

lex

وزارة التعليم العالي  
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR

المدرسة الوطنية المتعددة التقنيات  
BIBLIOTHEQUE — المكتبة  
Ecole Nationale Polytechnique

ECOLE NATIONALE POLYTECHNIQUE

DEPARTEMENT : ELECTRONIQUE

**PROJET DE FIN D'ETUDES**

**SUJET**

ETUDE ET REALISATION  
D'UN SIMULATEUR  
DE FONCTIONS LOGIQUES  
A PROM

Proposé par :

Farah

Etudié par :

B. Mébarki

Dirigé par :

Farah

PROMOTION : JUIN 90

## ECOLE NATIONALE POLYTECHNIQUE

DEPARTEMENT : ELECTRONIQUE

# PROJET DE FIN D'ETUDES

### SUJET

ETUDE ET REALISATION  
D UN SIMULATEUR  
DE FONCTIONS LOGIQUES  
A PROM

Proposé par :

Farah

Etudié par :

B. Mébarki

Dirigé par :

Farah

PROMOTION : JUIN 90

REMERCIEMENTS

Je tiens à exprimer ma profonde gratitude à Monsieur FARAH pour l'intérêt et surtout pour son aide précieuse tout au long de l'élaboration de ce travail .

Mes sincères remerciements à tous mes professeurs pour leur bonne disponibilité durant toute ma formation .

Mes vifs remerciements à mes amis Ahmed , Djamel et M. Sayah pour leurs soutiens matériels .

Sans oublier bien évidemment ma très chère amie Malika pour son soutien moral .

Qu'ils trouvent ici l'expression de ma profonde reconnaissance :

L'élève ingénieur :

B . Mebarki

DEDICACES

*Je dédie ce modeste travail*

*A vous*

- Mes très chers parents*
- Mes chers frères et ma chère soeur F/Z*
- Mes amis et tous ceux qui me sont chers*

*Bencherki*



## S O M M A I R E

### I N T R O D U C T I O N

### C H A P I T R E I

#### RAPPEL SUR LA TECHNOLOGIE DES MEMOIRES MORTES

I.1 )	Introduction.....	2
I.2 )	Généralités.....	2
	I.2.1/ Mémoires externes.....	2
	I.2.2/ Mémoires internes.....	2
	I.2.3/ Mémoires vives.....	2
	I.2.4/ Mémoires mortes.....	3
I.3 )	Technologie des mémoires mortes.....	3
	I.3.1/ Les mémoires mortes ( ROM ).....	3
	I.3.2/ Les mémoires programmables ( PROM ).....	5
	I.3.2.a/ PROM à fusibles.....	6
	I.3.2.b/ PROM à diodes en tête-bêche.....	7
	I.3.2.c/ Techniques de programmation des PROMs	
	I.3.3/ Les mémoires REPROMs.....	7
	I.3.3.a/ Mémoires EPROMs.....	7
	I.3.3.b/ Mémoires EEPROMs.....	10
I.4 )	Comparaison entre les différentes technologies des mémoires à semi-conducteur. ....	11
	I.4.1/ Technologie MOS.....	11

I.4.2/ Technologie CMOS . . . . . 11

I.4.3/ Technologie bipolaire . . . . . 11

I.5 ) Organisation des cellules mémoires . . . . . 11

    I.5.1/ Structure matricielle . . . . . 11

    I.5.2/ Structure linéaire . . . . . 10

    I.5.3/ Capacité et format d'une mémoire . . . . . 12

    I.5.4/ Vitesse d'une mémoire . . . . . 13

    I.5.5/ Consommation d'une mémoire . . . . . 13

    I.5.6/ Alimentation des mémoires . . . . . 13

C H A P I T R E    I I

SIMULATION DES FONCTIONS LOGIQUES PAR DES SYSTEMES COMBINATOIRES

II.1 ) Conception des systèmes combinatoires . . . . . 15

II.2 ) Utilisation d'une PROM comme générateur de fonctions logiques . . . . . 18

    II.2.1/ Exemple . . . . . 18

    II.2.2/ Généralisation de la méthode . . . . . 20

C H A P I T R E    I I I

MATERIALISATION DES FONCTIONS LOGIQUES PAR UTILISATION DES SYSTEMES SEQUENTIELS PROGRAMMABLES

III.1 ) Divers formes d'un système combinatoire . . . . . 24

III.2 ) L'arbre de décision logique . . . . . 24

    III.2.1/ L'obtention de l'arbre de décision logique . 24

        III.2.1.b/ Représentation tabulaire de l'arbre . . . . . 31

III.2.1.c/ Códage du tableau . . . . . 33

III.2.1.d/ Implantation de la machine séquentielle . 36

**CHAPITRE IV :**

DESCRIPTION DE LA REALISATION PRATIQUE

IV.1 ) Introduction . . . . . 39

IV.2 ) Description . . . . . 39

IV.2.1/ Le décódeur ( 4 vers 16 ) . . . . . 39

IV.2.2/ Le multipléxeur ( 8 vers 1 ) . . . . . 39

IV.2.3/ Les quadruples multipléxeurs ( 2 vers 1 ) . . . . 40

IV.2.4/ Les quadruples bascules "d" avec Reset . . . . . 41

IV.2.5/ Less quadruples portes NAND buffers . . . . . 41

IV.2.6/ Trigger de Schmit . . . . . 42

**CHAPITRE V :**

TEXTES PEDAGOGIQUES

V.1) Partie théorique . . . . . 44

V.1.a/ Matérialisation des fonctions logiques par utilisation des systèmes séquentiels programmables

\* Arbre de décision

\* Représentation tabulaire de l'arbre

\* Codage du tableau

V.1.b/ Matérialisation par utilisation des systèmes combinatoires

V.2/ Questions théoriques . . . . . 44

Réponses

V.3 ) Manipulation . . . . . 46

V.3.a/ Systemes séquentiels

V.3.b/ Systemes combinatoires purs

ANNEXE . . . . . 55

BIBLIOGRAPHIE . . . . . 67



## I N T R O D U C T I O N

Avant la venue des circuits intégrés à large et moyenne intégration , la simulation de fonctions logiques se faisait à l'aide de circuits à base de composants discrets .

Profitant des avantages procurés par l'intégration , on a essayé de réduire toute cette complexité et on a ainsi étudié et réalisé un dispositif de simulation , plutôt, de matérialisation de fonctions logiques à base d'une PROM qui elle est discrete .

Ce travail n'a donc pas la prétention de présenter ici , une maquette réalisant les meilleurs performances mais cela dans un but purement pédagogique .

Il complètera ainsi les cours d'électronique digital dispensés ici à E N P , et fera une illustration précieuse de la logique combinatoire d'une part et séquentielle d'autre part . Il permettra aux étudiants d'acquérir des connaissances physiques de la programmation ( matérialisation et programmation des fonctions logiques) .



# CHAPITRE : I

RAPPELS SUR LA TECHNOLOGIE

DES

MEMOIRES MORTES

## I-1) INTRODUCTION

Les mémoires sont des circuits électroniques qui peuvent enregistrer, conserver ou restituer plusieurs informations binaires. Dans tous les calculateurs électroniques, les systèmes de mémoires occupent une place de choix. On distingue les mémoires externes et les mémoires internes. Les mémoires externes sont réalisées jusqu'à présent sur des bandes magnétiques ou sur des disques magnétiques. Les mémoires internes (c'est à dire les mémoires réunies, au de vue construction, aux blocs électroniques de l'appareillage), qui étaient réalisées jusqu'à ces derniers temps avec des noyaux de ferrites utilisent actuellement presque exclusivement des circuits intégrés. Les mémoires assurent dans leurs applications secondaires le rôle de générateur de fonction logique.

## I 2) GENERALITES

I 2 1/ Les mémoires externes à support magnétiques se caractérisent par une durée de stockage indéfinie de l'information enregistrée, sans être alimentées, ainsi que par une capacité pratiquement illimitée en bits.

I 2 2/ Les mémoires internes sont destinées essentiellement à la conservation des données intermédiaires au cours des opérations arithmétiques et logiques (mémoires opérationnelles ou vives), ainsi qu'à la conservation des petits programmes standards nécessaires à la résolution des problèmes types sur un ordinateur numérique donné. (mémoires permanentes ou mortes).

I 2 3/ Les mémoires vives à lecture et écriture à accès aléatoire ou encore R A M (de l'anglais Random Access Memories) se caractérisent par la possibilité d'effectuer rapidement et alternativement l'entrée et la sortie

(l'écriture et la lecture) l'information et ceci avec l'accès à toute cellule distincte de la mémoire .

I 2 4 / Les memoires mortes à l'écriture seule ou encore R O M (de l'anglais Read Only Memories) sont utilisées principalement pour la lecture de l'information qui y est enregistrée .Quant à l'écriture d'une information,elle se fait soit<<une fois pour toute>>,soit en tout cas très rarement.

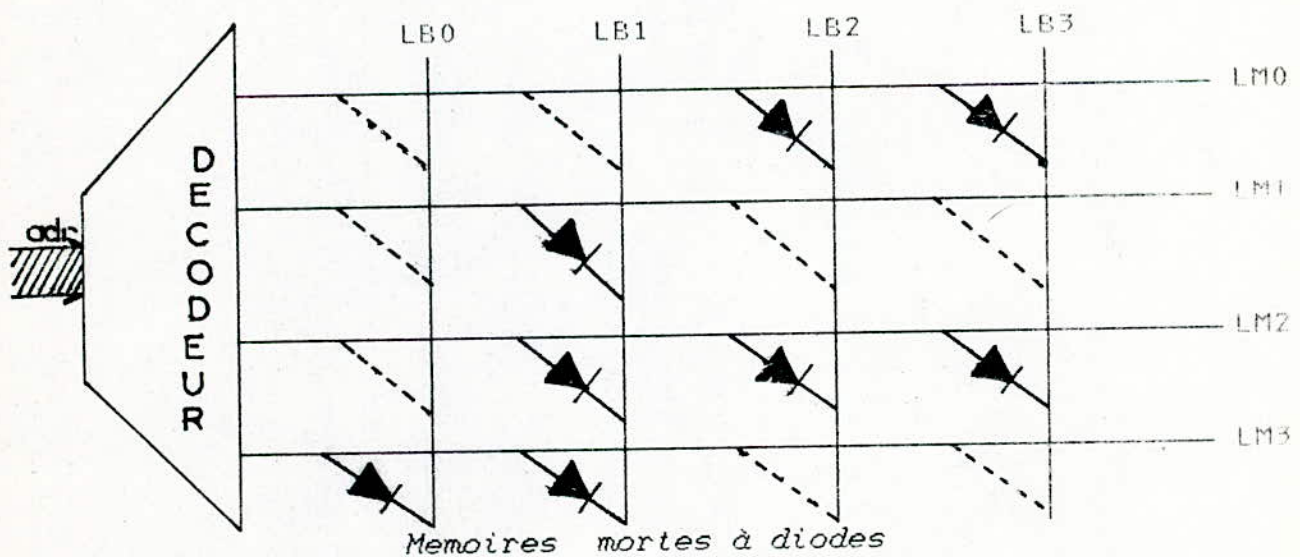
### I.3 / TECHNOLOGIE DES MEMOIRES MORTES

#### I.3.1./ Les memoires mortes (R O M):

Ces memoires sont encore appelées mémoires permanentes,mémoire passives ou mémoires fixes,se caractérisent par le fait qu'il enregistrement d'une information se fait soit une fois pour toute,soit représente une opération spéciale effectuée très rarement.

L'utilisation d'une mémoire morte consiste donc à lire une information enregistrée.

Le schéma type d'une mémoire morte à diodes est représenté comme suit:





Elle a la structure d'une matrice dont les rangées sont constituées par les lignes de mot (LM0, LM1, LM2 et LM3) et les colonnes par les lignes de bit (LB0, LB1, LB2 et LB3). Chacune des lignes de mot conserve un code déterminé :

0011, 0100, 0111, 1100 (pour cet exemple).

L'enregistrement du code (effectué en principe une seule fois) est obtenu à l'aide de diodes connectées entre les lignes de mot et les lignes de bit qui doivent contenir (lors de la lecture) un 1 logique

Supposons que le décodeur d'adresse ait choisi la ligne de mot LM1, alors une tension appliquée à cette ligne sera transmise à la ligne de bit LB2, tandis que les tensions sur les lignes de bits LB0, LB1 et LB3 seront nulles. Ceci signifie que la lecture parallèle de l'information sur toutes les quatre lignes de bit donne le code (ou le mot) 0100 enregistré dans la rangée choisie. Les diodes sont réalisées dans tous les noeuds de la matrice qui est livrée vierge. Il ya deux types de R O M :

- \* R O M programmable à la fabrication
- \* R O M programmable par l'utilisateur (PROM , EPROM , .... )

Remarque :

Bien que les mémoires mortes à diodes soient très simples, elles ont un inconvénient : le courant nécessaire dans les lignes de bit doit être assuré par le décodeur qui transmet ce courant à travers la ligne de mot

Pour rendre plus facile le fonctionnement du décodeur, on remplace les diodes par des éléments amplificateurs, c'est-à-dire par des transistors :



\* / Par emploi des transistors biolaires

Dans ce cas la de mot assure le passage du courant de base qui est  $(\beta+1)$  fois plus petit que le courant de l'émetteur qui alimente la ligne de bit. Par conséquent la puissance du décodeur se trouve réduite de plusieurs dizaines de fois .

\* / Par emploi des transistors MOS

Dans ce cas, on a un gain encore plus élevé, car le circuit de grille ne consomme en général aucun courant .

L'emploi des MOS ouvre de nouvelles perspectives dans l'utilisation des memoires mortes :

Premierement :

Pour l'enregistrement d'une information , on peut utiliser non la destruction des connexions mais un procédé plus "fin" : La memoire morte est livrée par le constructeur sans métallisation des grilles, c'est alors à l'utilisateur d'assurer la métallisation des grilles seulement des transistors qui doivent transmettre un "1" à la ligne de bit. Les grilles des autres transistors ne seront pas reliées aux lignes de mot de sorte que les transistors seront inopérants .

Deuxiemement

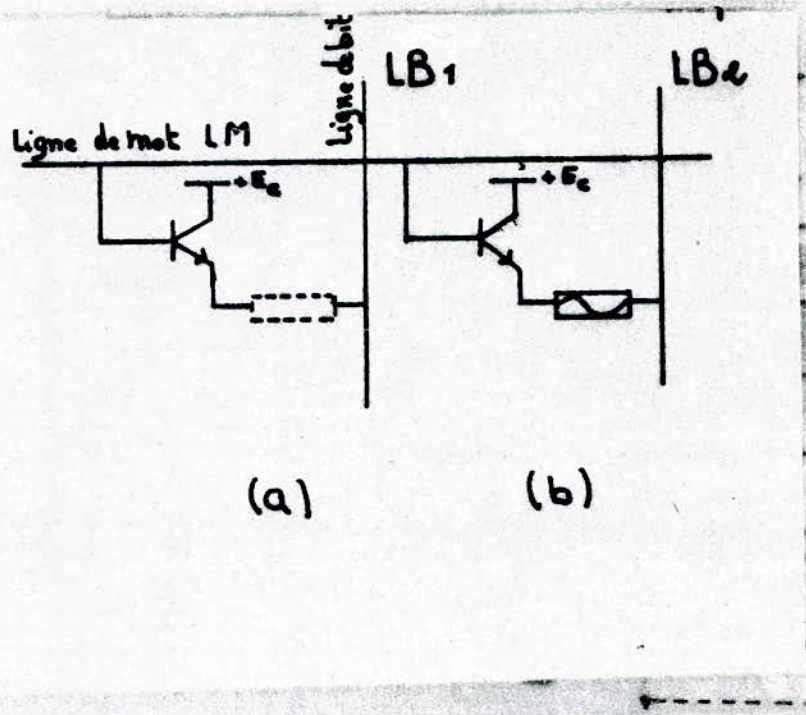
L'emploi des transistors MOS permet la réalisation des memoires dites semi-permanentes ou programmables (ROM programmable ou PROM) dans lesquelles on peut changer , de temps à autre l'information enregistrée .

### I.3.2. / Les memoires programmables P R O M

Il existe deux types de PROM :

- \* ) PROM à fusibles
- \* ) PROM à diodes en tete-beche

I.3.2.a/ PROM à fusibles

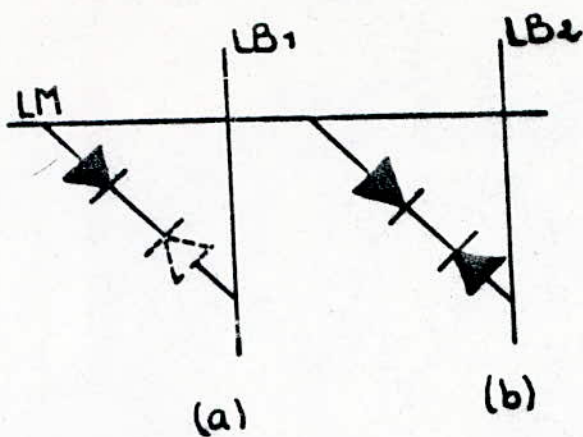


Pour le (a) ,le fusible est brulé, donc le courant ne passera pas entre LM et LB1 (ce qui donnera un "0" logique sur LB1 (dans le cas ou LM sera sélectionnée par le decodeur d'adresses ) Pour le (b) ,le fusible n'est pas brulé ,donc le courant passera et on aura un "1" logique sur la sortie LB2 (dans le cas ou LM est sélectionnée par le decodeur d'adresses ).

Remarque :

Les transistors bipolaires peuvent etres remplacés par des MOS pour économiser plus d'énergie car le circuit de grille ne consomme en general aucun courant .

### 1.3.2.b/ PROM à diodes en tete-bêche



Dans le cas (a), une diode a claqué donc on aura un circuit ouvert entre LM1 et LB1 (un "1" logique en sortie ).  
Dans le cas du (b), les deux diodes sont intactes, il n'y aura donc pas un passage de courant, ce qui donnera un "0" logique en sortie (si LM est sélectionnée )

### 1.3.2.c / Technique de programmation des PROMs

Dans les deux types , la programmation se fera par l'utilisateur en se servant d'un programmeur de PROM qui brulera soit le fusible , soit entrainera le claquage d'une jonction (diode) et cela par application d'un bref courant de quelques mA que bien sur , ni le fusible ni la jonction ne supportera son intensité .

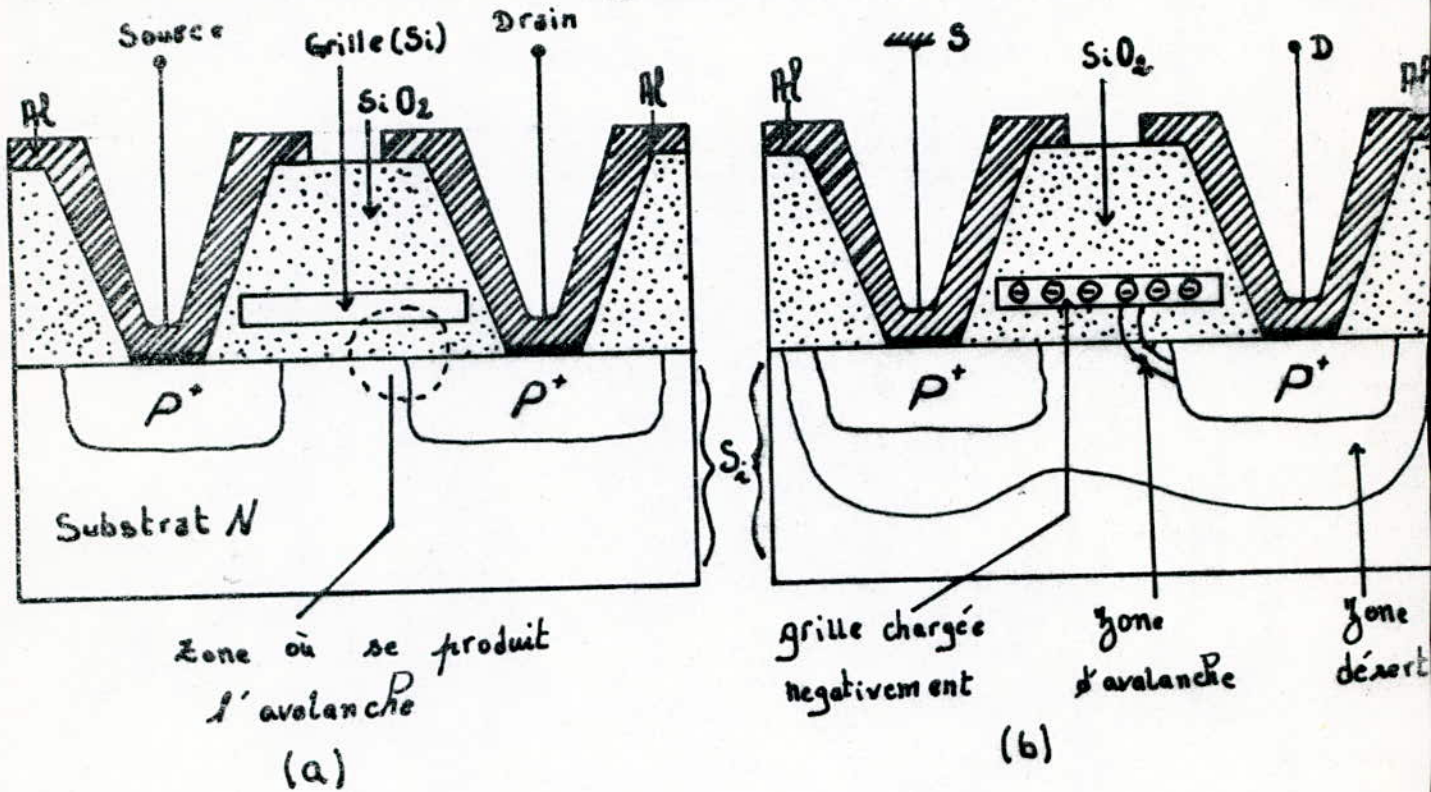
### 1.3.3.) Les memoires R E P R O M

Les REPRM renferment les EPROM et les EEPROM



### I.3.3.a / Memoire EPROM

Ce sont des ROMs effaçables programmables .La figure suivante montre une cellule de memoire unitaire de type F.A.M.O.S (Floating gate Avalanche injection MOS ),c'est à dire MOS à grille flottante chargée par un phénomène d'avalanche à travers le dielectrique



(a)

(b)

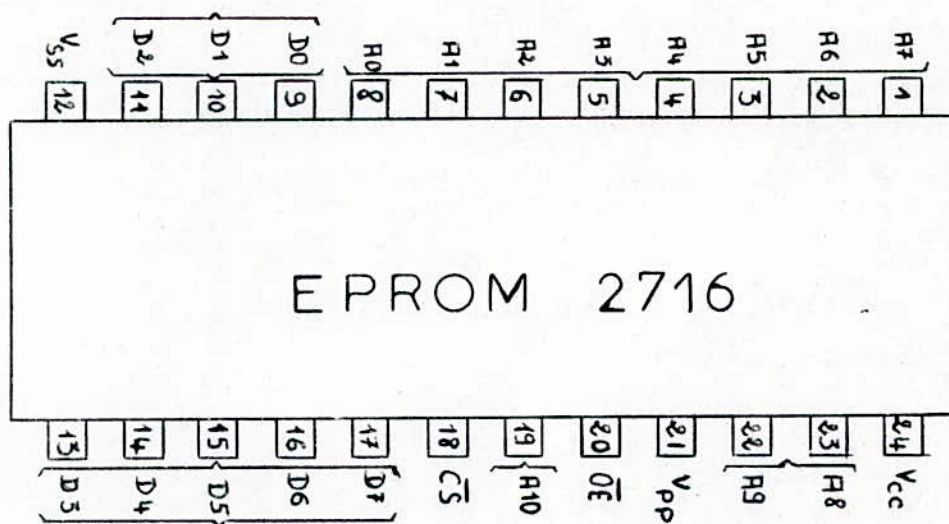
C'est un MOS dont la grille est isolée de contact électrique .Elle est isolée du substrat par une couche de SiO<sub>2</sub> .Elle est au silicium polycristalin . Elle est donc flottante .

La figure (b) montre comment on charge la grille :on applique une tension  $-V < -30$  Volts entre le drain et la source , ce qui fait passer en regime d'avalanche la jonction P N drain-substrat .Pendant ce régime , des électrons d'énergie suffisante arrivent à traverser l'épaisseur de  $0,1 \mu\text{m}$  de SiO<sub>2</sub> et chargent la grille négativement . Après l'avalanche , la grille reste chargée négativement et crée dans le substrat N une couche d'inversion de



type P qui relie la source et le drain .

La presence ou l'absence de charges sur la grille est détectée par la mesure de la conductance entre la source et le drain . Ainsi donc pour programmer , on utilise une impulsion de  $-50V$  et  $1mA$  ; ce qui donne un courant de  $5mA$  et permet de stocker une charge de  $3 \cdot 10^{-7}$  Coulomb par cellule . Il y a maintenant des mémoires ou on est pas obligé d'avoir des impulsions étroites . C'est le cas de la 2716 : qui est une EPROM de  $2K \times 8$ .son brochage est donnée par la figure suivante



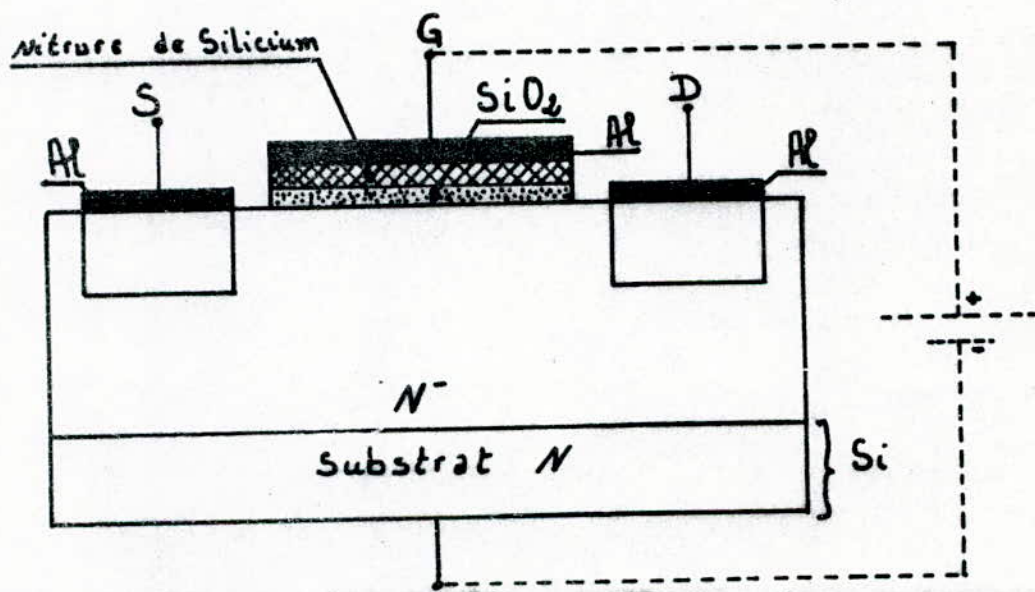
- \* A0 à A10 bus d'adresse de la mémoire. ce son des entrées.
- \* D0 à D7 bus de données de la mémoire.ils sont à lecture seul(sorie)
- \* Vcc et Vss (broche 24 et 12 ) : ligne d'alimentation de la mémoire Vcc=+5 V et Vss= masse.
- \* CS (BROCHE 18) ou ship select :broche de selection de boitier. Un niveau bas sur cette entrée signal à la mémoire que le micro-processeur veut dialogué avec elle . Cette broche a pour effet de passer le bus de données de la mémoire dans un état haute impédence lorsqu'elle est au niveau 1 logique (+5. v).
- \* OE (broche 20) ou output enable : lorsqu'elle est à 1,cette ligne permet une diminution de la consommation du boitier pour alimenter, par exemple , le boitier avec une pile ( sauvgarder de la RAM.alimentation coupée ). Au niveau bas, cette ligne à le même effet que le CS.

\*  $V_{pp}$  (broche 21) : cette ligne est utilisée pour programmer la mémoire 2716. En mode normal, cette ligne est au +5 V. En mode programme cette ligne est à +25 V.

Avant de reprogrammer, on fait disparaître les charges électriques stockées sur la grille en illuminant la cellule à l'aide de rayons UV (si la cellule est encapsulée) ou aux rayons X (s'elle ne l'est pas) ; un courant photoélectrique s'établit dans l'isolant et décharge la grille.

### I.3.3.b / Les memoires EEPROMs

Ce sont des ROM effaçables et programmables électriquement. Pour ce type de memoires, on fait appelle aux transistors MNOS (Metal Nitrure Oxyde Silicium) qui se presente comme suit :



L'écriture est effectuée en appliquant une importante tension positive entre la grille et le substrat. De ce fait des électrons se trouveront coincés dans la couche se trouvant entre le silicium et le SiO<sub>2</sub>. Ceci entraîne la création d'un canal conducteur entre la source et le drain et par conséquent le transistor devient conducteur.



L'effacement se fera par une simple application d'une tension inverse entre la grille et le substrat , ce qui aura pour effet la chasse des électrons emprisonnés.

L'avantage que présente les EEPROM sur les EPROM est que l'effacement est bien spécifique (ou selectif ) et non general .

## I.4 / COMPARAISON ENTRE LES DIFFERENTES TECHNOLOGIES DES MEMOIRES A SEMI-CONDUCTEUR

### I.4.1 / Technologie M O S

Les circuits MOS sont des transistors à effet de champ qui permettent , grace à leur taille réduite , des densités d'integration élevées et grace au hautes impédances mises en jeu , des consommations faibles . Mais les capacités parasites et ces hautes impedances donnent des constantes de temps plus élevées , ce qui diminue la vitesse de fontionnement .

### I.4.2 / Technologie C M O S ou MOS Complementaire

En associant des transistors MOS canal P et des MOS canal N , on obtient des strictures consommant très peu de courant , ce qui amène des économies importantes vue que la memoire presente la majeure partie d'un systeme . Mais la taille des cellules est grande , ce qui ne permet pas de grandes densités d'integration des CMOS .

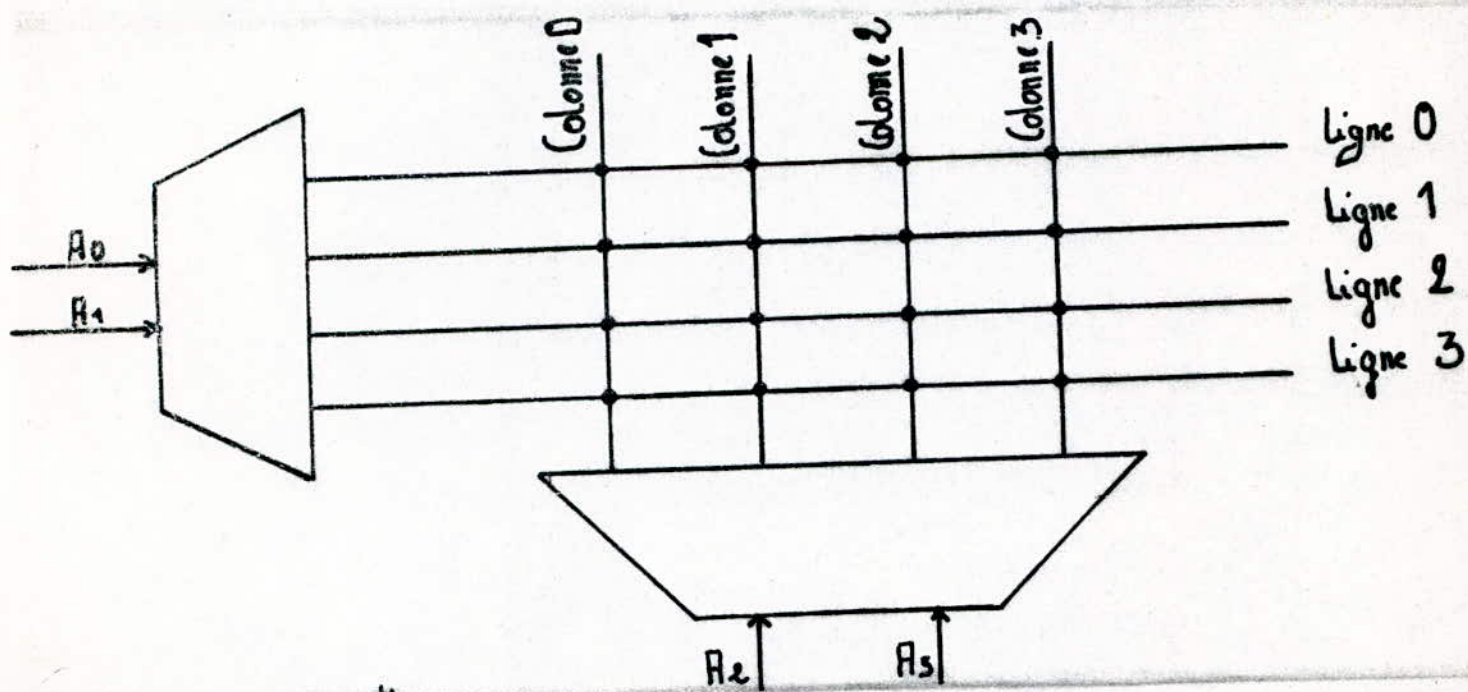
### I.4.3 / Technologie biolaire

Elle se caracterise par :La consommation est importante , la vitesse élevée ,et des densités d'integration plus faibles qu'en technologie MOS .

## I.5 ) ORGANISATION DES CELLULES MEMOIRES

### I.5.1 / STRUCTURE MATRICIELLE

Les cellules sont organisées en matrices ( M, N ) . Chaque cellule est repérée par un numéro de ligne et un numéro de colonne ( les deux numéros donneront l'adresse de la cellule )



structure matricielle à  $2^4$  elements binaire

Le découpage en matrice carrée n'est pas une obligation , et, un découpage en matrice rectangulaire peut exister aussi .

### 1.5.2 / STRUCTURE LINEAIRE

Une memoire F I F O ( First In First Out ) a une structure lineaire . Il s'agit d'une juxtaposition de registre à decalage . On a accès au premier étage et non au dernier . :

### 1.5.3 / CAPACITE ET FORMAT D'UNE MEMOIRE

Le nombre total d'éléments binaires donnera la capacité d'une memoire et leur arrangement donnera le format . .



C H A P I T R E

II

MATERIA LISA TION DES FONCTIONS  
LOGIQUES COMBINATOIRES PAR DES  
MEMOIRES MORTES

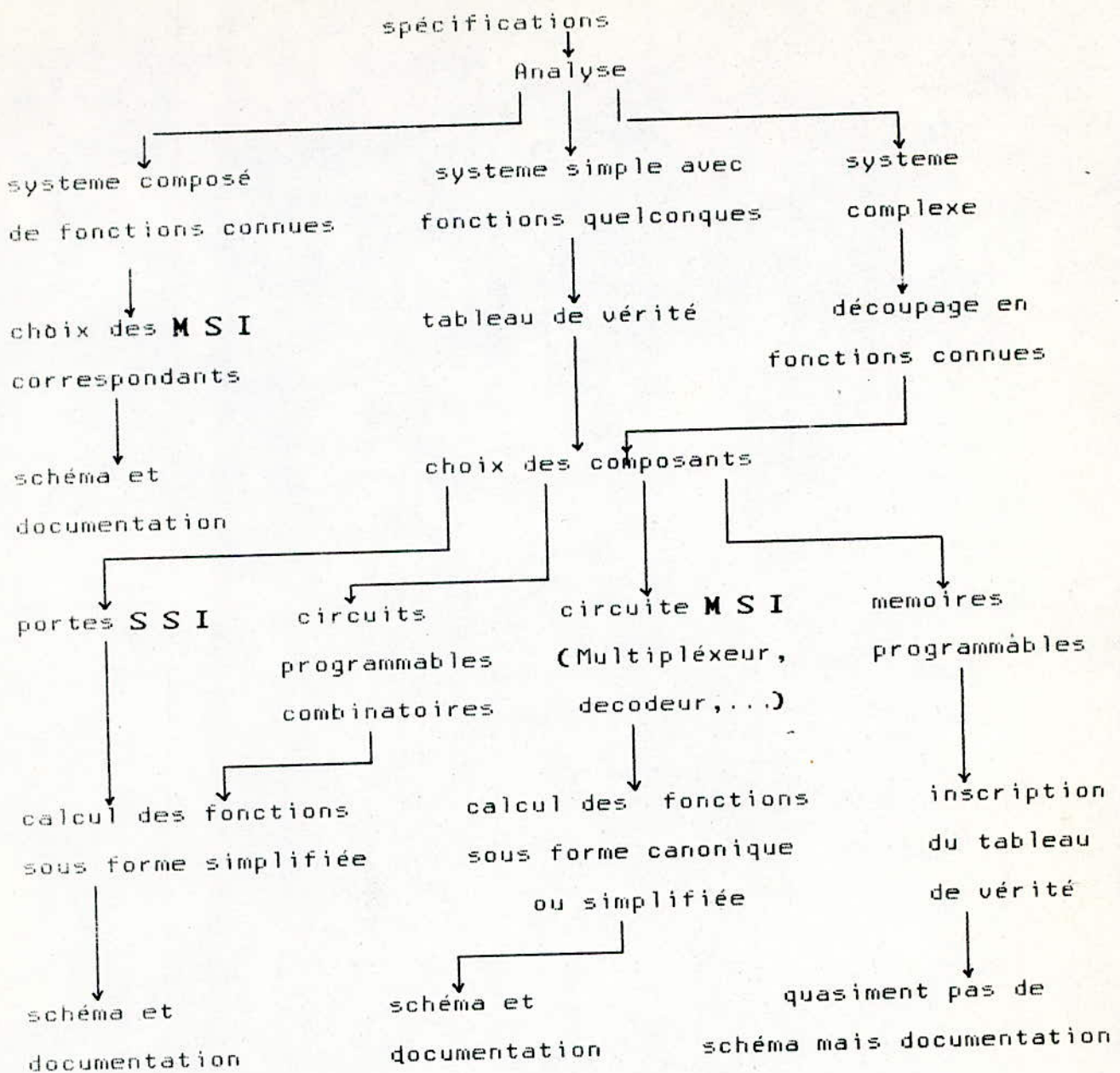
## II .1 ) Conception des systemes combinatoires

La réalisation des fonctions logiques à l'aide de memoires programmables peut être obtenue de deux façons différentes :

- . Soit sous forme combinatoire pure
- . Soit par utilisation de systeme séquentiels programmables .

Dans tous les cas qui se présentent , les spécifications constituent le point de départ pour la conception d'un systeme combinatoire ou séquentiel .

Il existe différentes méthodes de synthèse d'un systeme combinatoire comme le montre la figure suivante :



### Synthèse d'un systeme

Après les spécifications , vient l'analyse qui détermine les types de fonctions constituants le systeme . Trois cas peuvent se présenter :

II.1.1 / L'analyse met en évidence un systeme composé de fonctions booléennes connues ou des extensions de ses fonctions , par conséquent les circuits MSI (decodeur , multiléxeur,...) sont



utilisés .

II.1.2 / L'analyse met en évidence un système simple , c'est à dire ayant peu d'entrées et peu de sorties mais dans lequel n'apparaissent pas des fonctions booléennes connues. Dans ce cas , on établit la table de vérité du système , on passe ensuite au choix des composants en ayant toujours en tête l'idée d'avoir un schéma final optimal c'est à dire comprenant le nombre minimum de boitiers .

II.1.3 / L'analyse met en évidence un système complexe c'est à dire ayant beaucoup d'entrées et beaucoup de sorties . Il est pratiquement impossible de l'étudier par un tableau de vérité car un système à  $n$  variables d'entrées donnera un tableau de vérité à  $2^n$  lignes . Pour cela , on essaie de décomposer le système en plusieurs sous-systèmes combinatoires dans lesquels apparaissent , soit des sous-systèmes simples qu'on peut étudier à partir d'un tableau de vérité , soit des fonctions connues , disponibles sous forme de boitiers (décodeur, multiplexeur, additionneur, ...) . Il n'existe pas de méthode générale pour effectuer cette découpe , mais on doit s'appuyer sur notre intuition et notre connaissance des fonctions combinatoires MSI . On passera ensuite comme dans le cas précédent au choix de composants .

Pour le choix de composants , on se trouve devant quatre possibilités qu'on peut utiliser , séparément , soit conjointement. ces quatre possibilités sont :

\* / Utilisation des portes **ET** , **OU** , **NAND** , .... :

Le système est obtenu en transposant les formules (qui sont calculées et simplifiées à partir de la table de vérité ) avec les portes . Cette méthode est la plus courante .

\* / Utilisation des circuits programmables combinatoires :

Ce sont des circuits constitués de réseaux très denses de portes ET-OU .

\* / Utilisation des circuits MSI (décodeur, multipléneur...):  
Ils sont très utilisés comme générateur de fonctions booléennes .

\* / Utilisation des mémoires programmables PROM :  
Elles sont utilisées comme générateur de fonctions booléennes  
. Dans ce cas , il n'y a aucun calcul booléen , la table est directement inscrite dans la mémoire .

## II . 2 ) Utilisation d'une P R O M comme générateur de fonctions logiques

### II.2.1 /Exemple

Dans notre cas , on ne s'intéressera qu'à l'utilisation des mémoires PROM . Pour cela étudions l'exemple suivant :

Soit un système à trois variables d'entrées X0 , X1 et X2 et à quatre sorties Y0 , Y1 , Y2 et Y3 qui sont liées par le tableau de vérité suivant :

X0	X1	X2	Y0	Y1	Y2	Y3
0	0	0	0	1	1	0
0	0	1	1	1	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	0
1	0	0	1	0	1	0
1	0	1	0	0	1	0
1	1	0	1	1	0	1
1	1	1	1	0	1	0



On tire les sorties Y0, Y1, Y2 et Y3

$$Y_0 = \overline{X_0} \overline{X_1} \overline{X_2} + \overline{X_0} X_1 \overline{X_2} + X_0 \overline{X_1} \overline{X_2} + \overline{X_0} X_1 X_2 + X_0 X_1 \overline{X_2}$$

$$Y_1 = \overline{X_0} X_1 \overline{X_2} + \overline{X_0} \overline{X_1} X_2 + X_0 X_1 X_2$$

$$Y_2 = \overline{X_0} \overline{X_1} X_2 + \overline{X_0} X_1 X_2 + X_0 \overline{X_1} X_2 + X_0 X_1 \overline{X_2} + X_0 X_1 X_2$$

$$Y_3 = \overline{X_0} X_1 X_2 + X_0 \overline{X_1} X_2$$

On remarque que Y3 peut prendre la valeur "1" seulement dans deux cas : Dans le cas où X0=0 ; X1=0 et X2=1 ou dans le cas où X0=1 , X1=1 et X2=0 et que les deux cas ne peuvent se présenter simultanément . Et c'est le même cas pour les autres sorties Y0, Y1, et Y2 , mais il n'y a que le nombre de mintermes qui diffère. Chacun de ces mintermes peut être donné par la sortie d'un circuit MSI qu'est le décodeur . Donc on peut grâce à un décodeur de 3 vers 8 spécifier un et un seul des mintermes en même temps . Chaque minterme est en effet l'une des 2<sup>3</sup> combinaisons possibles des 3 variables d'entrées X0, X1 et X2 .

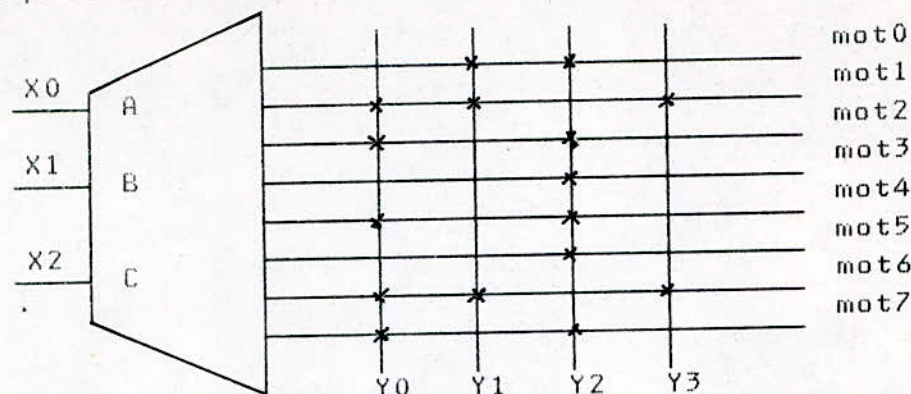
Il ne nous reste donc plus qu'à fixer les sorties Y0, Y1, Y2 et Y3 puisqu'elles ne sont pas variables dans le temps mais varient seulement en fonction des entrées X0, X1 et X2 . Par conséquent , une mémoire PROM ayant trois éléments binaires d'adresse et des mots à quatre éléments binaires fera l'affaire .

		0	1	1	0	mot0
X0	A	1	1	0	1	mot1
X1	B	1	0	1	0	mot2
		0	0	1	0	mot3
X2	C	1	0	1	0	mot4
		0	0	1	0	mot5
		1	1	0	1	mot6
		1	0	1	0	mot7
		Y0	Y1	Y2	Y3	

Fonctions y0, y1, y2 et y3 calculées avec une PROM



On peut aussi les représenter comme suit :



Où les "x" représentent des connexions ou des métalisations et donc des "1" logiques lorsqu'ils sont validés par le décodeur .

Le mot0	correspond à la combinaison	$\overline{X0}$	$\overline{X1}$	$\overline{X2}$	d'où	000	qui est	0
Le mot1	"	"	"	"	$\overline{X0}$	$\overline{X1}$	$X2$	" 001 " 1
Le mot2	"	"	"	"	$\overline{X0}$	$X1$	$\overline{X2}$	" 010 " 2
-----								
Le mot7	"	"	"	"	$X0$	$X1$	$X2$	" 111 " 7

Donc tout revient à mettre des croix "x", qui seront des connexions en pratique, au point d'intersection des lignes de sorties (Y0, Y1, Y2 ou Y3) et les lignes de mots correspondants .  
 Par exemple ,pour la fonction sortie Y3 du cas précédant qui est donnée par  $Y3 = \overline{X0} \overline{X1} X2 + X0 X1 \overline{X2}$

$$1 \quad \quad \quad 6$$

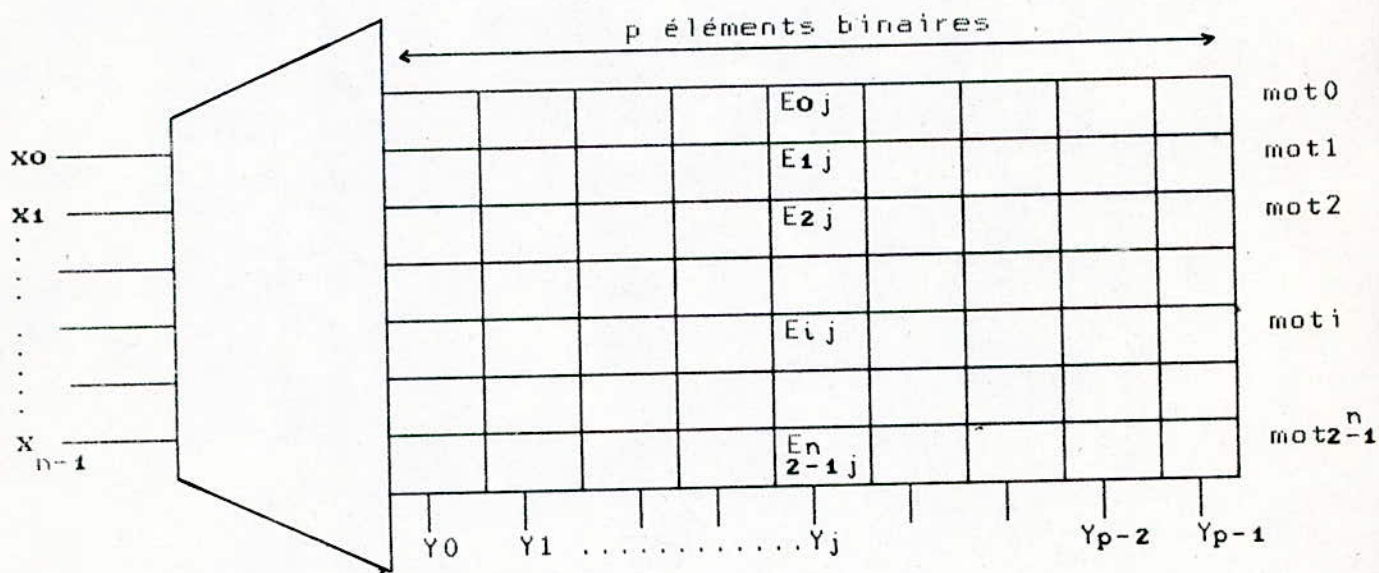
On obtiendra donc Y3 en assurant la connexion de la ligne de sortie Y3 avec la ligne de mot1 et la ligne de mot6 .

On note que la mémoire PROM contiendra la forme canonique ,c'est-à-dire le tableau de vérité d'une fonction . Par conséquent la réalisation de fonctions par une PROM n'implique donc aucun calcul booléen (aucune simplification) .

### 11.2.2 / GERALISATION DE LA METHODE

L'utilisation d'une telle mémoire programmable peut être généralisée

dans le cas où on a un système à  $n$  variables d'entrées et  $p$  fonctions de sorties. donc une mémoire PROM ayant  $n$  éléments binaires (digits) d'adresse et des mots à  $p$  éléments binaires, est nécessaire. La mémoire PROM se présentera alors comme une matrice à  $2^n$  lignes (qui sont les lignes de mots) et à  $p$  colonnes (qui sont les lignes de sorties), d'où la configuration suivante :



Fonction  $y_j$  calculées avec une PROM.

Si l'on examine l'état de la sortie  $Y_j$  de rang  $j$ , elle est égale :

- . A l'état de l'élément binaire  $j$  du mot0
- . ou à l'état de l'élément binaire  $j$  du mot1
- 
- . ou à l'état de l'élément binaire  $j$  du mot  $2^n-1$ .

Donc, la sortie  $Y_j$  s'exprime comme une somme :

$$Y_j = E_{0j} (Ad=0) + E_{1j} (Ad=1) + E_{2j} (Ad=2) + \dots + E_{2^n-1j} (Ad=2^n-1)$$

Avec,

$Ad$ : adresse validée par le décodeur ( $n$  vers  $2^n$ )

$E_{ij}$ : termes valant "0" ou "1" et déterminent les monômes de la fonction suivant qu'ils existent ( $E_{ij}=1$ ) ou non ( $E_{ij}=0$ )

La sortie  $Y_j$  peut s'écrire sous la forme condensée suivante :

$$Y_j = \sum_{i=0}^{2^n - 1} E_{ij} \quad (Ad = i)$$

Cette forme correspond à l'écriture canonique d'une fonction quelconque à n variables .

Finalement , la mémoire PROM contient la forme canonique , et donc cette méthode ne nécessite aucune simplification .



C H A P I T R E .....III

M A T E R I A L I S A T I O N   D E S   F O N C T I O N S

L O G I Q U E S   S E Q U E N T I E L L E S   P A R

U T I L I S A T I O N   D E S   S Y S T E M E S

S E Q U E N T I E L S   P R O G R A M M A B L E S

### III . 1 ) Divers formes d'un système combinatoire

Une fonction combinatoire se présente classiquement sous trois formes algébriques :

- . La forme canonique :  $Y_1 = a \bar{b} \bar{c} + a b \bar{c} + a \bar{b} c + a b c$
- . L'écriture simplifiée en somme d'intersections premiers ( ou implicants premiers ) :  $Y_1 = a \bar{b} + a c + b \bar{c}$
- . La forme avec mise en facteur :  $Y_1 = a ( \bar{b} + c ) + b \bar{c}$

$Y_1$  est la même dans les trois cas , mais son écriture diffère d'un cas à l'autre . d'autres écritures sont évidemment possibles .

Dans les cas les plus complexes , la fonction se présente comme une séquence d'opérateurs tels que **ET, OU, OUEX,** de parenthèses et de variables logiques .

### II . 2 / L 'arbre de décision logique

Partons d'un exemple très simple , on décrira les différentes étapes d'élaboration de la structure dans le cas de l'utilisation d'un circuit de mémorisation de type PROM . Ces étapes sont :

- . Obtention d'un arbre(ou graphe) de décision logique
- . Représentation tabulaire de l'arbre
- . Codage du tableau
- . Implantation de la machine séquentielle .

Soit à matérialiser une structure séquentielle de traitement de la simple fonction suivante :

$$Y = a ( c + b \bar{d} )$$

#### II.2.1 / Obtention de l'arbre ( ou graphe) de décision logique

il s'agit d'une représentation comprenant deux types d'éléments :

- . des assignations de valeur aux fonctions
- . des tests portant sur des variables et indiquant , suivant sa valeur le chemin à suivre pour aller du point de départ de l'arbre jusqu'à un point de d'assignation . Par convention , si la valeur de la variable testée est égale à zéro , la sortie de

l'élément de test est indiqué par un petit cercle de complémentation .Si elle est égale à 1 , elle correspondra à l'autre sortie .

un arbre booléen comprend un et un seul point de départ et ne peut y avoir qu'une seule liaison entre une sortie d'un test et l'entrée de l'élément suivant .

L'obtention du de l'arbre de décision logique découle directement de l'ordre dans lequel les variables sont testées . Cette obtention n'est donc pas unique . on peut donc construire un arbre de décision avec un peu de bon sens ,en ayant toujours le souci qu'il est toujours intéressant d'obtenir un développement par rapport à une ou plusieurs variables .

dans l'exemple proposé , on commencera évidemment par tester la variable "a", car elle est en facteur , et car, s'elle prend la valeur "0" , en déduit directement que  $Y=0$  ( sans tester les autres variables ) .On testera ensuite la variable "c" , et finalement , on testera indifféremment les variables "b" et "d" .  
Après condensation de l'arbre de décision , on obtient un graphe de décision logique comme le montre la figure III.1 .



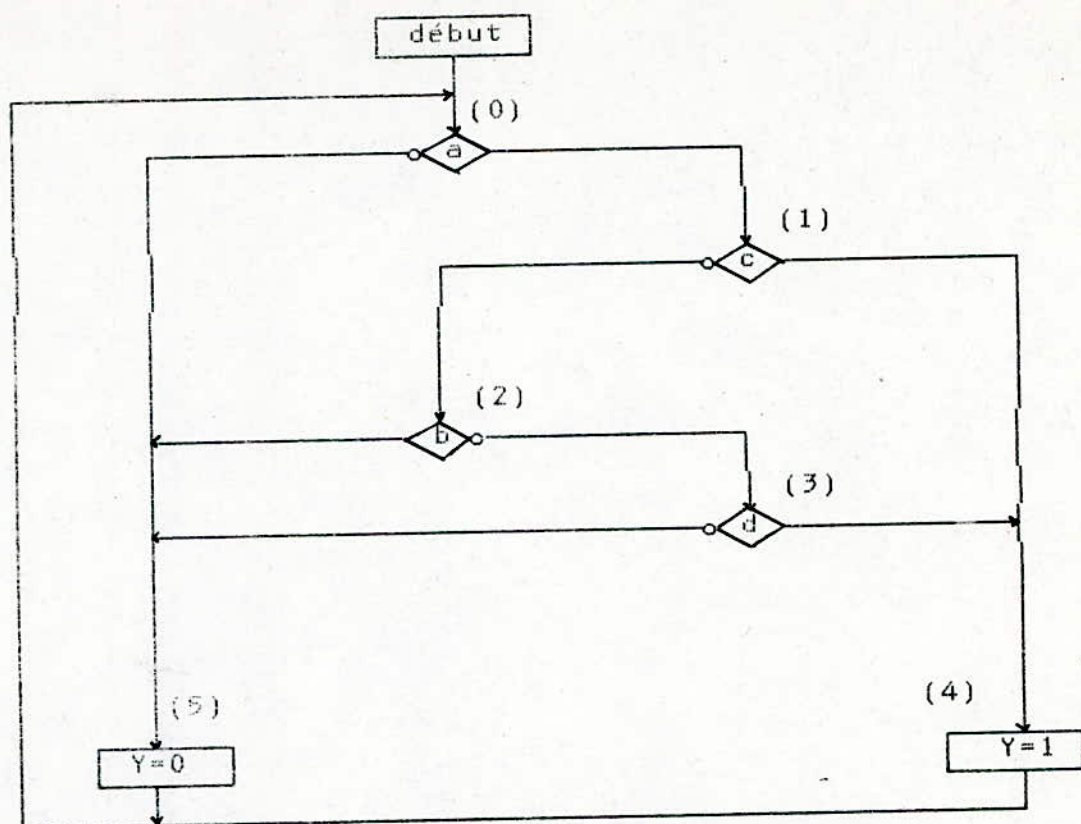


Figure III 1: *graphe de décision logique*

Il faut ensuite numéroter toutes les instructions ( tests et sorties ) . Par exemple,

l'instruction n°0 correspond au test de "a"

" n°5 " à la sortie Y

On remarque qu'on peut reboucler le graphe c'est-à-dire ,après avoir obtenu le résultat , on recommence de nouveau le même processus pour d'éventuels changements des valeurs des variables.

Il est facile de constater que sur un arbre de décision binaire que, tous les monômes sont disjoints . Les équations qui se présentent sous la forme d'une somme d'intersections premières ( ou implicants premiers  $Y = ab + ac + bc$  ) posent alors des problèmes de représentation . Dans ce cas il convient de se ramener à une forme utilisable .

La procédure pour rendre disjoints deux monômes booléens  $m_1$  et  $m_2$  d'une expression consiste à écrire :

$$F = m_1 + m_2 + m_3 + \dots = m_1 + m_2 \cdot \overline{m_1} + m_3 + \dots$$

Il est parfois nécessaire de répéter la procédure au niveau des monômes créés. Ainsi,

$$\begin{aligned} F &= a b + c d = a b + c d (\overline{a b}) \\ &= a b + \overline{a} c d + \overline{b} c d \end{aligned}$$

On remarque que  $\overline{a} c d$  et  $\overline{b} c d$  ne sont pas disjoints. Il convient donc de répéter la procédure en écrivant :

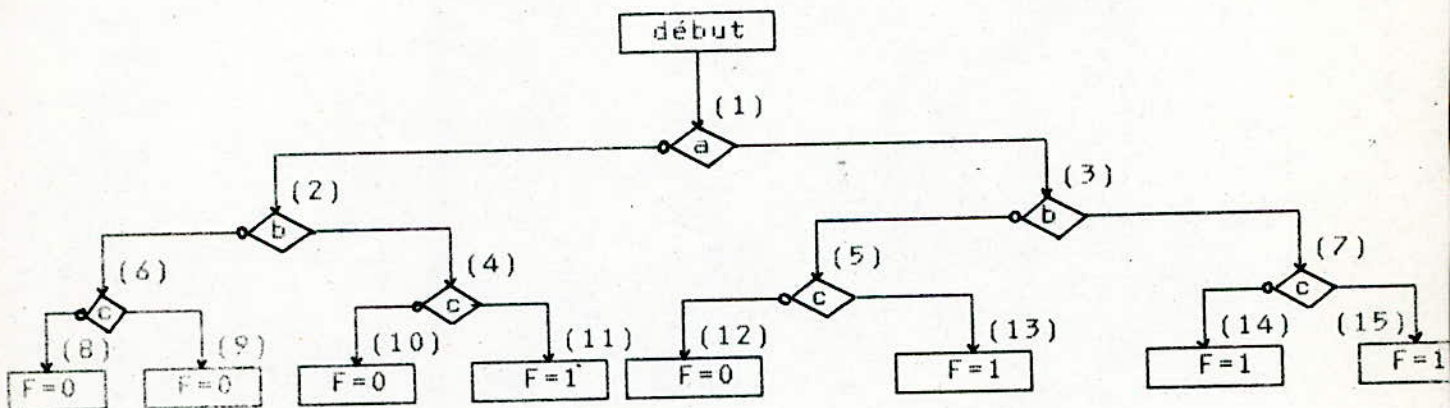
$$F = a b + \overline{a} c d + \overline{b} c d = a b + \overline{a} c d + \overline{b} c d (\overline{a c d})$$

$$F = a b + \overline{a} c d + a \overline{b} c d \quad \text{dont les monômes sont tous disjoints}$$

Il est à noter que dans certains cas, certaines branches mènent à un résultat identique, ce qui permet de condenser l'arbre de décision comme le montre l'exemple suivant.

Soit à représenter la fonction suivante,

$$F = a \overline{b} c + a b \overline{c} + a \overline{b} c + a b c$$



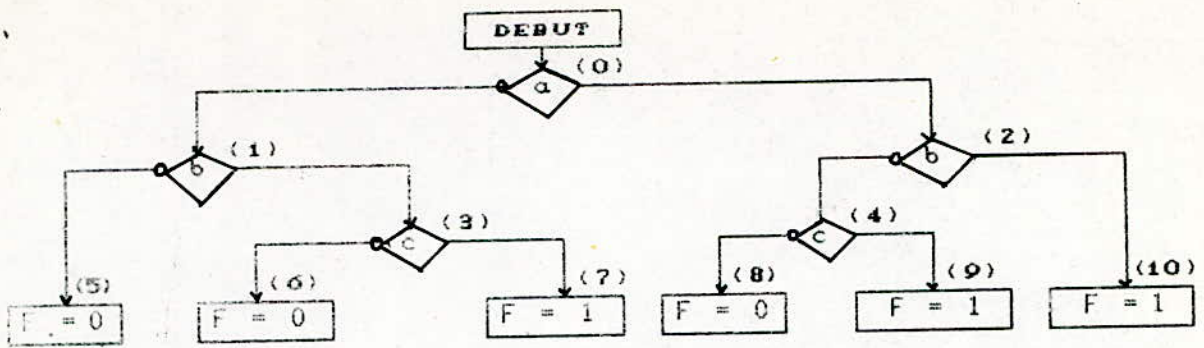
arbre de décision

F=0 indépendamment de la valeur de "c"

même résultat indépendamment de la variable "b"

F=1 indépendamment de la variable "c"

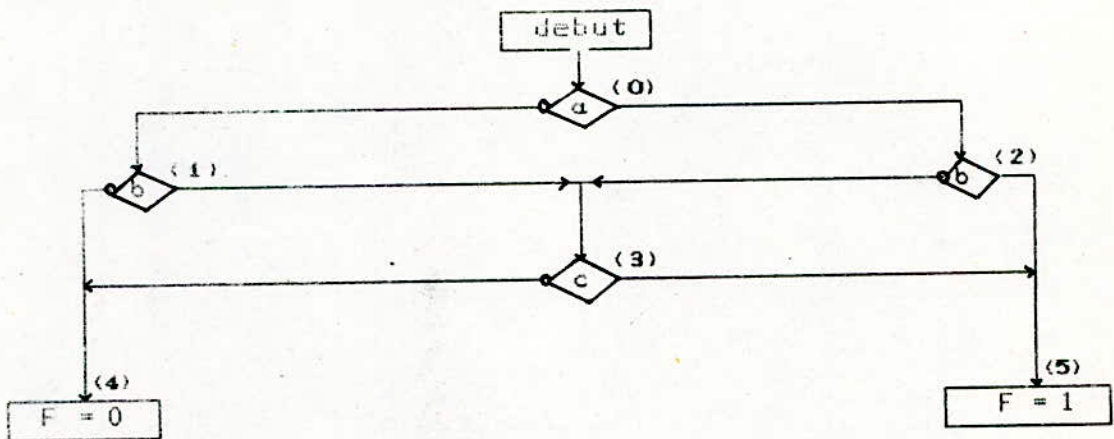
Après une première condensation, la fonction précédente se présente sous la forme de l'arbre de la figure suivante :



arbre de décision après une 1<sup>ère</sup> condensation

Il est évident que cette première condensation a permis de réduire de quatre le nombre total d'instructions à exécuter, donc cette condensation a entraîné un gain en temps et une économie en espace mémoire.

En ayant en tête le même souci d'économiser de l'espace mémoire et un gain en temps d'exécution, on fait une deuxième condensation et la fonction précédente se présentera sous la forme du graphe de la figure suivante :



graphe de décision

Après cette nouvelle condensation, on est descendu jusqu'à six instructions, soit une réduction de neuf par rapport à la représentation initiale qui en nécessitait quinze. Le temps nécessaire pour aboutir au résultat et ainsi, réduit de plus de sa moitié.

Le traitement des fonctions simultanées ne pose aucun problème si



celles ci sont disjointes car on est ramené au cas précédent .Si elle ne le sont pas ,il faut considérer les produits de fonctions .Ces produits faciliteront énormément la représentation de l'arbre binaire ,car en fonction des valeurs d'un ou quelques produits ,on tirera directement les valeurs des fonctions logiques .Soit par exemple deux fonctions non disjointes F1 et F2 .

En établissant les quatres produits :

$$P1 = F1 F2$$

$$P2 = F1 \overline{F2}$$

$$P3 = \overline{F1} F2$$

$$P4 = \overline{F1} \overline{F2}$$

Si P1=1 , on en déduit directement que : F1=F2=1

Si P4=1 , " " " " : F1=F2=0

Si P2=1 , " " " " : F1=1 , F2=0

Si P3=1 , " " " " : F1=0 , F2=1

Si P1=0 et P2=1 " " " " : F1=1 ,F2=0

Donc avec un peu de bon sens , on arrive facilement à dresser l'arbre de décision .

Il est à rappeler que le nombre de tests dépend du choix de l'ordre des tests .Dans le cas des traitements des fonctions simultanées non disjointes ,il est intéressant pour diminuer le nombre de tests , de commencer par la variable présentée sous sa forme normale et complémentée dans le maximum de produits Pi, puis de répéter cette procédure au niveau des fonctions résultantes du choix de la première variable en opérant de la même manière .

Soit à représenter les deux fonctions suivantes :

$$F1 = a \overline{d} + b + \overline{a} \overline{c}$$

$$F2 = \overline{a} \overline{b} + c d$$

Les Pi sont les suivants après application de la procédure pour rendre disjointes tous les monômes :

$$P1 = F1 F2 = b c d + \overline{a} \overline{b} \overline{c} = b c d + \overline{a} \overline{b} \overline{c}$$

$$P2 = F1 \overline{F2} = a \overline{b} + b \overline{c} + b \overline{d} = b \overline{c} + b \overline{c} \overline{d} + a \overline{b} \overline{d}$$

$$P3 = \overline{F1} F2 = \overline{a} \overline{b} c + \overline{b} c d = \overline{b} c d + \overline{a} \overline{b} c \overline{d}$$

$$P4 = \overline{F1} \overline{F2} = \overline{a} \overline{b} \overline{c} d = \overline{a} \overline{b} \overline{c} d$$

Dans ce cas , il faut commencer par tester la variable "b" qui est la plus complémentée dans les Pi .

Si b = 0 , c'est la variable "d" qui sera tester car elle sera la plus complémentée dans les Pi résultants .

Si b = 1 , c'est la variable c qui sera tester car elle sera la plus complémentée dans les fonctions résultantes .

De la même manière , dans le cas de trois fonctions F1 ,F2 et F3

non disjointes , l'arbre de décision binaire s'obtient en considérant les  $2^3$  produits  $F_1 F_2 F_3$  ,  $\overline{F_1} F_2 F_3$  , ...,  $\overline{F_1} \overline{F_2} \overline{F_3}$ .

En généralisant , pour un cas de n fonctions non disjointes , on considèrera les  $2^n$  produits pour l'obtention de l'arbre de décision logique .

### III.1.2.b ) REPRESENTATION TABULAIRE DE L ARBRE

En dénótant les instructions de test un élément de test , instruction de sortie un élément d'assignation de valeurs à la fonction (ou aux fonctions) et en utilisant la numérotation des éléments , il est possible de représenter l'arbre de décision logique de la figure (III.1) par le tableau équivalent donné par la figure suivante :

numéro instruc- tion	nature instruc- tion	instruction test			instruc.sortie	
		varia- ble	test 1	test 0	N° instruc	Y
0	test	a	1	5		
1	test	c	4	2		
2	test	b	5	3		
3	test	d	4	5		
4	sortie				0	1
5	sortie				0	0

Représentation tabulaire

Vu que la distinction entre "instruction test" et "instruction sortie" se fait par "nature instruction" , on peut donc condenser le tableau comme suit :

Numéro instruction	Nature instruction	Variable testée	test "1"	test "0"
0	test	a	1	5
1	test	c	4	2
2	test	b	5	3



4	sortie	-	0	1
5	sortie	-	0	0

### III.2.1.c ) CODAGE DU TABLEAU

Le codage du tableau découle directement de la nécessité d'une mémorisation dans la PROM .

Ce code doit pouvoir présenter :

- + La nature de l'instruction
- + Le numéro de l'instruction
- + La variable à tester .

#### \* / Codage de la nature de l'instruction

En gardant toujours l'exemple précédant , il n'y a que de deux types d'instructions : instruction test et instruction sortie . Un digit binaire "d" suffira pour ce codage .

instruction	d
test	0
sirtie	1

(a)

Il est bien sur indifférent du point de vûe théorique de coder,

instruction	d
test	1
sortie	0

(b)

Mais en pratique , un "1" correspondra à une réalisation d'une connection entre une ligne d'adresse et une ligne de sortie et comme le nombre d'instruction test est supérieur à celui d'instruction sorties (dans notre cas ) , on optera pour le code

(a)

\* / Codage des instructions

Il dépend du nombre total d'instructions . En gardant le même exemple de la figure (III.1) ou l'arbre booléen comporte :

- + 4 instructions de test
- + 2 instructions de sortie

Soit un total de six instructions , trois digits binaires  $(m_2, m_1, m_0)$  sont donc nécessaires pour ce codage tel que :

$$(m_2, m_1, m_0)_2 = (N^\circ \text{ instruction})_{10}$$

comme le montre le tableau suivant :

N° instruction	m2	m1	m0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
-	1	1	0
-	1	1	1

codage des instruction

Le compteur ordinal CO sera composé d'un ensemble de trois bascules .

\* / Codage des variables

Restons dans le même exemple , le nombre de variables à tester est égal à quatre . Une représentation codée de la variable à tester implique donc l'utilisation de deux digits binaires  $\alpha$  et  $\beta$  .



$\alpha$	$\beta$	Variable
0	0	a
0	1	b
1	0	c
1	1	d

codage des variables

Le codage des variables est vraiment quelconque .

En tenant compte de la représentation tabulaire et du codage , l'arbre booléen peut être représenté sur un seul tableau sous forme d'un tableau codé :

N° inst	N° instruction			nature d	variables		test "1"			test "0"			Adress suivante
	m2	m1	m0		$\alpha$	$\beta$	m2	m1	m0	m2	m1	m0	
	0	0	0		0	0	0	0	0	0	1	1	
1	0	0	1	0	1	0	1	0	0	0	1	0	
2	0	1	0	0	0	1	1	0	1	0	1	1	
3	0	1	1	0	1	1	1	0	0	1	0	1	
4	1	0	0	1	-	-	0	0	0	-	-	<b>1</b>	
5	1	0	1	1	-	-	0	0	0	-	-	<b>0</b>	
6	1	1	0	-	-	-	-	-	-	-	-		
7	1	1	1	-	-	-	-	-	-	-	-		

valeur de  
sortie

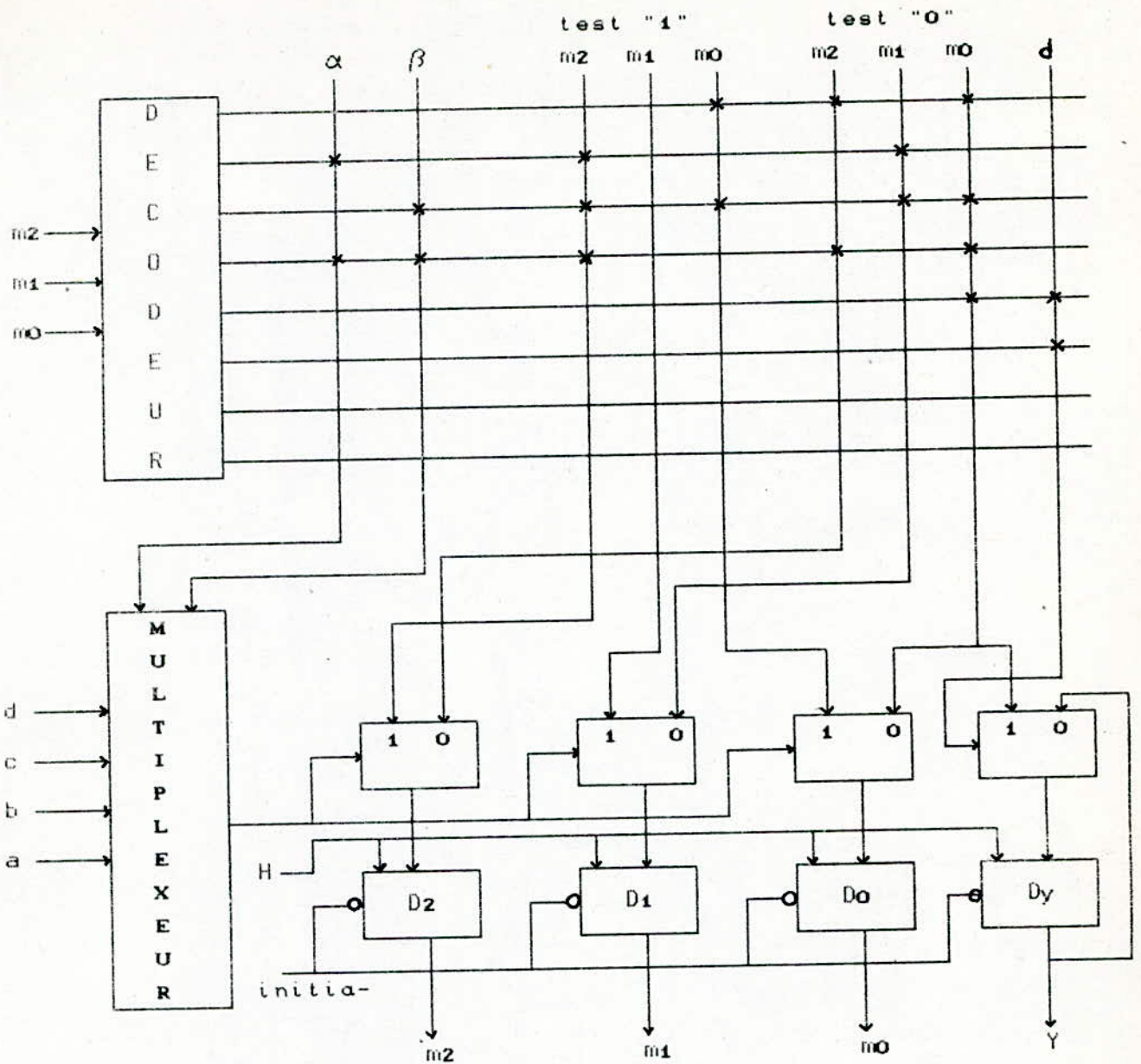
Représentation tabulaire

### III.2.1 d ) IMPLANTATION DE LA MACHINE SEQUENTIELLE

Une implantation d'une machine séquentielle pour la matérialisation de la structure combinatoire est donnée par la figure suivante qui comprend (dans le cas de notre exemple) :

- + Une mémoire PROM de format ( $8eb \times 9eb$ )
- + Un décodeur 3 vers 8 nécessaire pour sélectionner l'instruction qui doit être exécutée .
- + Un multiplexeur à deux entrées pilotées par  $\alpha$  et  $\beta$  ,mettant à sa sortie la variable à tester .
- + Un ensemble de trois bascules  $D_2$  ,  $D_1$  ,  $D_0$  mémorisant l'adresse de l'instruction en cours . Ces bascules constituent le compteur ordinal et sont initialisables à l'adresse (0,0,0) correspondant à l'instruction (0) .
- + Une bascule  $D_y$  de sortie .
- + Quatres multiplexeurs élémentaires de commande des bascules .

Le multiplexeur à deux variables d'entrées (  $\alpha$  et  $\beta$  ) et les quatres multiplexeurs élémentaires permettent un aiguillage correct vers le compteur ordinal constitué des bascules  $D_2$  ,  $D_1$  et  $D_0$  de l'adresse de l'instruction suivante , ainsi que l'affectation ou le maintien de la valeur de la sortie  $Y$  à l'entrée de la bascule  $D_y$  .



*Schéma d'implantation*

Il est à noter qu'en pratique :

- + Le nombre de variables est limité par le multiplexeur disponible
- + Le nombre total d'instructions est limité par le décodeur

Il est bien sûr possible de faire des extensions des fonctions décodage et multiplexage.



## CHAPITRE IV

### DESCRIPTION DE LA REALISATION PRATIQUE

IV 1 / INTRODUCTION

IV 2 / DESCRIPTION

IV.2.1) Le decodeur (4 vers 16)

IV.2.2) Le multipléxeur (8 vers 1)

IV.2.3) les quadruples multipléxeurs (2vers1)

IV.2.4) Les quadruples bascules "D" avec Reset

IV.2.5) Les quadruples portes NAND buffers

IV.2.6) Trigger de Schmit .

## IV.1 / Introduction

Pour le cas pratique , on se limitera à la matérialisation des fonctions logiques à 8 variables au maximum . Le schéma d'une telle réalisation est montré à la figure (IV) .

## IV.2 / Description

Ce circuit comprend :

IV.2.1 ) Un décodeur quatre vers seize . C'est le 74154 . Il mettra l'une de ses seize sorties  $E_0$  ,  $E_1$  ...  $E_{15}$  selon l'état de ses quatre entrées A , B , C et D . Ce circuit permettra d'obtenir seize lignes de mots , ce qui correspond à seize instructions ( 14 instructions "test" et 2 instructions "sortie" ) dans le cas où le circuit sera utilisé pour la matérialisation des fonctions combinatoires d'une façon séquentielle . Dans le cas où la mémoire sera utilisée comme générateur de fonctions combinatoires , ce décodeur nous limitera à quatre le nombre de variables ( d'où seize combinaisons c-à-d seize mintermes ) . Il n'est possible dans ce cas d'obtenir que des fonctions à quatre variables au maximum . Ces quatre variables seront les entrées adresses ( A , B , C et D ) même du décodeur .

Les caractéristiques électriques , le tableau de fonctionnement , ainsi que la table d'excitation sont donnés par la figure (IV .1)

IV.2.2 / Un multiplexeur huit vers un ( 8 vers 1 ) . C'est le 74151 . Il n'intervient seulement dans

le cas où le circuit sera utilisé pour matérialiser une fonction combinatoire d'une façon séquentielle . Il permettra de sélectionner la variable à tester à chaque cycle d'orloge . Les variables à tester sont représentées par les entrées D0 , D1 , ... , D7 du multiplexeur (741s151) . La variable sera sélectionnée par les entrées adresses A , B et C pilotées par  $\alpha$  ,  $\beta$  et  $\gamma$  qui représentent le code de la variable .

Les caractéristiques électriques , le tableau de fonctionnement , ainsi que la table d'excitation de ce multiplexeur sont montrés à la figure (IV.2).

IV.2.3 / Deux quadruples multiplexeurs deux vers un ( 2 vers 1 ) . C'est le 741s157 . A chaque cycle d'horloge deux adresses se présentent aux entrées  $A_i$  et  $B_i$  (  $i = 1, 2, 3, 4$  ) et la valeur de la variable à tester se présente sur l'entrée "Select" du 741s157 .

Où que :

$Y_i = G ( S A_i + \bar{S} B_i )$  avec G : entrée de validation (elle sera mise à la masse pour que le 741s157 sera toujours validé)

Si la valeur de la variable à tester est un "0" , c'est les valeurs des  $A_i$  qui seront transmises sur les sorties  $Y_i$  du même C I . Dans le cas contraire , c'est les valeurs des  $B_i$  qui seront transmises . Les sorties  $Y_i$  attaqueront les entrées des bascules . Le quadruple multiplexeur ( 2 vers 1 ) permet donc un aiguillage correct vers le compteur ordinal . Le deuxième quadruple multiplexeur ( 2 vers 1 ) ne sera exploité qu'à un quart de sa



capacité , et permettra suivant la valeur de "d" à changer ou à garder la valeur de la fonction en question .

Les caractéristiques électriques , le tableau de fonctionnement ainsi que la table d'excitation de ces C I sont données par la figure ( IV.4 ).

IV.2.4 / Deux quadruples bascules "D" avec entrée "Reset" ( le 741s175) . Le premier C I mémorise l'adresse de l'instruction en cours d'exécution et la gardera tant qu'un front montant de l'horloge "H" n'arrive sur son entrée "Clock" . Le deuxième C I ne sera utilisé qu' au quart de sa capacité . Il gardera ou changera la valeur de la fonction étudiée .

La figure ( IV.2) montre les caractéristiques électriques , le tableau de fonctionnement ainsi de la table d'excitation .

IV.2.5 / Trois quadruples portes NAND buffers ( c'est le 7428 ) . Ils joueront d'une part le rôle d'inverseurs pour rendre correct les informations provenant de la mémoire : quand le décodeur 74154 sélectionne une ligne d'adresse , il mettra cette sortie à l'état bas . Si une connection est effectuée entre cette ligne et ligne de bit , on aura un "0" logique sur cette ligne de bit , alors qu'on devait avoir un "1" . C'est pour cette raison qu'on est amené à utiliser les portes NAND ( dans le rôle d'inverseurs ) . Ils joueront d'une autre part le rôle d'amplificateurs ( buffers ) pour compenser les pertes .

Voir la figure (IV.3) pour les caractéristiques électriques , le tableau de fonctionnement et la table d'excitation .

#### IV.2.6 / Un trigger de Schmit ( NE 555 )

Il sera utilisé comme un bistable . Ce C I générera le signal H d'horloge .

En se basant sur les caractéristiques de ces C I , on arrive à déterminer la fréquence maximale du signal H de l'horloge . Le temps minimal entre deux fronts montants successifs d'horloge doivent être de 91 ns , soit une fréquence minimale d'environ 10 MHz . Cette valeur est bonne , vu que les bsscules travaillent jusqu'à une fréquence de 40 MHz .

\* détermination de la fréquence maximale de l'horloge

type de C.I	temps de réponse maximal (ns)
741s151	11
741s175	25
74157	18
7428	14
74154	23
	soit 91 ns au total

Supposons que les C I ne travaillent pas en parallele pour pouvoir déterminer le temps maximum entre deux fronts montants . Ce qui donnera un temps maximal d'environ 91 ns , soit une fréquence minimale de 10 MHz .

# CHAPITRE: V

TEXTES

PEDAGOGIQUES



## 0.2 / \*QUESTIONS THEORIQUES

- 1) Pourquoi l'obtention d'un arbre de decision n'est pas unique?
- 2) Pourquoi est-il necessaire de condenser la representation de l'arbre de decision ?
- 3) Pourquoi est-il necessaire de coder ?
  - 3.1/ Croyez-vous qu'il y a un choix privilegie pour le codage des variables ?
  - 3.2/ Et pour celui de la nature de l'instruction ?
- 4) Quels sont les avantages et les inconvenients de la materialisation par utilisation des systemes sequentiels programmables ?
- 5) Quels sont les avantages et les inconvenients de la materialisation des fonctions logiques par des systemes combinatoires ?

### \* REPONSES AUX QUESTIONS

- 1) L'obtention d'un arbre de decision n'est pas unique car il depend de l'ordre dans lequel les variables sont testees .
- 2) Il est necessaire de condenser la representation de l'arbre de decision pour diminuer le nombre total d'instructions qui est limite par le decodeur (16 instructions au maximum dans notre cas pratique ) . Cette condensation entraine aussi un gain en temps d'execution .

3) Le codage du tableau decoule directement de la necessite d'une memorisation dans la PROM .

3.1 / Il n'y a aucun choix privilegie pour le codage des variables mais il faut seulement respecter le tableau de fonctionnement du multiplexeur .

3.2 / Du point de vue theorique , il n'y a aucun choix privilegie pour le codage de la nature de l'instruction , mais en pratique , un 1 correspond a une connection . et comme le nombre d'instructions test est superieur a celui de sortie , il est donc preferable d'attribuer des 1 aux instructions sortie et des 0 aux instructions test .

4)\* Les avantages que presente l'utilisation des systemes sequentiels programmables sont :

- +Traitement des fonctions logiques a grand nombre de variables (8 variables dans notre cas pratique )
- +Initiation aux techniques de programmation surtout les instructions se presentant sous la forme If Then Else
- +La non necessite de passage par une table de verite .

\* Les inconvenients de cette methode sont :

- +Utilisation d'un circuit supplementaire
- +Traitement d'une seule fonction a la fois (dans notre cas )
- +Limitation du nombre d'instructions (16 dans notre cas )
- +Necessite de simplifier et de rendre disjoint les mintermes

5) Les avantages que presente la materialisation des fonctions logiques par des systemes combinatoires sont :

- +Ne necessite aucune simplification de la fonction
- +Traitement de plusieurs fonctions simultanement (12 dans notre cas )
- +Ne necessite que peu de materiel (un decodeur et une PROM )

Les inconvenients sont :

- +Le nombre de variables tres limite (seulement 4 dans notre cas )
- +La necessite de passer par une table de verite
- +Ne presente pas un grand interet pedagogique .

### 0.3 / PARTIE      PRATIQUE

#### 0.3.a ) Systeme séquentiel

Soit à matérialiser la fonction logique suivantes :

$$Y = ( X1 + X2 ) \overline{X5} + X1 + X2 ( \overline{X3} + X4 ) + X6 \overline{X7}$$

Indications :

- \* + denôte la somme modulo 2 " OUEX " tel que :  
 $a + b = a \bar{b} + \bar{a} b$
- \* Appliquer la procedure de rendre deux mintermes disjoints à :

$$\overline{X2} + X3 \overline{X4}$$

- \* Commencer par le test de X6 ,X8 puis X0 ,....

- (1) Donner le graphe de décision final .
- (2) Donner la représentation tabulaire de l'arbre..
- (3) Donner la représentation tabulaire codée .



- (4) Programmer la PROM selon votre représentation .
- (5) Donner des valeurs aux  $X_i$  respectivement sur les entrées  $D_i$
- (6) Mettre les commutateurs K sur les positions 1 pour faire fonctionner le système séquentiel programmable .
- (7) Mettre le commutateur  $K_1$  en position 1 puis le remettre en position 2 pour initialiser les bascules  $D_2, D_1, D_0$  et  $D_y$  afin que l'exécution du programme commence par l'instruction (0) ainsi que la sortie Y qui prendra la valeur initiale 0 .

Après obtention des résultats , changer les valeurs des variables  $X_i$  et répéter (6) .

Que constater vous ? interpréter .

CORRECTION

$$Y = X_0 \quad X_1 + X_2 \quad \overline{X_5} \quad + \quad \overline{X_1 + X_2} \quad \overline{X_3} + X_4 \quad + \quad X_6 \quad \overline{X_7}$$

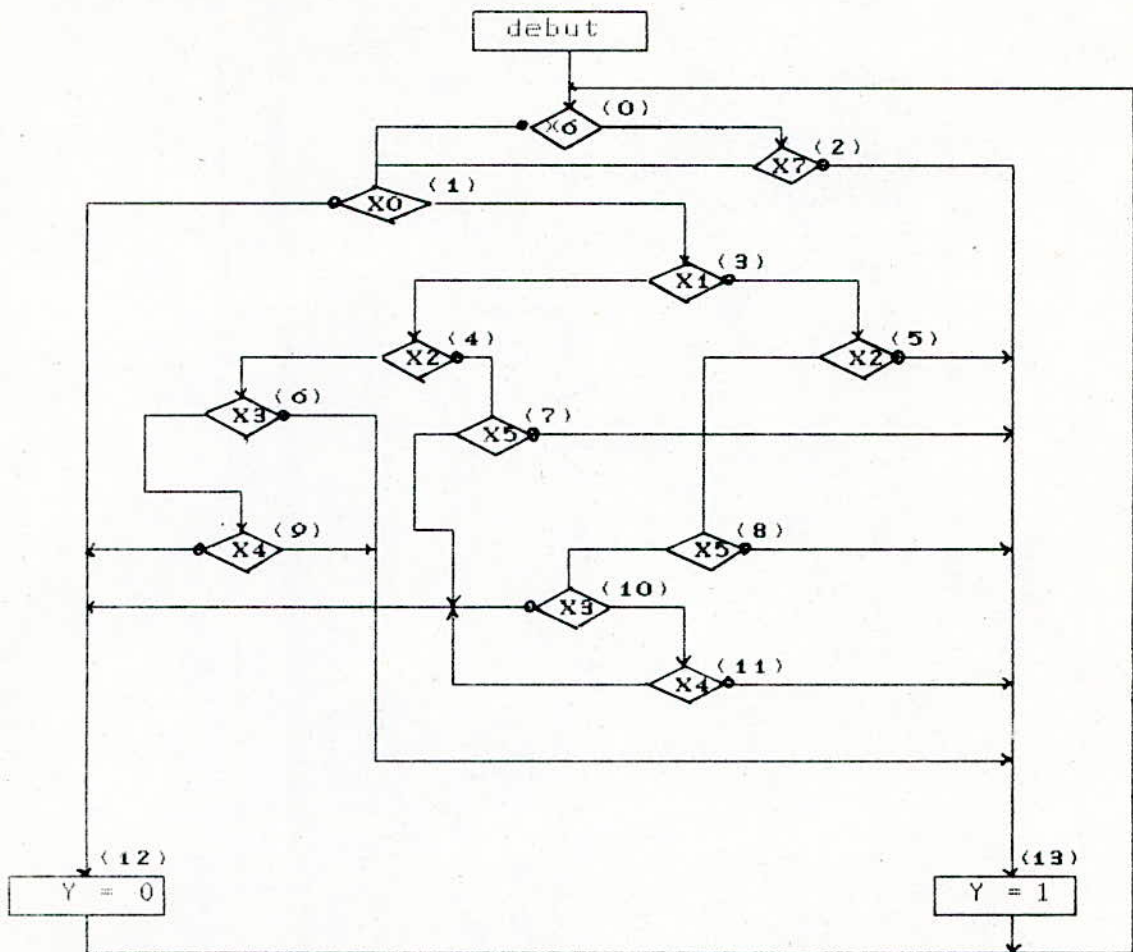
Par développement :

$$Y = X_0 \quad X_1 \quad \overline{X_2} \quad \overline{X_5} \quad + \quad \overline{X_1} \quad X_2 \quad \overline{X_5} \quad + \quad \overline{X_1} \quad X_2 \quad \overline{X_3} + X_4 \quad + \quad X_1 \quad X_2 \quad \overline{X_3} + X_4 \quad + \quad X_6 \quad \overline{X_7}$$

Finalement :

$$Y = X_0 \quad X_1 \quad \overline{X_2} \quad \overline{X_5} \quad + \quad \overline{X_1} \quad X_2 \quad \overline{X_5} \quad + \quad X_3 \quad \overline{X_4} \quad + \quad X_1 \quad X_2 \quad \overline{X_3} \quad + \quad X_4 \quad + \quad \overline{X_1} \quad \overline{X_2} \quad + \quad X_6 \quad \overline{X_7}$$

(4) Graphe de décision :



Graphe de décision

(2) Représentation tabulaire du graphe de décision :

N°instruc.	Nature instruc	Variable testée	test "1"	test "0"
0	test	X6	2	1
1	test	X0	3	12
2	test	X7	1	13
3	test	X1	4	5
4	test	X2	6	7
5	test	X2	8	13
6	test	X3	9	13
7	test	X5	12	13
8	test	X5	10	13
9	test	X4	13	12
10	test	X3	11	12



11	test	x4	12	13
12	sortie	-	0	0
13	sortie	-	0	1

*Représentation tabulaire du graphe de décision*

(3) Représentation tabulaire codée :

N°	N° instruction				d	Variables			test "1"				test "0"			
	m3	m2	m1	m0		$\alpha$	$\beta$	$\gamma$	m3	m2	m1	m0	m3	m2	m1	m0
0	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0	1
1	0	0	0	1	0	0	0	0	0	0	1	1	1	1	0	0
2	0	0	1	0	0	1	1	1	0	0	0	1	1	1	0	1
3	0	0	1	1	0	0	0	1	0	1	0	0	0	1	0	1
4	0	1	0	0	0	0	1	0	0	1	1	0	0	1	1	1
5	0	1	0	1	0	0	1	0	1	0	0	0	1	1	0	1
6	0	1	1	0	0	0	1	1	1	0	0	1	1	1	0	1
7	0	1	1	1	0	1	0	1	1	1	0	0	1	1	0	1
8	1	0	0	0	0	1	0	1	1	0	1	0	1	1	0	1
9	1	0	0	1	0	1	0	0	1	1	0	1	1	1	0	0
10	1	0	1	0	0	0	1	1	1	0	1	1	1	1	0	0
11	1	0	1	1	0	1	0	0	1	1	0	0	1	1	0	1
12	1	1	0	0	1	-	-	-	0	0	0	0	0	0	0	0
13	1	1	0	1	1	-	-	-	0	0	0	0	0	0	0	1
-	1	1	1	0	-	-	-	-	-	-	-	-	-	-	-	-
-	1	1	1		-	-	-	-	-	-	-	-	-	-	-	-

Table de programmation

### U.3.b / Systèmes combinatoires purs

Soit à matérialiser les douzes fonctions suivantes :

$$F_0 = \overline{X_0} \overline{X_1} X_2 X_3 + \overline{X_0} X_1 \overline{X_2} X_3 + \overline{X_0} \overline{X_1} \overline{X_2} X_3 + X_0 X_1 X_2 X_3$$

$$F_1 = X_0 X_1 + X_0 \overline{X_1} + X_3$$

$$F_2 = \overline{X_0} \overline{X_1} X_2 X_3 + \overline{X_0} X_1 \overline{X_2} X_3 + X_2 \overline{X_3}$$

$$F_3 = X_0 \overline{X_3} + X_1 X_4 + X_0 \overline{X_1}$$

$$F_4 = X_0 \overline{X_1} X_2 X_3 + X_0 X_1 X_2 X_3$$

$$F_5 = X_0 ( X_1 + X_2 ) + X_2 + X_3$$

$$F_6 = X_0 X_1 ( X_2 + X_3 ) + X_2 \overline{X_3}$$

$$F_7 = X_2 X_3 ( X_0 + X_1 )$$

$$F_8 = ( X_0 X_3 ) + ( X_1 X_2 )$$

$$F_9 = X_0 + ( X_1 + X_2 )$$

$$F_{10} = X_0 + X_1 + X_2 + X_3$$

$$F_{11} = X_1 + \overline{( X_1 + X_2 )} + \overline{X_3}$$

Indications :

Revenir toujours à la forme cannique après les développements .

Pour cela , effectuer les développements de façon à n'avoir que de simples sommes .

Après avoir porter les fonctions dans la PROM , mettez les commutateurs K en position 2 pour déconnecter le système séquentiel . Dans ce cas la PROM se comportera comme un générateur de fonctions logiques .



Mettre les valeurs des variables  $X_0$ ,  $X_1$ ,  $X_2$  et  $X_3$  respectivement sur les entrées A, B, C et D du décodeur.

Tirer l'intérêt de cette méthode par rapport à la programmation séquentielle ainsi que les inconvénients.

---

# A N N E X E

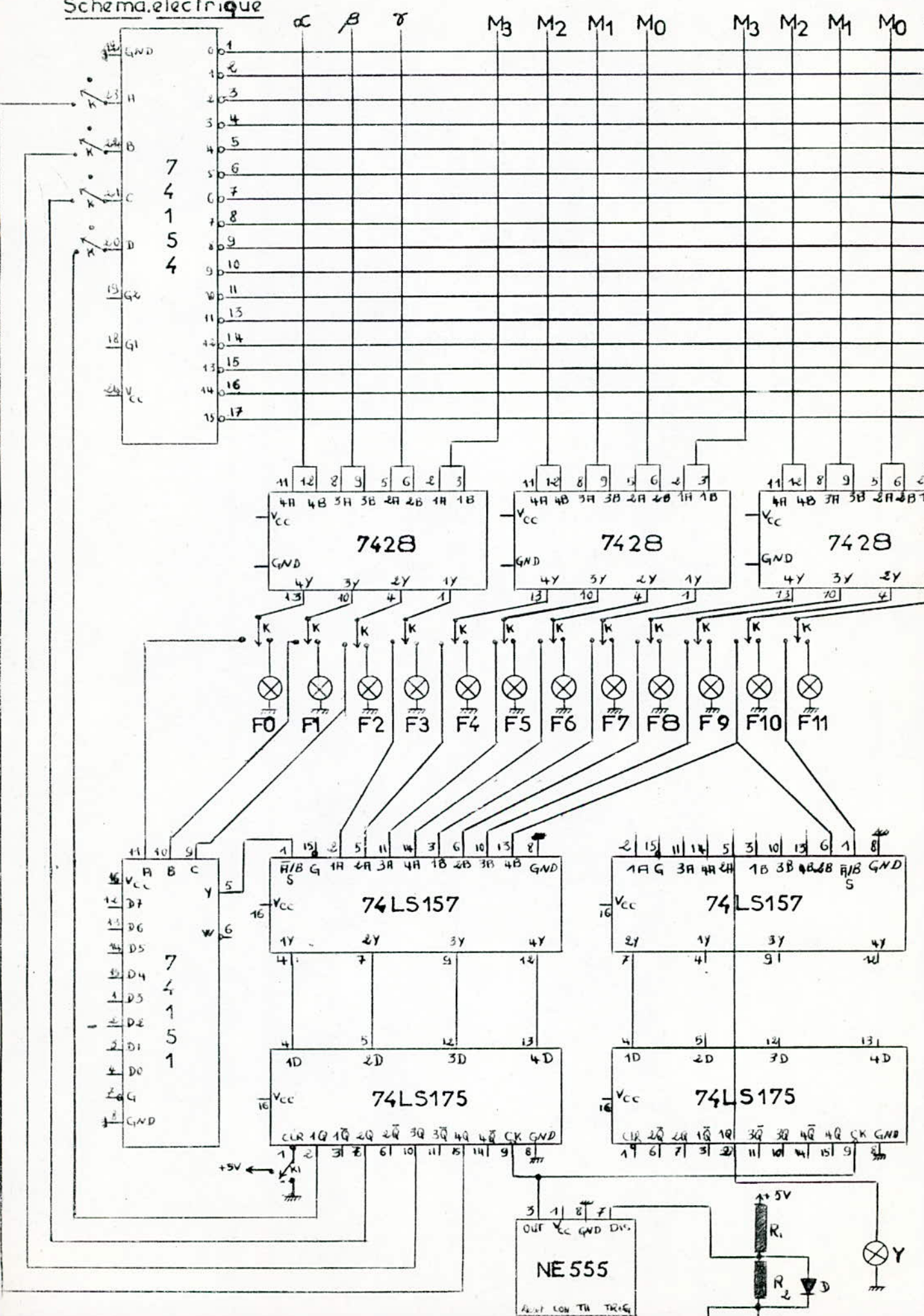
SCHEMA ELECTRIQUE

BROCHE DES CIRCUITS INTEGRES

CARACTERISTIQUES ELECTRIQUES DES C.I

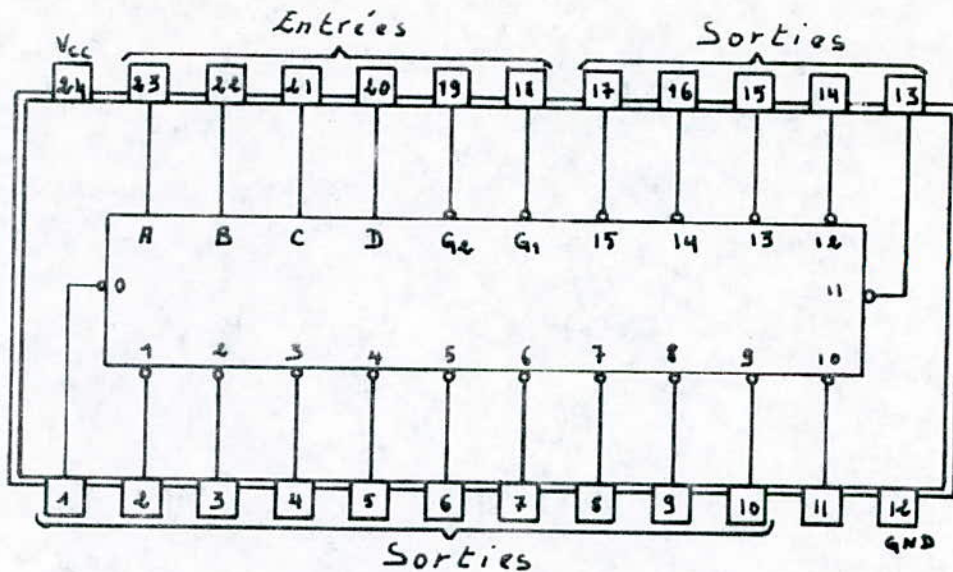
TABLES D'EXCITATION DES C.I

# Schéma électrique





# 74154 (DECODEUR 4 → 16)



brochage du 74154

## conditions de fonctionnement recommandées

	MIN	NOM	MAX	unit
Supply Voltage, $V_{cc}$	4.75	5	5.25	V
High-Level Output Current, $I_{OH}$			-800	$\mu A$
Low-Level Output Current, $I_{OL}$			16	mA
Operating free-air temperature, $T_A$	0		70	$^{\circ}C$

## caractéristiques électriques

	MIN	TYP	MAX	unit
High-Level Input Voltage, $V_{IH}$	2			V
Low-Level Input Voltage, $V_{IL}$			0.8	V
High-Level Output Voltage, $V_{OH}$	2.4	3.4		V
Low-Level Output Voltage, $V_{OL}$		0.2	0.4	V
Input Current at maximum Input Voltage			1	mA
High-Level Input Current, $I_{IH}$			40	$\mu A$
Low-Level Input Current, $I_{IL}$			-1.6	mA
Short-circuit Output Current, $I_{OS}$	-18		-57	mA

Toutes les valeurs typiques sont données pour  $V_{cc} = 5V$  et  $T_A = 25^{\circ}C$

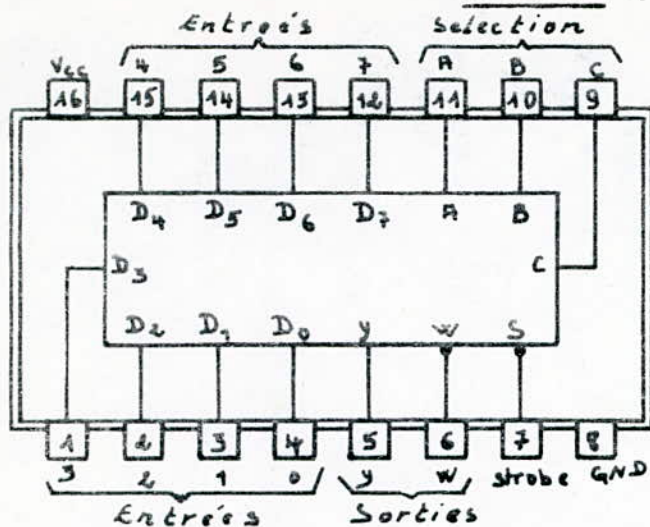
## temps de propagation et puissance dissipée

Temps typique de Propagation		Puissance typique dissipée
Etat Logique	Validation	
23 ns	13 ns	170 mW

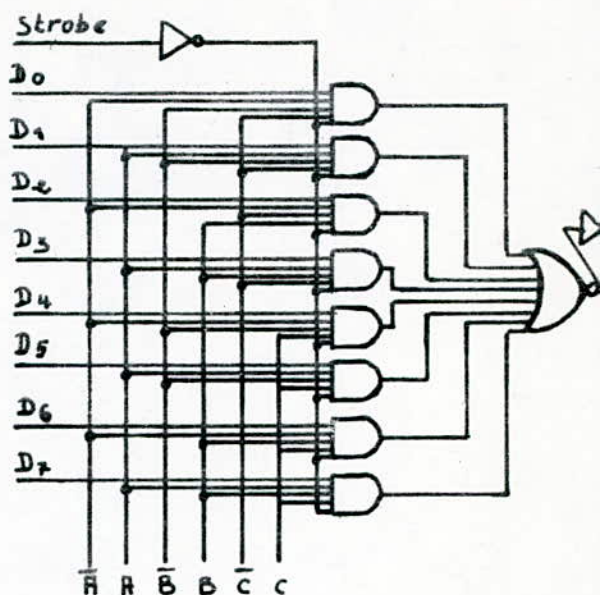




## 74LS151 (MULTIPLEUR 8→1)



brochage du 74ls151



### conditions de fonctionnement recommandées

	MIN	NOM	MAX	Unit
Supply Voltage, $V_{CC}$	4,75	5	5,25	V
High-Level Output Current, $I_{OH}$			-800	$\mu A$
Low-Level Output Current, $I_{OL}$			8	mA
Operating free-air temperature, $T_H$	0		70	$^{\circ}C$

### caractéristiques électriques

	MIN	typ	MAX	unit
High-Level Input Voltage, $V_{IH}$	2			V
Low-Level Input Voltage, $V_{IL}$			0,8	V
High-Level Output Voltage, $V_{OH}$	4,7	3,4		V
Low-Level Output Voltage, $V_{OL}$		0,35	0,5	V
Input Current at maximum input voltage			0,1	mA
High-Level input Current, $I_{IH}$			20	$\mu A$
Low-Level input Current, $I_{IL}$			-0,4	mA
Short-circuit Output Current, $I_{OS}$	-5		-42	mA

Toutes les valeurs sont données pour  $V_{CC} = 5V$  et  $T_H = 25^{\circ}C$

FIG: IV.2.a



# 74LS151

ENTRÉES				SORTIES	
Selection				Y	W
C	B	A			
X	X	X	H	L	H
L	L	L	L	$D_0$	$\overline{D_0}$
L	L	H	L	$D_1$	$\overline{D_1}$
L	H	L	L	$D_2$	$\overline{D_2}$
L	H	H	L	$D_3$	$\overline{D_3}$
H	L	L	L	$D_4$	$\overline{D_4}$
H	L	H	L	$D_5$	$\overline{D_5}$
H	H	L	L	$D_6$	$\overline{D_6}$
H	H	H	L	$D_7$	$\overline{D_7}$

*Table de fonctionnement (74LS151)*

*H: High level (Niveau Haut : "1" Logique)*

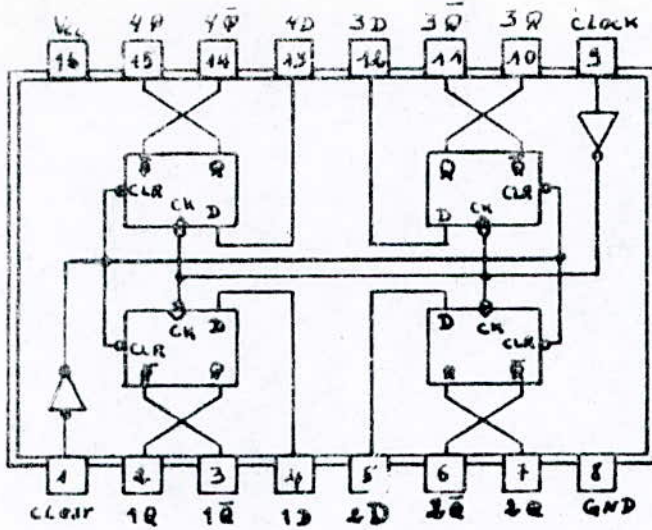
*L: Low level (Niveau bas : "0" Logique)*

*$D_0, D_1, \dots, D_7$ : Niveau respectif de l'entrée D.*

<i>Temps de propagation</i>	<i>Puissance typique dissipée</i>
11 ns	30 mW

FIGURE : IV.2. b

# 74LS175 (QUADRUPLE BASCULE D AVEC RESET)



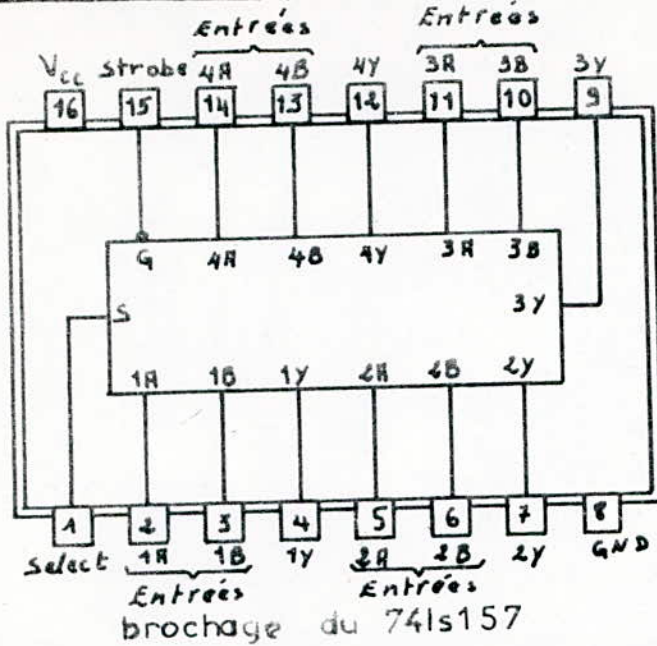
brochage du 74ls175

Entrées			Sorties	
clear	clock	D	Q	$\bar{Q}$
L	X	X	L	H
H	↑	H	H	L
H	↑	L	L	H
H	L	X	$Q_0$	$\bar{Q}_0$

Table de fonctionnement du 74LS175  
(pour chaque bascule)

FIGURE : IV.2.c

# 74LS157 (QUADRUPLE 2→1)



INPUTS				Sorties
Strobe	Select	A	B	Y
H	X	X	X	L
L	L	L	X	L
L	L	H	X	H
L	H	X	L	L
L	H	X	H	H

Table de fonctionnement (74LS157)

## conditions de fonctionnement recommandées

	MIN	NOM	MAX	UNIT
Supply Voltage, $V_{CC}$	4,75	5	5,45	V
High-Level Output Current, $I_{OH}$			-400	$\mu A$
Low-Level Output Current, $I_{OL}$			8	mA
Operating free-air temperature, $T_A$	0		70	$^{\circ}C$

## caractéristiques électriques

	MIN	Typ	MAX	UNIT
High-Level Input Voltage, $V_{IH}$	2			V
Low-Level Input Voltage, $V_{IL}$			0,8	V
High-Level Output Voltage, $V_{OH}$	2,4	3,4		V
Low-Level Output Voltage, $V_{OL}$		0,2	0,4	V
Input Current at maximum input Voltage			1	mA
High-level Input Current, $I_{IH}$			20	$\mu A$
Low-Level Input Current, $I_{IL}$			-0,8	mA
Short-circuit Output Current, $I_{OS}$	-9		-48	mA

## temps de propagation et puissance dissipée

temps typique de propagation	Puissance typique dissipée
18 ns	75 mW

FIG: IV.4.a



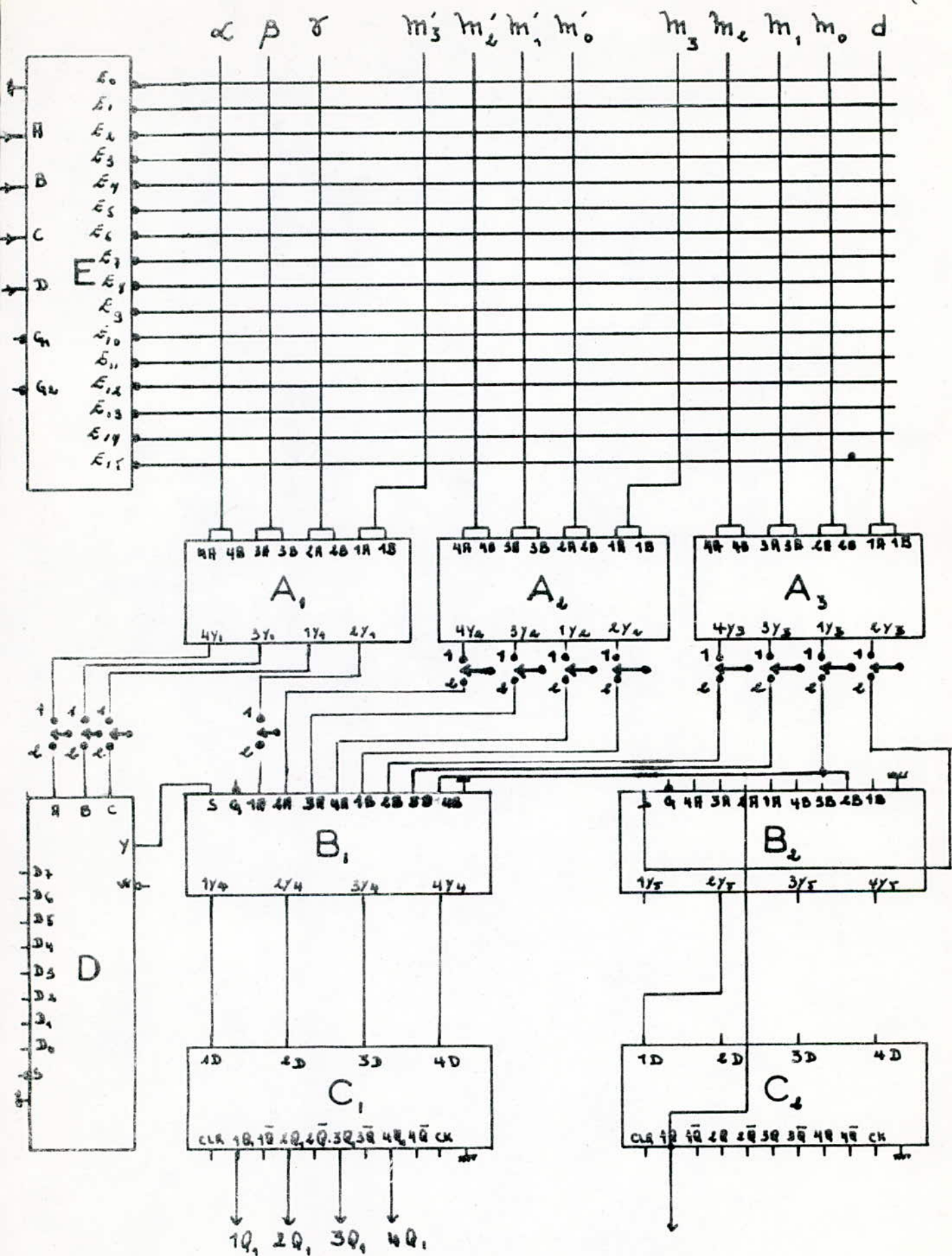


FIG IV Schéma pratique pour éllustrer les tableaux d'excitation des C. I

Table d'excitation du 7428 de la fig M

NOM	TYPE	ENTREES	ACTIVITE	CONDITIONS LOGIQUES
A1	7428	4A	D	$\alpha$
		4B	D	$\alpha$
		3A	D	$\beta$
		3B	D	$\beta$
		2A	D	$\gamma$
		2B	D	$\gamma$
		1A	D	M3
		1B	D	M3
A2	7428	4A	D	M2
		4B	D	M2
		3A	D	M1
		3B	D	M1
		2A	D	M0
		2B	D	M0
		1A	D	M3
		1B	D	M3
A3	7428	4A	D	M2
		4B	D	M2
		3A	D	M1
		3B	D	M1
		2A	D	M0
		1A	D	d
		1B	D	d

FIGURE: IV.3 . b

Table d'excitation du 74 ls 157 de la fig : IV

NOM	TYPE	ENTREES	ACTIVITE	CONDITIONS LOGIQUES
B <sub>1</sub>	74ls 157	1A	D	2 Y1
		1B	D	2 Y2
		2A	D	4 Y2
		2B	D	3 Y3
		3A	D	3 Y2
		3B	D	3 Y3
		4A	D	1 Y2
		4B	D	1 Y3
		S	D	Y
G	I	1		
B <sub>2</sub>	74ls 157	1A	D	
		1B	D	
		2A	D	1 Q
		2B	D	1 Y3
		3A	D	
		3B	D	
		4A	D	
		4B	D	
		S	D	2 Y3
G	I	1		

FIG : IV.4 . b



T

## Table d'excitation du 74ls175 et du 74151

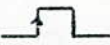

NOM	TYPE	ENTREES	ACTIVITE	CONDITIONS LOGIQUES
C <sub>1</sub>	74 ls 175	1D	D	1Y4
		2D	D	2Y4
		3D	D	3Y4
		4D	D	4Y4
		clear	1	1
		clock		H
C <sub>2</sub>	74 ls 175	1D	D	2Y5
		2D	D	
		3D	D	
		4D	D	
		clear	1	1
		clock		H
D	74151	D0	D	1
		D1	D	1
		D2	D	1
		D3	D	1
		D4	D	1
		D5	D	1
		D6	D	1
		D7	D	1
		A	D	4Y1
		B	D	3Y1
		C	D	1Y1
S	1	1		

Fig : IV.2 (d)

