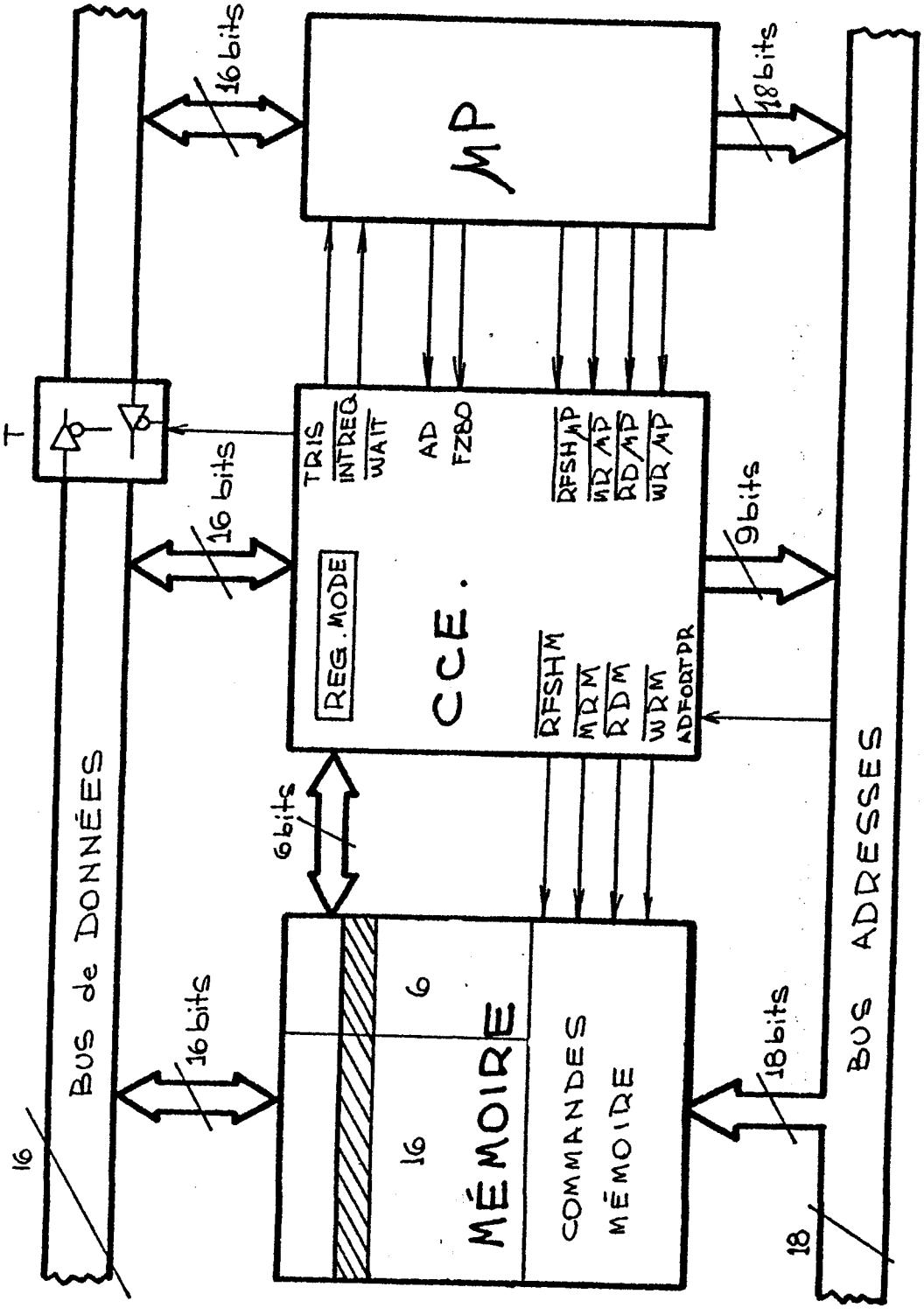


FIGURE III-1 : UTILISATION DU CIRCUIT CORRECTEUR D'ERREUR

Le circuit travaille systématiquement par lecture-correction-réécriture : L'information (16 bits) est lue dans la mémoire accompagnée de ses 6 bits de codage. L'ensemble est ensuite vérifié par le circuit, éventuellement corrigé, puis réémis sur le bus de données, pour être lu par le microprocesseur et simultanément réécrit dans la mémoire.

Cette opération est généralement possible durant le temps de l'accès mémoire des microprocesseurs NMOS. Les commandes reçues par la mémoire sont donc différentes de celles émises par le microprocesseur, et doivent être régénérées par le circuit de correction en fonction de l'opération demandée par le microprocesseur.

Ce mode de fonctionnement semble le plus efficace car il ne nécessite pas l'interruption du microprocesseur en cas d'erreur. En contrepartie, il fait courir davantage de risques à l'information en cas de panne du système de correction : toute la mémoire risque de se voir polluée de façon irréversible. C'est pourquoi il est utile de prévoir l'autotest de la fonction de correction, de façon à la mettre hors-service en cas de panne.

En cas de panne interne du CCE, une alarme est envoyée au microprocesseur, lequel peut forcer le CCE dans un état déconnecté. Les sorties de données du CCE sont alors placées en état de haute impédance et les signaux de contrôle arrivant du microprocesseur sont renvoyés vers la mémoire sans modifications.

III-2-2 Vérification périodique de la mémoire :

Le mode de correction exposé ci-dessus a l'inconvénient de ne corriger que les mots mémoire adressés par le programme en cours de déroulement. Or, certaines données peuvent n'être utilisées que très rarement. Dans ce cas, la probabilité d'apparition d'erreurs doubles affectant un même mot devient non négligeable. Le code n'étant pas capable de les corriger, il est utile d'effectuer périodiquement la lecture complète de toute la mémoire à des intervalles suffisamment rapprochés. Cette opération peut être effectuée sans perte de temps en profitant des cycles de rafraîchissement nécessaires aux mémoires dynamiques.

Actuellement les mémoires dynamiques à grande capacité sont organisées en matrices de "l" lignes et "c" colonnes (c ou l sans rapport avec la taille du mot adressable) : les lignes sont adressées par les poids forts de l'adresse, les poids faibles servant à sélectionner une colonne.

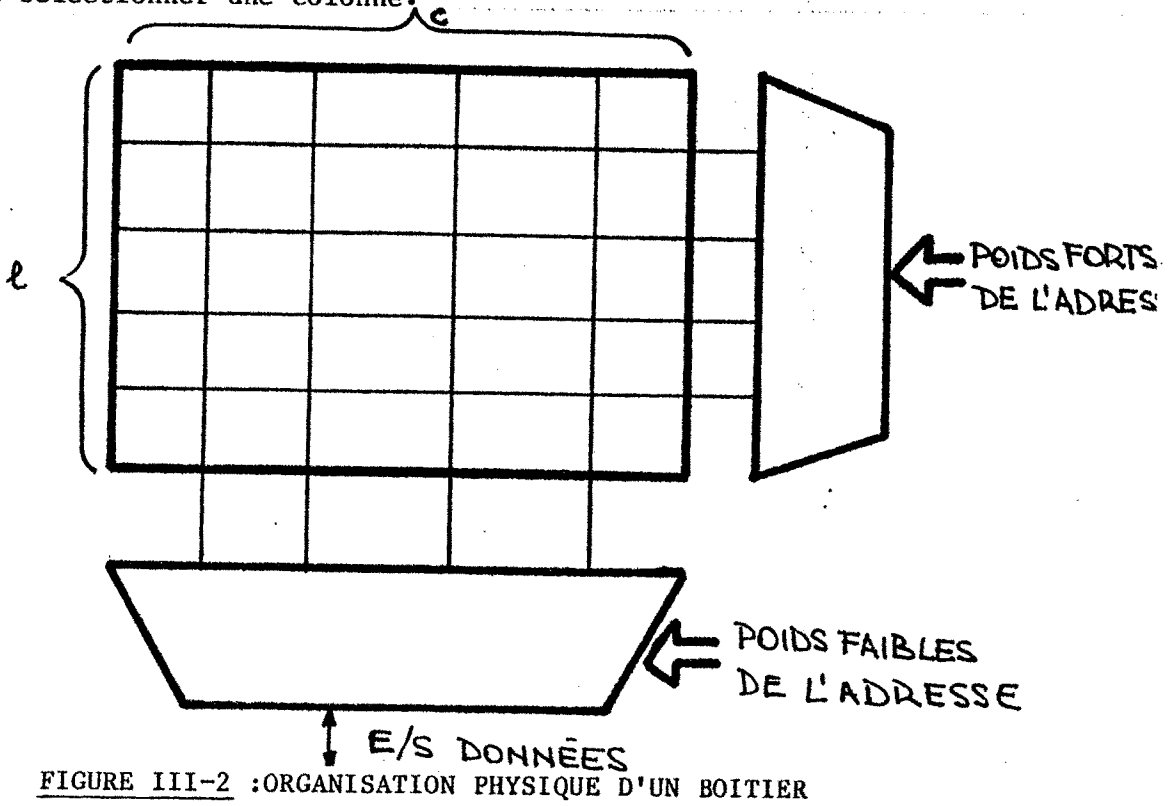


FIGURE III-2 :ORGANISATION PHYSIQUE D'UN BOITIER

Le rafraîchissement (régénération de niveaux corrects dans les capacités de stockage des cellules mémoires) s'effectue ligne par ligne : la lecture d'un bit régénère toute la ligne à laquelle il appartient.

Quelques microprocesseurs disponibles sur le marché, comme le Z8000 par exemple, fournissent un cycle spécial pour le rafraîchissement de la mémoire. Dans ce cas, l'utilisation des opérations de rafraîchissement pour vérifier les données gardées en mémoire, peut être facilement mise en place.

Le système de rafraîchissement ne générant que les bits de poids forts de l'adresse, le circuit de vérification devra pouvoir compléter cette adresse à l'aide d'un compteur incrémenté à la fin de chaque cycle de rafraîchissement.

Prenons le cas d'une carte mémoire organisée de la façon suivante :

Capacité : M mots de 16 bits.

Boîtiers utilisés : Capacité N mots de 1 bit

Nombre de lignes internes L

Période maximale de rafraîchissement t_r

L'intervalle maximum entre deux vérifications consécutives du même bit est donné par la formule :

$$T_r = \frac{M.t_r}{L}$$

Par exemple, si M = 64 K, L = 256; t_r = 5ms

On obtient T_r = 1,3 sec

Ce système de vérification périodique a l'avantage de permettre une détection plus rapide de pannes franches de la mémoire et en plus d'éviter l'accumulation d'erreurs fugitives sur des données faiblement utilisées.

III-2-3 Détection et localisation des pannes franches (mémoire) :

Il est nécessaire d'effectuer la distinction entre les pannes franches ("hard error") (bit de la mémoire en mauvais fonctionnement définitif) et les erreurs fugitives ("soft error") (décharge accidentelle d'une capacité de stockage par une particule α par exemple) de façon à réparer la mémoire dans le premier cas (reconfiguration...).

Il est nécessaire d'éviter l'accumulation d'erreurs dans la mémoire parce que les erreurs multiples, tout en étant possibles, peuvent dépasser la capacité de correction et même de détection du codage utilisé.

LES ERREURS SYSTEMATIQUES sont dues à une panne franche dans la mémoire : un ou plusieurs circuits ont cessé de fonctionner correctement. Chaque fois que l'on viendra stocker des informations dans les zones touchées, elles risquent d'être affectées d'erreurs. L'exemple typique serait le collage à "0" d'un bit en mémoire (il devient impossible de stocker et de récupérer un "1"), ou bien l'impossibilité d'accéder à une ligne de la mémoire (par suite d'une panne du système de décodage). Ces erreurs se répètent toujours au même endroit et dans les mêmes conditions. Elles correspondent à une altération matérielle du composant.

LES ERREURS FUGITIVES, au contraire, n'impliquent pas un mauvais fonctionnement permanent du composant. Dans les mémoires dynamiques, elles sont surtout causées par des particules de type α ou des rayonnements qui viennent décharger les capacités de stockage, sans provoquer de dommages permanents. L'information présente au moment de l'arrivée de la particule est détruite, mais toute information stockée ultérieurement n'a pas de raison particulière d'être erronée.

Ces erreurs peuvent affecter n'importe quel point mémoire, à n'importe quel moment, et sont non corrélées entre elles.

Cependant, les pannes franches d'un circuit mémoire ne sont pas toujours simples à mettre en évidence, et peuvent induire des erreurs "pseudo aléatoires" difficiles à distinguer des erreurs fugitives. On a cherché à rendre le circuit de vérification (CCE) aussi transparent que possible pour le microprocesseur qui ne sera alerté (par interruption) que dans les cas "graves". Cela suppose que le circuit soit capable de détecter la présence d'une panne à caractère répétitif, puis d'en prévenir le microprocesseur. Ce dernier placera alors le circuit (CCE) en mode surveillance : toutes les erreurs seront alors signalées au processeur, qui pourra procéder à une étude plus fine.

La mémoire est supposée organisée en tranche de bits. Chaque bit du mot stocké en mémoire provient d'un circuit intégré différent (les grosses mémoires dynamiques : 16K, 32K, 64K, 256K, sont d'ailleurs organisées généralement par "mots" de 1 bit).

Une panne d'un boîtier n'affectera donc que 1 bit d'un mot à la fois (d'où une efficacité maximale du code correcteur). Examinons la possibilité de distinguer les erreurs dues à des pannes franches des erreurs fugitives grâce à un système simple permettant de mémoriser les localisations (syndromes) des n dernières erreurs constatées, et émettant une alarme si elles sont les mêmes.

Une panne franche peut être distinguée d'une erreur fugitive du fait qu'elle ne peut être corrigée : une telle panne induira une série d'erreurs à la même adresse. Il suffirait donc de stocker les adresses des erreurs détectées et de déclencher l'alarme lorsqu'un nombre "suffisant" d'erreurs sont détectées consécutivement à la même adresse. Cependant :

- le stockage d'une adresse de 16 bits n'est pas économique ;
- une panne franche peut affecter plusieurs bits d'un même boîtier, rompant ainsi la répétitivité des adresses.

C'est pourquoi il est plus réaliste de ne stocker que le numéro du boîtier incriminé (n° du bit faux).

Dans ces conditions, un autre problème se fait jour : une particule α, par exemple, peut très bien provoquer plusieurs erreurs fugitives dans un même boîtier, amenant une alarme "panne franche".

Ce problème peut être résolu en ne stockant que l'adresse de la première erreur détectée dans chaque cycle de vérification de la mémoire complète (les autres erreurs étant corrigées seulement). Les erreurs fugitives correspondant à une particule α seront corrigées en deux cycles au plus. Il suffit de n'envoyer l'alarme qu'après trois répétitions au moins du même syndrôme pour éviter une fausse alarme.

Dans une vérification périodique de la mémoire, le taux d'erreurs fugitives sera extrêmement faible comparé au taux d'erreurs générées par une panne franche. Ainsi nous pouvons construire le système de détection de pannes franches avec une pile "FIFO" de trois éléments, chargeables avec les trois derniers emplacements d'erreurs. Si le contenu des trois registres est le même, une alarme est envoyée (INTREQ).

En réalité on garde dans chacun des trois registres le "syndrome" de l'erreur détectée, qui nous permet d'identifier le numéro de bit faux dans le mot mémoire (le boîtier dans notre exemple). Ces syndromes sont systématiquement comparés, la coincidence déclenche la demande d'interruption et c'est alors que le microprocesseur peut venir lire le syndrome incriminé et décider de la suite à donner. Par exemple, il pourra envoyer un petit programme de test pour vérifier la zone incriminée de la mémoire.

JUSTIFICATION STATISTIQUE DU MECANISME DE PILE ET DE SA PROFONDEUR (n = 3) :

On cherche ici à justifier le mécanisme de pile et sa profondeur adoptée.

Ce mécanisme est rudimentaire et ne permet pas toujours de discriminer l'origine des syndromes : panne franche ou erreur fugitive.

C'est pourquoi il ne se justifie que statistiquement en s'appuyant sur les taux probables de pannes franches et fugitives suivantes :

panne franche : $\lambda_0 = 10^{-7} h^{-1}$ par boîtier

erreur fugitive : $\lambda_1 = 10^{-5} h^{-1}$ par boîtier

durée de mission = T = 5 jours.

Il faut d'abord justifier que ce mécanisme va permettre de détecter quasi instantanément une panne franche dès qu'elle se manifeste. En effet, cette détection ne pourrait être différée d'une durée de plus de 3 cycles de vérification (Δ 3,9 sec) que si d'autres évènements : une panne franche ou une erreur fugitive sur d'autres boîtiers se produisaient pendant cette durée rendant les 3 syndrômes de la pile différents. Il est clair que ceci est très peu probable :

probabilité d'une erreur fugitive pendant 3,9 sec =

$\simeq 21 \times 10^{-5} \times h^{-1} \times 10^{-3} \times h = 2.1 \times 10^{-7}$.

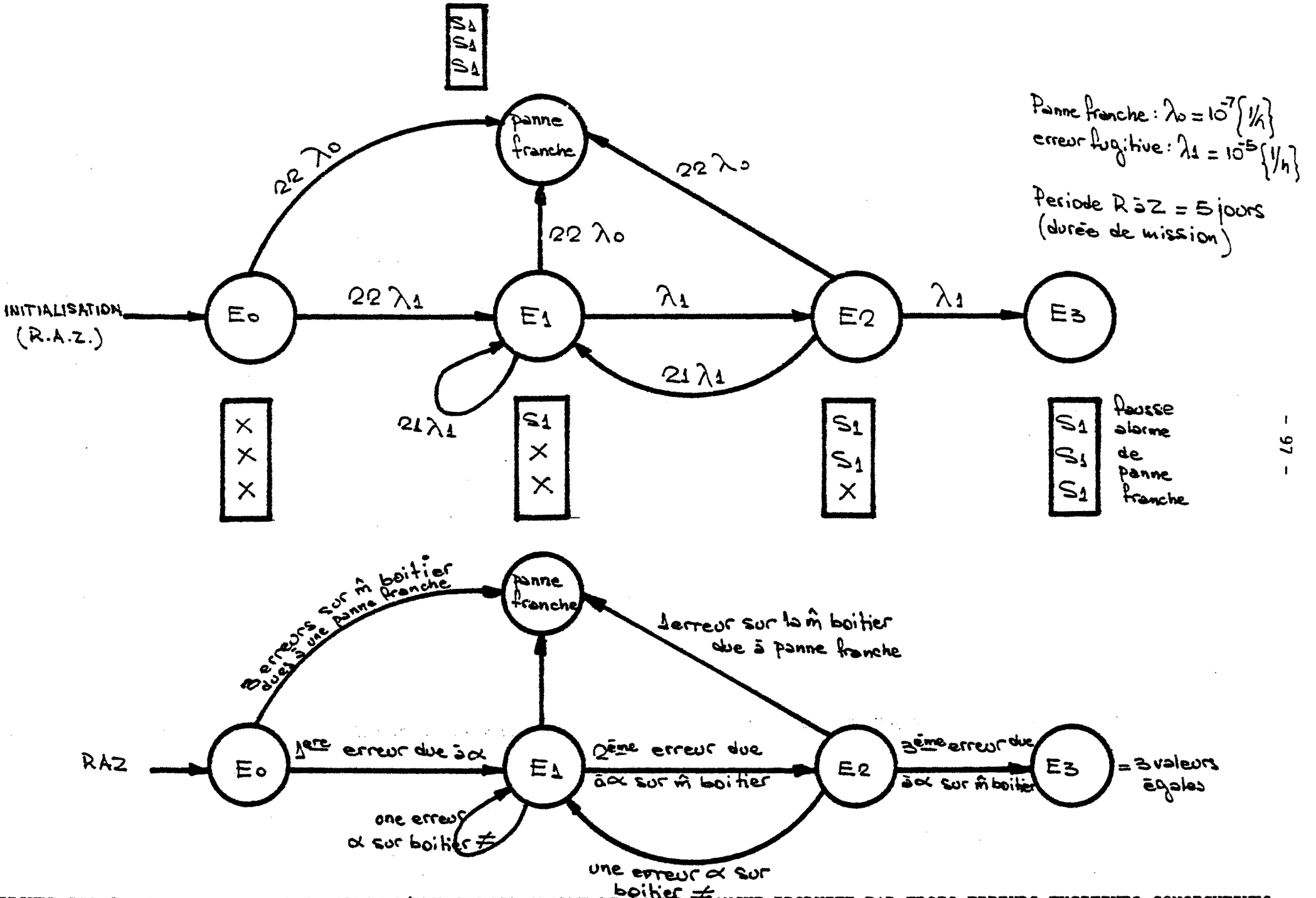


FIGURE III-3 : MODELISATION PAR MARKOV D'UNE FAUSSE ALARME DE PANNE FRANCHE PRODUITE PAR TROIS ERREURS FUGITIVES CONSECUTIVES

On considère donc par la suite que dès que une panne franche se produit dans un boîtier, elle sera détectée quasi certainement dans les 4 secondes suivantes.

Il reste à montrer qu'une succession d'erreurs fugitives pendant la durée d'une mission (5 jours) a un probabilité faible de conduire à une alarme correspondant à une panne franche.

Pour cela on propose la modélisation suivante par PROCESSUS de MARKOV.

Cinq états s'introduisent :

E_o : aucun syndrôme valide n'est stocké.

E_1 : le syndrôme de tête diffère de ses successeurs.

E_2 : les deux syndrômes de tête diffèrent du troisième.

E_3 : les trois syndrômes sont valides et égaux.

E_4 : panne franche détectée.

Le graphe résultant est donné à la figure III-3. Dans ce graphe, on a supposé qu'une panne franche était détectée instantanément, la durée de 3 cycles étant très faible vis à vis du problème traité (mission de 5 jours).

Il est clair que la probabilité de E_3 pour un temps de 5 jours correspond principalement à l'arrivée de 3 erreurs fugitives sur le même boîtier pendant cette durée de 5 jours (=T), les cas de figures correspondant à l'occurence d'un nombre supérieur de pannes ou d'erreurs étant négligeables.

On a donc :

$$\text{Probabilité de fausse détection} = P_{E3}(T) \cong \frac{22.\lambda_1^3.T^3}{6}$$

$$\text{Probabilité de détection de panne franche} \cong P_{E4}(T) = 22\ \lambda_0\ T$$

La comparaison montre que la probabilité de fausse détection est très faible pour une pile de taille 3, vis à vis de la probabilité de panne franche.

$$\frac{P_{E3}}{P_{E4}} \cong \frac{22.\lambda_1^3\ T^3}{6x22\ \lambda_0\ T} \cong \frac{\lambda_1^3\ T^2}{\lambda_0.6} \cong \frac{10^{-15} \times 14400}{10^{-7}\ .\ 6} = 2.4x10^{-5}$$

Si on avait pris une profondeur plus petite de la pile, n = 2, on aurait eu :

$$P_{E3} \cong 22\ \lambda_1^2\ T^2\ /\ 2.$$

$$\frac{P_{E3}}{P_{E4}} \cong \frac{\lambda_1^2\ T}{\lambda_0\ 2} \cong \frac{10^{-10} \times 120}{10^{-7} \times 2} = 0,06.$$

III-2-4 Modes de fonctionnement de ce circuit :

Le circuit de vérification de mémoire peut être placé par le microprocesseur dans divers modes de fonctionnement, par écriture d'un mot de commande dans un registre interne, à travers le bus de données. Sept modes d'opération sont programmables :

MODE NORMAL : Le circuit code vérifie et corrige les données passant sur le bus mémoire. En cas d'erreur non corrigible ou de panne interne au CCE ou de panne franche dans la mémoire, il émet une interruption vers le microprocesseur. Il corrige les erreurs fugitives sans les signaler.

MODE SURVEILLANCE : Périodiquement, ou en cas de doute sur le bon fonctionnement de la mémoire, le microprocesseur peut demander à se voir signaler toutes les erreurs, même corrigibles et non répétitives.

Dans ce mode là, l'indication d'erreurs fugitives au microprocesseur permet l'évaluation du taux de pannes dites "douces" de la mémoire.

MODE INITIALISATION : Il est prévu pour faciliter l'initialisation de la mémoire à la mise sous tension. En effet, le contenu de la mémoire n'est pas encore codé. Dans ce cas, le circuit n'émet pas d'interruption. Notons que le codage du contenu inital de la mémoire est fait de façon automatique par les cycles de vérification-rafraîchissement.

MODE "CYCLE LONG" : Ce mode permet de générer systématiquement un signal "WAIT", destiné à rallonger le cycle lecture-écriture dans le cas des mémoires plus lentes que les normales.

MODE BYTE : Ce mode permet de fonctionner par octet ("byte"), lors d'une écriture de 8 bits en mémoire à la place de mots de 16 bits.

Le problème apparaît avec les opérations d'écriture : le CCE lit en mémoire l'octet non-modifié du mot correspondant, de façon à en permettre le codage normal sur les 16 bits La gestion par octet du bus de données doit être assurée de l'extérieur.

MODE DECONNECTE : En cas de panne interne du CCE, le microprocesseur peut placer le circuit dans un état inopérant. Dans cet état, le microprocesseur peut continuer à travailler avec la mémoire qui n'est plus protégée.

MODE TEST : Ce mode est prévu pour faciliter le test du circuit, soit en test du boîtier nu, soit en cas de test périodique par le microprocesseur dans le système complet avec la mémoire.

Il existe un huitième mode de fonctionnement commandé directement à travers une broche, à la place du registre mode. Il s'agit du FETCH Z80 qui permet le traitement d'un cycle de lecture raccourci correspondant à la lecture d'une instruction par un microprocesseur du type Z80.

III-2-5 Messages d'erreur :

Les messages émis vers le microprocesseur le sont par l'intermédiaire du bus de données. En forçant AD à 1, le microprocesseur peut venir lire le contenu des registres de sortie qui contiennent le syndrome de la dernière erreur constatée et sa nature qui peut être :

a) erreur simple en mémoire ;

b) erreur multiple en mémoire ;

c) erreur répétitive en mémoire ;

d) erreur de transmission en entrée ;

e) erreur interne.

De plus, une réplique de la demande d'interruption est jointe au message d'erreur, principalement pour éviter le blocage du microprocesseur en cas du collage de la demande d'interruption à l'état actif. En cas de désaccord, le microprocesseur pourra masquer cette interruption douteuse ou mettre le circuit en mode déconnecté.

III-3 FICHE TECHNIQUE DU CIRCUIT CORRECTEUR D'ERREURS (C.C.E.)

III-3-1 Fonction des broches :

5V	Alimentation
0V	Alimentation
	Horloge maftresse du système à microprocesseur
$\overline{MR_{\mu P}}$	Demande d'accès mémoire, émis par le microprocesseur
$\overline{MR}$memo	Demande d'accès mémoire, émis par CCE en réponse à $\overline{MR_{\mu P}}$
$\overline{WR_{\mu P}}$	Top d'écriture en mémoire, émis par le microprocesseur
$\overline{WR}$memo	Top d'écriture en mémoire, émis par CCE
$\overline{RD_{\mu P}}$	Nature de l'opération mémoire (écriture/lecture), émise par le microprocesseur
$\overline{RD}$memo	Nature de l'opération mémoire, émise par CCE
$\overline{RFSH_{\mu P}}$	Cycle de rafrafchissement de la mémoire, émis par le microprocesseur ou le système de rafrafchissement
$\overline{RFSH}$memo	$\overline{RFSH_{\mu P}}$ retransmis tel quel en cas de travail en mode déconnecté
$\overline{TRIS}$	Commande des amplificateurs de "trois états" du bus de données. Ces "trois états" ne sont nécessaires que si l'on utilise les possibilités de test pendant le rafrafchissement ou de travail en mode "byte" - Signal émis par CCE
AD1-9	Partie "poids faibles" du bus adresse (utilisée sur CAS dans la mémoire)
$\overline{AD\text{fort}_{\mu P}}$	Bit de poids le plus fort du bus d'adresse (utilisé sur RAS)

D1-16	Bus de données
H1-6	Bus Hamming
FZ80	Demande de cycle court, émise par le microprocesseur
$\overline{\text{WAIT}}$	Demande d'un cycle "WAIT" au microprocesseur, émise par CCE
$\overline{\text{INTREQ}}$	Demande d'interruption au microprocesseur émise par CCE.

III-3-2 Horloges :

Le circuit est conçu pour s'adapter au séquencement des accès mémoire des principaux microprocesseurs actuels. Examinons le cas des principales opérations possibles :

- la lecture ;
- l'écriture ;
- le rafraîchissement.

LECTURE : Séquencement d'une lecture mémoire normale (Fig. III-4). Le microprocesseur commence par envoyer les adresses sur le bus d'adresses, puis fait retomber le signal $\overline{MR}/\mu P$, ce qui provoque l'échantillonnage successif des deux moitiés de l'adresse à l'intérieur de la mémoire (signaux RAS et CAS des mémoires dynamiques). La mémoire émet alors les données sur le bus (disponible car $\overline{RD}/\mu P = 0$), où elles peuvent être échantillonnées par le microprocesseur.

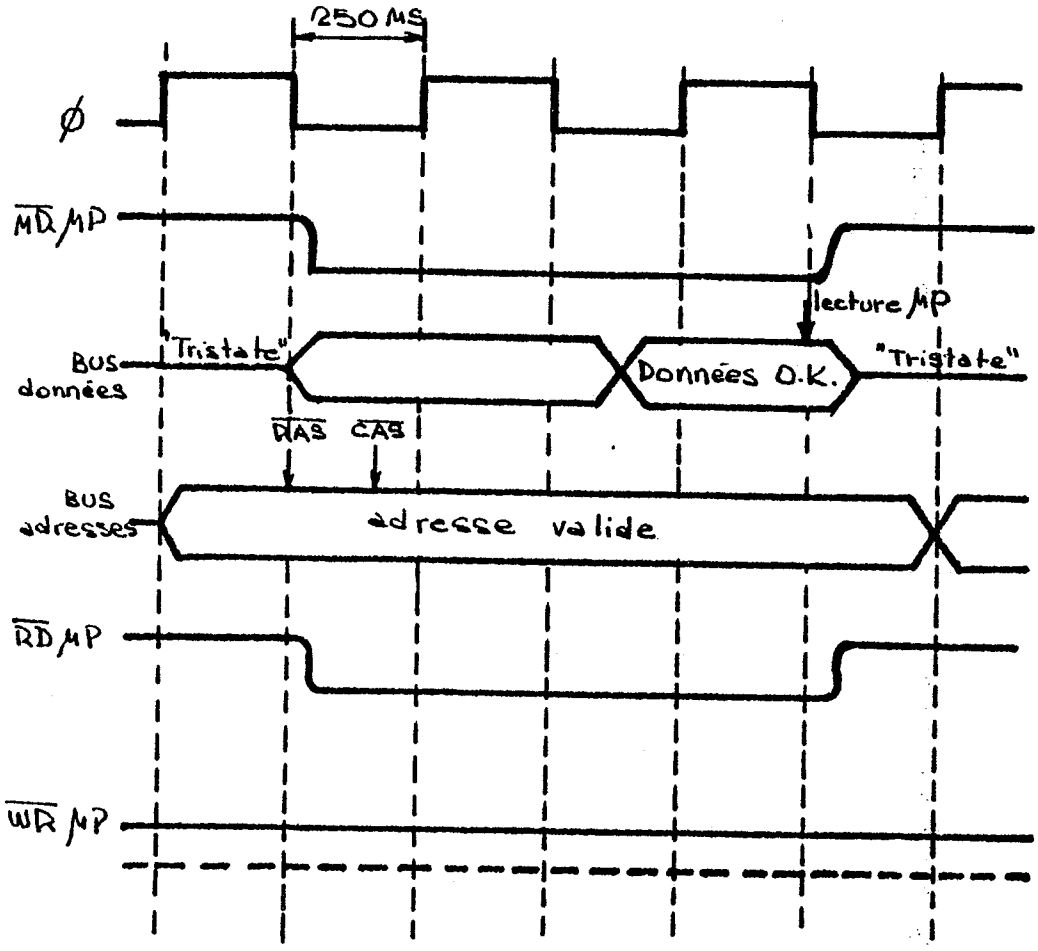


FIGURE III-4 : CYCLE LECTURE NORMAL

La Fig. III-5 montre ce qui se passe lorsque le circuit de correction est utilisé. Le cycle de lecture débute normalement : le MRmemo suit le MR P, les données sont émises sur le bus. Elles sont alors disponibles en entrée du système de vérification qui les corrige éventuellement et les stocke dans ses registres internes. Le cycle de lecture est alors transformé en cycle d'écriture, indépendamment des commandes mémoire émises par le microprocesseur. Les données corrigées sont donc réécrites dans la mémoire, et, simultanément, échantillonnées par le microprocesseur.

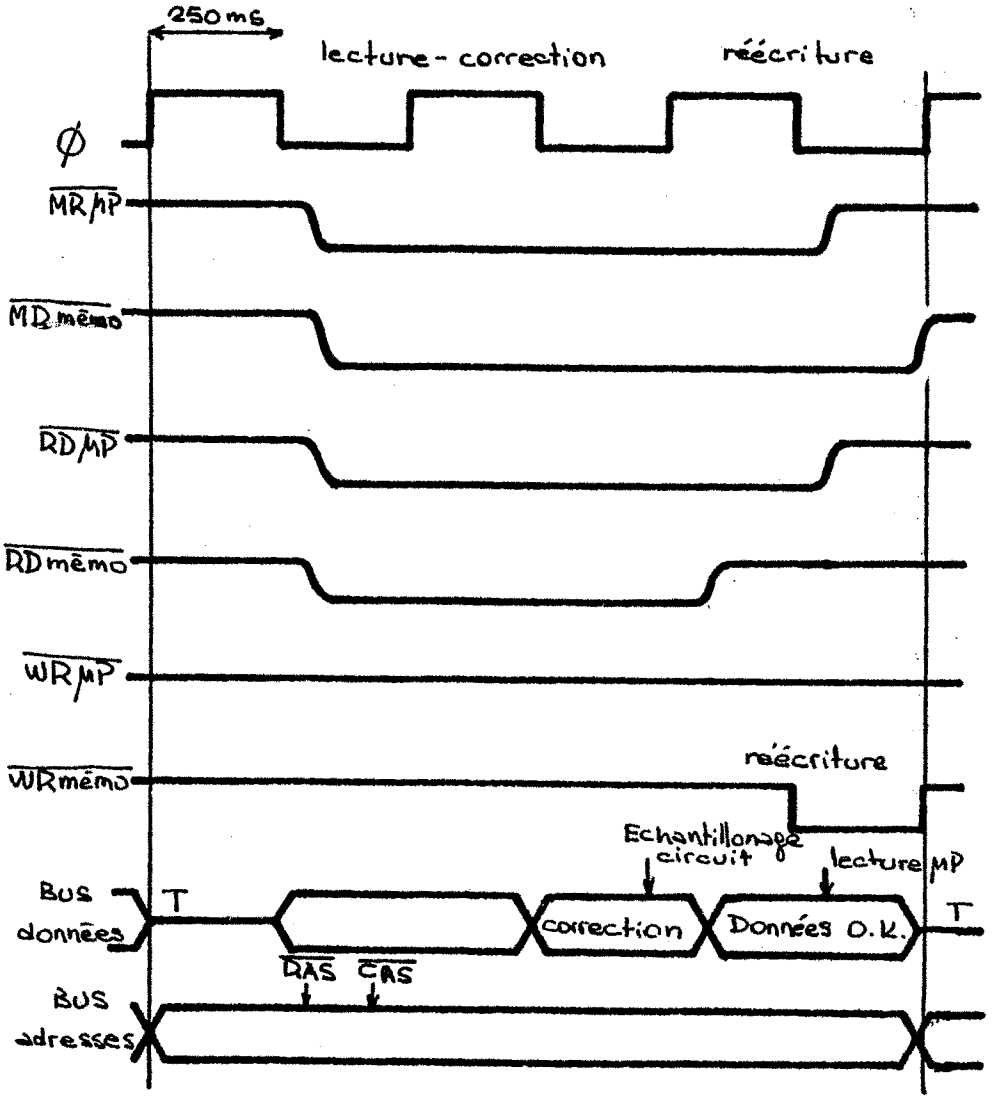


FIGURE III-5 : CYCLE LECTURE-CORRECTION-REECRITURE

ECRITURE : Les chronogrammes d'une écriture normale sont données en Fig. III-6 : le mot mémoire adressé est sélectionné par la retombée de MR P, mais cette fois le microprocesseur émet les données qui sont échantillonnées dans la mémoire au passage à "0" (zéro) de WR P.

Si l'on interpose le circuit correcteur, le cycle est légèrement modifié. Le WRmemo est retardé légèrement par rapport au WR P pour laisser au CCE le temps d'effectuer le codage (Cf. Fig.III-7).

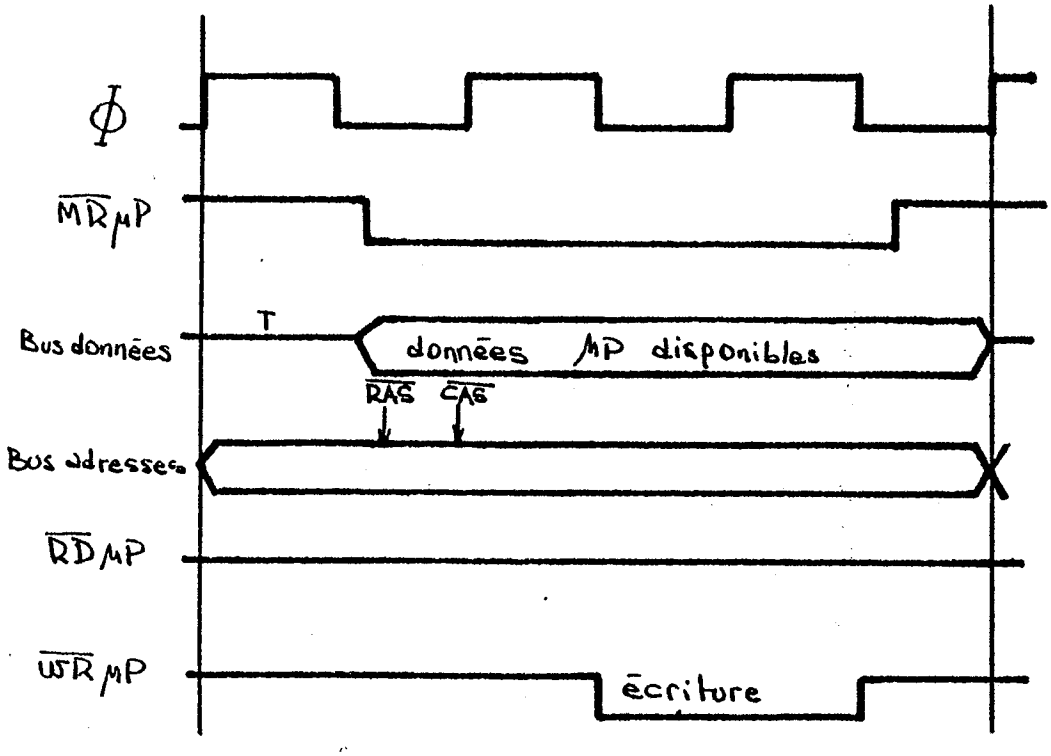


FIGURE III-6 : CYCLE D'ECRITURE NORMALE

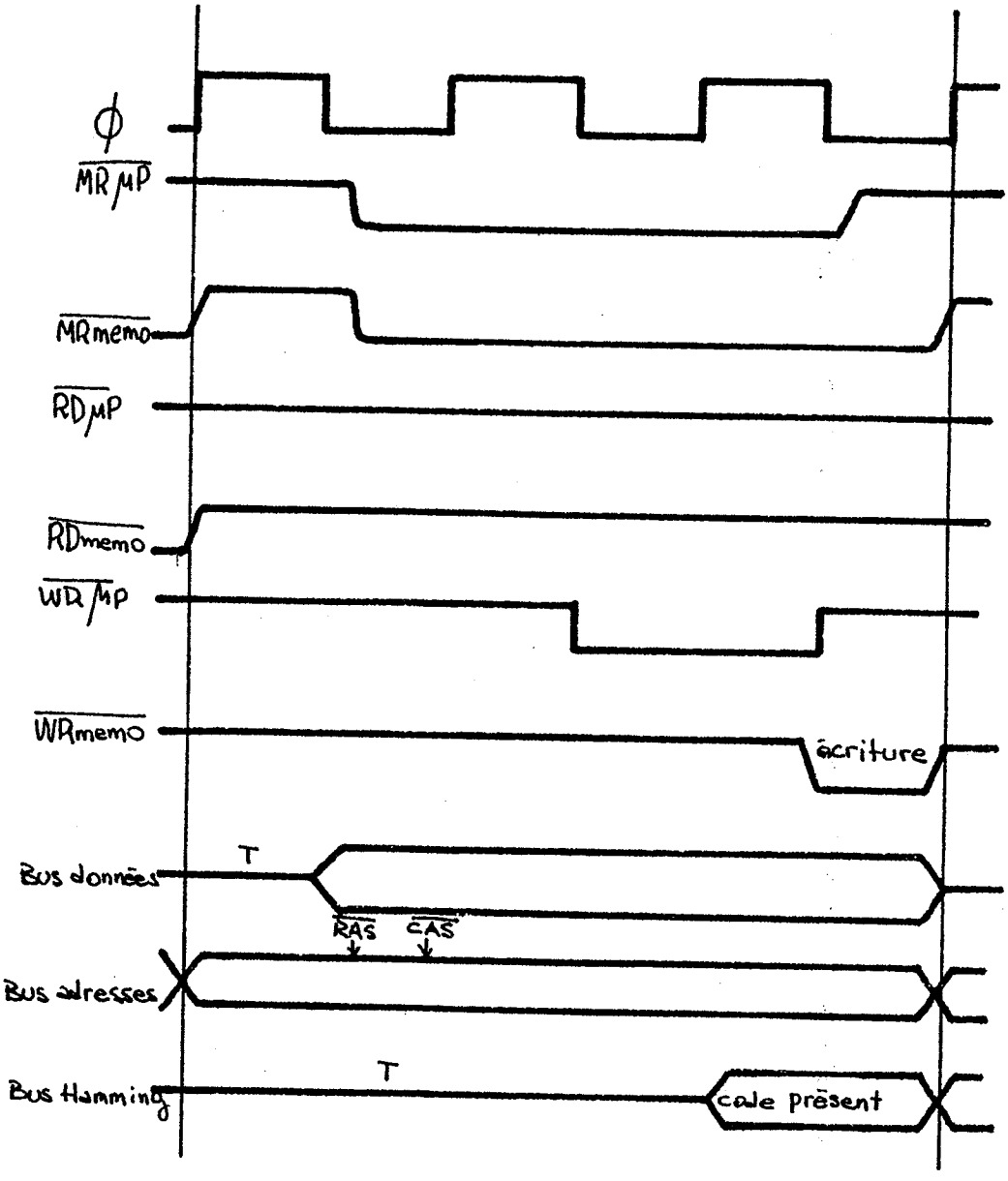


FIGURE III-7 : CYCLE D'ECRITURE MODIFIE

RAFRAICHISSEMENT : Certains microprocesseurs (Z80, Z8000) peuvent prendre eux- mêmes en charge le rafraîchissement de leur mémoire : ceci permet de profiter du temps de décodage de l'instruction pour effectuer le rafraîchissement. Le circuit CCE a été conçu pour fonctionner dans ce cas. Il reçoit (du microprocesseur ou bien du système de rafraîchissement) un signal $\overline{RFSH}/\mu P$ qui lui indique le démarrage (ou la possibilité) d'un cycle de rafraîchissement. Le microprocesseur émet les bits d'adresse de poids fort sur le bus d'adresse accompagné d'un $\overline{MR}/\mu P$ qui lance le rafraîchissement d'une ligne de la mémoire (Fig. III-2 et III-8).

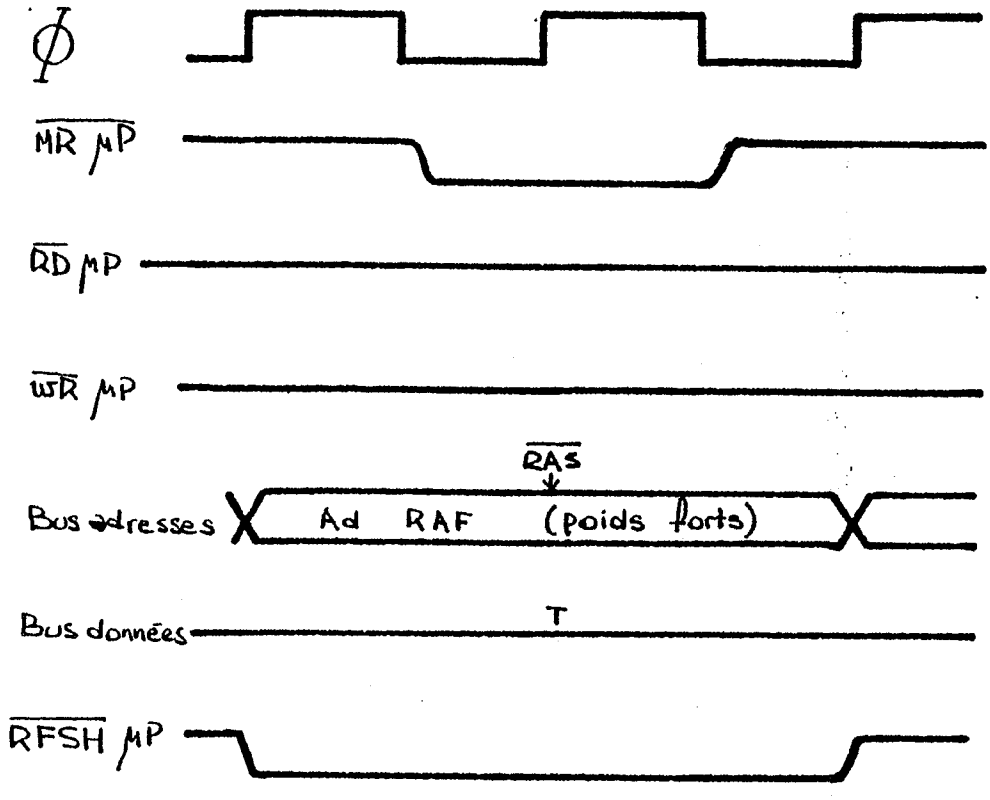


FIGURE III-8 : CYCLE DE RAFRAICHISSEMENT DU Z80

Le circuit CCE va profiter de ce cycle de rafraîchissement pour effectuer la vérification et éventuellement la correction d'un mot mémoire. Le temps alloué au cycle de rafraîchissement est suffisant pour effectuer la vérification de la donnée. Si elle est correcte (cas de la Fig. III-9), il n'y a pas de traitement particulier à effectuer.

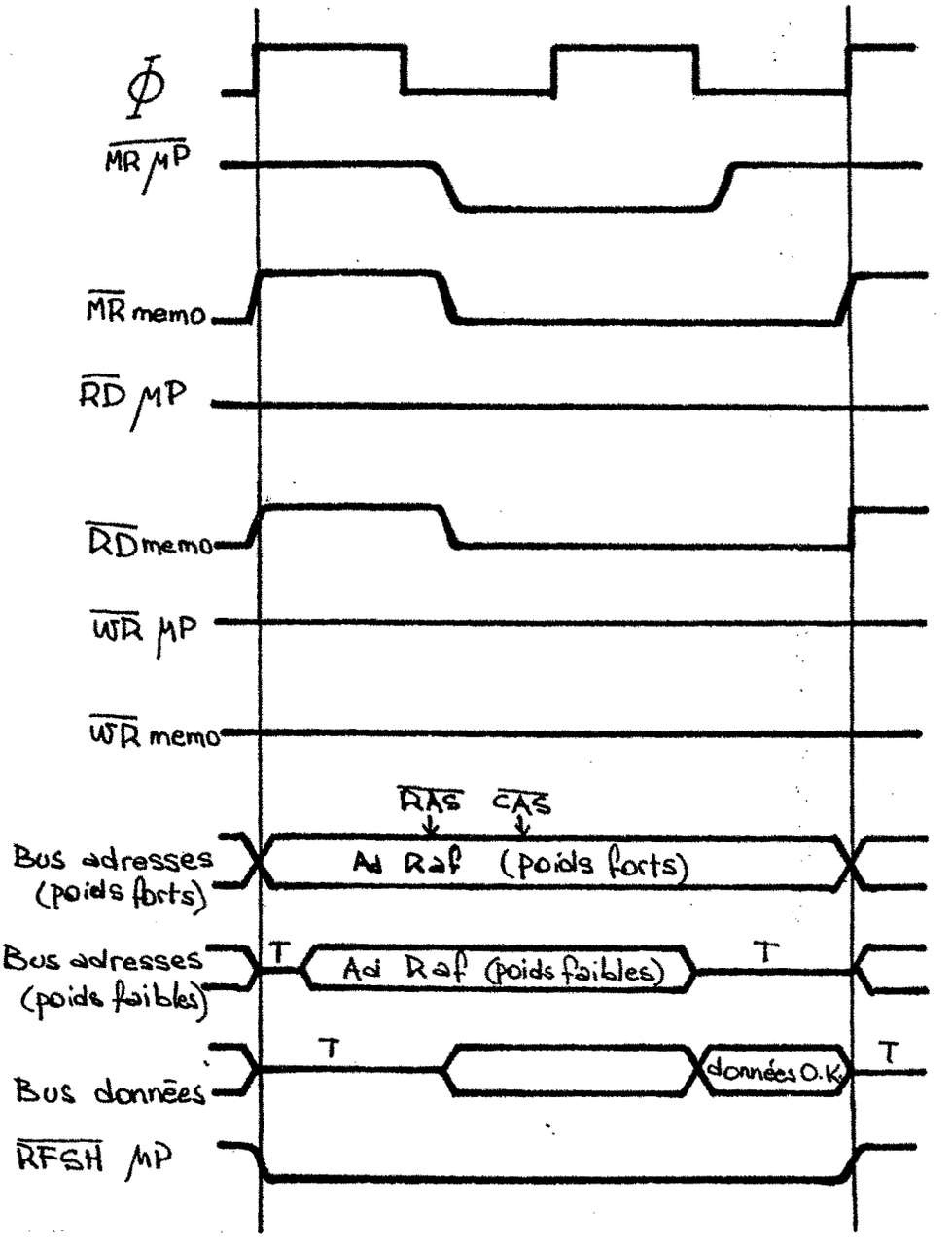


FIGURE III-9 : CYCLE DE RAFRAICHISSEMENT SANS ERREUR

Dans le cas contraire, il faut corriger la donnée en mémoire, et pour cela il est nécessaire d'arrêter le microprocesseur pendant le temps nécessaire pour effectuer la modification. La Fig. III-10 nous montre ce qui se passe dans ce dernier cas, le rafrafchissement étant suivi, par exemple, d'un cycle d'écriture : le MR P du cycle d'écriture n'est pas pris en compte (il causerait l'échantillonnage d'une nouvelle adresse par la mémoire).

Un signal TRIS permet de bloquer un système "trois états" interposé entre le bus mémoire et le bus microprocesseur (Fig. III-1), de façon à éviter le chevauchement des données corrigées émises par le CCE et des données émises par le microprocesseur. La réécriture terminée, le front de descente de génère un nouveau MRmemo qui initialise un cycle d'écriture ordinaire. Pour resynchroniser le microprocesseur, il est nécessaire de lui envoyer un signal WAIT, afin de regagner le cycle perdu par la correction.

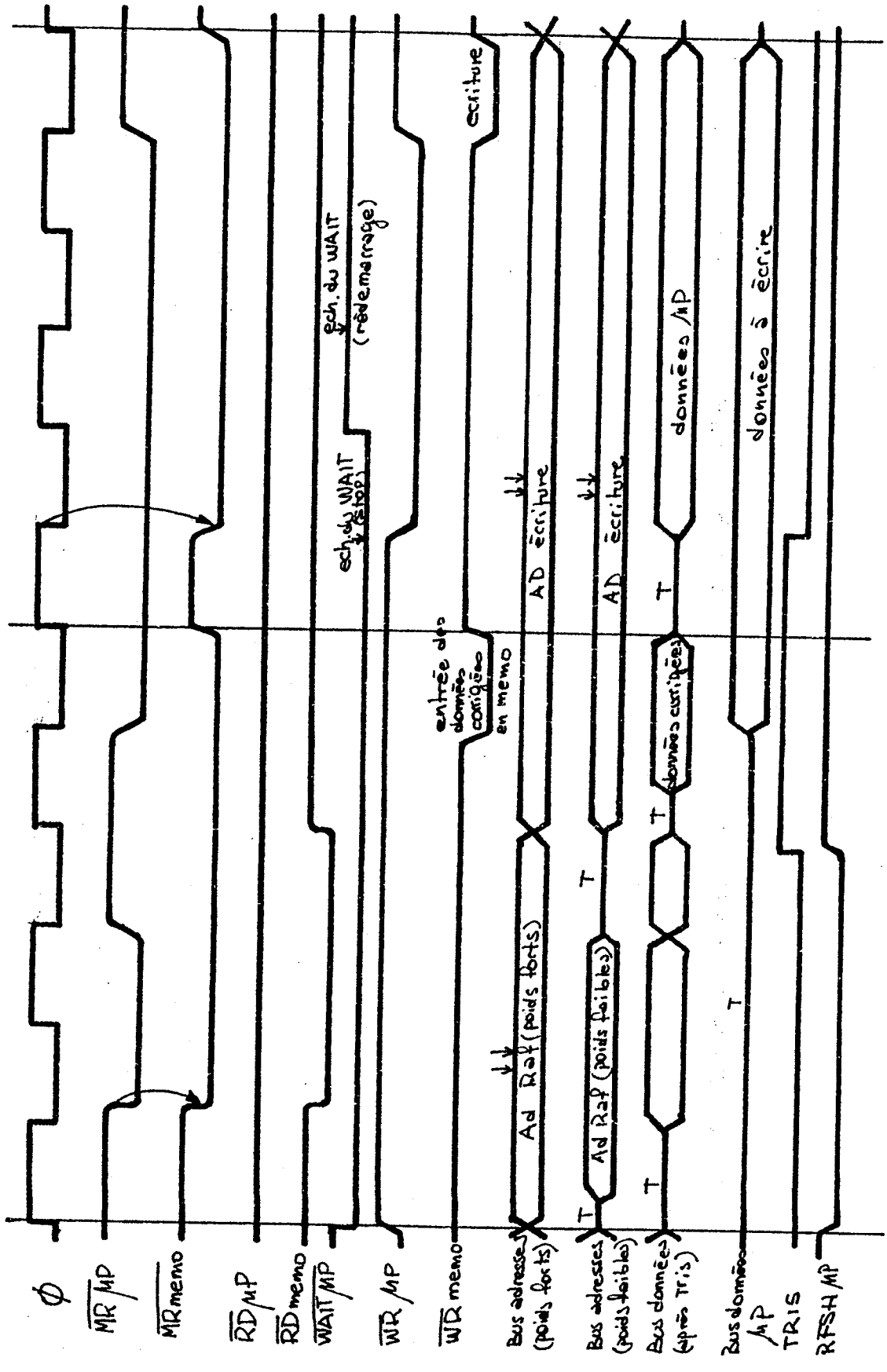


FIGURE III-10 : CYCLE DE RAFRAICHISSEMENT

AVEC ERREUR SUIVI D'UNE ECRITURE

III-3-3 Séquencement :

En plus des exemples exposés, il y a d'autres possibilités de cycles légèrement différents : Cycles de lecture, d'écriture et de rafraîchissement en mode "WAIT", cycle de lecture raccourci (type FZ80), cycle d'écriture en mode byte, cycles de chargement ou de lecture des registres internes. Ils sont le plus possible construits sur le cycle de lecture-écriture type.

Le séquencement obéit au graphe de la Fig. III-11. L'automate possède 8 états. R est l'état de repos.

Les cycles possibles sont :

R → 1 → 2 → 3 → 4 → 5 → 6 → 7 → R { E/L/RAF erreur avec WAIT

R → 1 → 2 ——→ 5 → 6 → 7 → R { RAF avec erreur
{ E/L sans WAIT

R → 1 → 2 ——→ 5 ——→ R { RAF sans erreur et sans WAIT

R → 1 → 2 → 3 → 4 → 5 ——→ R { RAF avec WAIT sans erreur.

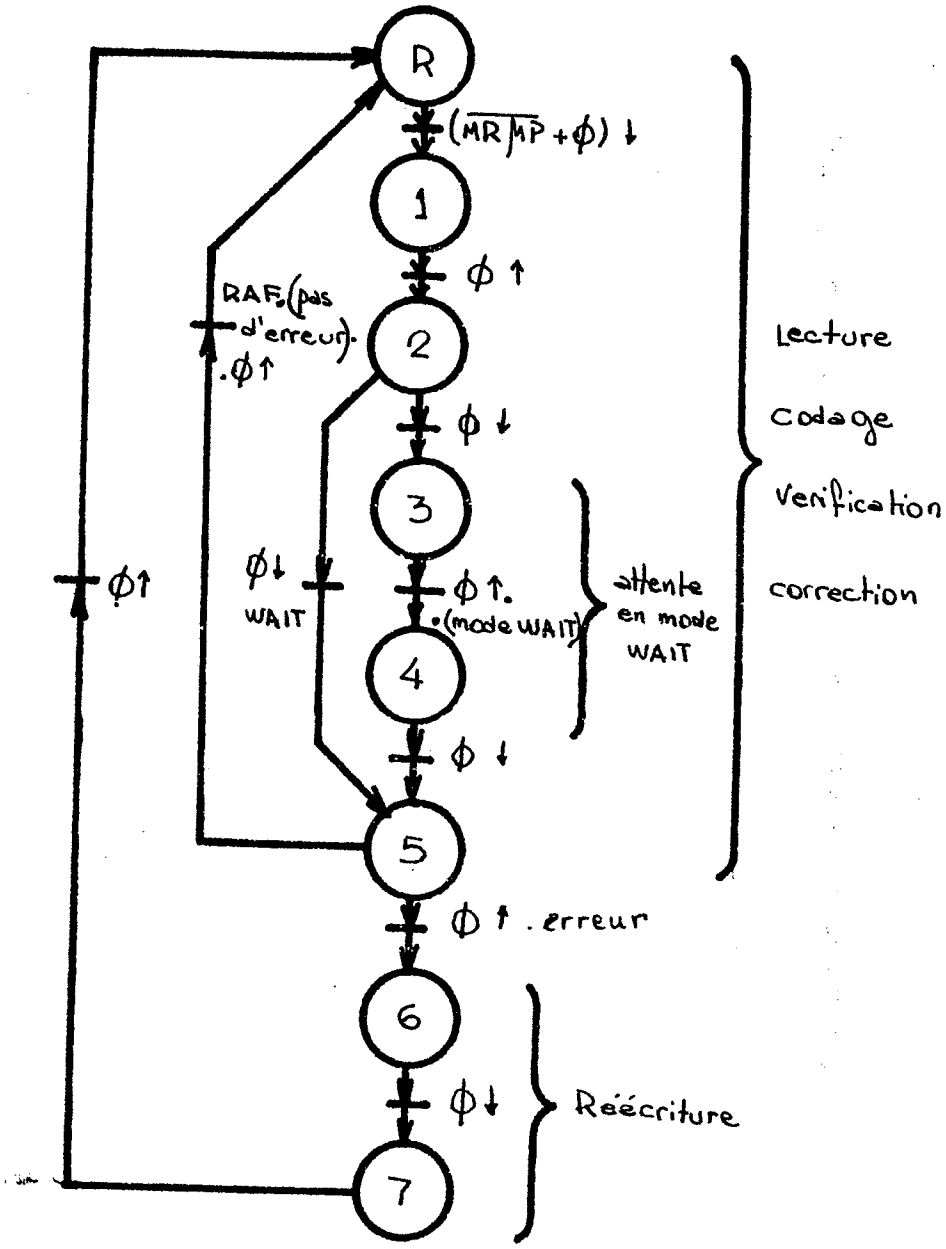


FIGURE III-11 : GRAPHE DE SEQUENCEMENT

III-4 DESCRIPTION FONCTIONNELLE INTERNE

III-4-1 Contrôle d'erreurs en mémoire :

Le code utilisé est un code SEC-DED ("Single Error Correcting - Double Error Detecting"). Les mots codés sont composés de 16 bits d'information plus 6 bits de codage. Chaque bit de codage (ou de "parité Hamming") est calculé comme la parité de 8 bits pris parmi les 16 de données.

Chaque bit de données rentre dans le calcul de trois parités Hamming différentes.

Pendant le cycle d'écriture, les bits de codage qui seront associés avec les données, sont générés par un module CODEUR (Fig. III-12) et envoyés vers le BUS HAMMING à travers un multiplexeur MUXSH et un registre RSH pour être stockés en mémoire.

Pendant le cycle de lecture, les bits de parité sont lus avec les données en mémoire, acheminés à travers le BUS HAMMING et comparés avec les bits de parité recalculés à partir de données lues.

Le résultat de cette comparaison, appelé SYNDROME est décodé par un circuit CORRECTEUR qui effectue la correction de données si il est nécessaire et le nouveau mot codé est échantillonné par les registres RSH et RSI.

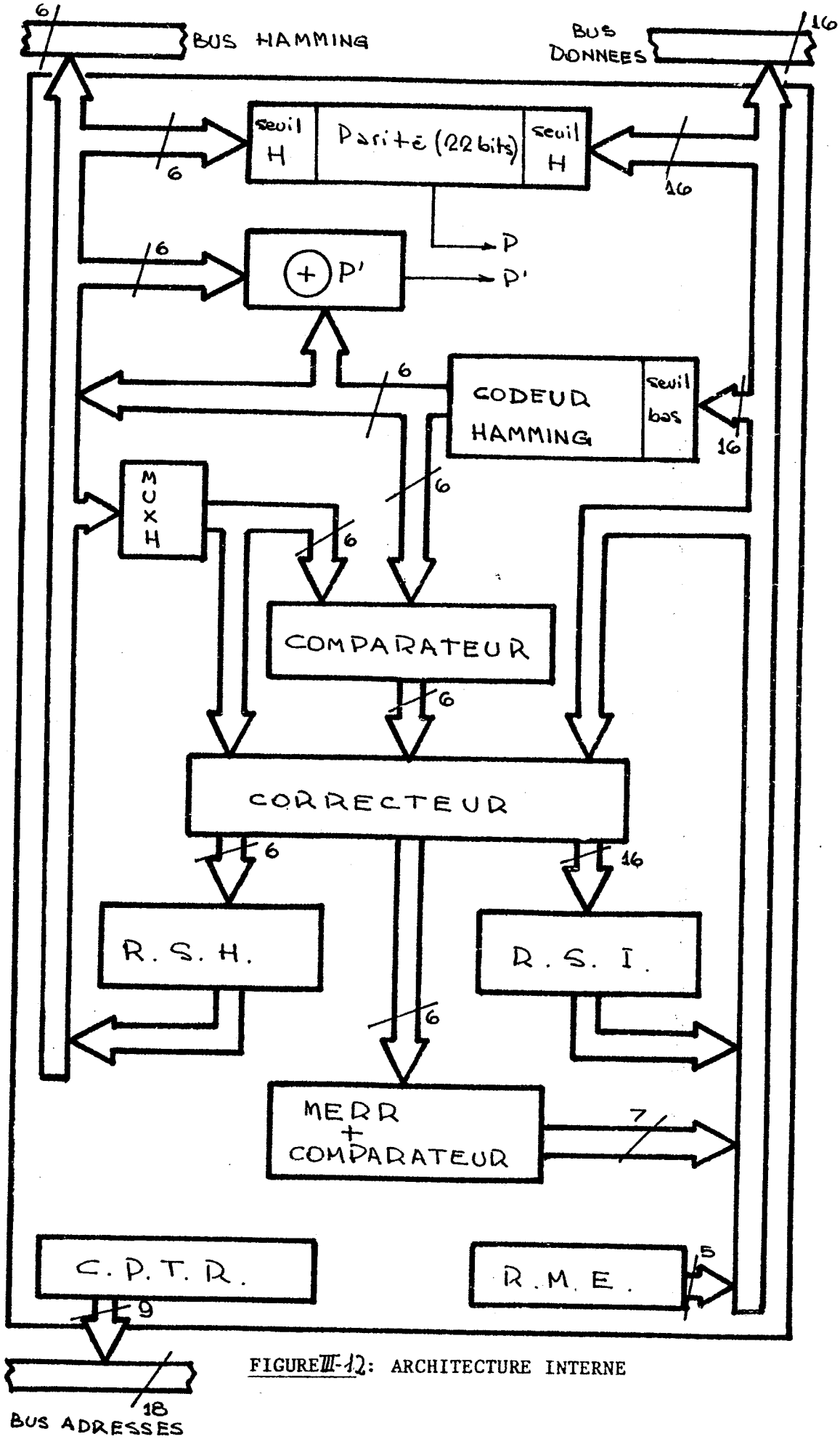


FIGURE III-12: ARCHITECTURE INTERNE

Le cycle de lecture est alors modifié pour permettre la ré-écriture de ces deux registres en mémoire : leurs contenus sont envoyés sur les bus pour être échantillonnés par la mémoire et le microprocesseur. Le temps du cycle des microprocesseurs disponibles suffit pour permettre ces opérations.

En cas d'erreur corrigée, le syndrome est stocké dans le système de détection de pannes franches, RADE. Un cycle de rafraîchissement est un cycle de lecture raccourci. Mais, si on détecte une erreur, il sera prolongé pour permettre la correction et la ré-écriture.

Les adresses mémoire à vérifier (poids faible) sont générées par le compteur CPTR.

III-4-2 Système de codage-correction :

Il est composé de trois blocs : un codeur (bloc Hamming), un comparateur et le décodeur-correcteur (Cf. Fig. III-12).

CODEUR : Il est chargé de calculer les 6 bits de codage associés au mot de 16 bits présent sur le bus de données. Les 6 bits sont envoyés vers le registre de sortie (cas de l'écriture d'un mot en mémoire) et au comparateur.

COMPARATEUR : Il ne sert qu'à la lecture. Il est chargé de comparer les bits de codage déjà stockés en mémoire à ceux recalculés à partir de la donnée, par le codeur. La sortie du comparateur (syndrome d'erreur) est envoyé au correcteur.

CORRECTEUR : Il décode le syndrome, détermine la nature de l'erreur (pas d'erreur, erreur simple corrigible, erreur multiple non corrigible) détermine la position du bit faux et effectue la correction. La commande CCOR permet la mise hors circuit du correcteur lors des opérations d'écriture en mémoire.

REGISTRES DE SORTIE : A la fin des opérations de vérification, le résultat correct est chargé dans les registres :

- RSI (16 bits d'information)

- RSH (6 bits codage Hamming).

Le contenu de ces registres peut être émis respectivement sur les bus de données et Hamming.

III-5 LES SYSTEMES DE TEST ET D'AUTOTEST.

III-5-1 Autotest du circuit :

D'une façon générale, il faut éviter qu'une panne du circuit n'entraîne la contamination catastrophique ("pollution") de la mémoire. On cherchera donc à éviter le plus possible la réécriture de mots erronés.

Les erreurs dues aux pannes internes du circuit devront être soit détectées rapidement, soit ne pas provoquer de réécriture en mémoire. Toute panne propre au CCE devrait aussi causer une alarme interne du circuit. Toutefois, cette propriété ne sera vérifiée ici que pour les parties critiques du circuit (génération des commandes mémoire et système de codage et de correction). Les pannes prises en compte pour effectuer l'étude de l'autotest sont des pannes du type collage d'une connexion interne. Dans la technologie étudiée cela recouvre tous les défauts réels sauf certains court-circuits.

III-5-2 Autotest du système de codage et de correction :

Il est basé sur une propriété du code de Hamming utilisé : chaque bit du mot à coder est utilisé pour générer 3 des bits de codage.

La somme modulo 2 des 6 bits de codage est égale à la parité des 16 bits de l'information à protéger. La parité du mot codé (22 bits) est toujours égale à "0". L'autotest peut donc être basé sur des vérifications de parité (Fig. III-13) :

A LA LECTURE, la parité du mot mémoire, calculée à l'aide d'un circuit spécial, sera comparée à celle obtenue à l'aide des 6 bits de codage.

A L'ECRITURE, on vérifiera que la parité du mot codé vaut bien "0".

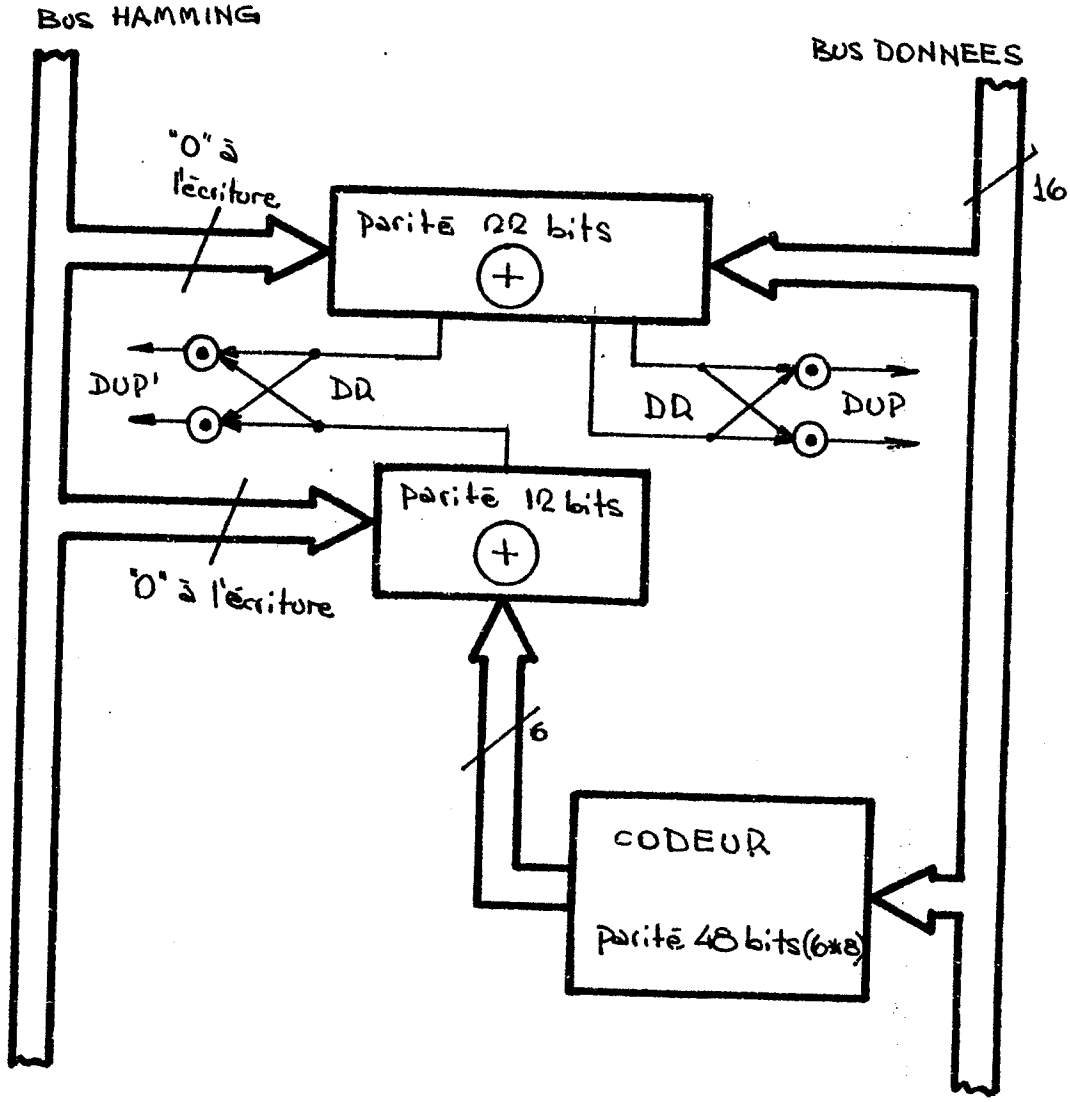


FIGURE III-13 : AUTOTEST DU SYSTEME DE CODAGE CORRECTION

L'analyse des effets des pannes simples (affectant une seule connexion) est assez simple : Une panne dans le circuit codeur va engendrer une erreur portant sur 1 bit seulement, qui sera détectée à la réécriture (parité globale fausse). Même chose pour le comparateur et le correcteur.

Le décodeur ne pose pas de problème non plus : Il s'agit d'un décodeur détectant des mots contenant
soit 3 "0" et 3 "1",
soit 5 "0" et 1 "1".

Il est évident que, pour passer d'un mot à l'autre, il faut changer au moins 2 bits Dans ces conditions, les pannes simples du décodeur ne peuvent provoquer que des erreurs portant sur 1 seule sortie, donc affectant 1 seul bit en sortie du correcteur. Là encore, le contrôle de la parité à l'écriture détectera le mauvais fonctionnement.

Le seul point délicat se trouve en entrée du circuit codeur (sur le bus de données) : Chaque bit du bus doit participer à la génération de 3 bits du code. Une panne du bus d'entrée (coupure) peut donc entraîner, selon son emplacement, des erreurs simples, doubles ou triples (Fig. III-14 a).

L'erreur simple ou triple entraînera la correction d'un bit, qui sera détectée par la vérification de parité. Par contre, l'erreur double ne sera pas détectée par l'autotest, et sera signalée au microprocesseur comme une erreur mémoire d'ordre 2, qui en provoquera toutefois pas de réécriture fausse dans la mémoire.

Il faudra donc, à l'implantation, interdire la possibilité de pannes ne provoquant que des erreurs doubles. Cela est heureusement possible (Fig. III-13 b).

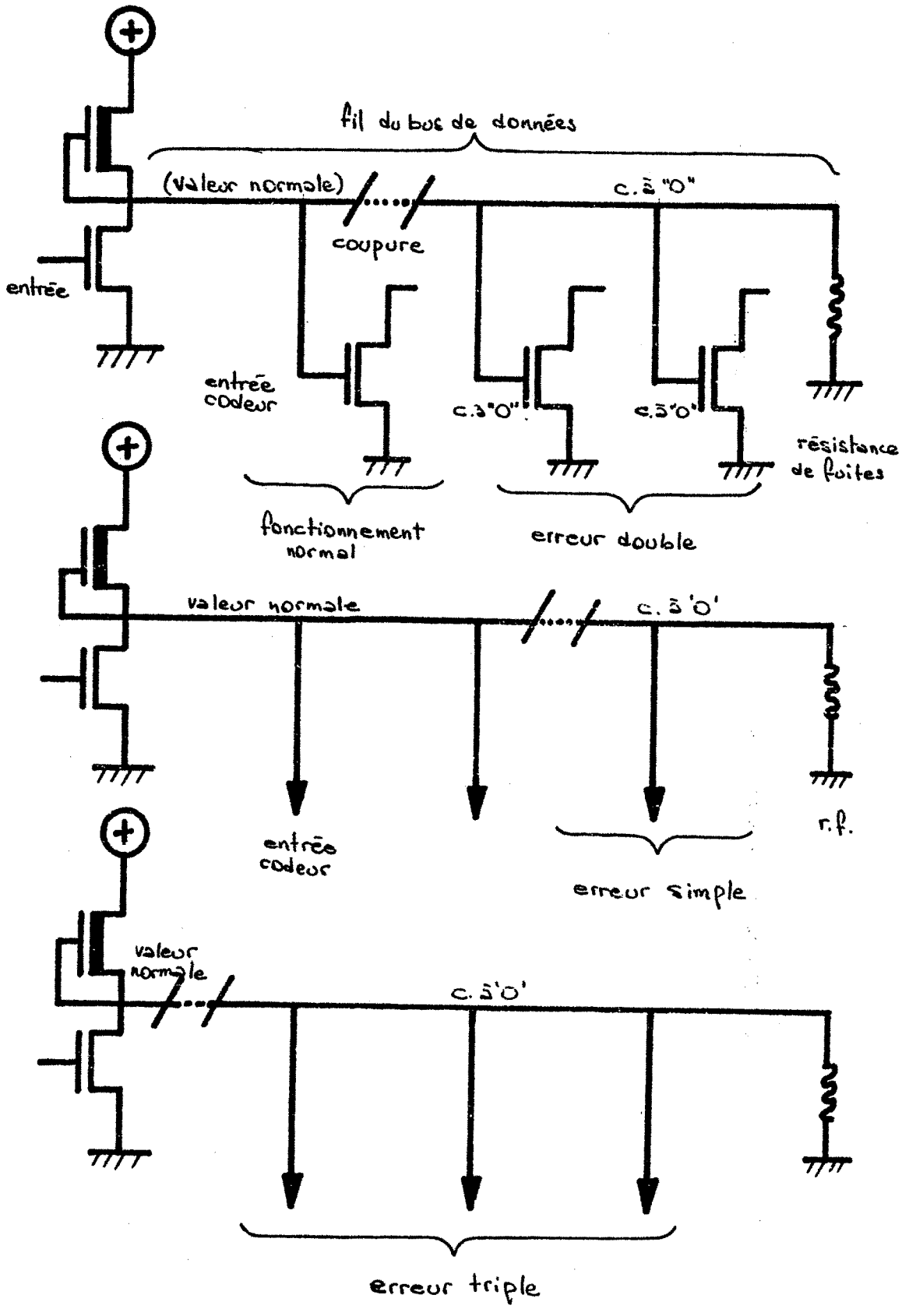


FIGURE III-14 a : ERREUR SIMPLE, DOUBLE OU TRIPLE SELON SELON L'EMPLACEMENT DE LA COUPURE. (PANNES LIEES A L'IMPLANTATION)

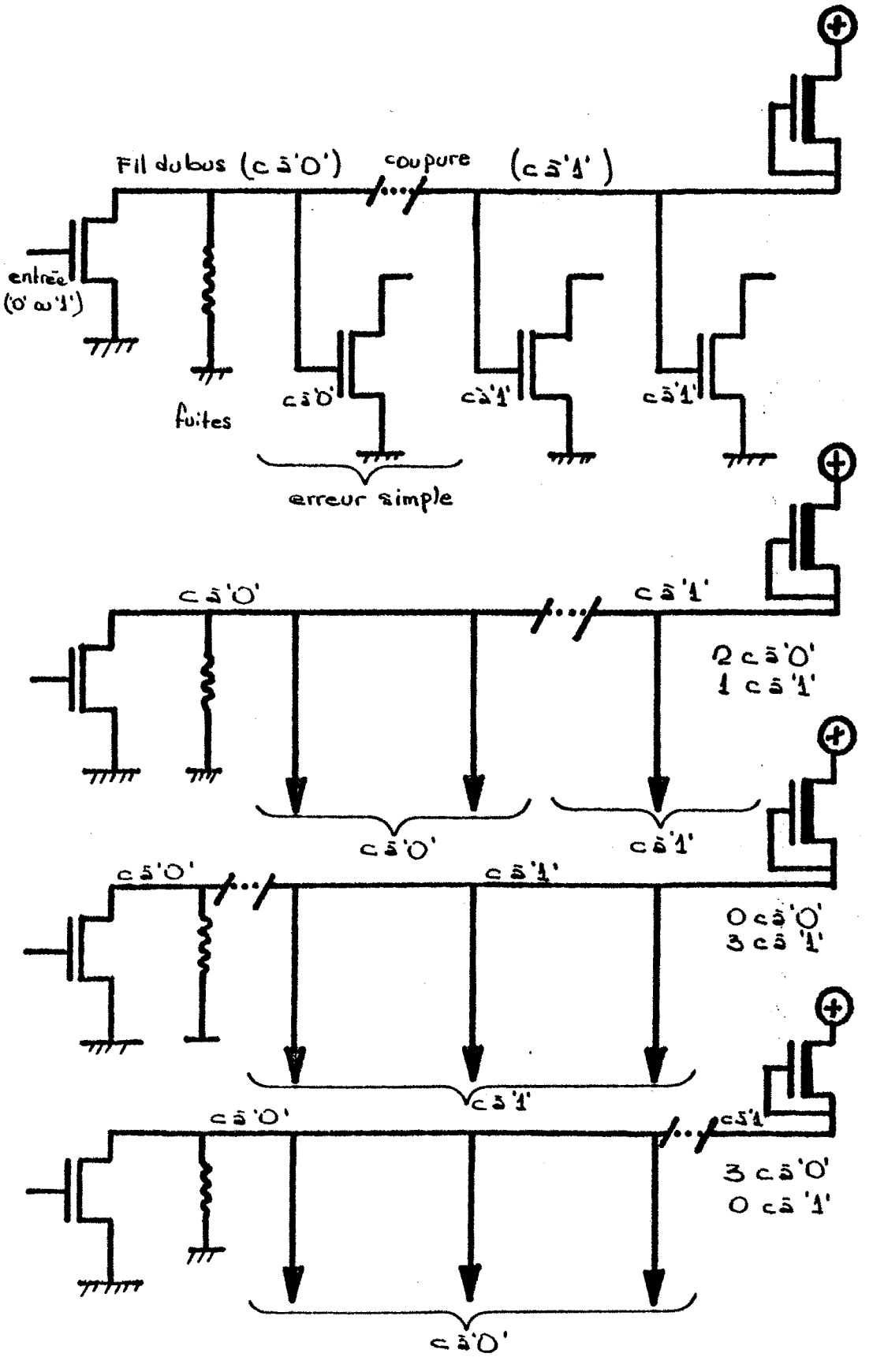


FIGURE III-14 b : TOUT COUPURE ENTRAINE OBLIGATOIREMENT UNE ERREUR SIMPLE OU TRIPLE (ERREUR IMPAIRE) (DANNES LIEES A L'IMPLANTATION)

Les pannes des circuits de vérification de parité ne causent pas de problèmes : Elles ne risquent pas de polluer la mémoire, et sont détectées à la première erreur.Les sorties du système de comparaison de parité sont dupliquées (système double-rail).

REMARQUE :

Toutes les erreurs causées par des pannes simples sont détectées ou bien ne causent pas de réécriture mémoire. Cependant, certaines pannes ne sont détectables que dans le cas (très rare) où l'on aura une erreur en mémoire. (De tels circuits, très rarement utilisés, sont appelés CIRCUITS DORMANTS).

Pour plus de sûreté, il est prudent de les "activer" périodiquement, en forçant des erreurs dans la mémoire. Sinon on pourrait arriver à une accumulation de pannes dans le système.

III-5-3 Autotest du séquenceur :

Le séquenceur est dupliqué. Cette technique est classique : Deux copies identiques travaillent en parallèle sur les mêmes données. Les résultats sont comparés à l'aide d'un comparateur autotestable. Cependant, l'utilisation d'un comparateur totalement autotestable pour les 34 signaux de commande coûterait trop cher (250 portes NAND). Pour des raisons d'économie, la comparaison systématique n'a été effectuée que par les signaux les plus critiques sortant du circuit. Des précautions particulières ont été prises pour la sortie "INTREQ" (demande d'interruption) dont le collage à l'état actif risquerait de gêner le microprocesseur : le circuit de génération de INTREQ est dupliqué, et sa sortie fait partie du message lu par le microprocesseur en cas d'interruption. Le microprocesseur pourra alors vérifier par lui-même la concordance des deux demandes (le fil INTREQ et le bit correspondant lu dans le message) et masquer la demande d'interruption en cas de désaccord. De même le microprocesseur pourra venir lire le message de façon arbitraire de temps en temps, de façon à détecter le collage de "INTREQ" à la valeur inactive.

Les autres commandes sont comparées par l'intémédiaire du système de vérification de parité.

Prenons l'exemple d'un système multiplexant, selon une commande CMUX, deux bus de 16 bits, B1 et B2 sur un bus de sortie de 16 bits, BS.

On pourra commander 8 bits à l'aide de CMUX, et les 8 autres à l'aide de sa réplique CMUX'.

Un collage de CMUX causera le mélange des bus B1 et B2 sur BS.

Une vérification de la parité de BS aura une probabilité de 0.5 de détecter la panne à chaque opération.

III-5-4 Autotest du BUS :

Le circuit comporte aussi un système simple permettant de tester la présence de certaines pannes des bus de données et Hamming ; en particulier le mauvais fonctionnement des "Trois états". Ce système fonctionne à l'entrée des données, et est capable de détecter des erreurs sur des mots non codés. Il est basé sur une vérification des niveaux logiques d'entrée : chaque bit du bus de données attaque 2 portes à seuils différents (1V et 3V) les deux sorties S et S' dépendent du niveau d'entrée (Fig. III-15 a).

$V_E \leqslant 1V$	S = 1	S' = 1	"0" correct
$V_E \geqslant 3V$	S = 0	S' = 0	"1" correct
$1V < V_E < 3V$	S = 0	S' = 1	Valeur douteuse.

Examinons maintenant les cas des pannes possibles du système "trois états".

- Deux sorties sont validées simultanément sur le même fil de bus.

Si elles ont la même valeur, la panne ne crée pas d'erreur. Si elles ont des valeurs différentes, alors le signal a de fortes chances d'être à une valeur intermédiaire ($\simeq$ 2V) à l'aide d'un diviseur à résistances (Cf. Fig. III-15 b).

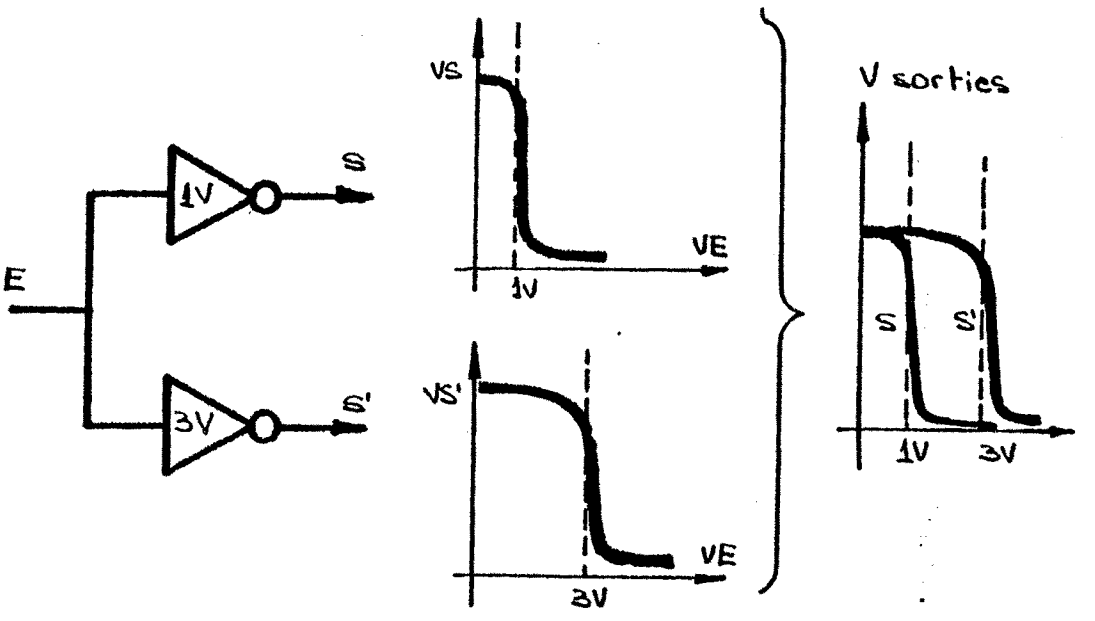


FIGURE III-15 a : PORTES A SEUIL POUR LE NIVEAU D'ENTREE.

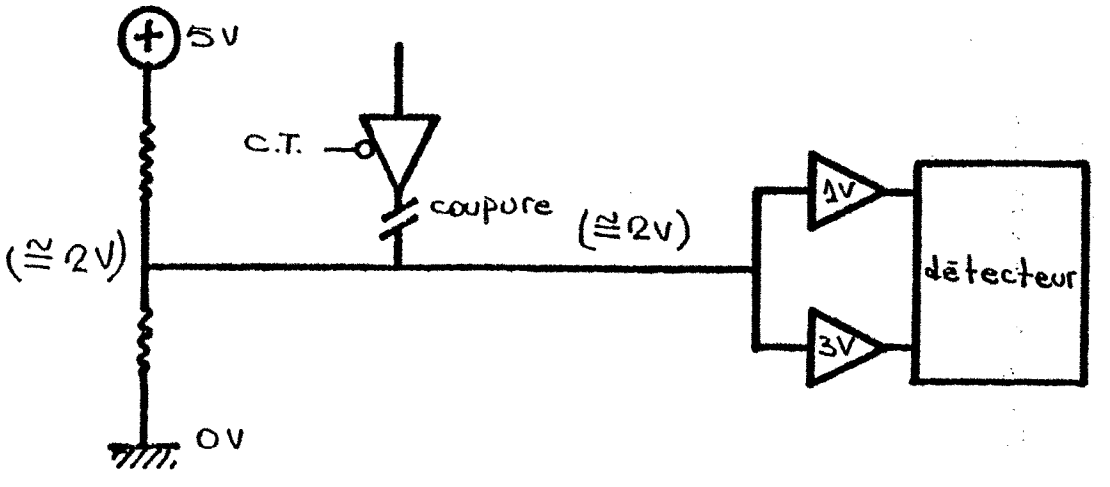


FIGURE III-15 b : VALEUR INTERMEDIAIRE SUR LE FIL DU BUS.

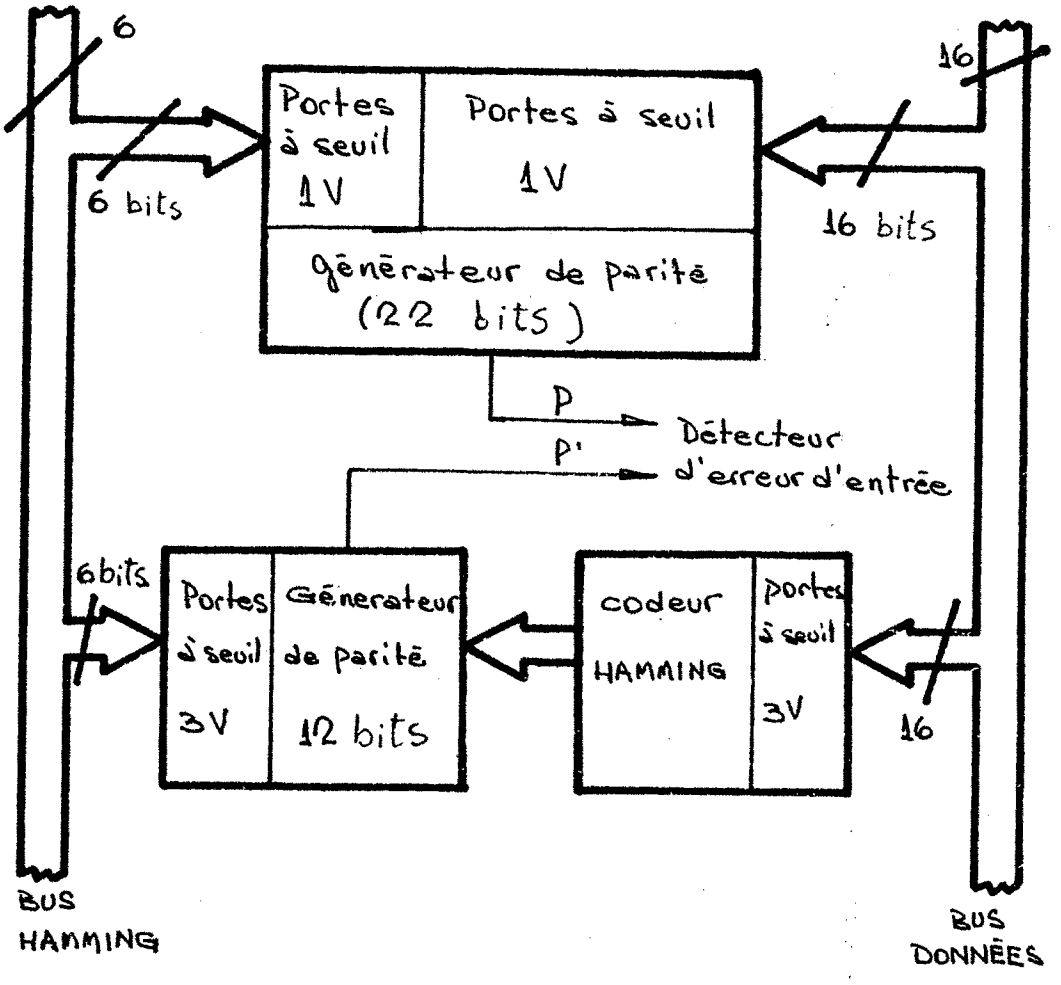


FIGURE III-15 c : PORTES A SEUIL DANS LE SYSTEME DE VERIFICATION.

Les résistances devront être assez fortes pour ne pas gêner le fonctionnement des amplis de sortie, et en même temps, suffisamment faibles pour permettre la charge du bus à la valeur 1,7V assez vite pour permettre la détection en temps nécessaire.

La vérification des sorties du système à seuil peut se faire d'une façon très économique par l'intermédiaire de la double vérification de parité effectuée lors de l'entrée de données : Les sorties provenant des portes à seuil haut attaqueront par exemple les circuits de codage Hamming, et celles des portes à seuil bas rentrent dans le circuit de calcul de parité (Cf. Fig. III-15 c).

Les portes à seuil pouvant être les portes d'entrée des circuits logiques, la vérification du bus ainsi obtenue est très économique. Elle permet de détecter :

- Les "trois états" coupés ou court-circuités.
- Les insuffisances des amplificateurs de sortie.

Par contre, cette vérification est insensible aux
- collages francs des fils du bus,
- coupures des connexions et
- court-circuits de deux connexions voisines.

Ces trois dernières pannes sont détectées, par contre, par le codage Hammming.

III-5-5 Test hors ligne :

Il n'a pas été possible d'éviter dans le circuit la présence d'un certain nombre de systèmes "dormants" : Ce sont des circuits qui ne fonctionnent et ne sont autotestés que lorsqu'une erreur mémoire doit être corrigée. Un tel évènement est normalement très rare, typiquement quelques fois par semaine.

On ne peut pas compter sur le fonctionnement normal pour assurer la détection des pannes internes avec une rapidité acceptable. Il est donc prudent d'effectuer de temps à autre, à l'initiative du microproceseur, un test hors ligne des parties dormantes.

Ce test nécessitant la lecture de mots erronés, un mode de fonctionnement particulier a dû être prévu pour permettre la simulation d'erreurs mémoire : Le mode "Test 2" permet la mise à "0" du bus de Hamming.

On peut aussi forcer des erreurs simples ou multiples dans la partie codage et tester les circuits associés à la réécriture.

Notons qu'une erreur triple des bits de codage est comprise par le circuit comme une erreur simple dans la partie données du mot mémoire.

Cette propriété permet d'effectuer un test H.L. de type aléatoire : Le microprocesseur place le circuit en mode TEST2, puis effectue la lecture d'une partie de la mémoire. Les bits de codage étant "forcés" à une valeur arbitraire, un certain nombre d'entre eux vont se trouver erronés.

- Si 1 seul bit du codage est faux : On a affaire à une erreur simple qui sera "corrigée" (test du système de correction des 6 bits de code).

- Si 3 bits du codage sont faux : L'erreur triple correspondante est confondue avec une erreur simple dans les bits de données du mot (test du système de correction des 16 bits d'information).

- Dans les autres cas : L'erreur (non corrigible) détectée par le circuit n'est pas traitée.

Un deuxième mode de test permet la simulation d'erreurs sur les bits de Hamming (les erreurs simples sont corrigées) : le mode TEST1 permet la vérification des signaux d'erreurs de parité sortie, en permettant l'émission sur les bus de mots "codés" impaires : Ces mots sont obtenus en forçant le registre RSH (Registre de Sortie Hamming) à "0". Le test se fera en simulant une écriture : Le microprocesseur enverra un mot à coder de parité impaire. Bien entendu, l'écriture réelle en mémoire ne se fera pas.

III-6 CONCLUSION.

Ce circuit est conçu en technologie NMOS canal 6 microns. Le nombre total de transistors est à peu près 5 000, et la surface est proche de 18 mm^2. Le coût des facilités de test et de l'autotest est environ 20 % de la surface totale de la puce. Il sera fabriqué en 1982 pour permettre l'expérimentation et des études statistiques sur ses possibilités de test et d'autotest.

III-7 BIBLIOGRAPHIE DU CHAPITRE III.

(1) J.L. RAINARD, J.M. GOBBI "CIRCUIT CODEUR MEMOIRES"
Note interne CNET-MEYLAN - 1981 (à paraître).

(2) T.C. MAY "SOFT ERRORS IN VLSI : PRESENT AND FUTURE"
IEEE Transactions on Components, Hybrids and Manufacturing Technology - VOL. CHMT-2, N°4 - Décembre 1979.

(3) J.F. ZIEGLER and W.A. LANFORD
"EFFECTS OF COSMIC RAYS ON COMPUTER MEMORIES"
Science, Vol. 206, 16 Novembre 1979.

(4) MC 68540 : "Memory error detection and correction circuit"
Ed. Préliminaire - Avril 1980.

(5) AM 2960 : "Cascadable 16 bit error detection and correction unit" - AMD - 1980.

(6) R.W. HAMMING "ERROR DETECTING AND ERROR CORRECTING CODES",
BSTJ - AVRIL 1950.

ANNEXE I

LES OUTILS DE CAO ET LE TEST

Un certain nombre d'outils informatiques d'aide à la conception matérielle existent. Ils permettent :

- de décrire un circuit, en général de façon modulaire afin que la modification d'un composant ne remette pas en cause la description entière ; de plus certains permettent une description à différents niveaux (structurel, fonctionnel, algorithmique).

- de simuler un circuit décrit, pour détecter les erreurs de conception de ce circuit.

Certains de ces outils CAO fournissent une aide spécifique à la détermination de séquences de test ; ils permettent alors :

- de décrire des pannes au niveau structurel et d'effectuer une simulation logique du circuit en panne ; cette simulation a pour but de déterminer toutes les pannes détectées par un ensemble de vecteurs de test : il s'agit d'analyse de vecteurs de test.

- de générer de façon automatique des vecteurs de test pour un circuit donné et un sous-ensemble de ses pannes potentielles, ceci pour certains des outils CAO : synthèse des vecteurs de test.

L'analyse d'un circuit avec pannes permet :

- soit de valider des séquences de test obtenues de façon manuelle ;
- soit de déterminer les pannes détectées par une séquence de test générée de façon aléatoire ;
- soit de déterminer l'ensemble des pannes détectées par une séquence de test générée par synthèse automatique pour une panne précise.

Nous allons ici présenter certains outils CAO classiques, relativement bien rodés :

- outils spécifiques au test, permettant l'analyse et la synthèse : TAU.
- outils permettant la simulation de pannes, donc l'analyse : EPISODE et TEGAS.
- outils permettant une simulation multiniveaux et en particulier la séparation PC-PO : AHPL et CASSANDRE-LASCAR.

On pourra voir qu'aucun de ces outils ne dispose de l'ensemble des capacités dont nous souhaiterions disposer pour le test de circuits complexes (Chapitre I), c'est-à-dire à la fois :

- d'une description et surtout d'une simulation multiniveaux ;
- d'une description séparée PC-PO. ;
- de la possibilité d'introduction de pannes.

A-I-1- PROGRAMME TAU.

Il s'agit d'un programme conçu en 1976 à SESCOSEM pour la génération de séquences de test pour circuits intégrés. Ce programme est capable de générer des séquences de test de manière autonome, mais laisse à l'utilisateur la possibilité d'orienter le déroulement des opérations et d'intervenir au cours de celles-ci.

L'ensemble de pannes potentielles du circuit est décomposable en sous-ensembles dont les vecteurs de test peuvent être élaborés séparément.

Chaque sous-ensemble de pannes peut être traité à l'aide d'un ou plusieurs des types de génération suivants :

a) Enchaînement de synthèses et d'analyses ;
b) Enchaînement de générations pseudo-aléatoires de vecteurs d'entrée et d'analyses ;
c) Analyse de vecteurs d'entrée donnés.

L'utilisateur peut générer la séquence de test en un ou plusieurs passages-machine en choisissant à chaque étape le sous-ensemble de pannes considéré. Entre chaque étape, il a accès à l'ensemble des résultats déjà obtenus.

ANALYSE est la simulation d'un circuit et de l'ensemble de ses pannes potentielles, cette simulation ayant pour but de déterminer toutes les pannes détectées par un ensemble de vecteurs de test.

On distingue généralement deux grandes familles d'analyse :

a) La SIMULATION PARALLELE : Simulation logique du circuit sain et parallèlement des N circuits erronés correspondants aux N pannes.

b) La SIMULATION DEDUCTIVE : Simulation logique du circuit sain puis déduction, à partir de son comportement, des pannes détectées.

SYNTHESE est le procédé d'élaboration de vecteurs de test détectant un sous-ensemble de pannes. Procédé itéré jusqu'à couverture de tout l'ensemble de pannes considéré.

A-I-1.1- Pannes considérées par le programme TAU :

Les pannes considérées par le programme TAU sont des pannes permanentes, des types suivants :

- les collages à 0 et à 1 ;
- un ensemble de courts-circuits logiques.

Elles peuvent affecter les entrées et les sorties des opérateurs, ainsi que les branches d'équipotentielles. Un ensemble de 9 valeurs a été défini pour représenter les valeurs possibles des connexions. La génération des vecteurs de test est faite en utilisant une méthode de chemins sensibles (D-Algorithme modifié).

A-I-1.2- Possibilités du programme TAU :

a) Enchaînement de synthèses de vecteurs de test et ses analyses :

On peut contrôler le temps maximum de génération ou le nombre maximum de vecteur d'entrée de chaque sous-séquence. Il sera possible ultérieurement de reconsidérer les pannes dont le test n'est pas assuré à l'issue de cette phase.

b) Enchaînement de générations pseudo-aléatoires et d'analyses :

Cette génération pseudo-aléatoire peut permettre de débuter économiquement l'élaboration d'une séquence de test.

Le contrôle de cet enchaînement est donné par :

- la longueur de la sous-séquence à générer aléatoirement (avec éventuellement certaines valeur prédéterminées). recherché
- le nombre maximum de répétitions de la génération de cette sous-séquence.
- le pourcentage de détection de pannes recherché par cette méthode.

Les valeurs pseudo-aléatoires sont obtenues à partir d'un registre à décalage rebouclé classique.

c) Analyse de vecteurs d'entrée données (commande ANALYSE) :

Il s'agit d'une commande répondant à trois besoins :

c-1) Validation des sous-séquences utilisateurs :

Le concepteur d'un circuit peut élaborer lui-même les vecteurs de test de certaines pannes.

Il suffit alors de pouvoir valider ces vecteurs par une phase d'analyse puis de compléter la séquence de test par un enchaînement de synthèse-analyse.

Cette méthode est efficace lorsque l'aspect fonctionnel de certaines parties d'un circuit, ou certaines expériences préalables, permettent la déduction rapide de vecteurs de test.

c-2) Positionnement INITIAL d'un circuit :

L'état initial d'un circuit lors de sa mise sous tension est généralement indéterminé. Pour le positionnement initial du circuit, il est préférable de spécifier les vecteurs d'entrée au programme plutôt que de lui en laisser l'élaboration. Commencer par l'analyse d'une sous-séquence utilisateur élaborée dans ce sens permettra donc l'initialisation du circuit tout en déduisant les pannes détectées par cette même sous-séquence.

c-3) Fusion des résultats de test partiels :

Dans le cas où l'on commence par l'élaboration des vecteurs de test pour des sous-ensembles de pannes distincts, l'ensemble total de pannes nécessite la fusion des résultats partiels.

Si nous considérons un sous-ensemble de vecteurs d'entrée V_1, élaboré pour le test d'un sous-ensemble de pannes P_1, il faut déterminer quelles pannes d'un autre sous-ensemble P_2 ($P_1 \cap P_2 = 0$) peuvent être détectées par les vecteurs d'entrée V_1.

L'analyse des vecteurs d'entrée déjà générés lorsque l'on débutera : le test d'un nouveau sous-ensemble de pannes permettra donc de NE PAS GENERER INUTILEMENT DES VECTEURS D'ENTREE DEJA OBTENUS.

Toutes les possibilités mentionnées sont nécessaires pour un test de CIRCUITS COMPLEXES.

Pour les circuits de FAIBLE COMPLEXITE, on pourra se contenter d'un enchaînement de synthèses et d'analyses effectué en un seul passage-machine et en considérant simultanément toutes les pannes potentielles du circuit. (Eventuellement on peut démarrer par l'analyse d'une sous-séquence de positionnement initial du circuit).

A-I-1.3 BILAN DE TAU :

L'avantage du programme TAU pour le traitement des circuits de grande complexité est son mode de traitement interactif qui permet l'intervention de l'utilisateur pour les parties du circuit dont les vecteurs de test sont déjà connus.

Les limitations du programme TAU sont dues à un nombre insuffisant des pannes détectées et à un coût prohibitif compte tenu du temps de calcul.

A-I-2 EPISODE

Les programmes EPISODE constituent un outil de description et de simulation de systèmes logiques développés à THOMSON-CSF et LETI.

Le logiciel comprend d'une part un langage de description et d'autre part un langage de simulation permettant respectivement de décrire rapidement les modèles et de les "faire fonctionner". En plus EPISODE permet de faire une SIMULATION DE DEFAUTS sur le circuit décrit. Elle sert à déterminer l'efficacité d'une séquence d'entrée vis-à-vis de la détection d'un ensemble de défauts susceptibles d'affecter ce circuit (collages ou court-circuit). Le souci des auteurs de ces programmes a été de prendre ces langages compréhensibles généraux et indépendants des méthodes de conception et de la technologie utilisée.

Nous avons utilisé une simulation EPISODE avant de faire la conception en symbolique MDMOS d'une partie importante du circuit intégré du chapitre III.

A-I-2.1- Le langage de description EPISODE :

Ce langage permet une description formelle, modulaire des circuits logiques à fin de pouvoir au besoin, décrire, tester et simuler les composants séparément.

Pour des raisons diverses la description doit être segmentable. Un circuit complexe peut se décomposer en réseaux de blocs interconnectés eux-mêmes contenus dans des blocs qui forment à leur tour un nouveau réseau, etc.

Avec EPISODE la description d'un de ces blocs est l'équivalent linguistique d'une "boîte noire".

A tout découpage du circuit complexe correspond une structure de blocs imbriqués respectant les contraintes de segmentation.

Différents niveaux de formalisme peuvent être utilisés suivant que l'on désire avoir plus ou moins accès à la structure interne.

Nous pouvons descendre, par exemple, d'un premier découpage très macroscopique du circuit, équivalent au schéma fonctionnel classique (avec l'introduction de MODELES de blocs écrits en FORTRAN), jusqu'à une description plus fine où est décrite la réalisation du circuit pour une technologie donnée (MDMOS canal N dans le cas du chapitre III).

L'archivage des descriptions permet la constitution de BIBLIOTHEQUES publiques ou privées.

Si au niveau de la conception l'utilisateur n'utilise que des fonctions logiques déjà définies (comme les "boîtiers" d'un catalogue de constructeur), la fonction de description se réduit alors à une fonction d'assemblage. Le concepteur puise dans la bibliothèsque de blocs et n'a plus qu'à assembler et connecter entre elles les descriptions des blocs utilisés. Le modèle ainsi obtenu est un équivalent rapidement monté de la maquette réelle du circuit complexe.

A-I-2.2- La simulation EPISODE :

Le but de toute simulation est d'éviter la mise en oeuvre d'une maquette physique (ou de la compléter) et de faciliter ainsi la prise de décisions sur le circuit complexe en projet.

En vue de la simulation, la description (réalisée à partir d'éléments standard reconnus par le langage de description et de sous-circuits mémorisés en bibliothèque). Elle est vérifiée puis compilée en une structure interne des variables et en un ensemble de relations liant ces variables entre elles.

L'ensemble "Description compilée" + "Simulateur" constitue un module simulable.

L'utilisateur dispose d'un langage de COMMANDES lui permettant de positionner les variables d'entrée, de charger les éléments de mémorisation et de connaître le contenu de n'importe quelle variable interne.

En mode conversationnel il est possible de sauvegarder instantanément l'état du circuit au milieu d'une séquence de simulation.

Il existe actuellement deux versions du simulateur. Sur une même description EPISODE du circuit il est possible de faire :

a) Une SIMULATION LOGIQUE utilisant quatre valeurs "0", "1", "ϕ" et "X".

ϕ rendant compte des indéterminations de transition d'un signal passant de 0 à 1 ou de 1 à 0 avec une certaine incertitude.

X rendant compte des indéterminations dues à l'initialisation ou des indéterminations dues à l'application de configurations ou de séquences interdite sur certains éléments.

Ce type de simulation sert à valider le circuit sur le plan temporel. Elle donne à l'utilisateur la possibilité de faire intervenir des notions temporelles plus proches de la technologie, telles que retards détaillés pour des blocs.

Il devient ainsi possible de vérifier que le schéma technologique réalise exactement le schéma logique prévu.

b) Une SIMULATION DE DEFAUTS utilisant trois valeurs "0", "1" et "X".

De manière interne il existe deux types de simulation de défauts :

- un simulateur déductif et
- un simulateur parallèle.

A-I-2.3- Caractéristiques générales d'EPISODE :

Les ENTREES A EPISODE : Elles comprennent :

a) La DESCRIPTION du circuit logique écrite en langage standard EPISODE ou sous forme d'algorithme en FORTRAN pour les modèles spéciaux.

La finesse de la description doit être faite en fonction du degré de précision souhaitée sur les résultats de la simulation. En particulier, c'est à l'utilisateur de choisir le niveau de finesse de la modélisation temporelle (pas de retard, retards typiques, retards typiques de montée et de descente, retards minimaux et maximaux de montée et de descente, retards fixes ou vriables en cours de simulation).

b) Les COMMANDES DE SIMULATION destinées à faire fonctionner le modèle de circuit décrit. Elles définissent la séquence d'entrée à appliquer au circuit ainsi que les conditions de déroulement de la simulation.

Les SORTIES EPISODE : Elles comprennent :

a) Au cours du traitement de la description, des TEXTES EDITES et des DIAGNOSTICS d'ERREURS (erreurs syntaxiques, sémantiques...).

b) Le déroulement de la simulation avec EDITION de RESULTATS. Les sorties graphiques (les chronogrammes) ont été les plus utilisées pour le circuit du chapitre III.

A-I-2.4-BILAN D'EPISODE :

L'AVANTAGE du programme EPISODE est sans doute sa possibilité de TRAITEMENT DES DEFAUTS sur un circuit logique de grande complexité. La gestion de listes de pannes (détectées avec la séquence de vecteurs de test envoyée sur les entrées) est de la plus grande importance pour notre travail. EPISODE a été largement utilisé pour choisir les séquences de test employées au chapitre III et en annexe II.

La LIMITATION du programme EPISODE est l'absence de l'aspect FONCTIONNEL dans la description d'un circuit complexe. Remarquons qu'il est théoriquement possible de faire une description FONTIONNELLE d'un bloc en utilisant FORTRAN pour la programmation. En pratique, ces MODELES ne sont pas faciles à faire, d'où l'absence d'utilisation par les concepteurs de circuits préoccupés par des problèmes de testabilité (plutôt que par des problèmes de programmation très spécialisée).

A-I-3- TEGAS

Le programme TEGAS a été développé aux Etats-Unis (Université de Houston et "Comprehensive Computing System and Services, Inc.", AUSTIN-TEXAS) à partir de 1979.

Il est orienté vers la simulation d'un circuit complexe conçu de façon modulaire en utilisant une méthode DESCENDANTE, ou ASCENDANTE, ou bien une combinaison des deux.

TEGAS permet la coéxistence dans une même bibliothèque, de plusieurs niveaux de description structurelle d'un même module d'un circuit. C'est à l'utilisateur de décider à quelle version de chaque module il fait appel lors de la simulation.

Le développement de ce système de programmes a été basé sur les outils d'aide à la programmation existants ("Design Tools to Software Engineers") et adapté en principe à la conception DESCENDANTE.

Au cours de l'élaboration d'un circuit les modules entrant dans sa conception passeront par différents stades de définitions, l'AVANTAGE de TEGAS est de permettre l'écriture de modèles à des niveaux de détail divers.

Le système TDL ("TEGAS DESIGN LANGUAGE") est un outil sophistiqué pour la conception des "VLSI". Son AVANTAGE le plus important pour nous est sa capacité à traiter les pannes simulées sur un circuit et sa génération automatique de vecteurs de test par la méthode du chemin sensible. Mais sa limitation est le manque d'une vraie simulation fonctionnelle de haut niveau.

A-I-4 AHPL

Il s'agit d'un programme développé aux Etats-Unis pour la description des circuits logiques complexes au niveau des transferts internes et de séquencement d'opérations avec parallémisme autorisé.

Le travail fondamental du concepteur de circuits est justement celui-ci : séquencement d'opérations et transfert d'informations à l'intérieur du circuit. Le reste du développement des diagrammes logiques, schémas et connexions est plutôt une procédure mécanique et automatisable.

A-I-4.1- Le langage AHPL :

Le langage choisi devait être suffisamment détaillé pour décrire des opérations bit par bit, et en même temps il devait être assez puissant pour permettre une description concise des opérations complexes (aspect fonctionnel).

Le langage de programmation le plus adapté est celui développé par K.E. IVERSON, connu simplement comm APL ("A Programming Langage", WILEY - 1962).

Le sous-ensemble de APL utilisée pour la description de circuits est appelé AHPL ("A Hardware Programming Language").

En AHPL une variable est le nom d'un REGISTRE ou plus généralement le nom d'un élément de mémorisation.

Par exemple : MQ ←——— B signifie le transfert du contenu de registre B dans le registre MQ. Bien entendu B et MQ doivent avoir le même nombre de bits.

MQ ←——— 10010111

signifie placer cette chaîne de bits donnée dans le registre MQ.

En AHPL la valeur de chaque élément d'un registre peut prendre seulement deux valeurs "0" et "1".

A-I-4.2- Le parallélisme en AHPL :

Il est possible d'exprimer le parallélisme entre opérations d'une manière très simple. Par exemple, le cas de deux transferts simultanés. Sur deux registres séparés peut s'exprimer :

A, B ←——— C, D

ou bien

A ←——— C ; B ←——— D

Ces instructions de transfert en AHPL demandent seulement une période d'horloge pour être exécutées.

A-I-4.3- Description AHPL d'un circuit en séparant PC-PO :

En AHPL on décrit dans la PC les transferts successifs entre registres, les opérateurs (P.O.) ayant été décrits préalablement et étant "appelés" dans la description de la P.C.

A-I-4.4- Synthèse de la P.C. à partir d'une description AHPL :

Il existe un programme pour faire la transformation automatique d'AHPL en diagramme de blocs logiques. La génération de ce diagramme par un tel programme est analogue à la génération de programmes en ASSEMBLEUR à partir de programmes écrits en langages de haut niveau. On peut parler alors d'un "compilateur circuit" ("hardware compiler"). Il y a eu des essais de ce type comme celui de Gentry : "A Compiler for AHPL Control Sequences" (Ph.D. disertation, Université d'Arizone - juin 1971) ; cela permet la synthèse automatique de la P.C. avec une réalisation 1 parmi n.

Mais un séquenceur de ce type n'est pas utilisé dans tous les ordinateurs, seulement les machines très rapides le justifient. Très souvent, de petites machines informatiques utilisent la technique de micro-programmation.

Dans ce cas, il est également possible d'utiliser AHPL pour concevoir la P.C. microprogrammée en faisant la correspondance entre micro-instructions et lignes de programme AHPL ("MICRAL : MICRO ASSEMBLING LANGUAGE").

D'autres types de P.C. utilisent une horloge en plusieurs phases ou bien distribuent des signaux de contrôle multi-niveau, ces cas peuvent également être décrits en AHPL, mais ce langage est plus spécialement adapté à la conception des P.C. câblées qui utilisent des éléments de retard pour bien séparer dans le temps, les signaux de commande.

A-I-4.5- Bilan AHPL :

Cet outil CAO très sophistiqué permet la séparation PC-PO, mais ne traite pas les pannes.

A-I-5- CASSANDRE ET LASCAR.

A-I-5.1-CASSANDRE :

Le programme CASSANDRE est un outil de description et de simulation de circuits logiques au niveau de transfert de registres, développé à l'ENSIMAG (Ecole Nationale Supérieure d'Informatique et de Mathématiques Appliquées de Grenoble).

Un circuit logique peut se décomposer en réseaux de blocs interconnectés. Avec CASSANDRE la description d'un de ces blocs s'appelle "unité" (équivalent linguistique de "boîte noire"). Différents niveaux de formalisme peuvent être utilisés suivant le niveau d'accès à la structure interne. La constitution de BIBLIOTHEQUES est possible et permet l'assemblage des "unités" disponibles.

L'utilisateur peut simuler le circuit en faisant fonctionner le modèle pas à pas en demandant des cycles de calcul.

En MODE CONVERSATIONNEL il est possible de sauvegarder instantanément l'état du circuit au milieu d'une séquence de simulation.

Il existe actuellement deux versions du simulateur se distinguant entre elles par la notion de "TIMING" :

A) Une version de TYPE SYNCHRONE permettant de vérifier un circuit sur le plan logique au niveau de fonctions booléennes et des algorithmes de calcul. Dans cette version, les impulsions de synchronisation proviennent essentiellement de l'extérieur du circuit simulé. Aucune impulsion de synchronisation ne peut être élaborée à l'intérieur de ce circuit.

B) Une version de TYPE ASYNCHRONE permettant la simulation logique à un niveau plus fin que ne permet pas la version SYNCHRONE. Elle donne à l'utilisateur la possibilité de faire intervenir dans son modèle des notions temporelles plus proches de la technologie, telles que retards des boîtiers, création d'impulsions sur transitions de signal, etc...

LES ENTREES CASSANDRE : Elles comprennent :

A) La DESCRIPTION du circuit logique écrite en langage CASSANDRE.

B) Les COMMANDES de SIMULATION destinées à faire fonctionner le modèle.

LES SORTIES DES PROGRAMMES : Elles comprennent :

A) Au cours du traitement de la description, des TEXTES EDITES et des DIAGNOSTICS D'ERREURS.

B) Le déroulement de la simulation. Des sorties graphiques (chronogrammes) sont possibles pour la version asynchrone. Dans CASSANDRE, comme dans APL, tous les opérateurs peuvent porter sur des scalaires aussi bien que sur des tableaux.

A-I-5.2-CASSANDRE-LASCAR :

LASCAR est une extension de la version synchrone des programmes CASSANDRE. Il permet des descriptions du circuit à différents niveaux de détail. Son avantage consiste à autoriser dans un même modèle la cohabitation de composants décrits au niveau fin en CASSANDRE avec des composants décrits de manière fonctionnelle en LASCAR.

Ainsi dans une CONCEPTION DESCENDANTE, le langage LASCAR sert de langage de spécification des composants que l'on décrira ensuite en CASSANDRE pour en vérifier le fonctionnement logique.

A l'inverse, dans une CONCEPTION ASCENDANTE, les composants seront d'abord décrits en CASSANDRE et vérifiés au niveau fin. Ensuite, dans une simulation plus globale du circuit tout entier, la réalisation matérielle interne des composants n'a plus lieu d'être décrite, et les modèles écrits en CASSANDRE seront remplacés par des modèles écrits en LASCAR. Le gain de temps en cette simulation est dans un rapport compris entre 50 et 200. Il est alors possible de valider des choix d'architecture de circuit avec LASCAR et d'effectuer des mesures de performances fines sur les composants du circuit décrit en CASSANDRE.

LE LANGAGE LASCAR :

Des variables et des opérateurs arithmétiques ont été introduits de manière à pouvoir exprimer de façon simple les algorithmes réalisés par une unité. Ainsi pour effectuer une multiplication, il n'est plus besoin de décrire un mutiplieur, qui est déjà un circuit logique relativement complexe.

Des opérateurs de test de variables arithmétiques et des opérateurs de conversion pour passer de valeur arithmétique à valeur booléenne, et vice-versa, ont été réalisés.

D'autres variables, les compteurs, permettent d'espionner le fonctionnement du système modélisé, en vue de relevés statistiques au cours de la simulation. Ces compteurs servent aussi à introduire des RETARDS pour exprimer le temps pris par des opérateurs complexes dans une "unité" décrite de manière entièrement algorithmique.

La compatibilité LASCAR-CASSANDRE est basée sur l'équivalence des "unités" décrites à différents niveaux. Deux unités sont EQUIVALENTES, vues de l'extérieur si elles sont interchangeables sans aucune modification du reste du modèle de circuit. De manière plus précise elles doivent avoir :

- les mêmes entrées-sorties,
- les mêmes valeurs sur les sorties pour des valeurs identiques de leurs entrées.

A-I-5.3-BILAN DE CASSANDRE-LASCAR :

L'AVANTAGE du système LASCAR-CASSANDRE pour la simulation de circuits de grande complexité est de pouvoir intégrer l'aspect FONCTIONNEL à la description structurelle et de permettre la coexistence des deux types de descriptions.

La LIMITATION de point de vue de notre travail est l'absence de simulation de défauts sur le circuit.

A-I-6 BIBLIOGRAPHIE DE L'ANNEXE I.

(1) CAILLAT Jacques, TULLOUE R., ZIRPHILE J.,
"TAU-1 : Un outil pour le Test des Circuits Intégrés Logiques"
Revue Technique THOMSON-CSF - 1976.

(2) CAILLAT Jacques
"Contribution au test des Circuits Intégrés Logiques"
Thèse de 3ème cycle - ENSIMAG - 1976.

(3) MUTEL J., RAULT J.C., THUEL J., TULLOUE R.
"EPISODE, un programme pour la simulation à plusieurs niveaux et l'analyse de la propagation des défauts dans les Circuits Intégrés complexes"
Tomes 1 et 2, LETI-THOMSON-CSF, 1977.

(4) Frederik-J. HILL, Gerald R. PETERSON
"DIGITAL SYSTEMS : HARDWARE ORGANISATION AND DESIGN"
Chapitre 5 - JOHN WILEY - 1978.

(5) BORRIONE Dominique
"DESCRIPTION ET SIMULATION D'UNE ARCHITECTURE MULTIPROCESSEUR A L'AIDE DU LANGAGE LASCAR"
ENSIMAG - 1977.

(6) Ed THOMPSON, Anthony LEKKOS, Don ROOS, Richard Von BLUCHER
"TEGAS DESIGN LANGUAGE (TDL) - A SYSTEM FOR MODULAR DESCRIPTION AND TOP-DOWN/BOTTOM-UP VERIFICATION OF LSI AND VLSI"
IEEE - 1980.

(7) "CC-TEGAS 4" - Comprehensive Computing Systems and Services, Inc., Austin, TEXAS - USA - 1979.

ANNEXE II

LE TEST DE LA MAQUETTE

A-II-1 DEROULEMENT DE LA CONCEPTION D'UN CIRCUIT COMPLEXE : LE CIRCUIT CORRECTEUR D'ERREURS MEMOIRE

Durant la conception du CCE (Chap. III), nous avons utilisé une méthode combinée de conception ascendante et descendante.

Nous avons commencé la conception après la rédaction d'un ensemble de SPECIFICATIONS PROVISOIRES. Ces spécifications résultent d'une part de l'étude du marché pour ce circuit et d'autre part de l'expérience acquise dans des projets antérieurs.

Certaines de ces spécifications se révèleront impossibles à mettre en oeuvre, leur impact sur la structure fine du circuit ne pouvant être que difficilement apprécié à ce stade de la conception. Cette impossibilité ne se manifestera que dans des étapes de conception ultérieures.

Une fois cette première étape terminée, l'étude architecturale du futur circuit peut commencer. Cette étude tient compte des connaissances antérieures acquises dans la technologie choisie et cela peut aller jusqu'à la réutilisation de dispositifs ou de blocs déjà réalisés.

A ce stade, les spécifications doivent être mises à jour pour conduire à une version plus élaborée, la viabilité du circuit étant assurée. Deux types d'études doivent être alors entamés :

- A partir de l'architecture proposée, faire une description détaillée du circuit pour sa simulation.

- A partir des spécifications, élaborer des séquences pour un test fonctionnel.

A-II-2 SIMULATION LOGIQUE ET MAQUETTE DU CIRCUIT

La simulation logique sur ordinateur ou la construction de la maquette en laboratoire sont les solutions possibles selon les moyens dont on dispose.

Dans notre cas, nous avons commencé par une simulation logique détaillée des blocs critiques du circuit en utilisant le programme EPISODE (Cf. Annexe I).

Une fois ceci complété par la simulation du reste du circuit nous avons fait une validation de l'architecture provisoire du circuit. Validation logique seulement sans prendre en compte tous les aspects temporels fins. Pour cela nous avons éliminé la plupart des RETARDS des blocs pour accélérer le procédé de SIMULATION et diminuer le coût en temps de calcul.

La simulation logique du circuit a été faite sur une description structurelle complète de la P.O. et sur une description de la P.C. très schématique.

Une fois corrigées, certaines erreurs de conception (5 erreurs trouvées sur une simulation de 5 000 transistors environ) nous avons décidé la fabrication d'une maquette en technologie TTL ("SSI-MSI") pour étudier les problèmes de séquencement posés par la génération des commandes et des horloges (P.C. détaillée).

La validation de cette maquette nécessitait des séquences de vecteurs d'entrée, séquences d'entrée ELABOREES MANUELLEMENT. Ces séquences ont pu être vérifiées grâce à la simulation logique précédente.

Nous avons démarré avec des séquences de longueur minimale et au fur et à mesure que l'architecture se précisait, nous avons augmenté progressivement la longueur des séquences.

La maquette nous a permis d'étudier les problèmes posés par l'initialisation du circuit. Pour la simulation logique, en effet, l'état initial des variables est : soit inconnu ("X"), soit positionné à "1" ou à "0".

Sur la maquette, par contre, l'état initial de ces mêmes variables dépend de phénomènes très complexes. En absence d'une remise à zéro générale à la mise sous tension, il a fallu améliorer la conception du coeur de la P.C.

Les erreurs dues aux retards réels des éléments ont été éliminées sur la maquette. Ensuite on a adapté les solutions trouvées sur la maquette en TTL à notre circuit qui sera fabriqué en NMOS.

A-II-3 VALIDATION DE L'ARCHITECTURE ET PROGRAMME DE TEST

Arrivés à cette étape, une conception presque définitive avait été validée mais le test restait à son niveau minimal.

Nous avons alors appliqué aux entrées, différentes séquences étudiées pour un test approfondi.

Rappelons que la maquette est une approximation du circuit réel, et de plus, une version réduite : la P.C. est complète et détaillée, mais la P.O. originale portant sur 16 bits a été réduite à 4 bits (la P.O. ayant été validée de façon complète par la simulation logique).

Le programme de test limité à une voie de 4 bits apparaît comme une version particulière du programme à appliquer plus tard sur le circuit final.

Les seuls signaux qu'on pouvait vérifier complètement sur cette maquette réduite étaient les signaux de séquencement. Signalons toutefois que la maquette fonctionnait à une fréquence d'horloge différente de celle des spécifications du circuit. Cette fréquence était ralentie par la vitesse de fonctionnement des appareils utilisés avec la maquette. Un synoptique résumant les étapes de notre démarche de conception est donné en Figure A-II-1.

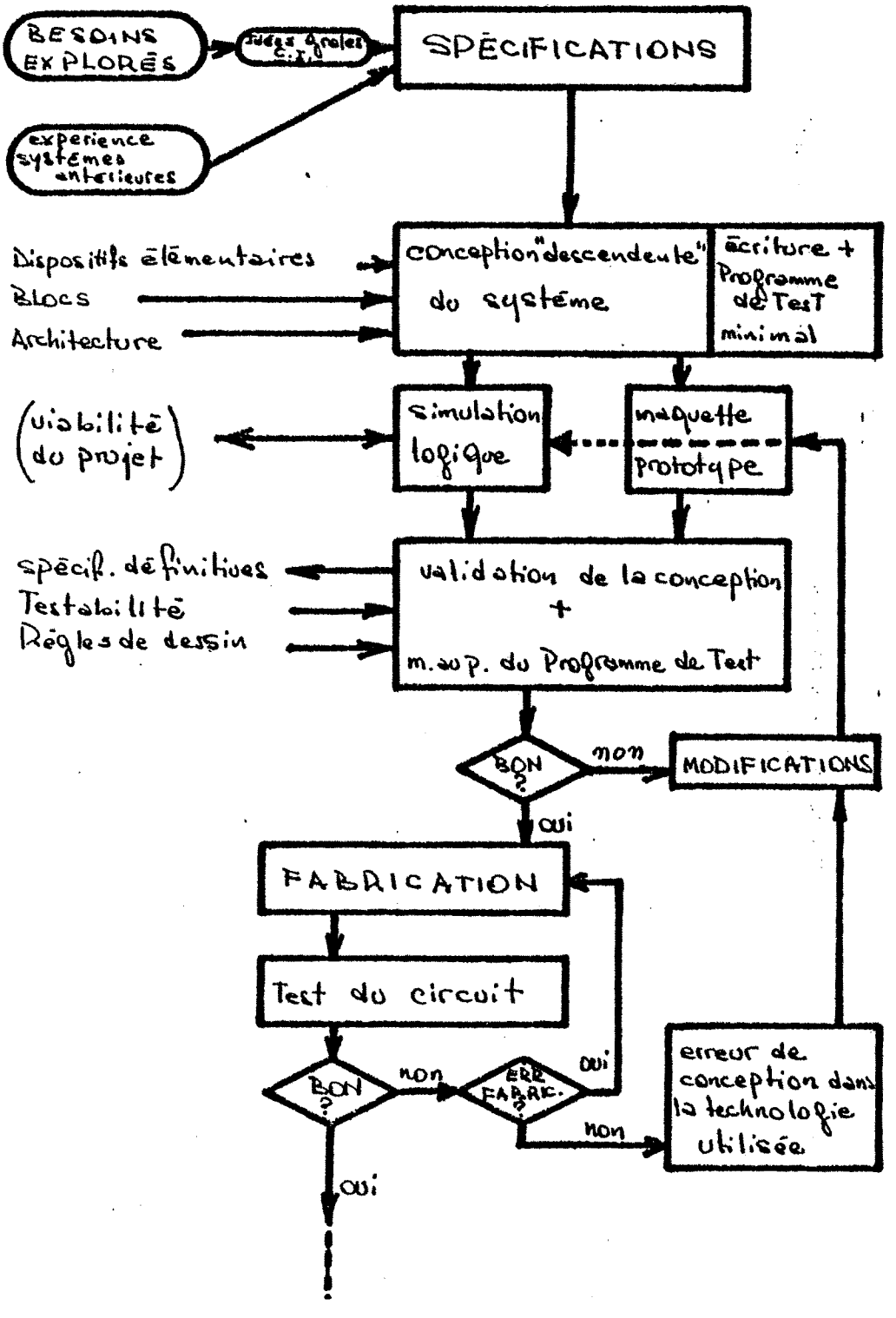


FIGURE A-II-1 : PROJET INDUSTRIEL D'UN CIRCUIT

A-II-4 VALIDATION DE LA MAQUETTE

La réalisation physique de la maquette a été faite avec des circuits intégrés disponibles dans le commerce, en majorité "TTL". Elle a été testée avec une séquence de vecteurs d'entrée et de commandes établie à la main.

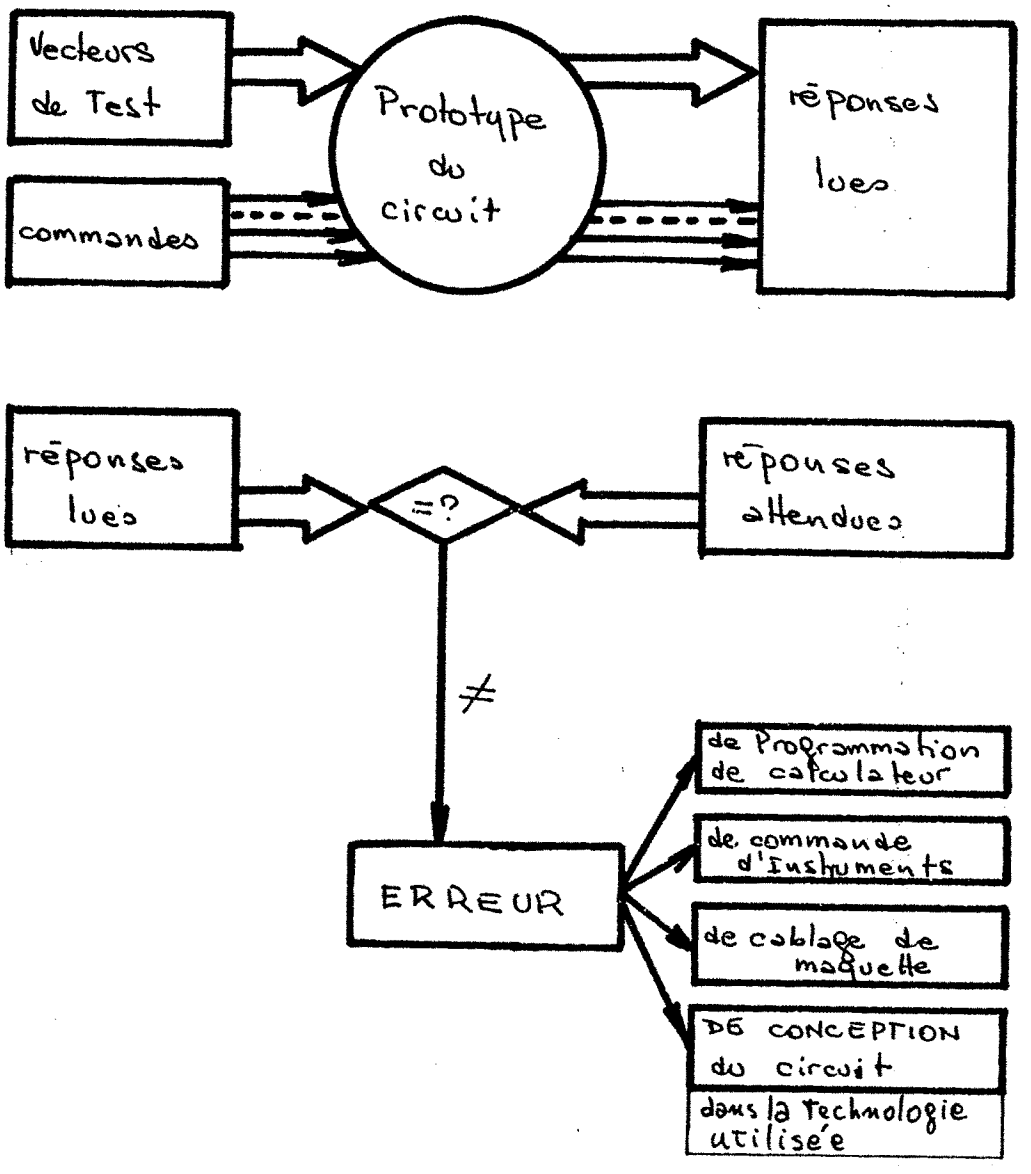


FIGURE A-II-2 : ESSAI DE LA MAQUETTE

Les réponses attendues du circuit sont celles obtenues lors de la simulation logique.

Si pendant la mise au point de la maquette nous avons trouvé des réponses autres que les réponses "attendues", nous avons mis en cause dans l'ordre suivant :

- la programmation du calculateur ;
- les commandes de contrôle envoyées sur les instruments à travers le bus IEEE 488, ou bien erreur de synchronisme;
- le câblage de la maquette ;
- la conception du circuit.

Cet ordre d'examen s'est révélé correct ; les erreurs détectées étaient en effet le plus souvent des erreurs de programmation. Nous avons ainsi éliminé deux erreurs de conception de la P.C.

Remarquons que sur la maquette, il y a peu de limitations pour rajouter des points de test, facteur important pour améliorer la testabilité (Cf. Chap. II); mais cette amélioration est illusoire : on ne peut pas implanter les mêmes points d'observation sur le circuit réel.

L'INSTRUMENTATION utilisée pour la validation de la maquette est assez classique (*).

a) GENERATEUR DE MOTS (pour données et commandes).

b) ANALYSEUR LOGIQUE pour visualiser directement les chronogrammes des réponses.

c) MINI-CALCULATEUR pour surveiller toute l'opération et pour générer aussi certaines données en synchronisme avec a).

d) BUS IEEE 488 pour relier générateur, analyseur et calculateur.

e) CARTES D'INTERFACE pour éviter des conflits sur les bus.

Le calculateur disposait en mémoire de l'ensemble des vecteurs de test envoyés sur la maquette, plus l'ensemble de réponses attendues de celle-ci.

L'ensemble maquette plus instruments de service simule un système : Microprocesseur + (circuit correcteur d'erreur) + mémoire (Cf. première figure du Chapitre III).

(*) Voir Page 178.

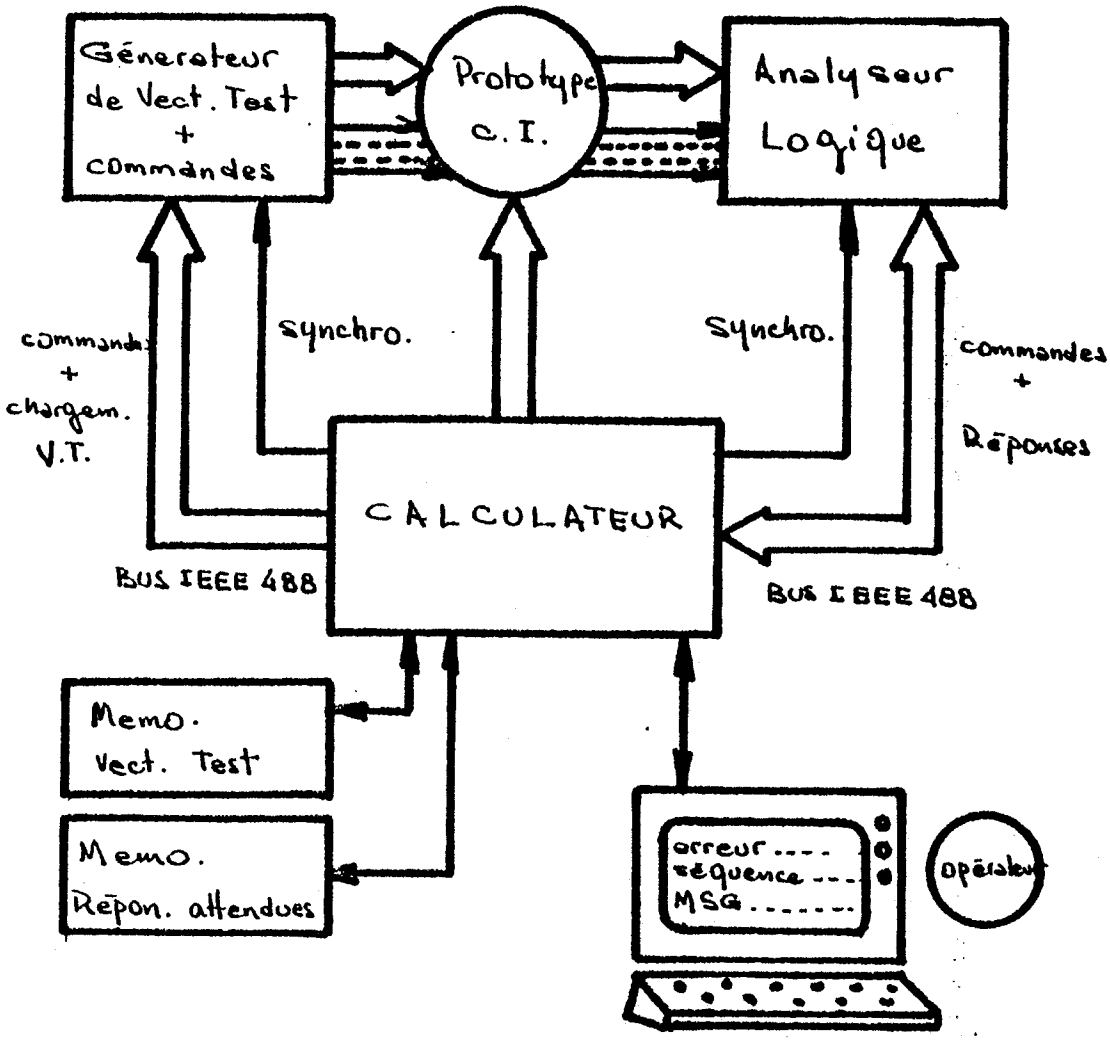


FIGURE A-II-3 : EQUIPEMENT POUR LA VALIDATION DE LA MAQUETTE

Pendant la simulation d'un cycle d'ECRITURE EN MEMOIRE, le générateur et le calculateur prennent la place du microprocesseur alors que l'analyseur logique simule la mémoire.

Pour la simulation d'un cycle de LECTURE DE MEMOIRE, générateur de mots et calculateur remplaçant la mémoire, l'analyseur joue le rôle du microprocesseur.

Dans chaque cas la carte interface aiguille les données et évite les conflits sur le bus.

La PREMIERE PARTIE de la validation de la maquette utilisait une programmation rigide de la séquence envoyée sur la maquette.

Cette séquence se composait, par exemple, de douze cycles d'horloge (deux cycles "vides" de machine) pour assurer (dans le pire des cas) à l'automate de contrôle, un état initial connu : "000000".

Puis on programmait le registre de mode d'opération de la maquette. Ensuite, on envoyait des cycles LECTURE/ECRITURE en mémoire avec/sans erreur et avec/sans lecture des registres internes de message dans la maquette.

La capacité de mémoire du mini-calculateur étant réduite nous avons dû segmenter notre programme. De plus, le fait d'utiliser du BASIC interprété comme langage nous procurait certains avantages de programmation du bus IEEE 488 mais réduisait la fréquence d'opération.

Nous n'avons pas attaché une grande importance à cette réduction de fréquence parce que de toute façon les fronts de montée et de descente de la maquette en TTL et du circuit réel en NMOS, ne seront pas les mêmes.

LA DEUXIEME PARTIE de la simulation mettait en oeuvre une programmation plus flexible. On pouvait changer l'ordre des cycles machine à envoyer sur la maquette.

Dans tous les cas, la programmation du calculateur comprenait les étapes suivantes :

- Vérification des instruments connectés sur le bus IEEE 488 pour éliminer une première cause de réponses fausses.
- Choix du vecteur de test à envoyer sur le bus de données pour chaque cycle de machine. A l'aide d'un menu "standard" on pouvait choisir le cycle et les bits à changer dans la séquence de vecteurs de test pour ce cycle.
- Programmation en deux étapes du générateur de mots à travers le bus IEEE 488 : d'abord les commandes pour le mode d'opération, fréquence, synchronisme etc, et après la séquence de vecteurs de test modifée.
- Programmation de l'analyseur logique.
- Déclenchement de la série de cycles machine.
- Lecture de l'analyseur logique.
- Comparaison entre réponses lues et réponses attendues.
- Diagnostic d'erreur sur les cycles.

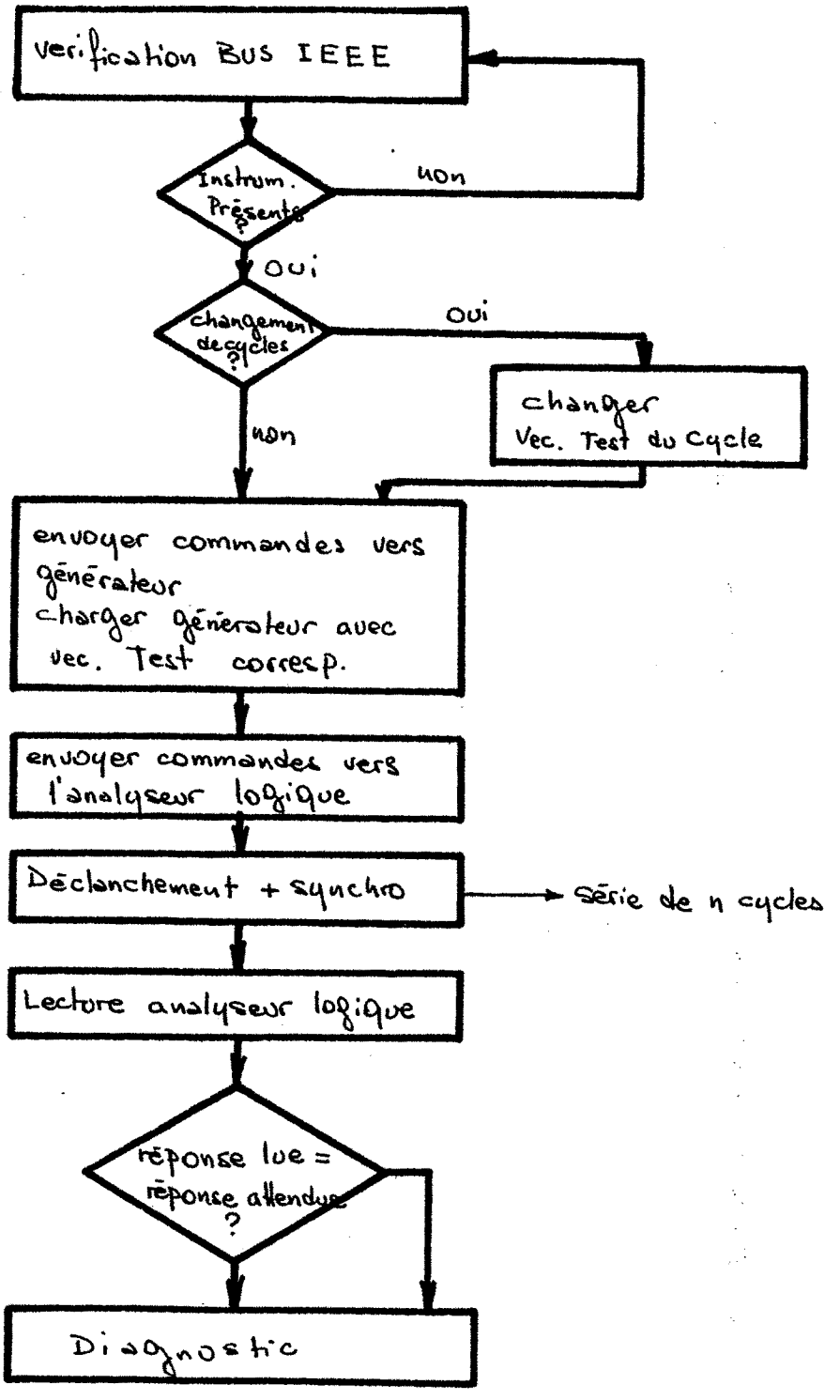


FIGURE A-II-4 : PROGRAMME DE VERIFICATION DE LA MAQUETTE

A-II-5 CONCLUSION

La mise au point de la maquette exige l'écriture d'un programme de test minimal. Les séquences de test utilisées nous permettent :

a) de valider l'architecture choisie, ce qui complète les résultats de la simulation logique sur ordinateur.

b) de bien roder la séquence de commandes à envoyer sur la P.C. du circuit pour le futur programme de test.

c) de déterminer la succession de cycles machine la plus intéressante pour envoyer les vecteurs de test obtenus lors de la simulation logique.

Mais le programme de test de la maquette ne constitue pas un programme de test du circuit réel :

a) On ne peut pas prendre exactement en compte les fronts de montée et de descente des signaux, étant donnée la différence de technologie entre maquette et circuit intégré définitif.

b) On ne peut pas essayer tous les vecteurs, et les VRAIS vecteurs de test, étant donnée la P.O. en version réduite construite pour la maquette.

c) On peut disposer sur la maquette de points de commande et d'observation dont on ne disposera pas sur le circuit réel.

Cependant, comme la maquette permet d'éliminer les erreurs de conception, nous pouvons espérer qu'il y aura seulement des erreurs de conception "locales" dans le circuit réel, dûes à l'implantation et à la fabrication.

INSTRUMENTS UTILISES POUR LA VALIDATION DE LA MAQUETTE

- Générateurs de mots de 16 bits : "H.P. 8170A".

- Analyseurs logiques : DOLCH-LAM. 4850

  SCHLUMBERGER-7600.

- Calculateur : MINC-11

  Console VT 105

  Unités de sortie numérique DIGITAL

  IMPRIMANTE

* * * * *

TABLE DES MATIERES

Page :

dernière page de la thèse

AUTORISATION DE SOUTENANCE

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

Vu les rapports de présentation de

. *Madame Gabrièle SAUCIER, Professeur*
. *Monsieur Christian KUBIAK, Ingénieur*

Monsieur GOBBI José Maria

est autorisé à présenter une thèse en soutenance pour l'obtention du diplôme de DOCTEUR INGENIEUR, Spécialité "Génie Informatique".

Fait à Grenoble, le 25 novembre 1981

Le Président de l'I.N.P.-G.

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