Conception logique et topologique en technologie MOS

Ghislaine Thuau

THESE

présentée à

l'Institut National Polytechnique de Grenoble

pour obtenir le grade de
DOCTEUR de 3ème cycle
«Génie Informatique»

par

THUAU Ghislaine

CONCEPTION LOGIQUE ET TOPOLOGIQUE EN TECHNOLOGIE MOS.

Thèse soutenue le 7 décembre 1983 devant la commission d'examen.

	P. GENTIL	Président
	M. DEPEYROT	Examinateurs
	M. LAURENT	
	C. PAYAN	
	J.L. RAINARD	
Mme	G. SAUCIER	

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Année universitaire 1982-1983

Président de l'Université : D. BLOCH

Vice-Président : René CARRE
Hervé CHERADAME
Marcel IVANES

PROFESSEURS DES UNIVERSITES :

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

.../...

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

PROFESSEURS ASSOCIES

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

PROFESSEURS UNIVERSITE DES SCIENCES SOCIALES (Grenoble II)

BOLLIET Louis
Chatelin Françoise

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

RIEU Jean
SOUSTELLE Michel

CHERCHEURS DU C.N.R.S.

FRUCHART Robert	Directeur de Recherche
VACHAUD Georges	Directeur de Recherche

.../...

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

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

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

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

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

.../...

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

.../...

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

LABORATOIRES EXTERIEURS

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

Je tiens à exprimer toute ma reconnaissance à Madame Gabrièle SAUCIER, Professeur à l'ENSIMAG, de m'avoir accueilli dans son équipe de recherche et d'avoir dirigé mes travaux.

Je tiens à remercier :

Monsieur P. GENTIL, Professeur à l'ENSERG, de m'avoir fait l'honneur de présider le jury de cette thèse,

Monsieur J.L. RAINARD, Ingénieur au CNS, d'avoir accepté d'être rapporteur de cette thèse et d'avoir bien voulu me faire profiter de ses remarques constructives.

Monsieur M. DEPEYROT, Directeur Scientifique à EFCIS,

Monsieur M. LAURENT, Directeur C.A.O. à Matra-Harris,

Monsieur C. PAYAN, Maître de Recherche au CNRS,

d'avoir accepté de faire partie du jury de cette thèse.

Je tiens également à remercier :

Tous mes collègues de l'équipe "Circuits et Systèmes", tout particulièrement C. BELLON et R. VELAZCO,

Toutes les personnes qui ont assuré la réalisation technique de cet ouvrage : Madame S. ROCHE pour la frappe, ainsi que Monsieur D. IGLESIAS et l'équipe de reprographie de l'IMAG pour le tirage.

Que soient aussi remerciés mes amis qui, par leur soutien et leur aide, ont contribué à ce travail.

TABLE DES MATIERES

III - PORTES COMPLEXES NMOS

III - 1. Généralités
III - 2. Implantation d'une porte complexe NMOS dont le réseau $\overline{R}$ est lexicographique
III - 3. Dimensionnement d'une porte complexe NMOS
III - 4. Portes complexes NMOS à précharge

IV - PORTES COMPLEXES CMOS

IV - 1. Généralités
IV - 2. Implantation d'une porte complexe CMOS
IV - 3. Synthèse logique et topologique (cas d'une porte)
IV - 4. Dimensionnement d'une porte complexe CMOS
IV - 5. Portes complexes CMOS à précharge
4.5.1. Précharge au niveau logique '1' de toutes les portes
4.5.2. Précharge aux niveaux logiques '0' et '1' de façon alternée

V - MACRO-CELLULES REALISANT DES FONCTIONS INDEPENDANTES

V - 1. Implantation de macro-cellules NMOS
V - 2. Implantation de macro-cellules CMOS

VI - MACRO-CELLULES REALISANT DES FONCTIONS DEPENDANTES

VI - 1. Composition de portes, générateurs de base
VI - 2. Décomposition d'une porte trop complexe
VI - 3. Réalisation de plusieurs fonctions des mêmes variables
6.3.1. Recherche de monômes communs à plusieurs fonctions
6.3.2. Sous-fonctions communes à plusieurs fonctions
Cas particulier : plusieurs fonctions réalisées à partir de l'une d'entre elles
VI - 4. Implantation de macro-cellules réalisant des fonctions dépendantes

CONCLUSION

CHAPITRE II : LOGIQUE DE TRANSFERT

INTRODUCTION

I - DEFINITION DE LA LOGIQUE DE TRANSFERT

I - 1. Principe de base

I - 2. Caractéristiques électriques et topologiques

II - FORME CANONIQUE DE LA LOGIQUE DE TRANSFERT

II - 1. Définition

II - 2. Réalisation systématique de la forme canonique

III - FORME ARBORESCENTE DE LA LOGIQUE DE TRANSFERT

III - 1. Intérêt de la forme arborescente

3.1.1. Position du problème

3.1.2. Bibliographie

3.1.3. Position par rapport à la bibliographie

III - 2. Construction de l'arbre de décision binaire à partir de la forme canonique

III - 3. Réalisation systématique de l'arbre canonique

III - 4. Construction de l'arbre de décision binaire à partir de la forme non canonique

III - 5. Optimisation de l'arbre de décision binaire

IV - LOGIQUE DE TRANSFERT A VARIABLE PRIVILEGIEE

IV - 1. Définition

IV - 2. Forme canonique et logique de transfert à variable privilégiée

IV - 3. Forme arborescente et logique de transfert à variable privilégiée

IV - 4. Application à la technologie CMOS

V - IMPLANTATION DE LA LOGIQUE DE TRANSFERT

V - 1. Réalisation systématique

V - 2. Différents types d'implantation

V - 3. Implantation automatisée

VI - EXTENSION A DES SYNTHESES A PLUSIEURS COUCHES

VI - 1. Composition de fonction réalisées en logique de transfert
VI - 2. Décomposition d'une fonction trop complexe
VI - 3. Réalisation de plusieurs fonctions des mêmes variables

CHAPITRE III : CONCEPTION DE CIRCUITS COMPLEXES

INTRODUCTION

I - LES SPECIFICATIONS INITIALES OU CAHIER DES CHARGES DU CIRCUIT

I - 1. Spécifications fonctionnelles initiales
I - 2. Les contraintes

II - PRINCIPES D'UNE METHODE DE CONCEPTION DESCENDANTE

II - 1. Principe général
II - 2. Cas de la conception d'un circuit

III - SPECIFICATIONS ALGORITHMIQUES

IV - CHOIX DE LA STRUCTURE DU CHEMIN DE DONNEES : SPECIFICATION DU CIRCUIT SOUS FORME DE GRAPHE DE CONTROLE INTERPRETE

IV - 1. Choix du chemin de données
IV - 2. Outil de spécification

V - PROBLEMES TEMPORELS, LOGIQUE STATIQUE ET BIPAHSEE : GRAPHE DE CONTROLE INTERPRETE ET TEMPORISE

V - 1. Logique statique
V - 2. Logique biphasée
5.2.1. Principe général
5.2.2. Modes fondammentaux
5.2.3. Modes non fondammentaux
V - 3. Les solutions dans le choix de la partie contrôle

INTRODUCTION

Ce travail concerne la conception logique et topologique en technologie MOS. Le premier chapitre propose une méthode de conception logique de cellules MOS facilement implémentables. L'élément de base est le réseau "d'interrupteurs" MOS, dit lexicographique. Il s'agit de réseaux dans lesquels l'ordre d'occurence des variables d'entrée sur chaque chemin est le même. Ceci permettra en effet des implantations systématiques sur des matrices dont les lignes sont des diffusions, les colonnes du Silicium Polycristallin ou de l'aluminium. Un ordonnancement optimisé des variables est proposé, c'est à dire un ordonnancement permettant une simplification du réseau par compactage des transistors. Cette simplification doit laisser le réseau planaire. Dans un premier temps une forme arborescente série-parallèle est recherchée, puis une 2ème étape de minimisation amène à un réseau planaire quelconque. Des extensions à des bandes de telles cellules sont ensuite étudiées.

Dans ce premier chapitre, la méthode de synthèse logique proposée prend en compte les aspects technologiques et les problèmes d'implantation (structure régulière, interconnexions, caractérisations des technologies NMOS et CMOS) et s'est définitivement éloigné des méthodes classiques de synthèse en portes NOR, NAND, etc....

Dans le deuxième chapitre, le point de départ est la spécification d'une fonction booléenne sous forme d'arbre de décision binaire. Après avoir montré les simplifications possibles, l'implantation sur silicium est expliquée, en technologie NMOS. Cette synthèse est originale en circuiterie et est intéressante car elle permet une traduction automatique d'une description fonctionnelle en circuit (compilateur de Silicium).

Dans le troisième chapitre, la méthodologie de conception de circuits à haute complexité du type microprocesseur est abordé ; la méthode de conception sûre par affinements de spécifications est rappelée et est illustrée sur deux exemples : la conception d'un circuit d'extraction de racine carrée en technologie NMOS, statique, la conception d'un multiplieur en logique biphasée. L'étude a essentiellement porté sur les problèmes de synchronisation entre le contrôleur et les ressources contrôlées ainsi que sur l'aspect temporisé de façon générale. Dans ce troisième chapitre, on peut estimer qu'une démarche de conception bien structurée a été proposée et appliquée à des réalisations pratiques. La formulation des problèmes de temporisation est très intéressante et permettra, peut être, de rendre plus claire et donc plus sûre la conception de tels circuits.

CHAPITRE I

PORTES COMPLEXES

INTRODUCTION

Les cellules MOS ont largement été étudiées dans le passé. Les résultats concernent principalement les deux aspects suivants :

(i) La synthèse logique

La synthèse logique étudiée dans (LIU 75) (EL-Z 77) (MUR 82) tient compte de la limitation du nombre de transistors dans une cellule (en particulier le nombre de transistors mis en série) et considère que les variables d'entrée n'apparaissent que sous une forme (directe ou complémentée). Les auteurs proposent une décomposition optimisée d'une fonction booléenne répondant à ces critères. Le résultat est une cellule complexe ou une macro-cellule faite de cellules élémentaires interconnectées.

(ii) La synthèse topologique

Partant d'une expression booléenne, les auteurs dans (UEH 81) cherchent une implantation optimale, c'est-à-dire une surface minimale. Une organisation optimale est souvent recherchée dans une seconde étape. D'autres approches donnent plus d'importance aux avantages d'un passage automatique de la description logique à l'implantation (compilateur de Silicium) (EIC 82)(KUB83). Si les cellules élémentaires sont des portes NOR ou NAND, une "bande" logique régulière est obtenue facilement et le dimensionnement des transistors peut se faire de façon automatique (CRO 82) (DUN 81) (MAJ 78).

L'approche présentée ici a les caractéristiques suivantes :

(i) Les problèmes de synthèse logique et topologique sont considérées simultanément pour obtenir une meilleure optimisation ; la synthèse logique est réalisée en se référant aux caractéristiques topologiques.

(ii) Une structure systématique des cellules MOS est proposée facilitant l'interconnexion de celles-ci et l'organisation générale d'un ensemble de cellules en une structure à bande logique.

(iii) Les variables directes et complémentées sont supposées être disponibles à l'entrée d'une cellule. En effet, nous sommes principalement concernés par la conception de logique pour calculateurs, il semble donc raisonnable de considérer que les cellules logiques des circuits sont reliées des deux côtés aux éléments de mémorisation d'une façon très régulière (SAU 82). Les sorties directes et complémentées sont alors disponibles des éléments mémoires.

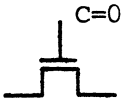
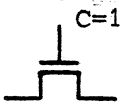
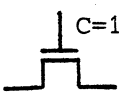
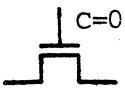
(iv) L'algorithme proposé ici est très simple et peut s'appliquer à des cellules de grande complexité (logique dynamique).

I - GENERALITES SUR LES RESEAUX D'INTERUPTEURS ET LES FONCTIONS DE TRANSFERT ASSOCIEES

Au niveau logique, l'étude des portes complexes se ramène à l'étude de réseaux d'interupteurs et des fonctions de transfert associées.

I - 1. Le transistor MOS vu comme un interrupteur

On considère ici comme élément de base le transistor MOS réalisant la fonction d'interrupteur. Les conventions d'ouverture et de fermeture d'un tel transistor suivant la condition C appliquée sur la grille du transistor sont données dans le tableau suivant :

	Transistor NMOS	Transistor PMOS
interrupteur ouvert		
interrupteur fermé		

avec la représentation habituelle du transistor MOS

Le transistor NMOS utilisé en interrupteur dégrade le niveau logique "1" d'une tension de seuil, le niveau logique "0" est transmis correctement. On a

$$Vseuil = VT + V(VBS)$$

avec Vseuil : tension de seuil du transistor
VT : tension de seuil du transistor à VBS = 0
VBS : tension Substrat (Bulk) Source

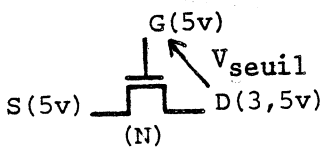
Pour un transistor NMOS : Vsubstrat (bulk) = VSS = 0

$$Vseuil = VT = 0,6 \text{ à } 0,7 \text{ V}$$

Si la tension VBS augmente, la tension V fonction de VBS augmente, ainsi que la tension de seuil Vseuil : c'est ce qu'on appelle l'effet substrat.

Pour un transistor NMOS utilisé en interrupteur :

$$VBS = 5V \quad = \quad Vseuil = 1,5V$$



La dégradation du niveau logique "1" est la même pour un ou plusieurs transistors en série.

(iii) Les variables directes et complémentées sont supposées être disponibles à l'entrée d'une cellule. En effet, nous sommes principalement concernés par la conception de logique pour calculateurs, il semble donc raisonnable de considérer que les cellules logiques des circuits sont reliées des deux côtés aux éléments de mémorisation d'une façon très régulière (SAU 82). Les sorties directes et complémentées sont alors disponibles des éléments mémoires.

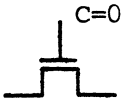
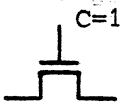
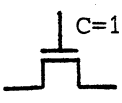
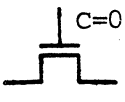
(iv) L'algorithme proposé ici est très simple et peut s'appliquer à des cellules de grande complexité (logique dynamique).

I - GENERALITES SUR LES RESEAUX D'INTERUPTEURS ET LES FONCTIONS DE TRANSFERT ASSOCIEES

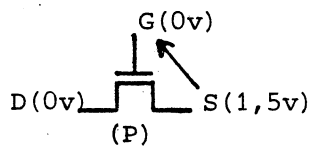
Au niveau logique, l'étude des portes complexes se ramène à l'étude de réseaux d'interupteurs et des fonctions de transfert associées.

I - 1. Le transistor MOS vu comme un interrupteur

On considère ici comme élément de base le transistor MOS réalisant la fonction d'interrupteur. Les conventions d'ouverture et de fermeture d'un tel transistor suivant la condition C appliquée sur la grille du transistor sont données dans le tableau suivant :

	Transistor NMOS	Transistor PMOS
interrupteur ouvert		
interrupteur fermé		

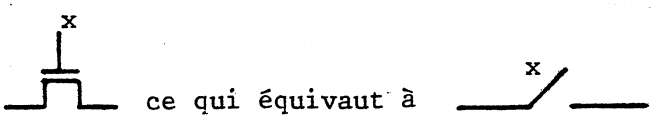
Pour la même raison, le transistor PMOS utilisé en interrupteur dégrade le niveau logique "0" d'une tension de seuil, le niveau logique "1" est transmis correctement.



I - 2. Réseaux d'interrupteurs et fonctions de transfert

La notion de fonction de transfert utilisée ici est la même que celle qui a été communément employée en théorie des relais (CAL 62).

Soit un réseau d'interrupteurs ayant une seule entrée et une seule sortie. A chaque interrupteur est associée une variable booléenne sous sa forme normale ou complémentée ; une variable x signifie que l'interrupteur est fermé (c'est-à-dire laisse passer l'information) si et seulement si $\overline{x} = 1$; x signifie que l'interrupteur est fermé si et seulement si $x = 0$. Cette convention correspond au fonctionnement du transistor NMOS, on représentera un interrupteur par :



Un chemin reliant l'entrée du réseau à sa sortie laisse passer l'information si tous les interrupteurs du chemin sont fermés ; il représente donc un produit de variables booléennes qui est un des monômes de la fonction de transfert réalisée par le circuit.

Le réseau de transfert laisse passer l'information de son entrée vers sa sortie si et seulement si au moins un des chemins laisse passer l'information. La fonction de transfert associée au réseau est une fonction booléenne qui est la somme des monômes associés à chaque chemin reliant l'entrée du réseau à sa sortie.

L'entrée et la sortie d'un réseau sont reliées si et seulement si la fonction de transfert du réseau vaut 1.

Exemple

Soit le réseau R (figure 1)

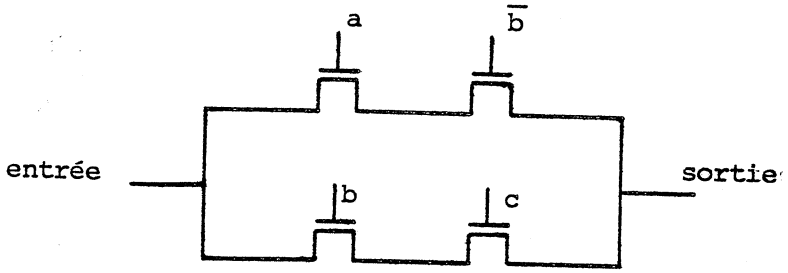


Figure 1 : Réseau d'interrupteurs

La fonction de transfert F est la somme de deux monômes correspondant aux deux chemins reliant l'entrée à la sortie.

$$F = a\,\bar{b} + b\,c$$

F est la fonction de transfert associée au réseau R, ou inversement le réseau R réalise la fonction F.

Remarque : La convention correspondant au fonctionnement du transistor PMOS est la suivante :
Une variable x signifie que l'interrupteur est fermé si et seulement si $x = 0$, $\bar{x}$ signifie que l'interrupteur est fermé si et seulement si $x = 1$.

I - 3. Réseau dual

On appelle dualité d'une quantité booléenne, l'application de l'ensemble des valeurs d'une variable booléenne simple sur lui-même définie par :

$0 \rightarrow 1$
$1 \rightarrow 0$

Soit x, une quantité booléenne simple, on désigne son dual par x^*. Il est immédiat de vérifier que :

$$(x^*)^* = x$$

Soit la fonction F(X), on nomme duale de cette fonction la fonction :

$F^*(X)$ définie par $F^*(X) = (F(\bar{X}))^*$ (KUN 68)

Remarque : La somme et le produit booléen sont des opérations duales :
$(a+b)^* = ab$, $(ab)^* = a + b$

Exemple

Soit $F = a\bar{b} + bc$

$F^* = (a+\bar{b})(b+c) = ab + ac + \bar{b}c$

En appliquant une méthode de minimisation quelconque on obtient $F^* = ab + \bar{b}c$

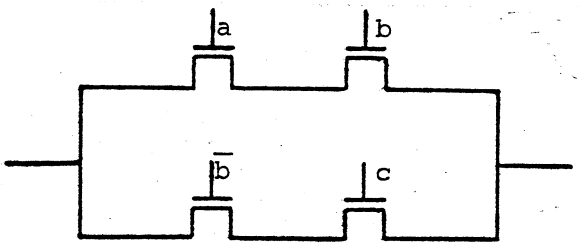


Figure 2a : Réseau dual R^* réalisant $F^* = ab + \bar{b}c$

Construction graphique et systématique du réseau dual

Dans certains cas simples (réseaux série-parallèle) le réseau dual peut être obtenu par construction directe sur le réseau. On n'obtient pas en général une solution minimale.

(i) Construction globale

Le réseau R^* dual topologique de R est le réseau dont les chemins sont l'ensemble des coupures de R (ensemble de chemins fictifs coupant la transmission entre l'entrée et la sortie).

Exemple

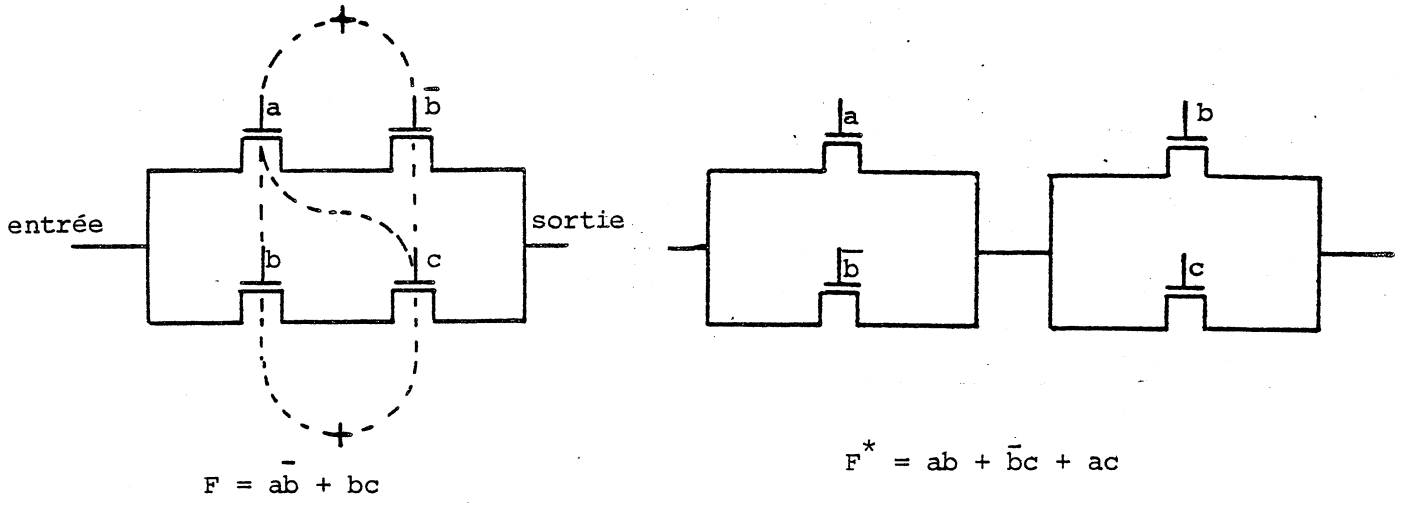


Figure 2b : Réseau R^* dual topologique de R

(ii) Construction par chemins

La remarque sur la dualité entre les fonctions somme et produit amène à une construction graphique systématique : chaque chemin ($a\bar{b}$ et bc) dans notre exemple) est transformé en une " coupure " sous forme de somme ($(a+\bar{b})$ et $(b+c)$). Ces coupures sont mises en série.

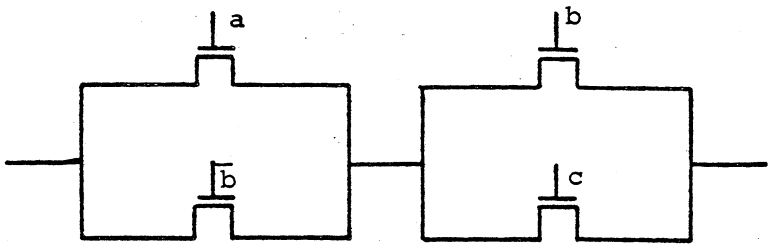


Figure 2c : Réseau dual R^*

Cette construction graphique systématique amène à un réseau appelé réseau dual topologique.

I - 4. Réseau complémentaire

On nomme complément d'une variable booléenne générale X de composantes xi, la variable booléenne $\bar{X}$ de componsantes $\bar{x}i$.

On nomme complément de la fonction F(X), la fonction :

$$\bar{F}(X) = \overline{F(X)}$$

Il est bien clair que :

$$x^* = \bar{x} \ , \ X^* = \bar{X}$$

d'où $(\bar{\bar{X}}) = X$ mais $\bar{F}(X) \neq F^*(X)$

En définissant, la fonction duale F^* à partir de la fonction complémentaire, on a $F^*(X) = \bar{F}(\bar{X})$ (KUN 68).

Exemple

$$F = a\bar{b} + bc$$
$$\bar{F} = (\bar{a}+b)(\bar{b}+\bar{c}) = \bar{a}\bar{b} + \bar{a}\bar{c} + b\bar{c}$$

après simplification on obtient $F = ab + bc$.

On note $\bar{R}$ un réseau réalisant la fonction $\bar{F}$ complémentaire de F ; $\bar{R}$ est appelé réseau complémentaire du réseau R.

Les constructions graphiques sont identiques à celles pour le réseau dual et on parlera de réseau complémentaire topologique.

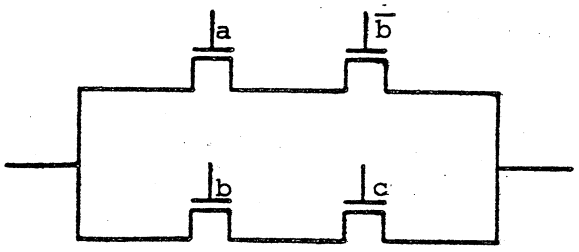


Figure 3a : Réseau R réalisant la fonction F = ab + bc

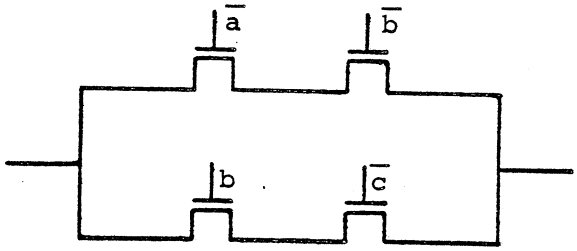


Figure 3b : Réseau R complémentaire minimisé

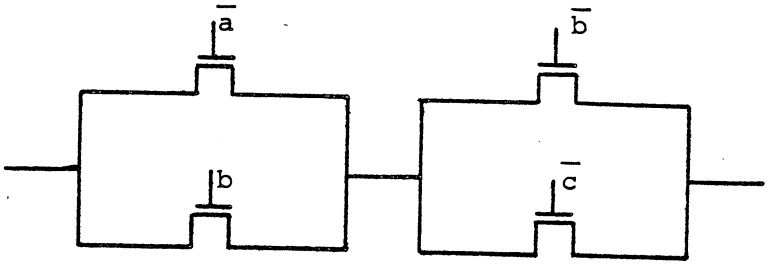


Figure 3c : Réseau R complémentaire topologique

Remarque : Avec des transistors PMOS, le réseau R de la figure 3a réalise la fonction de transfert :

$$F\begin{bmatrix}\bar{a}\\ \bar{b}\\ \bar{c}\end{bmatrix} = \bar{F}^{*}\begin{bmatrix}a\\ b\\ c\end{bmatrix}$$

I - 5. Portes complexes NMOS et CMOS en termes de réseaux

Dans ce chapitre, on étudiera les portes complexes dans les technologies NMOS et CMOS. Les portes logiques NMOS et CMOS sont définies par les schémas suivants, où $\bar{R}$ et $\bar{R}^*$ sont des réseaux d'interrupteurs (cf § III et IV) ; (figures 4a,b et 5a,b,c)

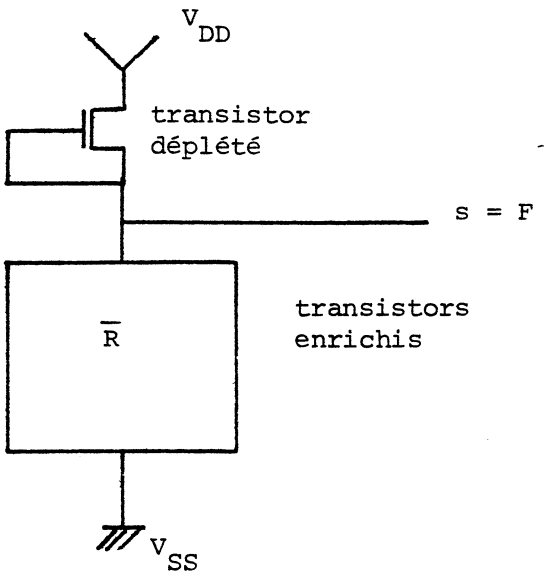


Figure 4a : Porte NMOS
à transistor de charge déplété

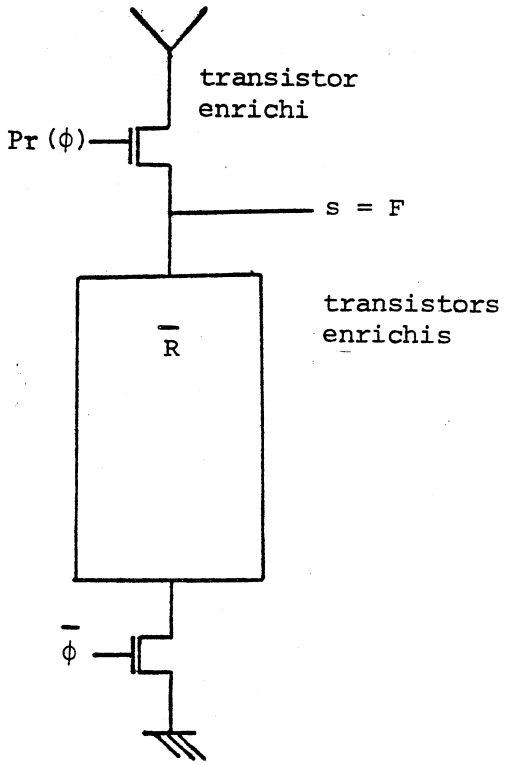


Figure 4b : Porte NMOS
à précharge sur la phase d'horloge ϕ

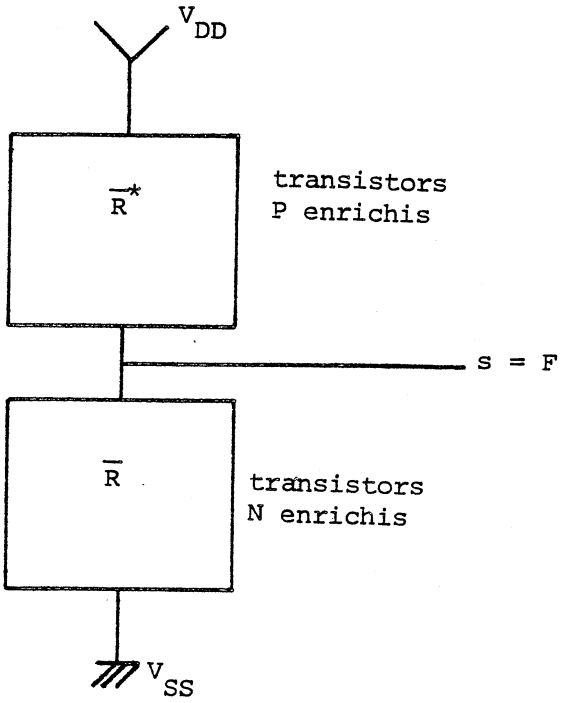


Figure 5a : porte CMOS classique

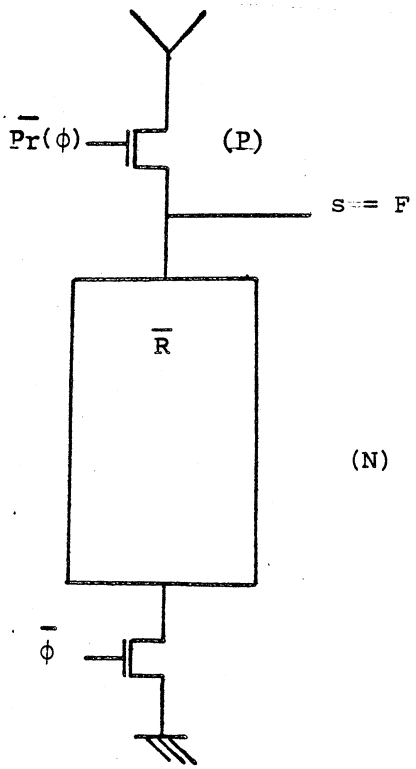


Figure 5b : porte CMOS à précharge au niveau logique '1'(pendant la phase ϕ)

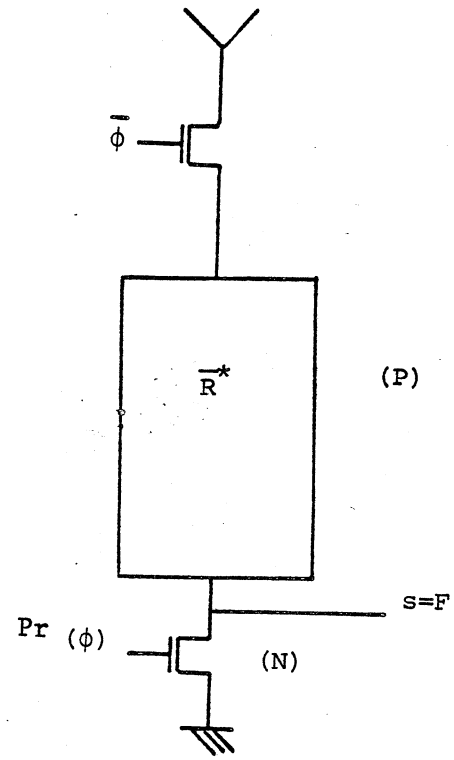


Figure 5 : porte CMOS à précharge au niveau logique '0' (pendant la phase ϕ)

Comme nous l'avons vu, l'élément de base des réseaux $\bar{R}$ et $\bar{R}^*$ est le transistor MOS qui joue le rôle d'interrupteur.

Pour obtenir des structures systématiques pour ces cellules MOS, nous définirons d'abord des structures systématiques mais néanmoins optimisées pour les réseaux élémentaires associés à ces cellules. Un réseau bien structuré est d'abord défini sur une grille virtuelle, le compactage réduira alors la surface globale de celui-ci.

I - 6. Réseau série-parallèle

Un réseau série parallèle est un réseau obtenu par construction (à partir d'un sous-réseau série-parallèle qui peut être réduit à un seul interrupteur) en appliquant les deux seules opérations de composition élémentaires :

- La mise en série ; un sous-réseau série-parallèle peut être remplacé par la mise en série de deux sous-réseaux série-parallèle
- La mise en parallèle : un sous-réseau série-parallèle peut être remplacé par la mise en parallèle de deux sous-réseaux série-parallèle.

Exemple

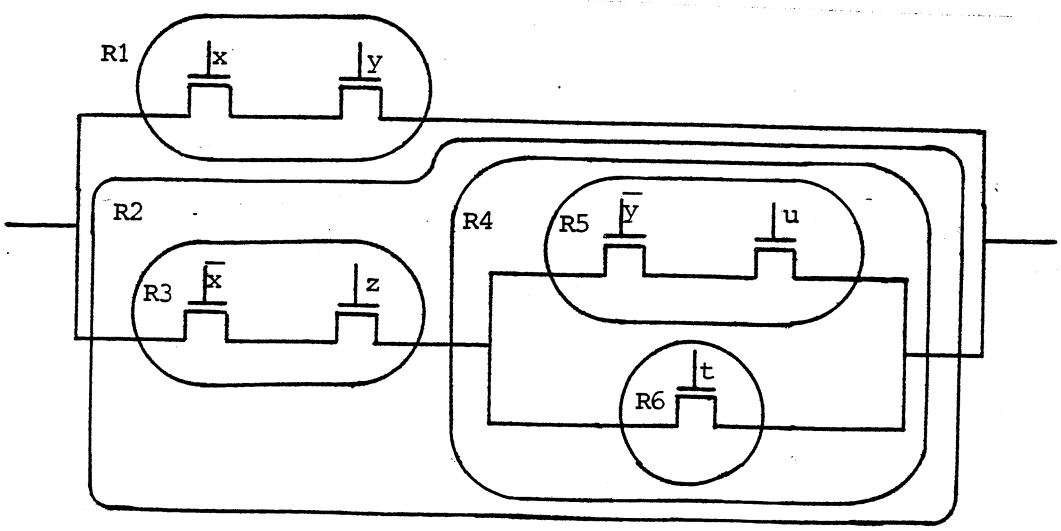


Figure 6 : Réseau R série-parallèle

En notant par II la mise en parallèle et o la mise en série, le réseau R peut s'écrire

$$R = R1 \text{ II } R2$$
$$R = R1 \text{ II } (R3 \text{ o } R4) = R1 \text{ II } (R3 \text{ o } (R5 \text{ II } R6))$$
$$R = (x \text{ o } y) \text{ II } ((\bar{x} \text{ o } z) \text{ o } ((\bar{y} \text{ o } u) \text{ II } t))$$

Il est clair que la fonction de transfert est automatiquement obtenue puisque la mise en série n'est autre que le produit booléen et la mise en parallèle la somme booléenne :

$$F = xy + (\bar{x}z\ (\bar{y}u + t))$$

I - 7. Réseau lexicographique

Un réseau a comme entrées les variables de la fonction de transfert et la valeur d'entrée du réseau proprement dite. Soit un réseau à réaliser sachant que (cf Figure 7) :

- chacune de ses variables d'entrée n'est disponible qu'à un seul endroit (qui reste à choisir),
- on ne dispose que d'une valeur de chacune des variables d'entrée, la valeur complémentaire étant générée à partir de la valeur disponible.

L'affectation des points d'entrée doit prendre en compte, lors de la réalisation du réseau, ses connexions avec d'autres réseaux.

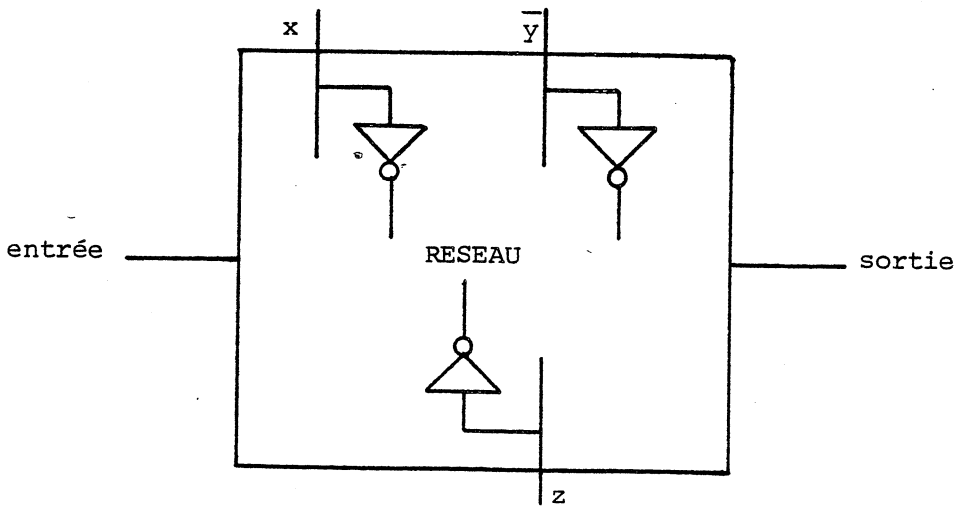


Figure 7 : Connexions du réseau avec l'extérieur

A l'intérieur du réseau à réaliser, il y a deux types de lignes :

- les lignes de variables
- les lignes de chemins entre l'entrée et la sortie du réseau.

Ces deux types de lignes sont réalisés aussi bien en technologie NMOS qu'en technologie CMOS par des matériaux différents (cf § III.2 et § IV.2). Les croisements entre deux lignes de types différents se réalisent sans problème par contre un croisement entre deux lignes de même type impose un changement de matériau.

Un réseau est facilement implantable s'il respecte la contrainte suivante : il n'y a pas de croisement entre deux lignes de même type (sachant que le croisement entre une ligne en Silicium Polycristallin et en diffusion créera un transistor).

Cette contrainte impose que les occurences des variables respectent le même ordre sur tous les chemins du réseau définissant ainsi un ordonnancement appelé ordonnancement lexicographique des variables, une variable et sa variable complémentée étant adjacentes. Un tel réseau sera appelé réseau lexicographique.

Un réseau lexicographique est facilement implantable avec (figure 8) :

- les variables sur des lignes verticales (colonnes),
- les chemins horizontaux (aux coudages près).

Exemples

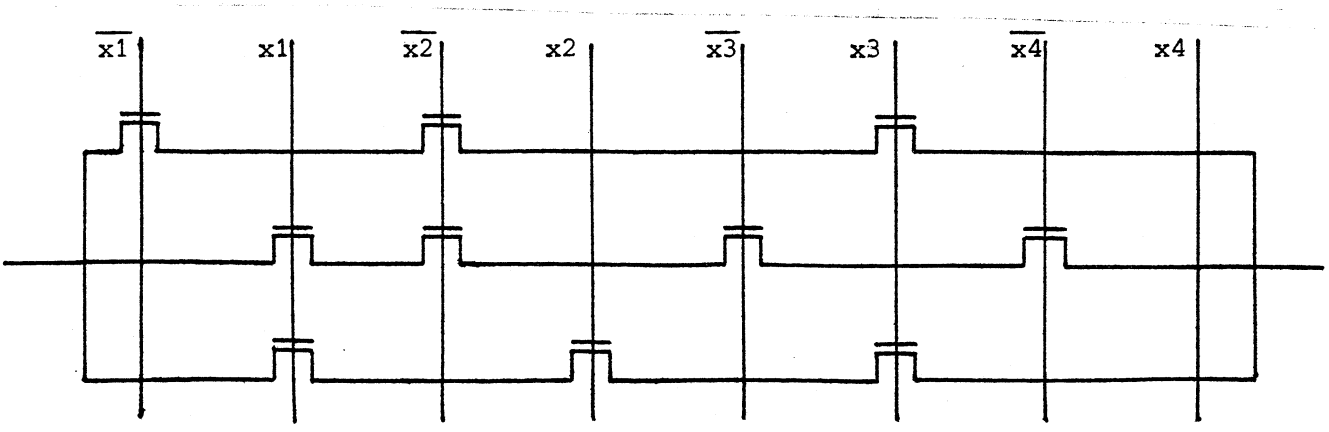


Figure 8 : Réseau lexicographique

Le réseau de la figure 8 réalise une fonction de transfert sous la forme : somme de monômes.

le réseau de la figure 1 peut être réalisé sous forme lexicographique avec, par exemple l'ordre suivant : a, b, c, (figure 9)

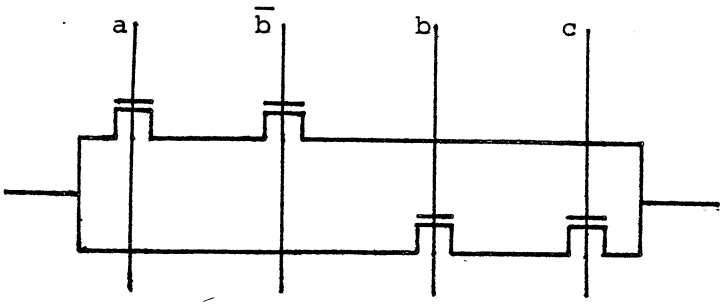


Figure 9 : Réseau de la figure 1 réalisé sous forme lexicographique

II - MINIMISATION D'UN RESEAU LEXICOGRAPHIQUE

II - 1. Réseau optimisé

Un réseau optimisé est un réseau facilement implantable (lexicographique) et à nombre minimal de transistors.

Il s'agit de réaliser un compromis entre :

- la facilité d'implantation
- le nombre de transistors.

En effet, le nombre minimal de transistors n'est pas forcément atteint pour une solution facilement implantable.
Un réseau bien structuré sera implanté sur une grille virtuelle et tassé dans une seconde étape.

Exemple

Soit la fonction F = ab + ac + ad + bcd à réaliser. Elle peut être réalisée avec 5 transistors (réseau de la figure 10a), mais sous forme non lexicographique. La réalisation sous forme lexicographique nécessite 7 transistors (figure 10b).

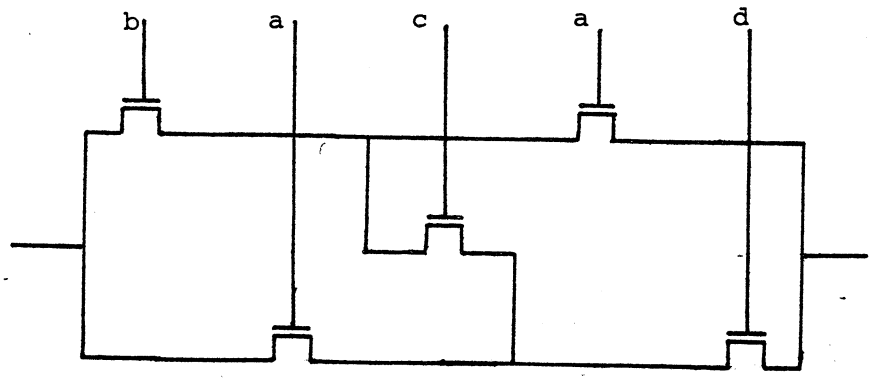


Figure 10a : Réseau non lexicographique réalisant F = ab + ac + ad + bcd

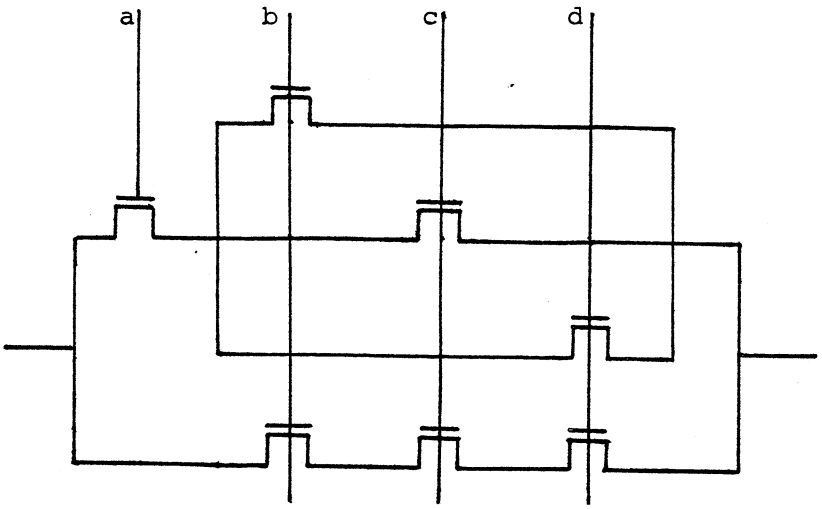


Figure 10b : Réseau lexicographique réalisant F = ab + ac + ad + bcd

Le problème étudié ici sera la recherche d'un ordonnancement des variables et d'un ordonnancement des lignes permettant d'atteindre un nombre minimal de transistors pour un réseau lexicographique.

On discutera d'abord (§ II.2) le choix de la forme initiale de la fonction à réaliser ; puis on étudiera la minimisation du réseau.

II - 2. Choix de la forme initiale de la fonction

2.2.1. Minimisation logique

De nombreux travaux ont concerné la minimisation logique des fonctions booléennes, le but étant l'écriture sous forme d'une base irrédondante de monômes ou monaux premiers, c'est à dire l'écriture à l'aide d'un nombre minimal de monômes ou de monaux (KUN 68).

L'écriture sous forme de base irrédondante est intéressante pour une réalisation à base de portes simples (NAND ou NOR) : elle minimise le nombre de ces portes.

Dans notre cas, la fonction est réalisée par une seule porte complexe sous forme de réseaux ; il faut donc rechercher une expression de la fonction qui minimise le nombre d'occurences de variables pour rechercher ensuite un réseau à nombre minimal de transistors ; ce nombre de transistors sera inférieur ou égal au nombre d'occurences de variables dans l'expression de la fonction.

Exemple

Soit la fonction F exprimée sous forme d'une base irrédondante en somme de monômes premiers :

$$F = A\bar{B}C + ABD + \bar{B}CD$$

elle est réalisable à l'aide de 4 portes simples à 3 entrées ; le nombre d'occurences de variables est 9.

Si on met B en facteur dans chaque monôme :

$$F = B(A\bar{C} + AD + \bar{C}D)$$

elle est alors réalisée avec 5 portes simples à 2 entrées ; cette expression comporte 7 occurences de variables.

Si on met A, C ou D en facteur dans cette nouvelle expression de F, la fonction

$F = B(A(\bar{C} + D) + \bar{C}D)$ est réalisable avec 5 portes simples à 2
$F = B(\bar{C}(A + D) + AD)$ entrées, cette expression a 6 occurences
$F = B(D(A + \bar{C}) + A\bar{C})$ de variables.

Remarque : Il est à noter que l'introduction de variables redondantes dans la fonction peut permettre d'obtenir une expression contenant moins d'occurences de variables (CAL 62).

Exemple

$F = AB(E + F) + CD(AE + BF)$ 10 occurences de variables

On peut remarquer alors :

$AB(E + F) = AB(AE + BF)$

d'où finalement :

$F = (AB + CD)(AE + BF)$ 8 occurences de variables

On a réduit de 10 à 8 le nombre d'occurences de variables de la fonction.

Il n'existe pas de méthode systématique permettant de trouver l'expression de la fonction ayant un nombre minimal d'occurences de variables. De plus, il faut noter qu'une telle expression :

- ne conduit pas toujours à une réalisation sous forme lexicographique
- ne conduit pas non plus à un réseau ayant un nombre minimal de transistors, elle donne simplement une borne supérieure au nombre de transistors.

Exemples

L'expression $(AB + CD)(AE + BF)$ de la fonction de l'exemple précédent a un nombre minimal d'occurences de variables, mais elle ne conduit pas à une réalisation lexicographique.

Le réseau de la figure 10a a un nombre minimal de transistors, mais la fonction qu'il réalise ne peut être exprimée par une expression avec le même nombre d'occurences de variables. En effet, seuls les réseaux série-parallèle implantent "directement" une expression logique et réalisent une bijection entre variable et transistor.

Nous nous proposons donc de rechercher un réseau optimisé (réseau lexicographique à nombre minimal de transistors) en partant d'une expression initiale de la fonction sous forme de base irrédondante. La minimisation sera faite de façon topologique, c'est à dire directement sur le réseau.

2.2.2. Choix entre les expressions somme de monômes et produit de monaux

On étudie ici la réalisation sous forme lexicographique des deux types d'expression.

Exemple 1

Soit la fonction F1 ayant quatre monômes premiers $a\bar{c}$, $\bar{b}\bar{c}$, ad, $\bar{b}d$. Sans mise en facteurs, un réseau lexicographique à 8 transistors est obtenu directement (figure 11a).

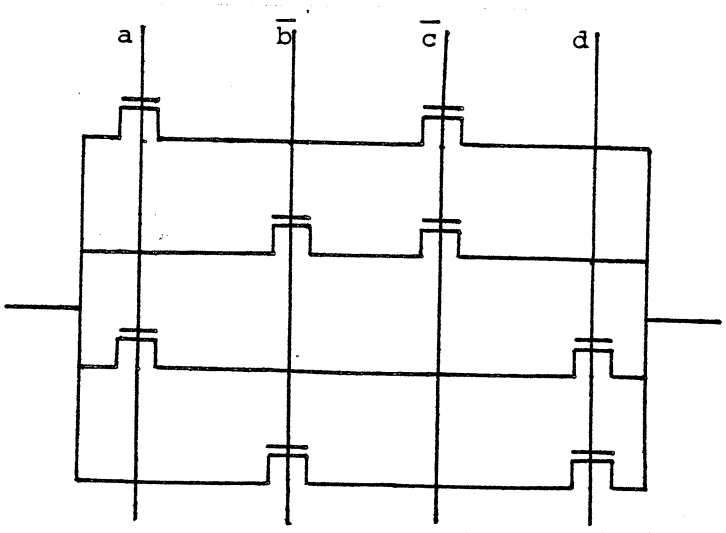


Figure 11a : Réseau réalisant F1 sous forme somme de monômes

La base irrédondante en produit de monaux donne $F1 = (a + \bar{b})(\bar{c} + d)$.

En choissant l'ordonnancement a, b, c, d des variables, on obtient directement la réalisation lexicographique à 4 transistors (figure 11b)

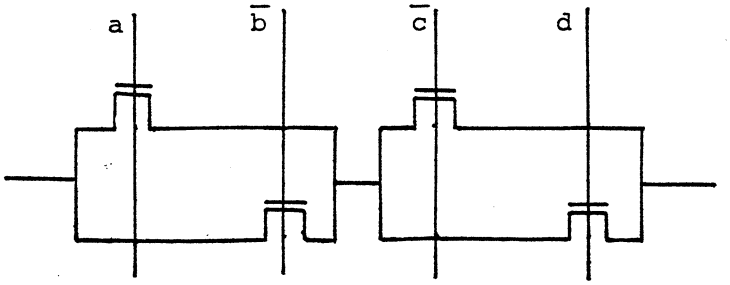


Figure 11b : Réseau réalisant F1 sous forme produit de monaux

Exemple 2

Soit la fonction F2 ayant une base irrédondante composée des quatre monômes premiers WY, $W\bar{Z}$, $\bar{W}\bar{X}\bar{Y}$, $\bar{W}\bar{X}Z$; cette forme conduit à une réalisation lexicographqiue à 10 transistors si l'on n'opère aucune mise en facteur (figure 12a)

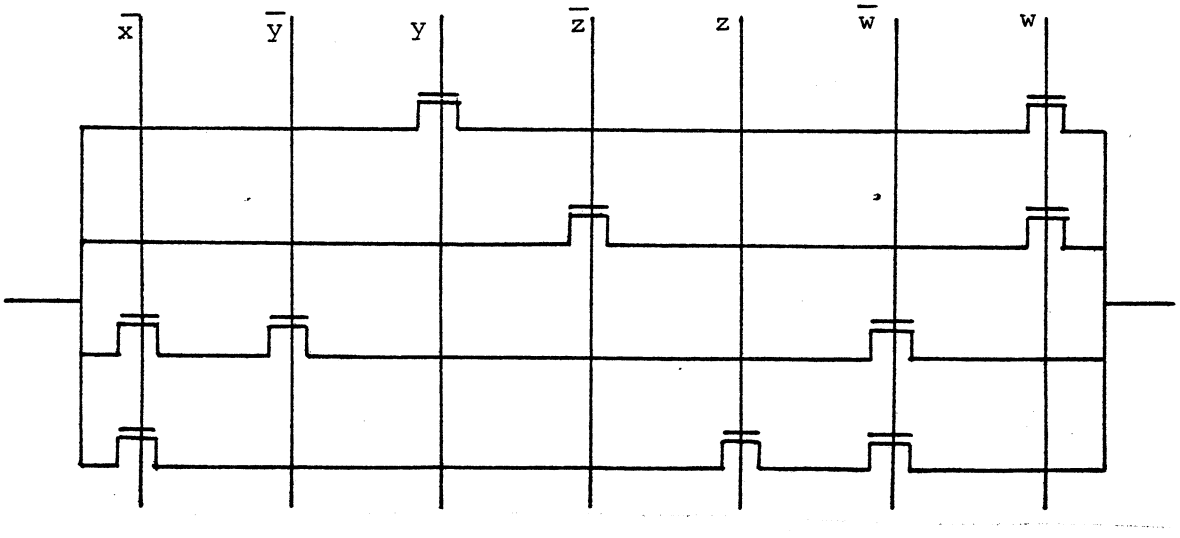


Figure 12a : Réseau réalisant F2 sous forme somme de monômes

L'écriture minimale sous forme de produit de monaux est :

$$F2 = (W + \bar{X})(W + \bar{Y} + Z)(\bar{W} + Y + \bar{Z})$$

Cette écriture conduit à une réalisation à 8 transistors. Pour réaliser ce réseau sous forme lexicographique, on sera conduit à plier le réseau (monaux les uns sous les autres) (figure 12b)

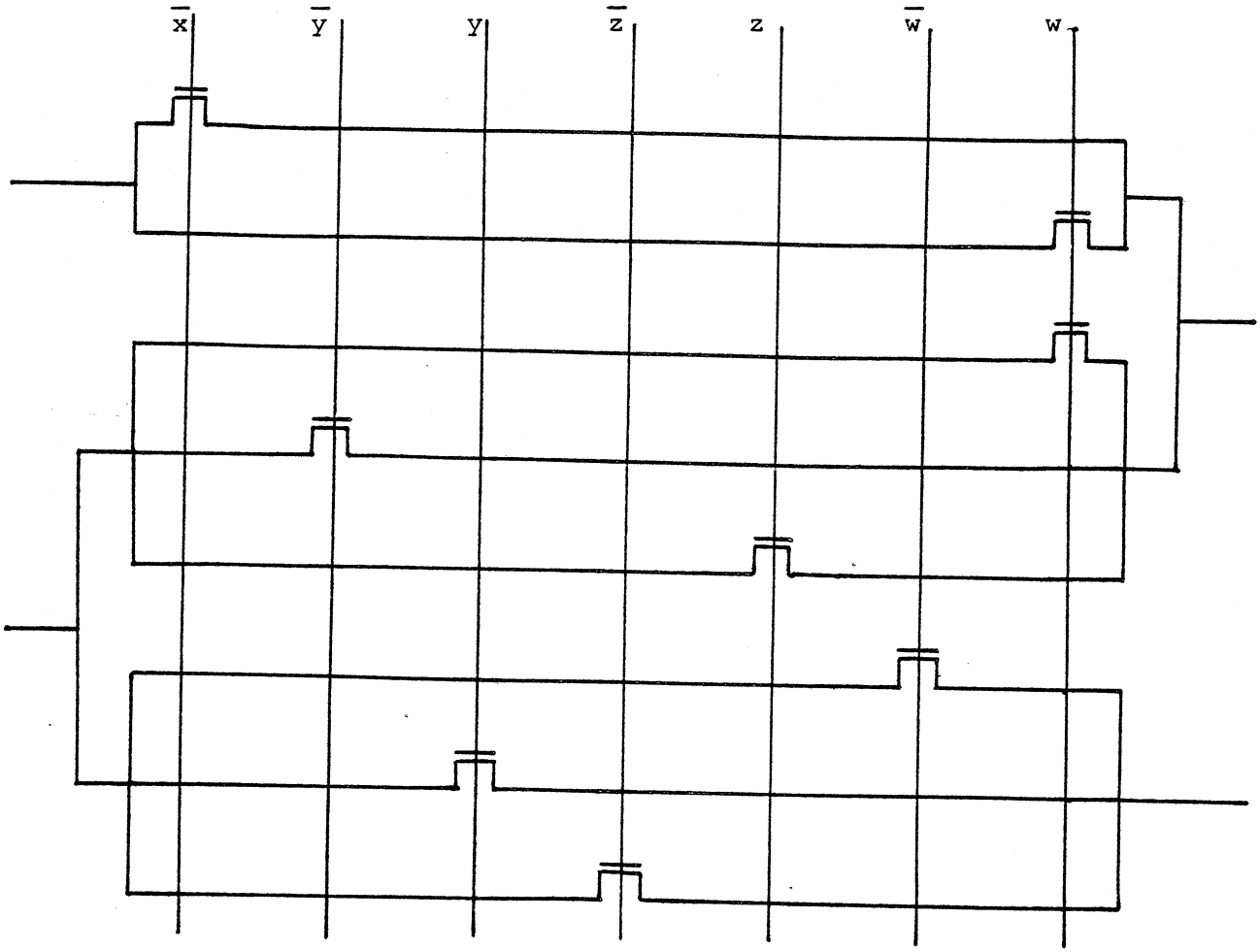


Figure 12b : Réseau réalisant F2 sous forme de produit de monaux

Les deux formes initiales : somme de monômes, produit de monaux, présentent donc une différence importante. Une base irrédondante en somme de monômes premiers pourra se traduire aisément par un réseau lexicographique.

La forme en produit de monaux premiers peut toujours se traduire sous forme lexicographique en pliant le réseau. Dans ce cas,on a des longueurs de chemins (diffusion cf. § III.2 et IV.2) dont il faut tenir compte pour le calcul du niveau logique "0". De plus, la surface du réseau est fonction du nombre de transistors en série et du nombre de diffusions en parallèle, important ici.

On procédera donc à l'écriture en base irrédondante sous les deux formes. On ne choisira la forme en produit de monaux que si celle-ci donne un réseau lexicographique non replié et avec moins de transistors. Sinon on préférera travailler sur le réseau exprimé sous forme de somme de monômes.

On se limitera dans les deux cas à 4 ou 5 transistors en série pour des raisons électriques et topologiques comme on le verra plus loin (cf. § III.1, IV.1).

On étudiera les réductions supplémentaires de transistors en travaillant directement sur le réseau.

II - 3. Optimisation d'un réseau réalisant une somme de monômes

L'optimisation d'un réseau se fera en cherchant à remplacer plusieurs transistors par un seul quand cela est possible, c'est à dire en fusionnant les transistors.

Exemple

Soit le réseau (figure 13a)

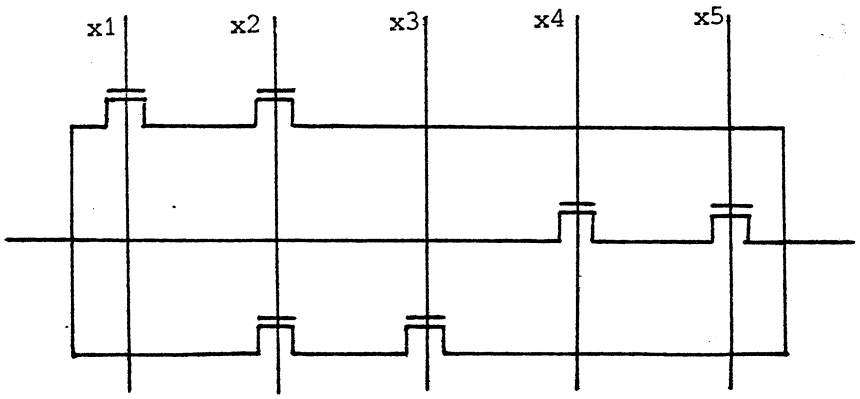


Figure 13a : Réseau initial

Après fusionnement des transistors de variables x2, en changeant l'ordre des chemins et avec l'ordre lexicographique suivant : x1, x3, x4, x5, x2 ; on obtient le réseau (figure 13b).

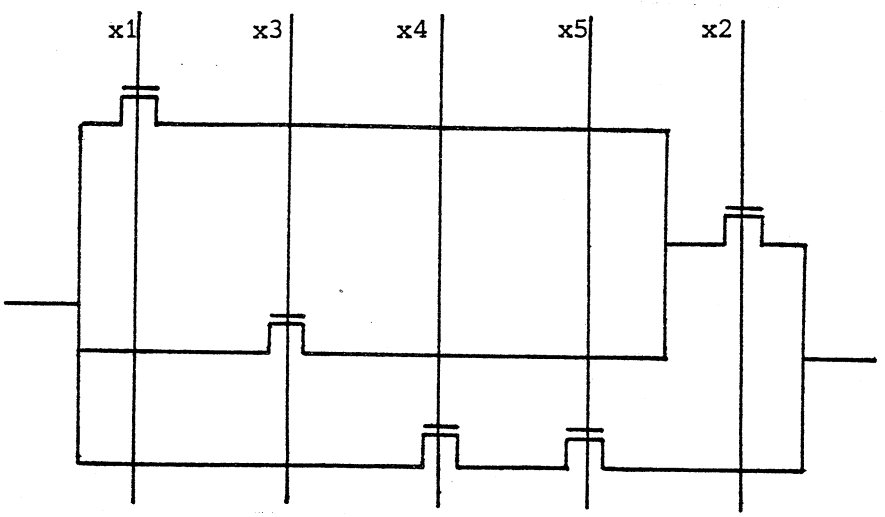


Figure 13b : Réseau obtenu après fusionnement (x2)

2.3.1. Fusionnement de deux transistors

Deux transistors sont dit fusionnables si les trois conditions C1, C2, C3 sont vérifiées.

Condition C1 : La même variable, sous la même forme directe ou complémentée, apparaît sur leurs grilles.

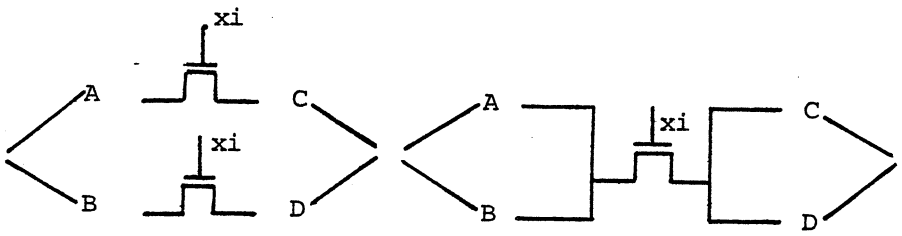
Condition C2 : Leur fusion conserve la propriété de planéarité. Ceci est vérifié si les fusionnements se font sur des transistors figurant sur des lignes adjacentes. Nous chercherons donc à permuter les lignes ou chemins de façon à rendre adjacents les transistors fusionnables.

Condition C3 : La fonction F n'est pas modifiée par l'introduction de chemins parasites.

Nous définissons les propriétés suivantes pour le fusionnement de transistors.

Propriété 1 : condition nécessaire de fusionnement

Considérons deux transistors de même variable apparaissant sur des lignes adjacentes. Soit A et B les fonctions de transfert des sous-réseaux reliant (respectivement) l'entrée du réseau aux deux transistors, C et D les fonctions de transfert des sous-réseaux reliant (respectivement) les deux transistors aux sorties :



Les deux transistors sont fusionnables si :

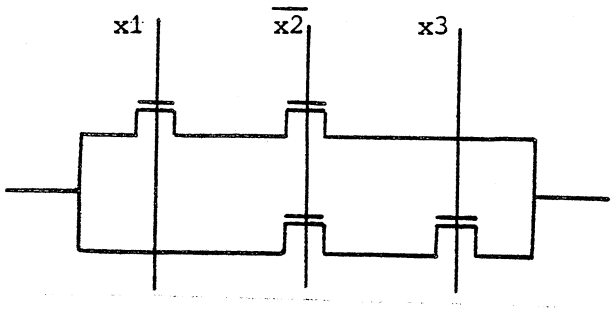
AxiD $\leqslant$ F c'est à dire (AxiD)F = AxiD
et BxiC $\leqslant$ F c'est à dire (BxiC)F = BxiC

En effet, dans ce cas, les chemins "parasites" AxiD et BxiC créent des termes absorbés par la fonction initiale F.

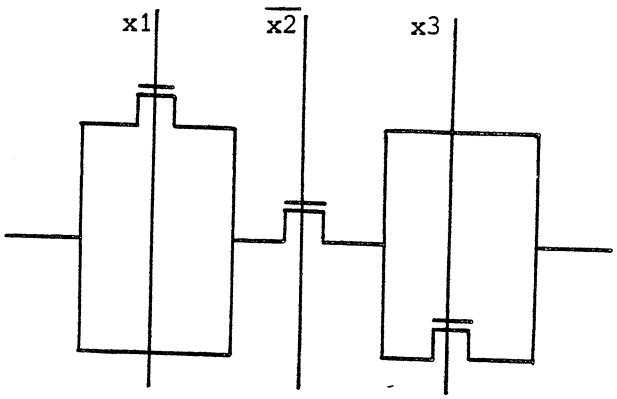
Remarque : La condition nécessaire précédente est remplie dans le cas particulier où (C et D) ou (A et B) sont vides c'est à dire si les transistors xi apparaissent "à l'extrémité" de 2 chemins. Dans ce seul cas de figure, les vérifications de non introduction de chemins parasites sont inutiles. On procède à de simples mises en facteur.

Exemple

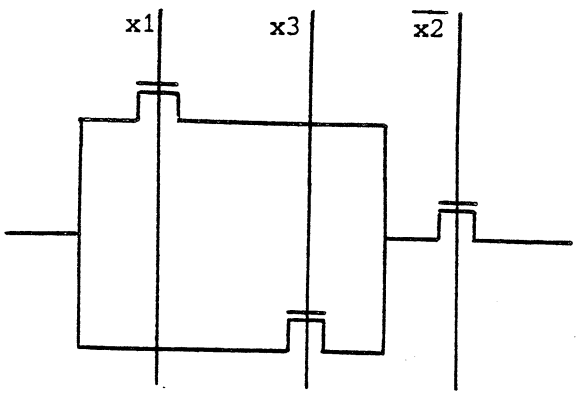
Soit le réseau dont la fonction de transfert est $F = x1\overline{x2} + \overline{x2}x3$



Les transistors correspondants à $\overline{x2}$ ne sont pas fusionnables car il crée le terme $\overline{x2}$ ($x1\overline{x2}x3$ est un terme absorbé)



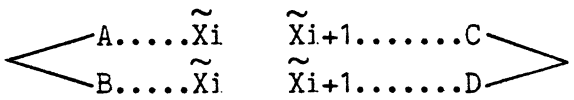
En modifiant l'ordre lexicographique (ordre x1x3x2) les transistors $\overline{x2}$ apparaissant à l'extrémité des chemins deviennent fusionnables.



Extensions

- La condition nécessaire précédente peut être étendue à un ensemble de chemins. La vérification de non introduction de chemins critiques devient complexe. Elle ne se fera en pratique que pour des ensembles de transistors mis aux extrémités.

- La condition nécessaire précédente peut également être étendue à des chaînes de plusieurs transistors apparaissant sous la même forme dans deux ou plusieurs chemins différents.



Il est clair que les conditions de non introduction de chemins critiques sont à vérifier une seule fois. Nous dirons que les fusionnements des couples (xi),(xi+1).... sont des fusionnement compatibles.

Exemple

Soit le réseau de fonction de transfert F = x3x4x5 + x2x4x5 + x1x4

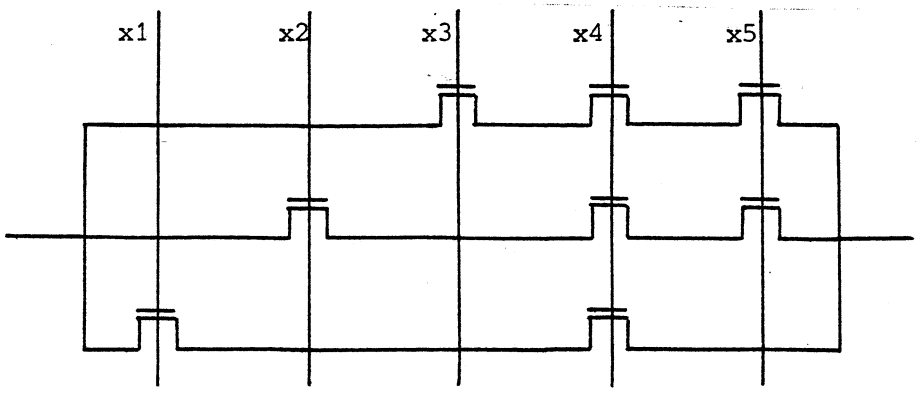


Figure 14 : Réseau réalisant F = x3x4x5 + x2x4x5 + x1x4

La chaine x4x5 est fusionnable dans les premiers chemins sans vérification (C et D sont vides).

Un compromis devra être trouvé entre l'utilisation de ces deux types d'extensions dans les algorithmes pratiques.

Propriété 2

La relation transistor i (ti) fusionnable transistor j (tj) est une relation non transistive.

ti fus tj et tj fus tk n'impliquent pas forcément ti fusionnable avec tk.

Le problème d'optimisation consiste donc à trouver un ordonnancement des colonnes et des lignes permettant de fusionner un nombre maximal de transistors tout en gardant un réseau facilement implantable. Si n est le nombre de variables, l le nombre de lignes initiales, l'examen des solutions exhaustives conduirait à l'examen des n!l! permutations possibles. Ceci correspond pour n = 4 et l = 6 à 17 280 configurations. Il est donc intéressant de définir une heuristique.

2.3.2. Recherche des fusionnements par mise en facteurs à partir d'un réseau de chemins distincts

Il s'agit du cas simple où les fusionnements sont faits à partir d'une extrémité et où la non introduction de chemins parasites est évidente dans une première étape.

On recherche un ordre des colonnes permettant de gagner un nombre maximal de transistors et un ordre des lignes laissant le réseau planaire.

Première étape : passage à une forme arborescente

Pour ceci, on associe à chaque variable xi une partition $\overline{\Pi}$i, des lignes en blocs telle que pour les lignes d'un même bloc cette variable apparaît sous la même forme directe ou complémentée ou n'apparaît pas. e gain en transistors est le nombre d'éléments d'un bloc où la variable apparaît -1. Le gaind'une partition est la somme des gains de chaque bloc.

Partant d'une variable initiale xi, le choix d'une variable "suivante" xj amènera à une partition qui sera l'intersection des partitions $TTi \cap TTj$. es choix successifs des variables amèneront à faire des affinements de partition par intersections successives. On choisira à chaque pas une variable amenant à une partition de gain maximal. a structure finale est un arbre.

Définition 1 : partition initiale associée à une variable

A chaque variable xi est associée une partion initiale TTi des lignes en 3 blocs (Bi1, Bi2, Bi3) telle que :

. Bi1 est l'ensemble des lignes où xi apparaît
. Bi2 est l'ensemble des lignes où $\bar{x}i$ apparaît
. Bi3 est l'ensemble des lignes où cette variable n'apparaît sous aucune forme.

Pour des raisons de concision, on pourra omettre Bi3 qui est directement déduit de Bi1 et Bi2.

Définition 2 : Partition associée à une suite de variables (xi1, xi2, ..., xij)

A une suite de variables ordonnancées, on associe la partition $TTi \cap TTi2.. \cap TTij$. Les blocs de cette partition sont les lignes dans lesquelles la même chaine $\tilde{x}i1\tilde{x}i2...\tilde{x}ij$ apparaît sous la même forme. Cette partition sera notée TTi1...ij et sera associée à la j° colonne.

La partition initiale TTi est un cas particulier où cette chaîne est réduite à une seule variable xi.

Définition 3 : Gain associé à une partition $\tilde{TT}i1...ij$

On appelle gain associé à une partition le nombre de transistors gagnés par fusionnement des lignes appartenant aux mêmes blocs de la partition. Ce gain sera dont égal pour la j° colonne à (nombre d'éléments (transistors fusionnés) du 1er bloc-1) + (nombre des éléments du 2ème bloc-1) +.....

Définition 4 : Ordonnancement optimal des variables

Un ordonnancement des variables est optimal s'il permet un gain maximal (en transistors fusionnés). Un ordonnancement xi1xi2...xin est donc optimal $\Leftrightarrow$ gain ($\widetilde{TT}$i1) + gain ($\widetilde{TT}$i1,2) + ... gain ($\widetilde{TT}$i1,2...n) = gain ($\widetilde{TT}$i1) + gain ($\widetilde{TT}$i1 $\widetilde{TT}$i2) + gain ($\widetilde{TT}$i1 $\widetilde{TT}$i2 + $\widetilde{TT}$i3 + ...) est maximal.

Recherche de l'ordonnancement optimal

Préliminaire

Une partition $\widetilde{TT}$i $\subset$ $\widetilde{TT}$j si chaque bloc de $\widetilde{TT}$i est inclus dans un bloc de $\widetilde{TT}$j, c'est-à-dire pour chaque Bin $\in$ $\widetilde{TT}$i, $\exists$ Bjn tel que Bin $\leq$ Bjn, et $\widetilde{TT}$i $\neq$ $\widetilde{TT}$j.
La recherche exhaustive de l'ordonnancement optimal se fait en comparant le gain des solutions générées de la façon suivante :

Arbre de choix

- Partir de partitions initiales maximales
- Considérer à chaque pas les variables induisant des partitions maximales pour une suite donnée xi1 ... xij, on ne considérera jamais 2 candidats xij+1 et x'ij+1 si $\widetilde{\widetilde{TT}}$i1...ij+1 et $\widetilde{\widetilde{TT}}$'i1...ij+1 sont comparables. Seule la solution où $\widetilde{\widetilde{TT}}$i1...ij+1 est maximale est adoptée.
- Dans l'arbre de choix, les solutions optimales sont celles de gain maximal.

Exemple 1

Soit le réseau initial (figure 15a)

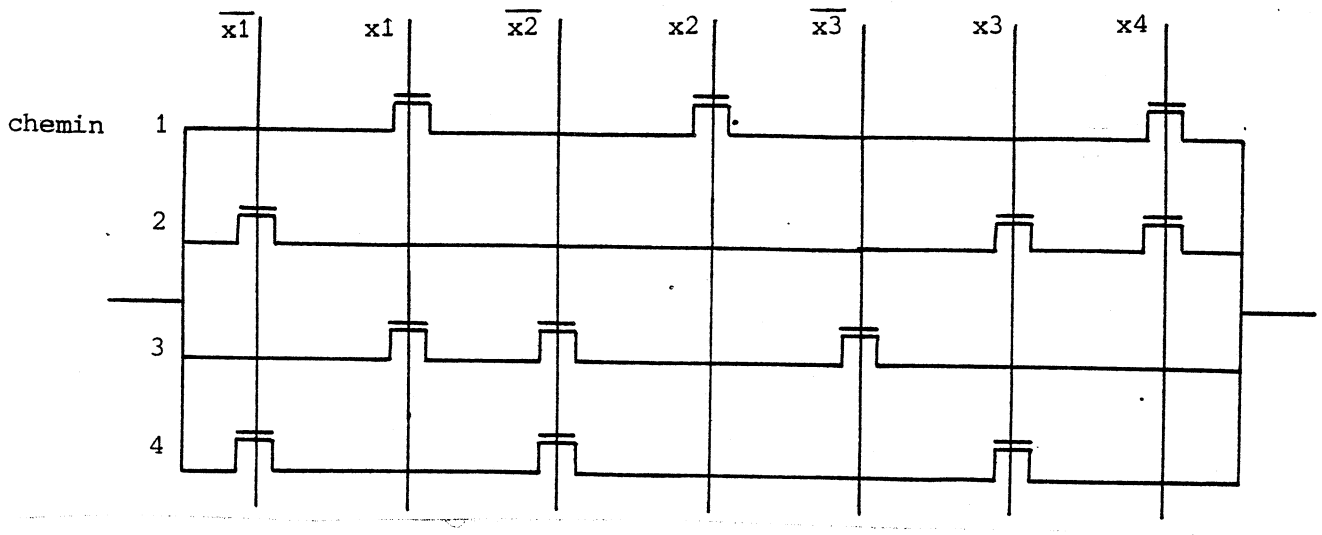


Figure 15a : Réseau de fonction de transfert

$F = x1x2x4 + \overline{x1}x3x4 + x1\overline{x2}x3 + \overline{x1}\overline{x2}x3$

Les partitions initiales sont données sur le tableau suivant :

variable xi	x1	x2	x3	x4
partition $\overline{TT}i$	(13)(24)()	(1)(34)(2)	(24)(3)(1)	(12)()(34)
gain	2	1	1	1

Les partitions initiales à considérer sont $\overline{TT}1$ et $\overline{TT}4$. La solution débutant par $\overline{TT}1$ conduit à un meilleur gain, elle est donnée sur le tableau suivant :

Ordonnancement	x1	x3	x2	x4
partition $\widetilde{TT}1...j$	$TT1$ = (13)(24)	$TT1,3$ = (1)(3)(24)	$TT1,3,2$ = (1)(3)(2)(4)	$TT1,3,2,4$ = (1)(3)(2)(4)
ensemble de fusionnements	(13)(24)	(24)	0	0
gain	2	1	0	0

Le gain final est de trois transistors.

Le réseau de la figure 15b traduit ces partitions successives :

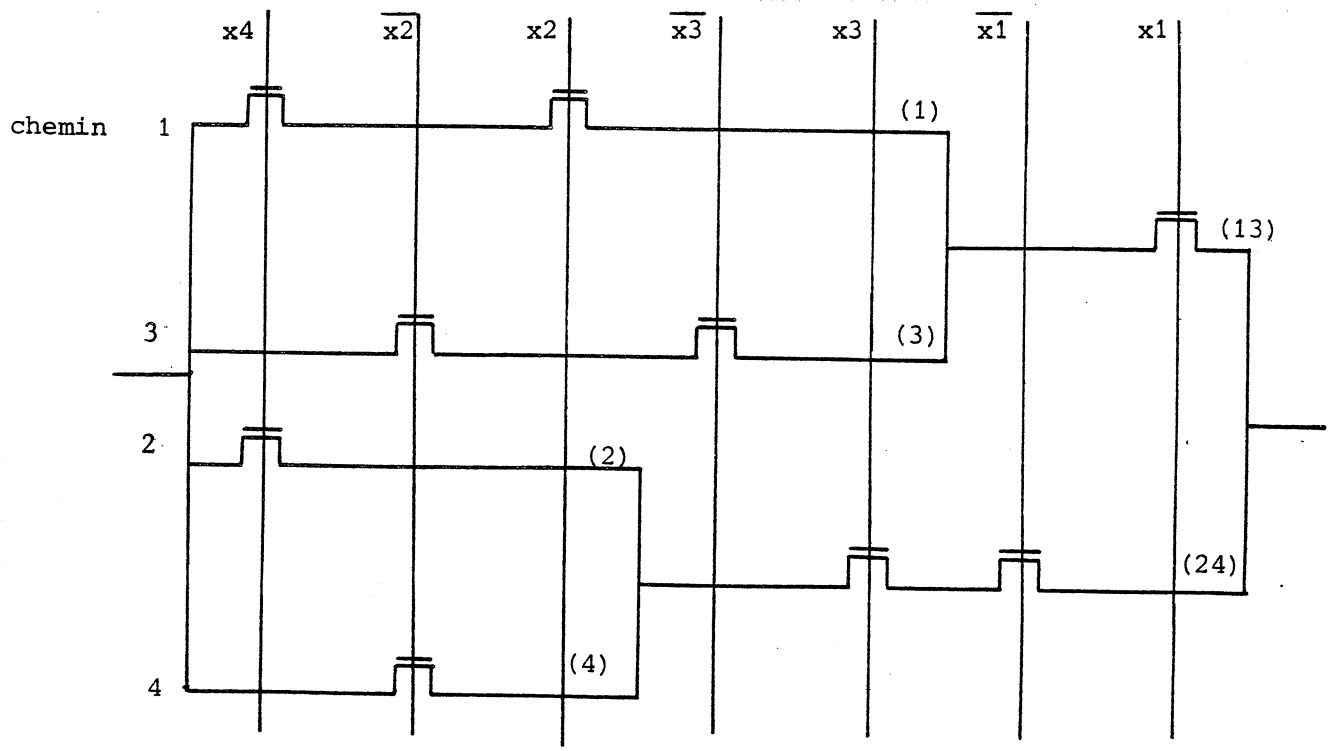


Figure 15b : Réseau obtenu après la 1ère étape

Exemple 2

Soit le réseau initial (figure 16a)

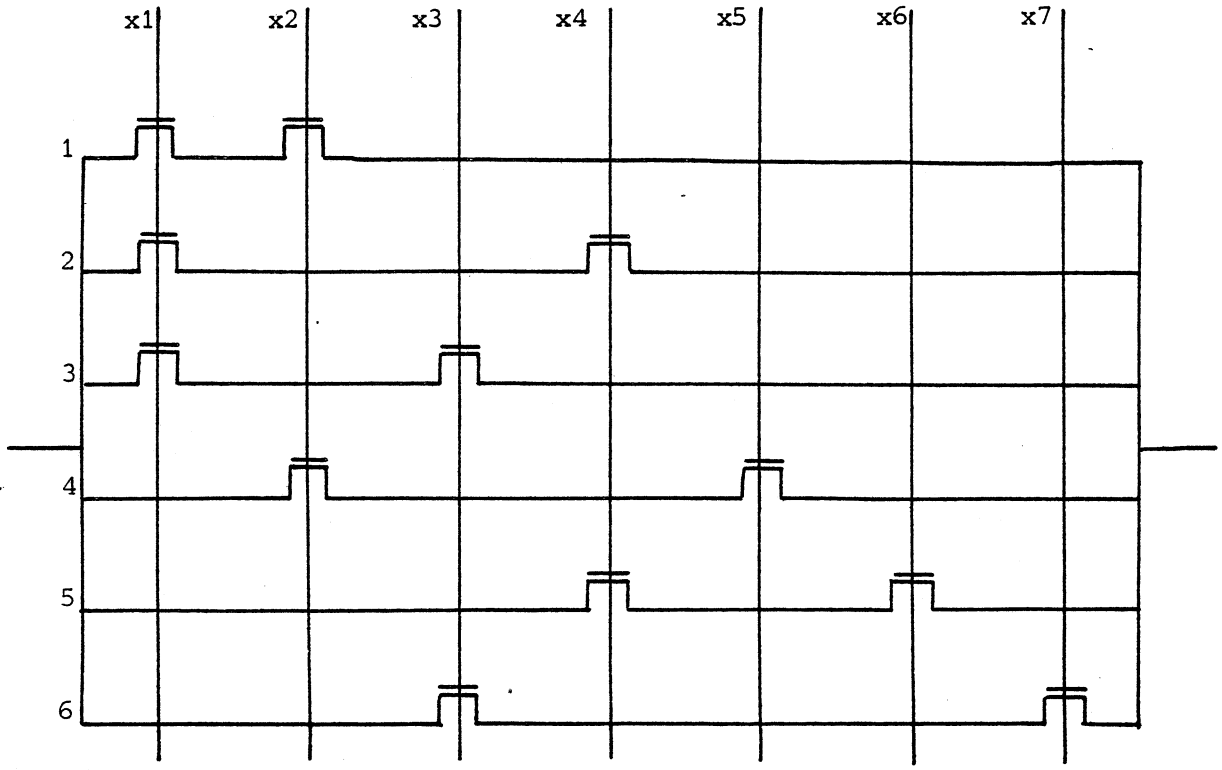


Figure 16a : Réseau initial

xi	x1	X2	X3	x4	x5	x6	x7
TTi	(123)	(14)	(36)	(25)	(4)	(5)	(6)
gain	2	1	1	1	0	0	0

Remarque : Les variables complémentées n'apparaissent pas sur le réseau, on ne note que le bloc où la variable xi apparaît dans chaque partition TTi.

Les partitions initiales à considérer sont $\overline{TT1}$, $\overline{TT2}$, $\overline{TT3}$, $\overline{TT4}$. Le choix des trois premières partitions ($\overline{TT2}$,$\overline{TT3}$,$\overline{TT4}$) quel que soit leur ordre conduira à la solution optimale.

Ordonnancement	x2	x4	x3	x5	x6	x7	x1
ensemble de fusionnements	(14)	(25)	(36)	0	0	0	0
gain	1	1	1	0	0	0	0

Le gain final est de trois transistors.

Le réseau de la figure 16b traduit ces partitions successives :

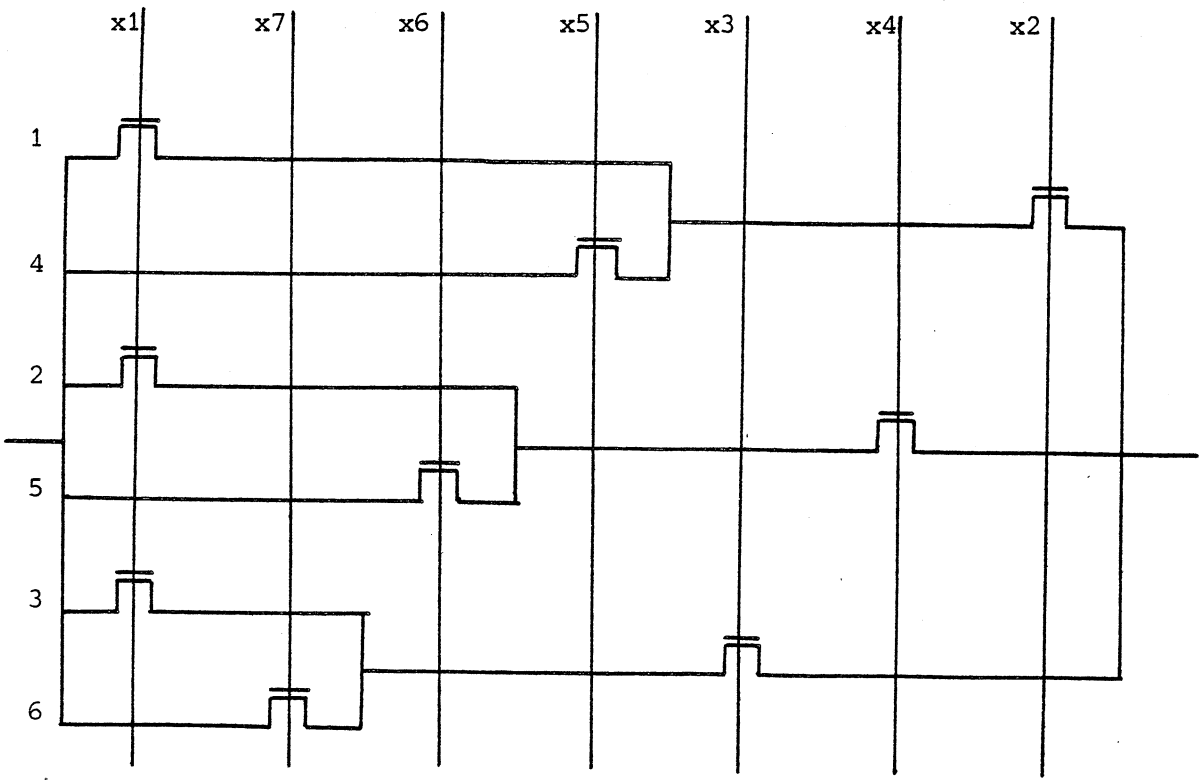


Figure 16b : Réseau obtenu après la 1ère étape

Deuxième étape : réduction de l'arbre à partir de ces feuilles

On recherche le fusionnement de points origine de sous-arborescences identiques dans l'arborescence précédente, à condition de laisser le réseau planaire et de ne pas introduire de chemins parasites.

Exemple 1

On peut effectuer un fusionnement sur les transistors commandés par la variable x4. On obtient le réseau de la figure 17 :

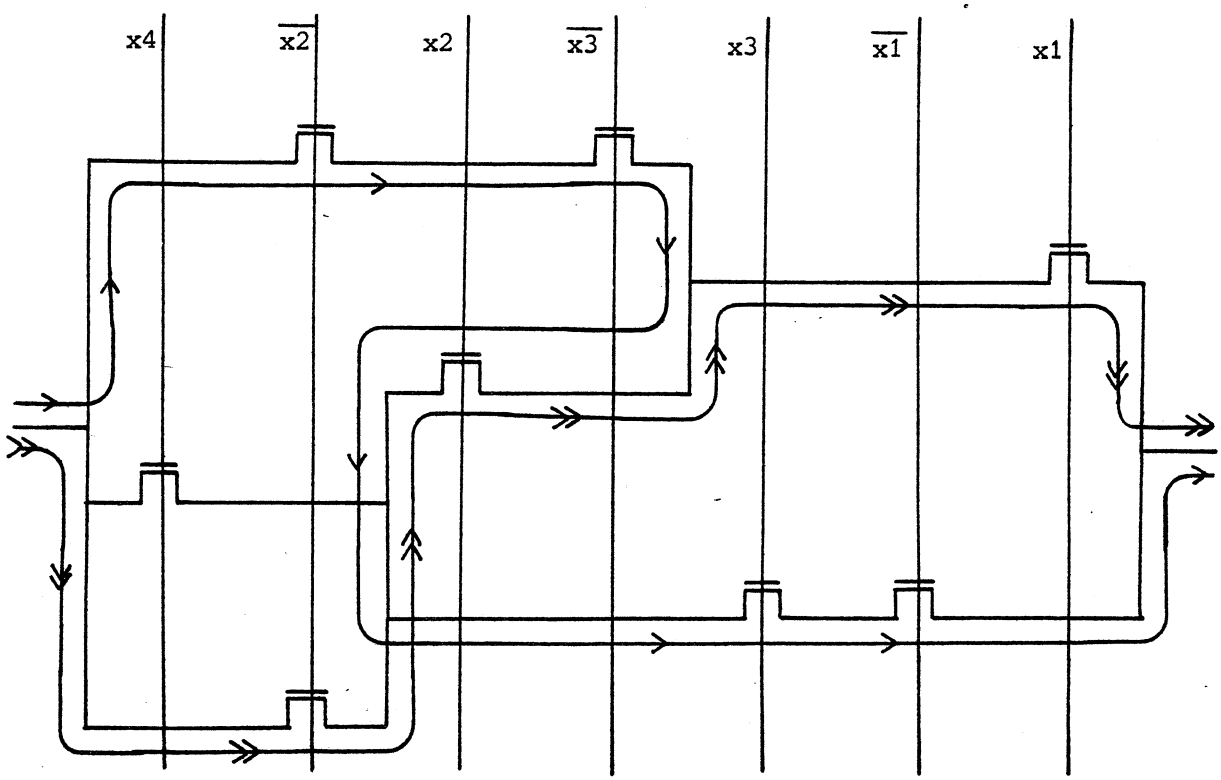


Figure 17 : Réseau final

Les nouveaux chemins indiqués sur le schéma sont nuls (présence de x2 et $\overline{x2}$ sur ces chemins).

L'étude des chemins parasites se fait de la façon suivante :

Soient A et B, deux points candidats au fusionnement reliés à une des extrémités du réseau à des sous-réseaux identiques (figure 18).

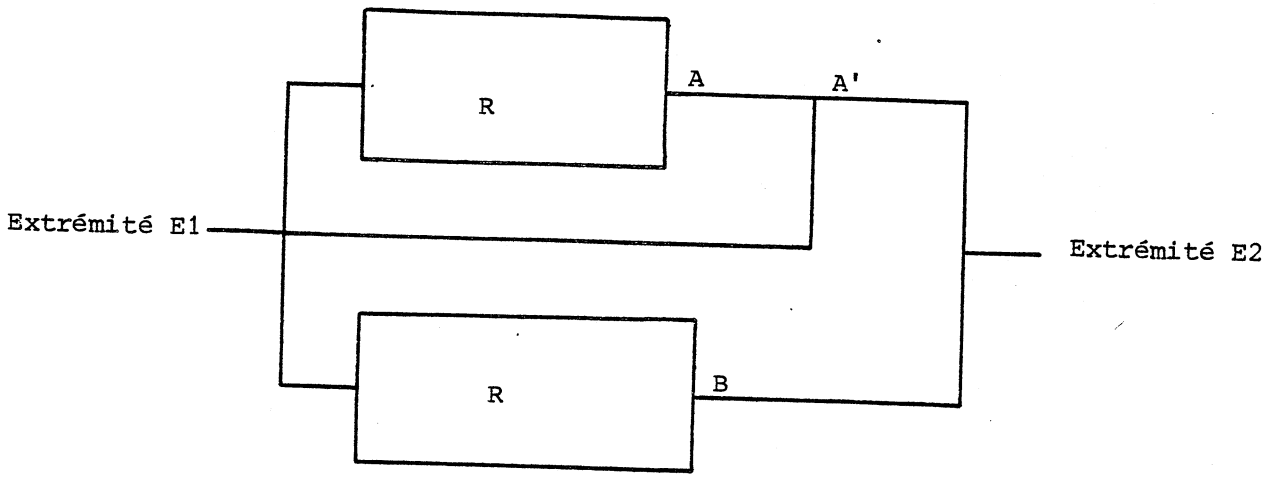


Figure 18 : Sous réseaux R candidats au fusionnement

Un chemin (E1AE2) correspondant à un monôme de la fonction de transfert du réseau peut se décomposer en une séquence de 2 chemins appelés chemins partiels (E1A) et (AE2). Un chemin parasite (E1A) est un chemin joignant E1 et A, qui n'est pas un chemin partiel du chemin (E1AE2) de la fonction de transfert, le chemin (E1A'A) est un chemin parasite.

Le fusionnement des points A et B crée les nouveaux chemins entre E1 et E2 définis comme la composition série de :

chemin parasite (E1A'A) et chemin partiel (BE2)
chemin parasite (E1B'B) et chemin partiel (AE2)

Si un des chemins modifie la fonction de transfert, le fusionnement n'est pas possible.

Exemple 1

Soit les réseaux de la figure 15b obtenu après la première étape. En premier lieu, les deux transistors commandés par $x4$ ou $\overline{x2}$ peuvent être fusionnés.

Si R = x4, les chemins créés sont :

(E1A'A) : $\bar{x}2\ \bar{x}3\ x2$ et (BE2) : $x3\ \bar{x}1$
(E1B'B) : $\bar{x}2$ et (AE2) : $x2\ x1$

Les deux monômes correspondants aux chemins créés sont $\bar{x}2\bar{x}3x1x3\bar{x}1$ et $\bar{x}2x2x1$, ils sont tous les deux égaux à zéro.

Si N = $\bar{x}2$, les deux monômes correspondants aux chemins créés sont $x4\bar{x}3x1$ qui modifie la fonction de transfert et $x4x2\bar{x}3x3x1$ égal à zéro.

Le réseau final sera donc celui de la figure 17

Exemple 2

Soit le réseau de la figure 16b obtenu après la première étape. On peut effectuer des fusionnements sur les transistors commandés par la variables x1, ces fusionnements se feront deux à deux à cause de l'adjacence des chemins.

Les monômes créés par le fusionnement de x1 sur les chemins 1 et 2 sont : (E1A'A)(BE2) = x5 x4 et (E1B'B)(AE2) = x6 x2 ; ces monômes modifient la fonction de transfert.

Pour le fusionnement sur les chemins 1 et 3, on aura les monômes créés : x5x3 et x7x2 qui modifient la fonction de transfert.

Les monômes créés : x6 x3 et x7 x4 pour le fusionnement sur les chemins 2 et 3 modifient aussi la fonction de transfert.

On n'effectuera donc pas de fusionnement au cours de la deuxième étape. Le réseau final est donc celui de la figure 16b.

II - 4. Optimisation d'un réseau réalisant un produit de monaux

On considère d'abord les cas simples où des solutions lexicographiques existent sans pliage.

2.4.1. Cas simples sans pliage

Cas 1

Il existe une partition des monaux en ensembles de monaux tels que les variables apparaissant dans les différents ensembles sont disjointes. Une optimisation se fait alors à l'intérieur de chaque ensemble de monaux sur les variables apparaissant sous la même forme dans les différents monaux de l'ensemble.

Exemple

$$F = (a + \bar{b} + c)(a + \bar{b} + d)(e + f)(e + \bar{g})(e + h)$$

On obtient le réseau de la figure 19, réalisant la fonction minimisée :

$$F = (a + \bar{b} + cd)(e + f\bar{g}h)$$

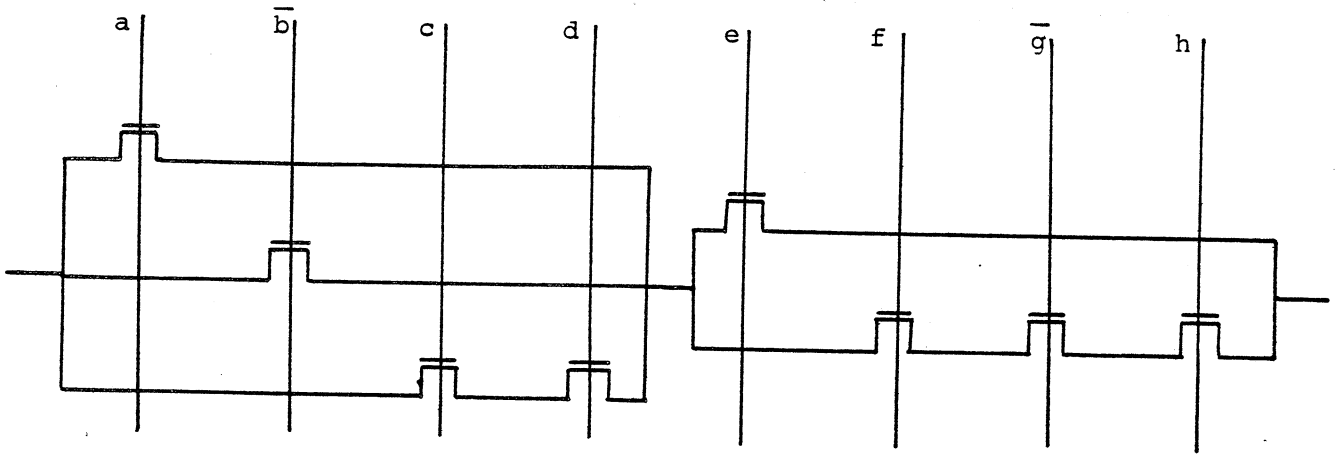


Figure 19 : Réseau réalisant $F = (a + \bar{b} + cd)(e + f\bar{g}h)$

Cas 2 : extension au cas 1

Il existe une partition des monaux en ensembles ordonnés de monaux. Chaque ensemble peut avoir au plus, une variable commune avec chacun de ses voisins. Cette variable est distincte à droite et à gauche. Deux des ensembles n'ont qu'une variable en commun avec les autres ensembles, ils se situeront aux deux extrémités du réseau.

Exemple

$$F = (a + \bar{b} + c)(a + \bar{b} + \bar{d})(d + e)(d + f)(d + \bar{g} + h)$$

On obtient le réseau de la figure 20, réalisant la fonction minimisée :

$$F = (a + \bar{b} + c\bar{d})(d + ef(\bar{g} + h))$$

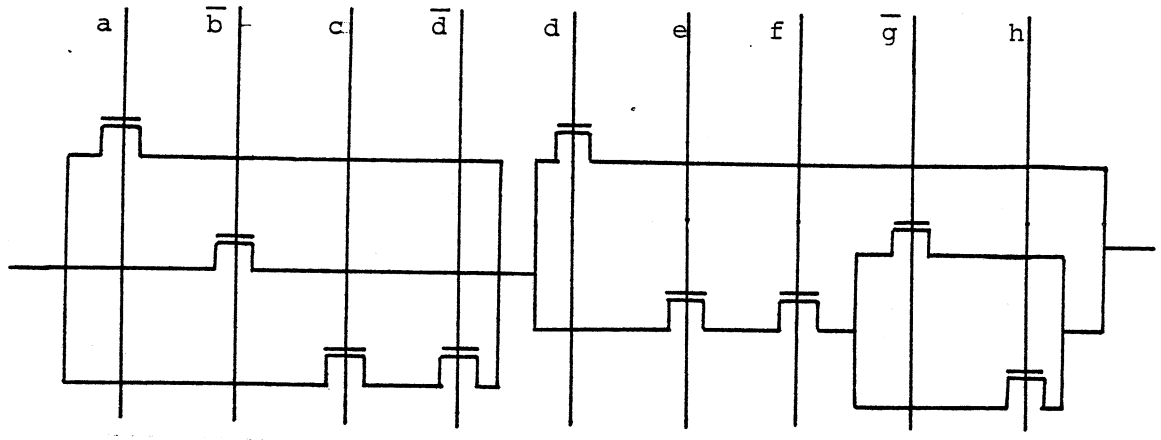
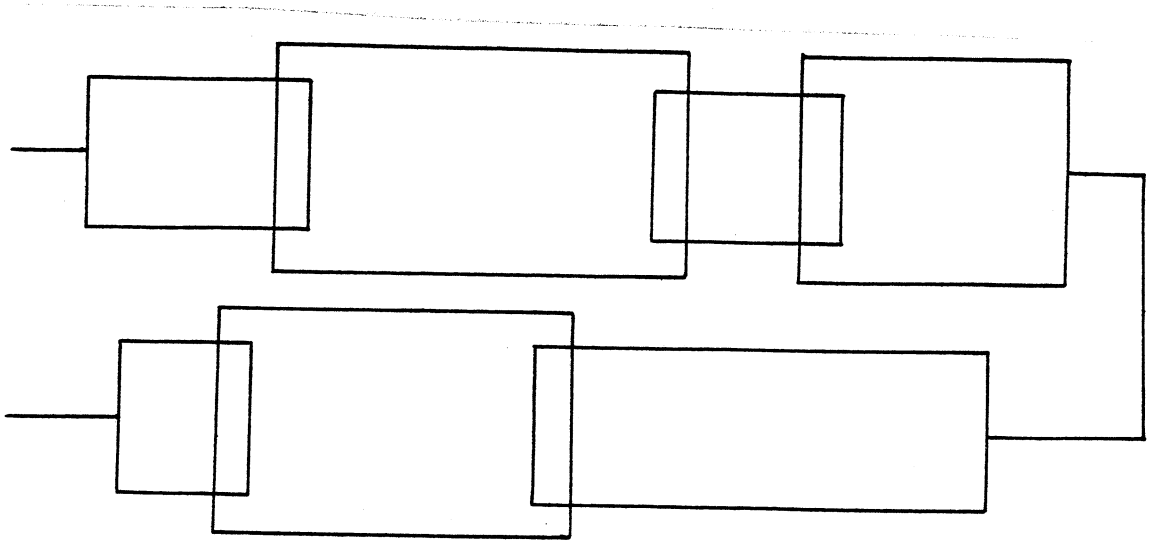


Figure 20 : Réseau réalisant $F = (a + \bar{b} + c\bar{d})(d + ef(\bar{g} + h))$

2.4.2. Cas général

Dans le cas général, il s'agira essentiellement de minimiser les pliages. Pour cela on recherche des suites maximales de monaux remplissant les conditions précédentes. Les pliages seront réalisés "entre" ces suites :



Exemple

Prenons le cas simple de la porte

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

Une suite de trois monaux peut être extraite (a+b)(b+c)(c+d), le quatrième monal introduit un pliage

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

recouvrement pliage

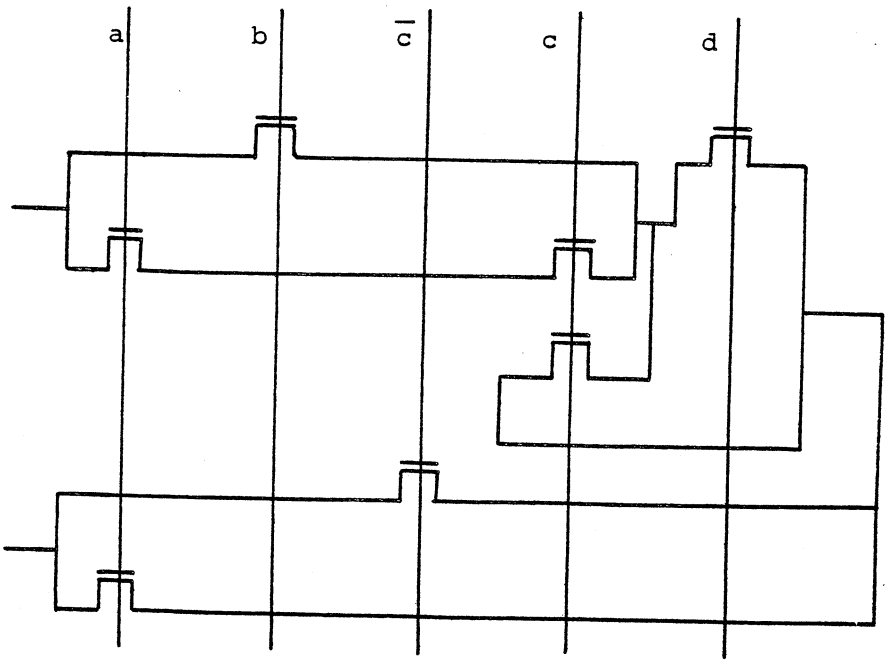


Figure 21 : Réseau réalisant $F = (b + ac)(c + d)(a + \bar{c})$

III - PORTES COMPLEXES N MOS

III - 1. Généralités

La porte logique NMOS est définie par le schéma suivant (figure 22) :

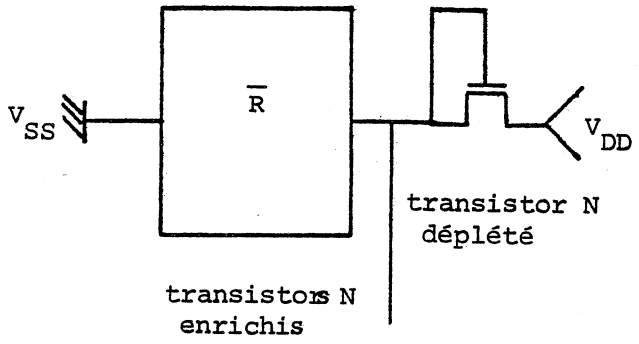


Figure 22 : Porte NMOS

La fonction F est réalisée en reliant la sortie s à l'alimentation VDD par un transistor de charge déplété et à la masse VSS par le réseau $\overline{R}$ dont la fonction de transfert est $\overline{F}$. Si tous les chemins de R sont coupés, la sortie sera reliée à VDD par l'intermédiaire du transistor de charge. Si un des chemins de $\overline{R}$ est fermé, la sortie sera reliée à la masse.

Porte simple et porte complexe

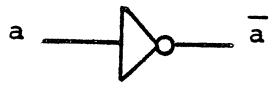
On appelle, en général, porte simple, une porte où le réseau R est constitué par un seul transistor, 2 transistors en parallèle ou 2 transistors en série. Les portes simples réalisent en conséquence les fonctions classiques :

$F1 = \overline{x}$: inverseur
$F2 = \overline{x + y}$: porte NOR
$F3 = \overline{xy}$: porte NAND

Dans les autres portes dites portes complexes, le nombre de transistors mis en série dans $\overline{R}$ sera limité à 4 ou 5, sinon la taille des transistors devient trop importante. En effet, pour garder un niveau logique "0" correct en sortie de la

porte, les résistances équivalentes des transistors mis en série doivent diminuer, (les résistances en série s'ajoutent) ce qui correspond à une augmentation de surface des transistors (§ III.3 Dimensionnement). Le nombre maximal de transistors mis en parallèle dans $\overline{R}$ sera de 10 environ pour la rapidité de la porte (les capacités en parallèle s'ajoutent).

Portes simples : Schémas logiques



Schémas électriques

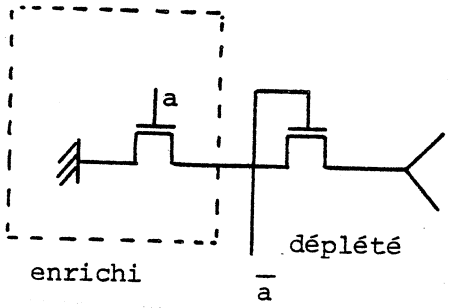


Figure 23a : Inverseur

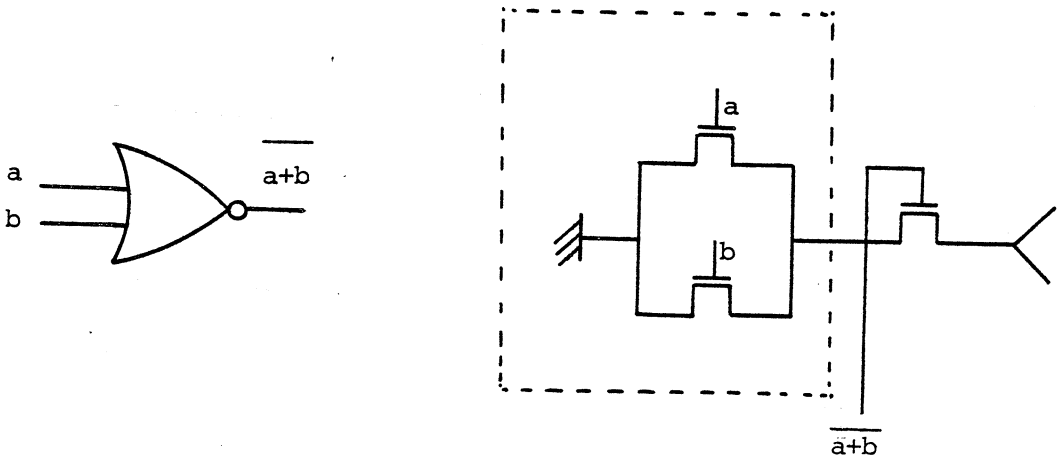


Figure 23b : Porte NOR à 2 entrées

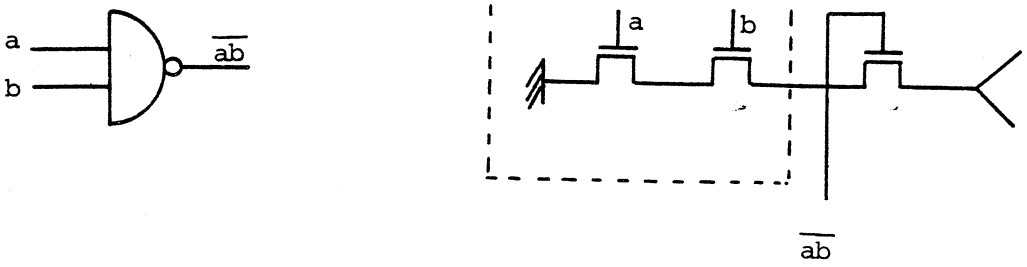


Figure 23c : Porte NAND à 2 entrées

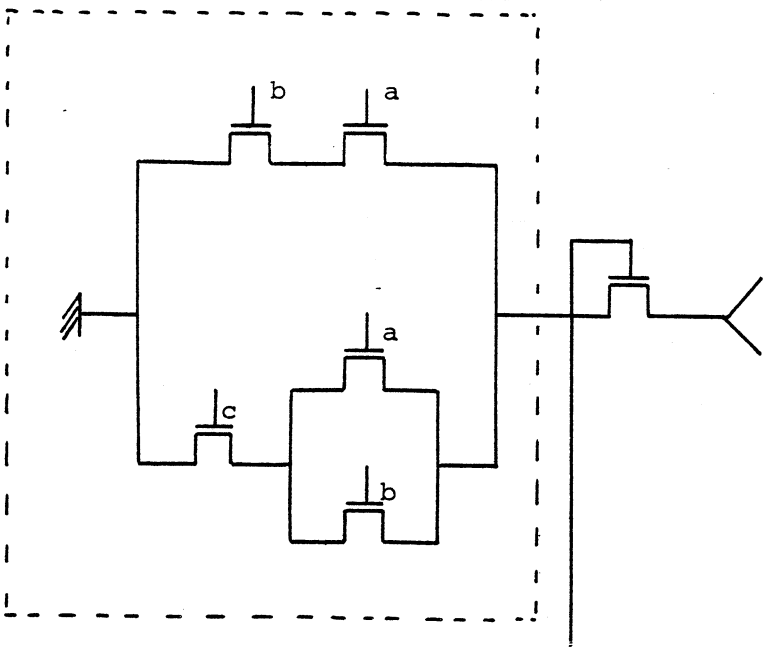
Le réseau $\overline{R}$ réalise la fonction de transfert $\overline{F}$ complémentaire de F :

$\overline{F}$	fonction réalisée (F)
a	inverseur $F = \overline{a}$
a + b	NOR $F = \overline{a + b}$
ab	NAND $F = \overline{ab}$

Portes complexes

Exemple

Soit la fonction $F = \overline{ab + (a + b)c}$



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

Figure 24 : Porte complexe

Le réseau $\overline{R}$ ci-dessus réalise la fonction $\overline{F} = ab + (a + b)c$

Les portes complexes NMOS courrament utilisées sont données en annexe 1.

On s'intéressera ici aux portes complexes NMOS dont le réseau $\overline{R}$ est un réseau optimisé, une telle porte sera appelée porte NMOS optimisée. Pour une porte complexe réalisant la fonction F, on partira donc de la fonction $\overline{F}$. Le réseau $\overline{R}$ réalisant $\overline{F}$ sera optimisé. On étudiera :

- l'implantation d'une porte et d'un ensemble de portes
- le dimensionnement.

Enfin, le problème de la précharge est abordé.

III - 2. Implantation d'une porte complexe NMOS dont le réseau $\overline{R}$ est lexicographique

On va d'abord passer du schéma électrique au schéma topologique de l'inverseur pour définir la technologie utilisée et les différents matériaux. Puis on proposera trois types d'implantation pour une porte complexe NMOS.

On utilise une technologie NMOS à déplétion en charge (un niveau de Silicium polycristallin, pré-contact pour les transistors de charge). On définit les matériaux employés de la façon suivante (figure 25a et 25b) (MEA 80) :

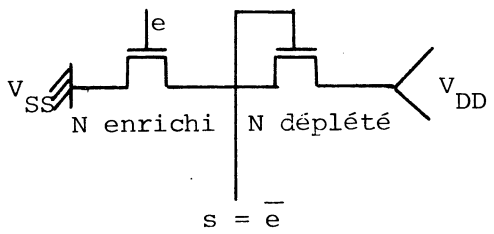


Figure 25a : Schéma électrique de l'inverseur

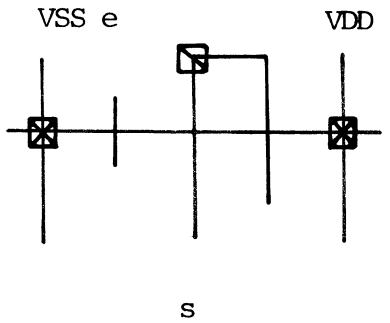


Figure 25b : Matériaux utilisés pour l'inverseur

On définit les couleurs standards (MEA 80) :

—— aluminium
—— silicium polycristallin
—— diffusion N
: : implantation ionique

et les contacts :

⊠ contact alu-autre matériau
⊠ pré-contact Silicium polycristallin-diffusion.

On a (figures 25a et 25b) une ligne de diffusion N entre les deux points de contacts avec les alimentations VDD et VSS. Le croisement du Silicium polycristallin avec la diffusion crée un transistor (transistor signal ou transistor de charge). Le transistor de charge déplété nécessite une implantation ionique, le contact entre le Silicium polycristallin et la diffusion définit le pré-contact.

On désire implanter une porte complexe NMOS optimisée. L'organisation topologique pour une porte sera la suivante (figure 26) :

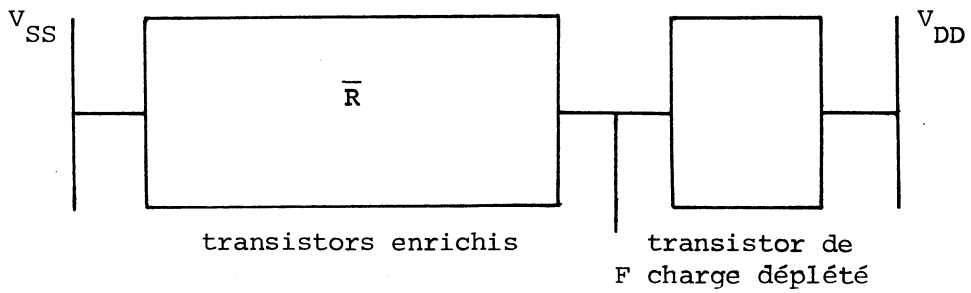


Figure 26 : Organisation topologique pour une porte

On part du réseau $\overline{R}$ lexicographique optimisée. Le réseau est un quadrillage : les variables d'entrée sur les lignes verticales, les chemins horizontaux. On propose trois types d'implantation qui seront traités sur le réseau optimisé précédemment (schéma électrique de la figure 17).

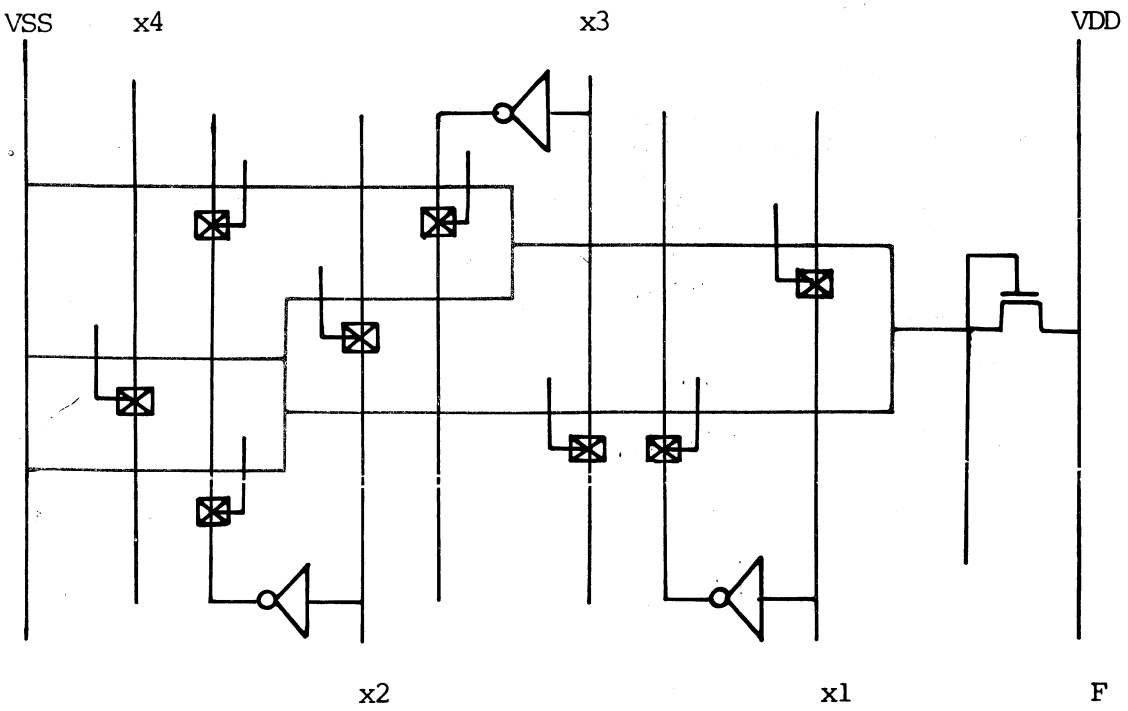


Figure 27a : 1er type d'implantation d'une porte NMOS optimisée

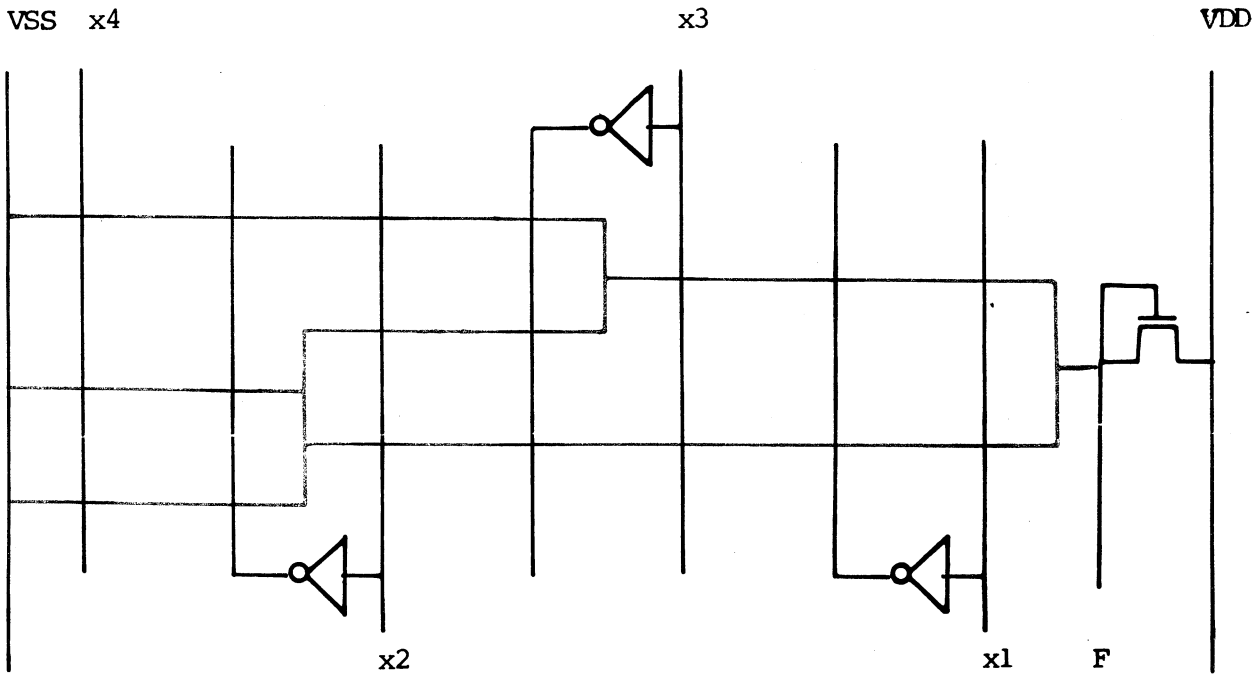


Figure 27b : 2ème type d'implantation

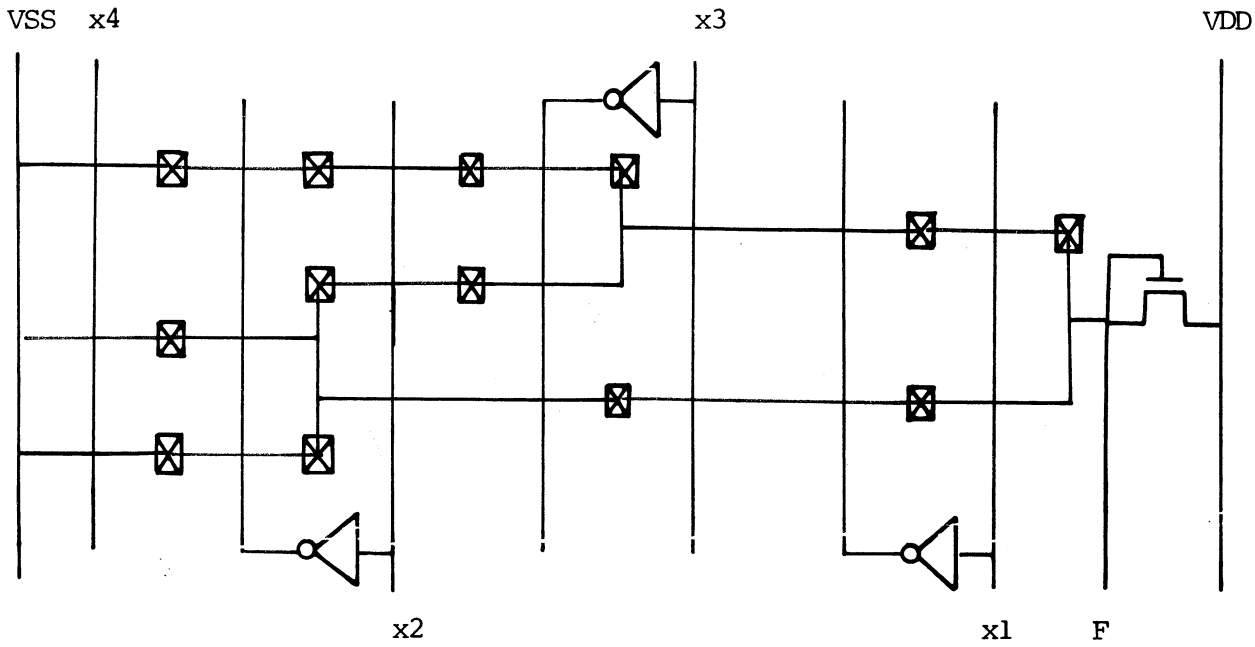


Figure 27c : 3ème type d'implantation

Remarque importante :

Sur ces schémas implantés, on a considéré des transistors de taille identique, ils ne sont pas dimensionnés. En fait, la taille des transistors d'une porte complexe peut être importante (cf. III.3 dimensionnement). L'implantation d'une porte complexe NMOS ne se fera donc pas de façon automatique. Suivant la taille des transistors, la place dont on dispose, les portes peuvent être implantées différemment : on peut tordre les lignes de diffusion, imbriquer les portes. Les lignes de variables traversant le réseau peuvent être coupées, on peut rentrer des variables supplémentaires.
Dans ces trois types d'implantation, les lignes horizontales sont en diffusion N (chemins).

Pour le 1er type, les lignes verticales sont en aluminium (variables d'entrée). S'il existe un transistor au croisement de deux lignes, le silicium polycristallin en contact avec l'aluminium (variable d'entrée) viendra au-dessus de la diffusion (figure 27a). Une telle solution est donnée pour les fonctions symétriques, réalisées en logique de transfert de la page 79 du (MEA 80), planche 7b, elle est différente car dans ce cas tous les transistors sont de taille identique.

Pour les 2ème type et 3ème type, les lignes verticales sont en silicium polycristallin (variables d'entrée). On a donc un transistor à chaque croisement silicium polycristallin-diffusion. S'il n'existe pas de transistor on peut :

- soit fabriquer un transistor déplété (implantation ionique), ce transistor est toujours passant (commande "0" ou "1" sur sa grille), le chemin en diffusion est toujours connecté (figure 27b). Une telle solution est donnée pour la forme canonique, réalisée en logique de transfert de la page 77 du (MEA 80) planche 7a, elle est différente car dans ce cas tous les transistors sont de taille identique.

- soit court-circuiter celui-ci avec de l'aluminium (figure 27c)

Suivant la nature du matérieu souhaité pour les variables, on choisira un schéma implanté parmi ces trois schémas proposés.

Le 2ème type est intéressant pour le passage de connexions au-dessus du réseau, effectivement on n'a pas utilisé le niveau aluminium. Mais les transitors MOS déplétés introduisent des résistances sur les chemins : la rapidité sera moins grande.

Dans le 3ème type, on a le niveau aluminium à rajouter pour court-circuiter les transistors (dernier masque).

III - 3. Dimensionnement d'une porte complexe NMOS

On calcule d'abord les dimensions de l'inverseur répondant aux objectifs de fonctionnement, puis on en déduira les dimensions de la porte complexe équivalente.

Les modes de fonctionnement, caractéristiques et équations du courant du transistor MOS canal N sont données dans l'annexe 3.

On note les dimensions des transistors MOS comme suit (MEA 80) :

- Ws, largeur de la grille du transistor signal
- LS, longueur de la grille du transistor signal

- Wc, largeur de la grille du transistor de charge
- LC, longueur de la grille du transistor de charge.

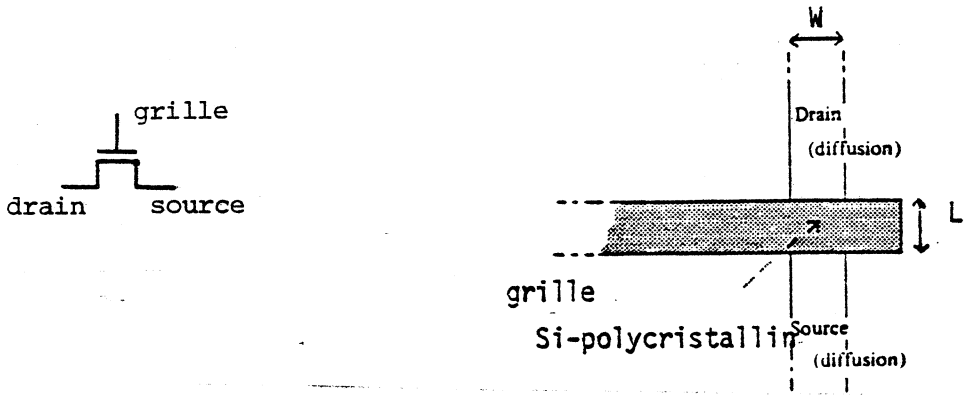


Figure 28a : Transistor

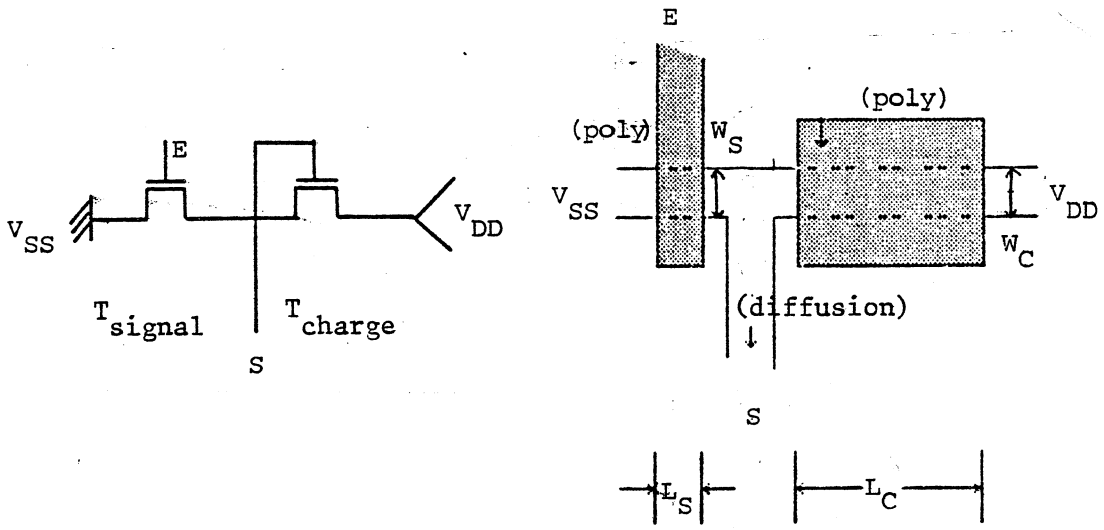


Figure 28b : Inverseur

On cherche à déterminer les quatre paramètres de dimensions d'un inverseur, à savoir :

- WS, LS pour le transistor signal
- WC, LC pour le transistor de charge

de façon à atteindre les objectifs de fonctionnement fixés.

II n'est pas nécessaire de calculer ces quatre paramètres, il suffit d'évaluer s, c, r définis par :

- $\gamma s = Ws/Ls$, rapport du transistor signal

- $\gamma c = Wc/Lc$, rapport du transistor de charge

- $\gamma r = s/c$, facteur de forme de l'inverseur

On présente ici une étude simplifiée du calcul des dimensions. On trouvera dans (MAL 82) le calcul des dimensions d'un inverseur et sa généralisation à tous les types de portes logiques : en particulier, pour les portes complexes, les rapports des transistors γc et γs sont calculés précisément.

(i) Dimensions du transistor de charge

Soit tm le temps de montée désiré pour cet inverseur. Le passage du niveau logique "0" au niveau "1" en sortie de l'inverseur se fait en chargeant la capacité de la sortie C. On considère le schéma suivant (figure 29) pendant la charge de la capacité.

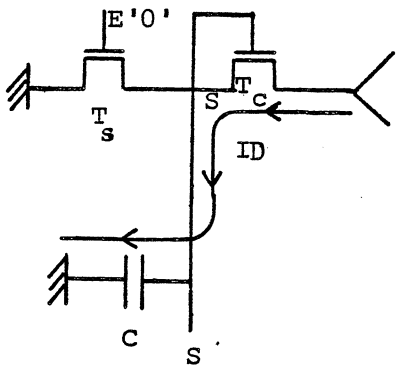


Figure 29 : Charge de la capacité C en sortie d'un inverseur

C est la capacité équivalente en sortie de l'inverseur.

Le transistor signal Ts est bloqué : en effet, sa tension de seuil Vseuil est positive et la tension appliquée E = "0" = VGS.

La charge de la capacité C se fait par l'intermédiaire du transistor de charge Tc, déplété, donc passant : sa tension de seuil Vseuil est négative et sa grille est reliée à son drain, VGS = 0

On a :

$$VDS = VGS - Vseuilc = - Vseuilc$$
$$VDS = VDD - S$$

Le transistor fonctionne dans le mode saturé si :

$$VDS > VDS\ sat \Rightarrow VDD - S > - V\ seuilc$$

c'est à dire : $S < VDD + Vseuilc$

soit $Sat = VDD + Vseuilc$

Pour $0<S<Ssat$, le transistor fonctionne dans le mode saturé :

$$IDsat = 1/2\ \bar{u}n\ Cox\ \gamma c\ (VGS - Vseuilc)^2$$
$$IDsat = 1/2\ \bar{u}n\ Cox\ \gamma c\ (Vseuilc)^2 = \text{constante } X\ \gamma c$$

Pour $S>S\ sat$, le transistor fonctionne dans le mode conducteur :

$$IDcond = \bar{u}n\ Cox\ \gamma c\ ((VGS - Vseuilc)\ VDS - VDS^2/2))$$
$$IDcond = \bar{u}n\ Cox\ \gamma c\ (-Vseuilc\ VDS - VDS^2/2)$$
$$IDcond = IDsat\ (-2VDS/Vseuilc - VDS^2/Vseuil^2)$$

On calcule le temps de montée de la façon suivante :

$$tm = C \int_{0,1VDD}^{0,9VDD} dS/ID$$

$$tm = C \left(\int_{0,1VDD}^{Ssat} dS/IDsat + \int_{Ssat}^{0,9\,VDD} dS/IDsat \right)$$

Après calcul, on trouve :

$$\int_{0,1VDD}^{Ssat} dS/IDsat = 1/IDsat\ (Ssat - 0,1VDD)$$

$$\int_{Ssat}^{0,9VDD} dS/IDcond = (Vseuilc/2IDsat)\ (Log\ VDS/(VDS + 2Vseuilc))_{VDSsat}^{0,1VDD}$$

$$= (Vseuilc/2IDsat)\ Log - 0,1VDD/(0,1VDD + 2\ Vseuilc)$$

d'où

$$tm = (Ssat - 0,1VDD + (Vseuilc/2)\ log\ 0,1VDD/(0,1VDD + 2Vseuilc))\ C/IDsat$$

$$tm = Constante\ x\ C/\gamma c$$

A partir du temps de montée tm donné et de la capacité C calculée, on trouve le rapport du transistor de charge

$$\gamma c = Wc/Lc$$

La valeur de la capacité de la sortie dépend de trois facteurs :

$$C = CDS + Cl + CGS$$

CDS, capacité du transistor signal
Cl, capacité des lignes de connexions
CGS, capacité des grilles à relier avec la sortie.

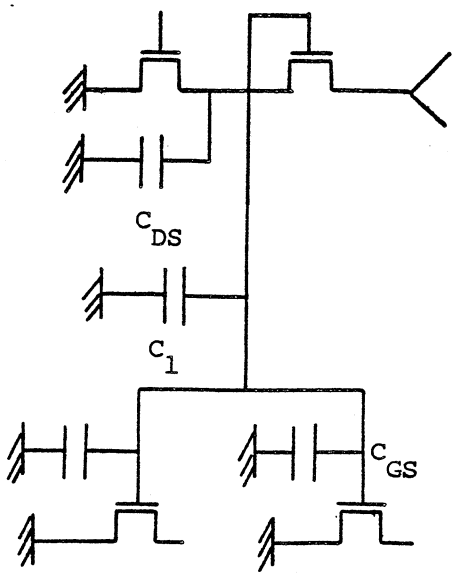


Figure 30 : Capacité de la sortie

(ii) Dimensions du transistor signal

On désire un niveau logique "0" correct en sortie de l'inverseur. On considère le schéma suivant (figure 31) quand l'entrée du transitor signal est au niveau "1", la sortie est au niveau "0".

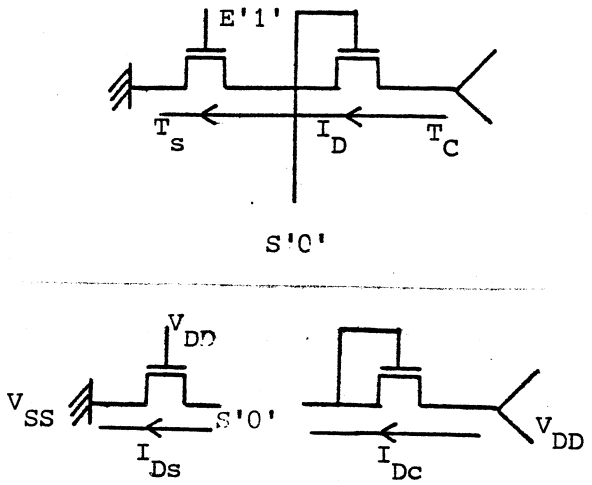


Figure 31 : Inverseur avec E = "1"

Pour avoir un niveau logique "0" correct, on aura la résistance équivalente du transistor de charge bien supérieure à celle du transistor signal, c'est pourquoi on ne s'intéresse pas au temps de descente négligeable devant le temps de montée.

Le niveau "0" correct en sortie correspond à : S"0" << Vseuil. Soit

S "0" = Vseuils/k avec k grand

Le principe du calcul consiste à égaler les courants : IDc = IDs, on en déduit le facteur de forme de l'inverseur : γ r.

Le transistor de charge Tc est déplété donc passant : sa tension de seuil Vseuilc est négative et sa grille est reliée à son drain, VGS = 0

On a :

VDSsat = VGS - Vseuilc = - Vseuilc

VDS = VDD - S"0" = VDD - Vseuils/k

d'où : VDS > VDS sat

Le transistor de charge fonctionne dans le mode saturé :

$$IDc = 1/2\ \bar{u}n\ Cox\gamma c\ (Vseuilc)^2$$

Le transistor signal Ts est passant, la tension VGS appliquée est égale à VDD, donc supérieure à Vseuils

On a :

VDSsat = VGS - Vseuils = VDD - Vseuils.

VDS = S"0" - VSS = S "0" = Vseuils/k

d'où : VDS << VDSsat

Ce transistor fonctionne dans le mode conducteur, dans la région ohmique :

$$IDs = \bar{u}n\ Cox\gamma s(VGS - Vseuils)\ VDS$$
$$IDs = \bar{u}n\ Cox\gamma s\ (VDD - Vseuils)\ S"0"$$

L'égalité des courants IDs = IDc donne le facteur de forme de l'inverseur $\gamma r = \gamma s/\gamma c$:

$$\gamma r = Vseuilc/2S'0'(VDD-Vseuils))$$

$$\gamma r = constante/S"0"$$

Le rapport du transistor de charge γc = Wc/Lc a été calculé précédemment en (i), on peut donc calculer le rapport du transistor signal :

$$\gamma s = \gamma r \times \gamma c$$

(iii) Application numérique

On fait les calculs avec les paramètres électriques de CMP pour la technologie NMOS (CMP 82)

VDD = 5 V

Vseuilc =-1,4 V

Vseuils = 0,6 V

$\overline{un}$ = 800 x 10^{-4} m^2/Vs

Cox = 5,5 x 10^{-4} pF/μ^2

S"0" = 0,1 V

On calcule le rapport du transistor de charge γ c à partir de l'expression tm=constante X C/γc

On a :

Ssat = VDD + Vseuilc = 3,6 V

VDSsat = VDD - Ssat = 1,4V

$\overline{\mu n}$ Cox = 4,4 x 10^{-5}

IDsat = 1/2 $\overline{un}$ Cox γc(Vseuilc)^2 = 4,3 x 10^{-5} γ c

d'où tm(ns) $\#$ 96 C(pF)/γc

Soit C = 0,125 pF, d'où tm(ns) = 12/γc

Soit tm = 18ns, le temps de montée désiré, on obtient : γ c = 2/3

On calcule maintenant le facteur de forme : γ r = 2,25

On obtient alors γ s = γr x γc : γs = 3/2

On a déterminé les quatre paramètres de dimensions d'un inverseur, à savoir (pour CMP) :

- pour le transistor signal :
  la longueur de la grille Ls = 2λ
  la largeur de la grille Ws = 3λ
- pour le transistor de charge :
  la longueur de la grille Lc = 3λ
  la largeur de la grille Wc = 2λ

(iv) Dimensions de la porte complexe

On garde les mêmes dimensions pour le transistor de charge. Pour les transistors signaux :

- les transistors mis en parallèle auront la même taille que le transistor seul : Ws et Ls (figure 32a)
- les transistors mis en série auront une largeur de grille égale à nWs si n est le nombre de transistors en série, la longueur de grille Ls est la même : nWs et Ls (figure 32b)

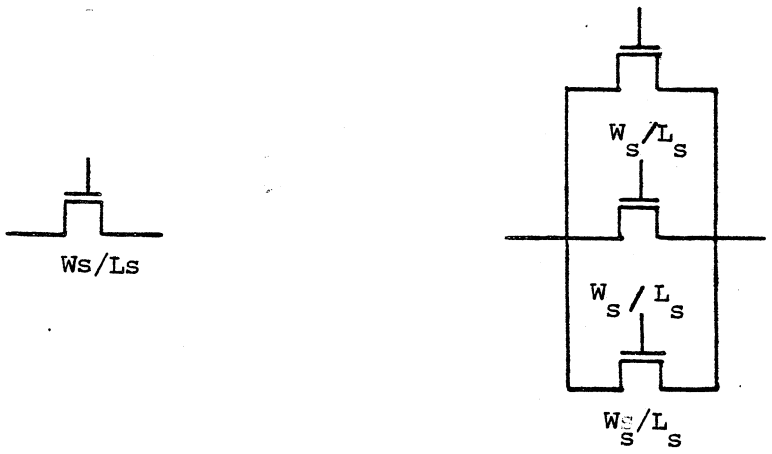


Figure 32a : mise en parallèle de transistors

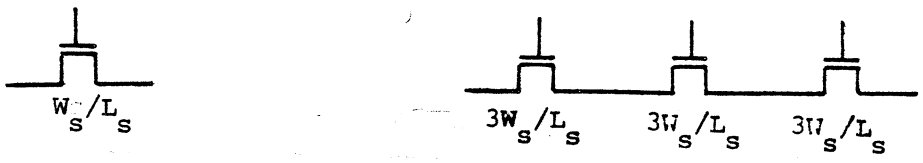


Figure 32b : mise en série de transistors

On prendra une borne maximum pour les réseaux non série-parallèle.

Remarque : Pour un calcul moins simplifié, on pourra calculer la valeur de la capacité C = CDS + Cl + CGS en(i) avec CDS, capacité des transistors signaux de la porte complexe équivalente.

Exemple 1

Soit la porte complexe NMOS dont le réseau $\overline{R}$ a été simplifié précédemment (figure 33) ; les dimensions des transistors signaux sont notées sur cette figure, Ws et Ls étant les dimensions calculées pour l'inverseur.

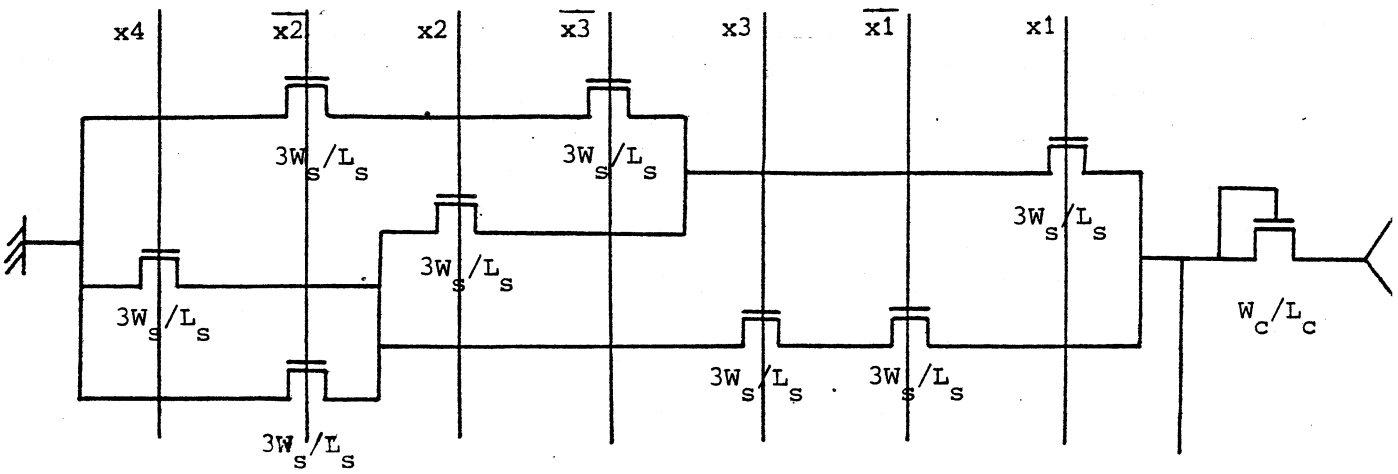


Figure 33 : Porte complexe et dimensions

Exemple 2

Soit la porte complexe donnée en annexe 1, réalisant la fonction $F = \overline{(A+B)C + D}$ (figure 34). les dimensions des transistors signaux sont notés sur cette figure, Ws et Ls ayant les dimensions calculées pour l'inverseur.

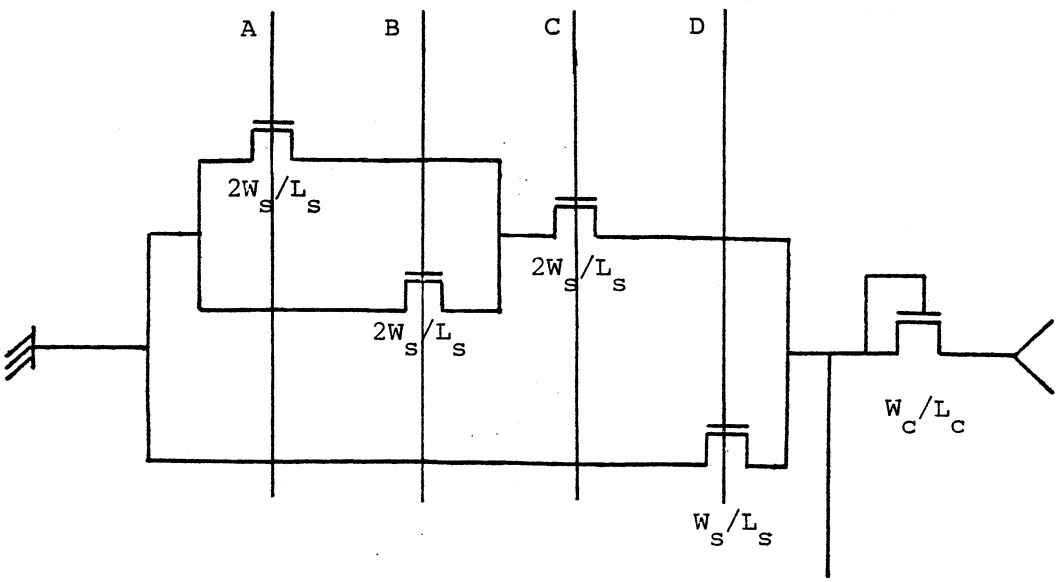


Figure 34 : Porte complexe et dimensions

Avec les valeurs calculées précédemment, on a :

- Lc = 3λ, Wc = 2λ
- Ls = 2λ, Ws = 3λ, 2Ws = 6λ

III-4. Portes complexes NMOS à précharge

La technologie NMOS a comme inconvénient une forte consommation statique et un temps de charge (passage du niveau logique "0" à "1" en sortie) long par rapport au temps de décharge : temps de commutation dissymétrique (§ III-3). Pour y remédier, on utilise des dispositifs à précharge : précharge au niveau logique "1".

(i) Le principe de cette précharge pour une porte est le suivant (figure 35a et 35b)

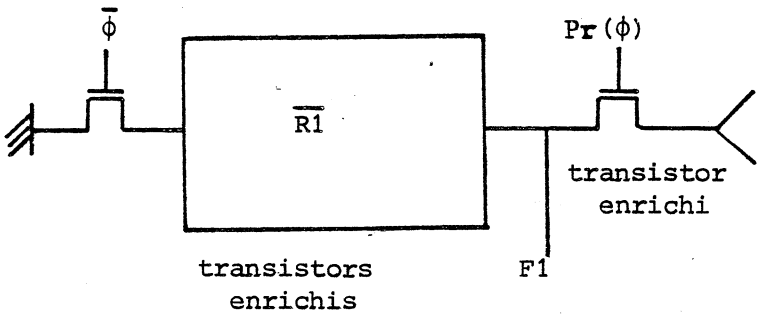


Figure 35a : Schéma de principe d'une porte à précharge en NMOS

Pr est le signal de précharge, présent pendant ϕ. La validation des entrées du réseau $\overline{R}$ se fait pendant $\overline{\phi}$:

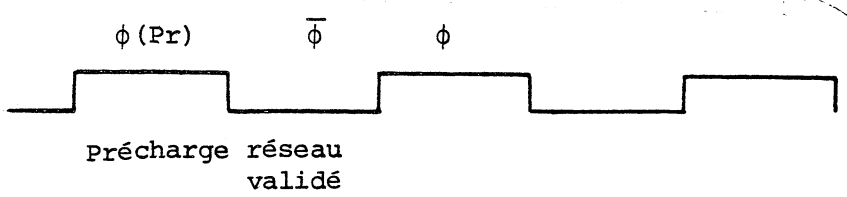


Figure 35b : Principe de fonctionnement d'une porte à précharge

(ii) L'inconvénient de la précharge en NMOS est de porter la sortie à "3,5V" au lieu de 5V : le transistor NMOS dégrade le niveau logique "1" d'une tension de seuil (§I-1).

Pour se rapprocher d'un niveau logique correct en sortie, il existe plusieurs solutions.

Une solution consiste à fabriquer des transistors enrichis avec une tension VT proche de 0V. On a habituellement VT ≠ 0,6 V, d'où Vseuil ≠ 1,5 V ; avec VT proche de 0V, on a Vseuil qui décroit, on se rapproche donc du "5V" en sortie.

Souvent, on précharge la sortie à "3,5V", on augmente alors la taille des transistors attaqués par cette tension pour récupérer un niveau logique correct.

(iii) Toute précharge s'accompagne d'un problème de fuite de précharge. En effet, la précharge est maintenue pendant un temps maximal (capacité et résistance de la ligne cf. figure 36) ; on a donc une fréquence minimale de fonctionnement.

Pour y remédier, on peut la maintenir avec une forte résistance (figure 36), d'où un faible courant : il faut faire attention à la consommation statique.

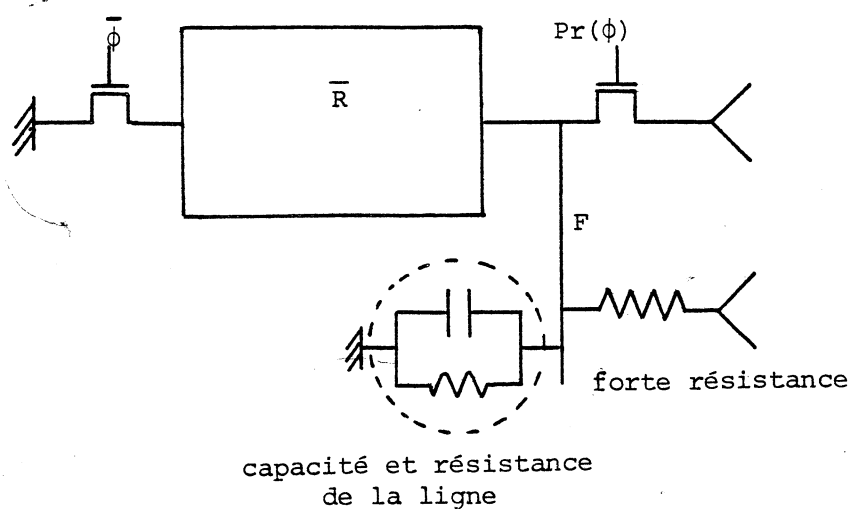
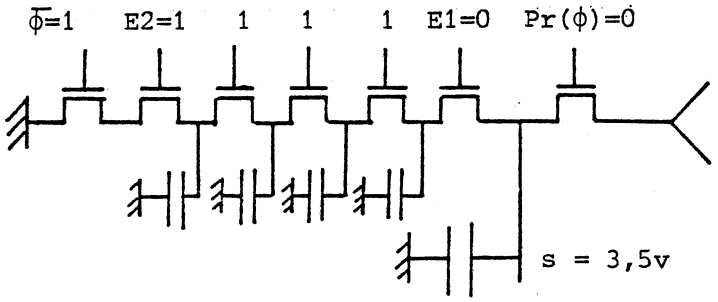


Figure 36 : Principe de maintien de la précharge

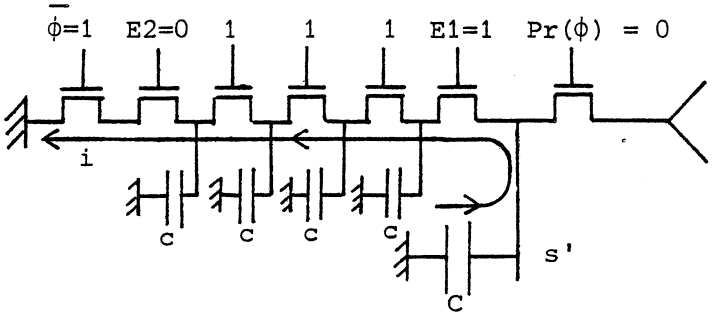
On peut obtenir une forte résistance de différentes façons, par exemple :

- en utilisant du Silicium polycristallin peu dopé
- avec un transistor déplété ayant un rapport $\gamma = W/L$ faible (surface importante de ce transistor)
- en utilisant un transistor légèrement conducteur (en appliquant sur sa grille une tension V proche de sa tension de seuil).

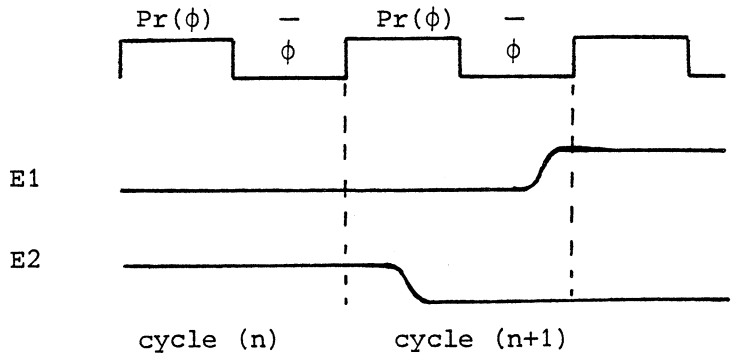
(iv) Un problème important de la précharge est celui du partage des charges, si les transistors sont en série dans le réseau $\overline{R}$. (figure 37).



$\overline{\phi}$, cycle (n) les capacités intermédiaires se déchargent



$\overline{\phi}$, cycle (n+1) la capacité de précharge se décharge partiellement dans les capacités intermédiaires



A.N : soit C = 0,15pF c = 0,03pF

$$s' = 3{,}5V \frac{C}{C+4c} \# 2V$$

Figure 37 : Problème du partage de charges

On y remédie de différentes façons :

- en évitant les transistors série
- en implantant les transistors série de façon à ce qu'ils aient une faible capacité
- avec des capacités des transistors du réseau très inférieures à celles de la capacité de précharge, c'est le cas des mémoires
- en préchargeant aussi les capacités du réseau, on a alors des cycles de précharges complexes
- en n'autorisant le changement des entrées du réseau que pendant la précharge.

(v) La précharge de la sortie au niveau logique "1" rend conducteurs les transistors N attaqués. Le signal $\overline{\phi}$ appliqué en même temps sur toutes les portes implique la fuite de la précharge : non-réalisation de la foncion apparition d'une consommation statique. Il existe plusieurs solutions pour rendre la précharge compatible.

La première méthode consiste à isoler deux étages consécutifs de la façon suivante (figure 38a).

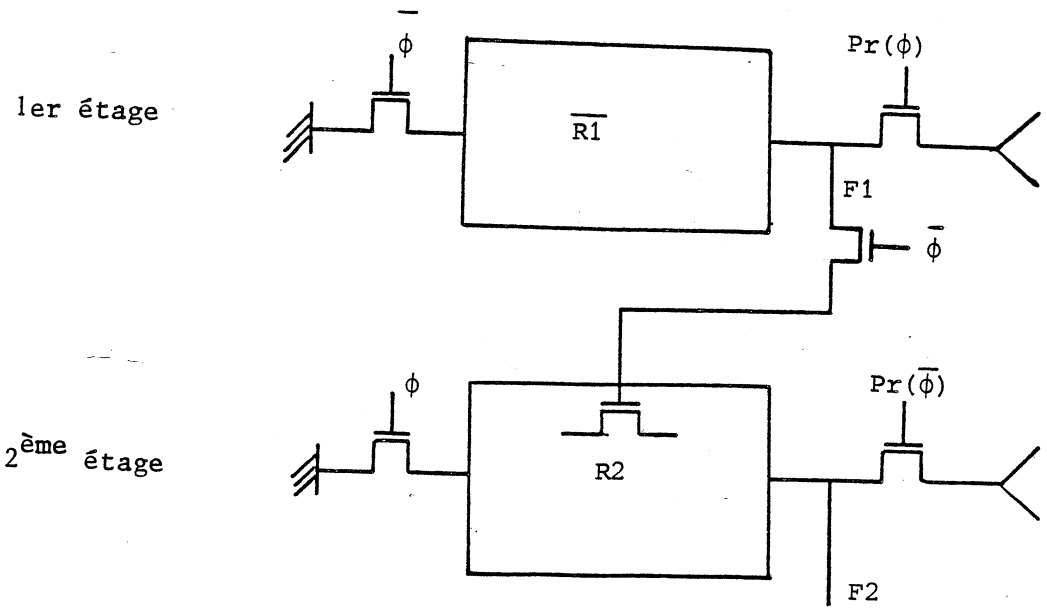
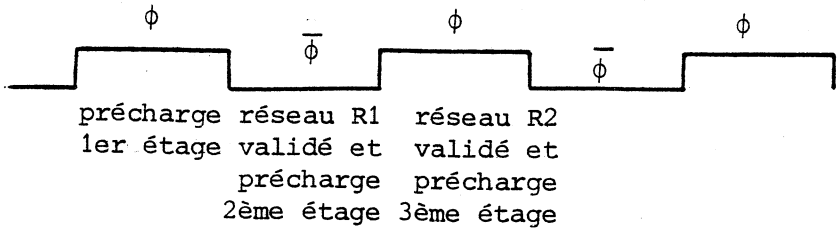


Figure 38a

Le principe de fonctionnement est le suivant :



De cette façon, les entrées du réseau ne changent que pendant la précharge de celui-ci, on a donc pas de problème de partage de charges.

L'inconvénient de cette solution est d'être une logique séquentielle (un top d'horloge par étage : faible vitesse), on devra donc travailler à une fréquence importante. L'avantage est la réalisation de toutes les fonctions logiques.

La deuxième méthode consiste à introduire un inverseur entre deux étages consécutifs (figure 38b).

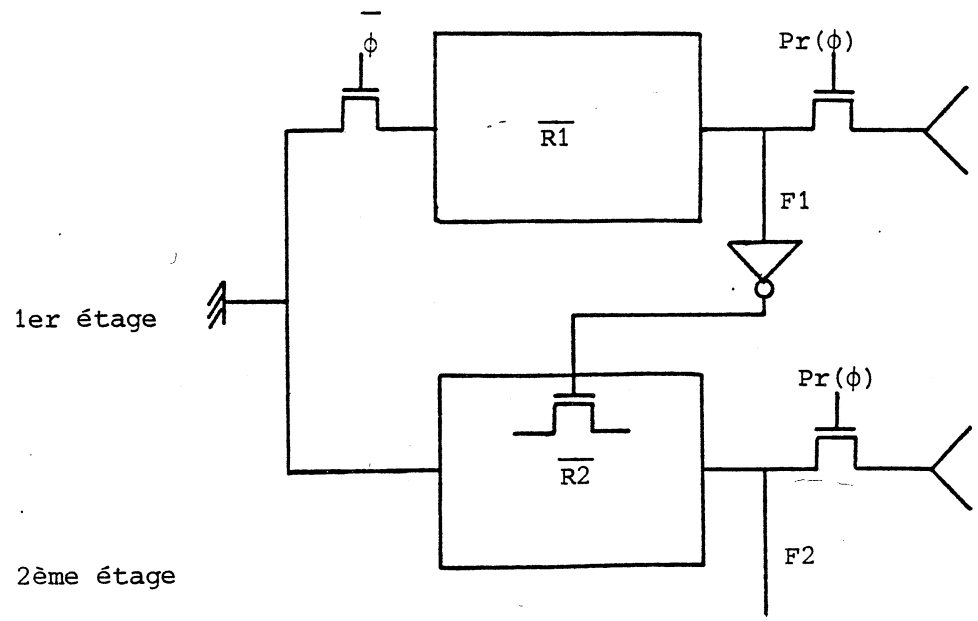


Figure 38b

Le signal de précharge est présent pendant ϕ, les entrées sont validées pendant $\overline{\phi}$.

L'avantage de cette solution est d'avoir un top d'horloge pour plusieurs étages (bonne vitesse), on travaille à une fréquence basse.

L'inconvénient réside dans la limitation des fonctions réalisées (fonction complémentaire en sortie de l'inverseur). Cependant, cette méthode reste tout à fait utilisable dans le cas où le précharge est la plus intéressante : mémoires et PLA.

La troisième méthode consiste à appliquer le signal $\overline{\phi}$ avec un retard ΔT entre deux étages consécutifs. Ce temps ΔT est le temps nécessaire à la stabilité des entrées au moment où le signal $\overline{\phi}$ est appliqué (figure 38c).

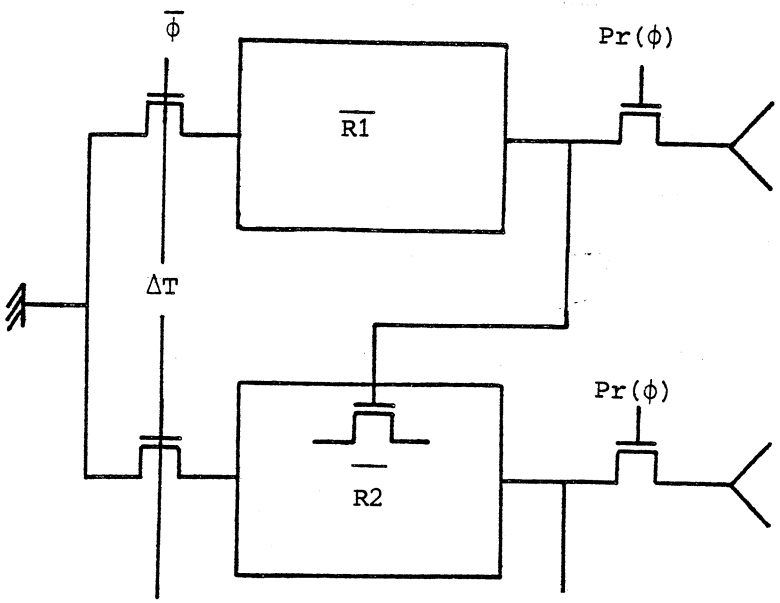


Figure 38c

Cette méthode est utilisée dans le cas des mémoires et PLA.

IV - PORTES COMPLEXES CMOS

IV - 1. Généralités

La porte logique CMOS est définie par le schéma suivant (figure 39).

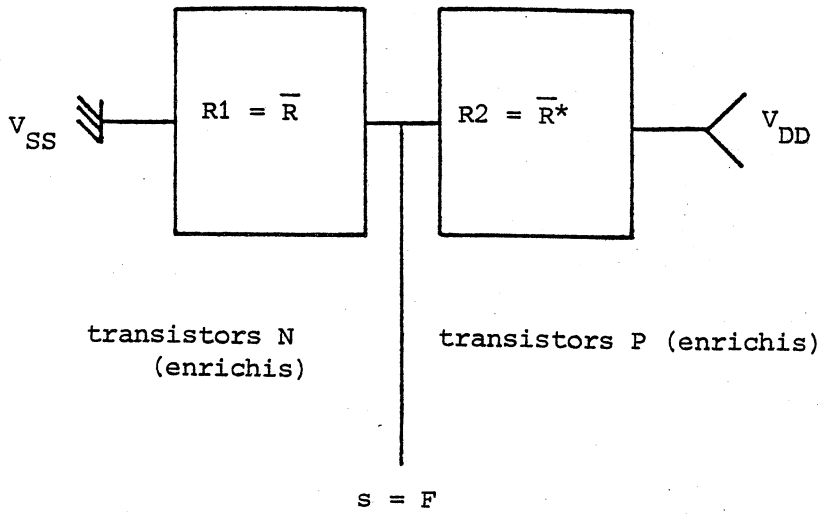


Figure 39 : Porte CMOS

La sortie de la porte réalisant la foncion F est reliée à VSS par un réseau à base de transistors N de fonction de transfert $\bar{F}$ et à VDD par un réseau à base de transistors P de fonction de transfert $\bar{F}^*$ (duale de $\bar{F}$). On vérifiera aisément que :

- cette porte réalise la foncion F
- que la sortie est toujours reliée soit à VDD, soit à VSS
- qu'aucun chemin ne peut relier VDD à VSS

Les avantages de la technologie CMOS sont d'éliminer la consommation statique, d'avoir un temps de charge équivalent au temps de décharge. La masse VSS est transmise par des transistors N, l'alimentation VDD par des transistors P : on a donc des niveaux logiques corrects en sortie de la porte (§ I-1.)

Porte simple et porte complexe

Pour les portes simples le réseau dual $\bar{R}^*$ peut être obtenu graphiquement à partir du réseau $\bar{R}$ en associant à chaque chemin de $\bar{R}$, le réseau "coupure" correspondant dans $\bar{R}^*$. Ce réseau dual topologique est le même que le réseau dual calculé à partir de l'expression de la fonction pour une porte simple.

Exemple

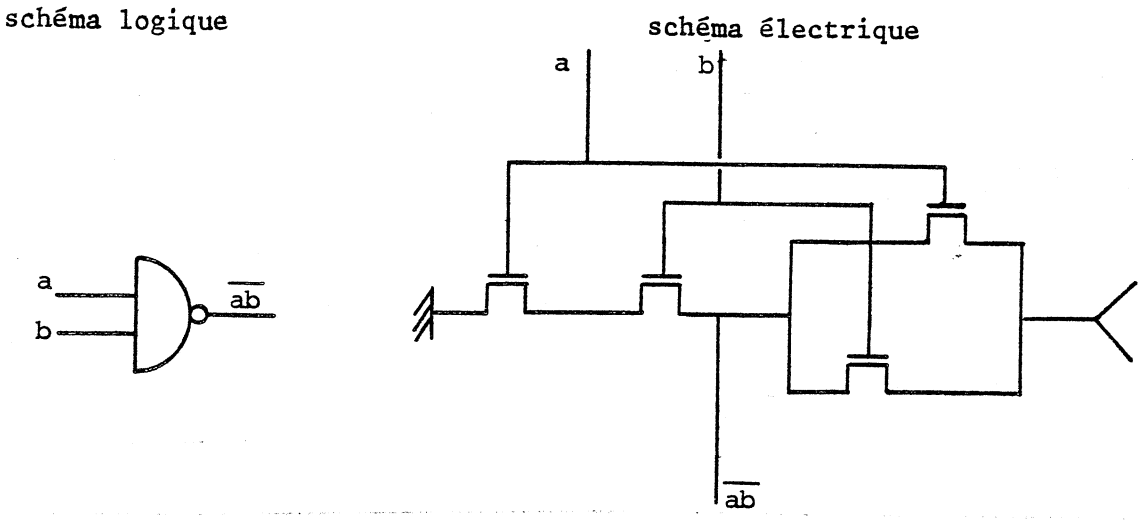


Figure 40 : Porte NAND à 2 entrées : $F = \overline{ab}$

Pour les portes complexes, le réseau dual topologique peut être différent du réseau dual calculé à partir de l'expression de la fonction.

Exemple

Soit la foncion $\overline{F}$ minimisée : $\overline{F} = \bar{x}y + \bar{y}\bar{z}$, on a directement le réseau $\overline{R}$ associé et le réseau $\overline{R}^*$ dual topologique (figure 41a).

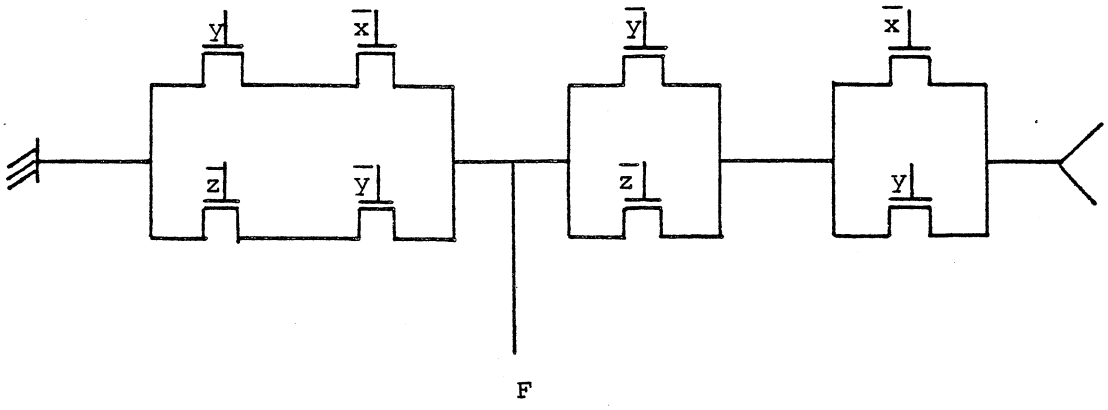


Figure 41a : Porte CMOS réalisant la fonction F, le réseau $\overline{R}^*$ étant le réseau dual topologique

Le réseau dual est différent s'il est obtenu par calcul algébrique de la fonction duale et minimisation classique (figure 41b).

$$\bar{F}^* = (\bar{x} + y)(\bar{y} + \bar{z}) = \bar{x}\bar{y} + \bar{x}\bar{z} + y\bar{z} = \bar{x}\bar{y} + y\bar{z}$$

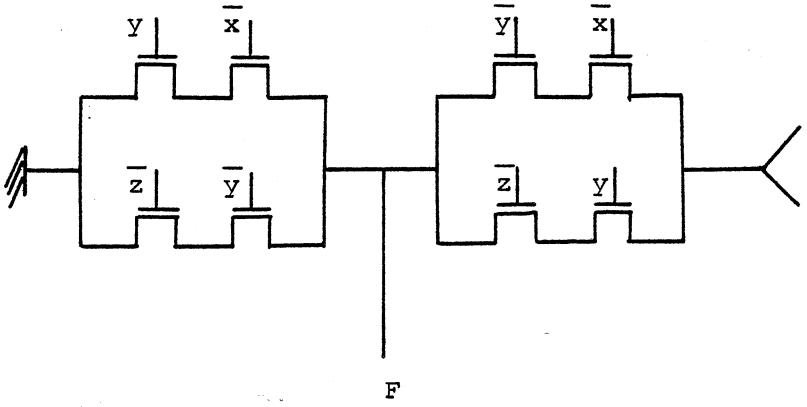


Figure 41b : Porte CMOS réalisant la fonction F, $\bar{R}^*$ réalisant la fonction $\bar{F}^*$calculée

Pour les portes complexes, le nombre de transistors mis en série dans $\bar{R}$ ou dans $\bar{R}^*$ sera limité à quatre ou cinq. En effet, pour garder la même rapidité à la porte, la taille des transistors va augmenter avec la mise en série de ceux-ci, les résistances équivalentes des transistors s'ajoutent (§ IV-3 Dimensionnement). Le nombre maximal de transistors mis en parallèle dans ces réseaux $\bar{R}$ et $\bar{R}^*$ sera de 10 environ pour la rapidité aussi (les capacités en parallèle s'ajoutent). Les portes complexes CMOS couramment utilisées sont données en annexe 2.

On s'interresera ici aux portes complexes CMOS dont les réseaux $\bar{R}$ et $\bar{R}^*$ sont optimisés séparément ou globalement, une telle porte sera appelée porte CMOS optimisée. Pour une porte complexe réalisant la fonction F, les réseaux $\bar{R}$ et $\bar{R}^*$ réalisant la fonction seront optimisés. On étudiera :

- l'implantation d'une porte
- le dimensionnement.

Enfin on abordera le problème de la précharge.

IV - 2. Implantation d'une porte complexe CMOS

Comme pour la technologie NMOS nous allons nous intéresser à des synthèses systématiques donnant des structures régulières et à entrée-sortie des deux côtés des portes.

On définit d'abord la technologie uilisée et les différents matériaux, en passant du schéma électrique au schéma topologique de l'inverseur. Puis on proposera deux organisations topologiques différentes pour une porte complexe CMOS et deux types d'implantation pour les réseaux.

On utilise une technologie CMOS avec un seul niveau d'aluminium et de silicium polycristallin. On définit les matériaux employés de la façon suivante (figures 42a et 42b)

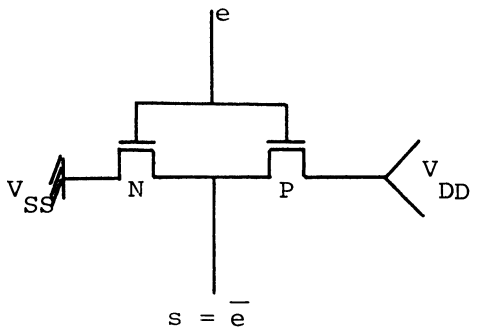


Figure 42a : Schéma électrique de l'inverseur

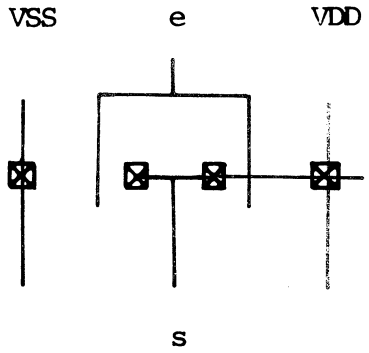


Figure 42b : Matériaux utilisés pour l'inverseur

On définit les couleurs standards suivantes par la technologie CMOS :

- aluminium
- Silicium polycristallin
- diffusion N
- diffusion P
- ⊠ contact alu-autre matériau.

On a (figure 42a et 42b) une ligne de diffusion P puis N entre les deux points de contacts avec les alimentations VDD et VSS. Le croisement du silicuim polycristallin avec l'une des diffusions N ou P crée un transistor. Les grilles des deux transistors qui reçoivent le même signal d'entrée sont reliées en silicium polycristallin.

On propose deux organisations topologiques pour une porte (figure 43a et 43b)

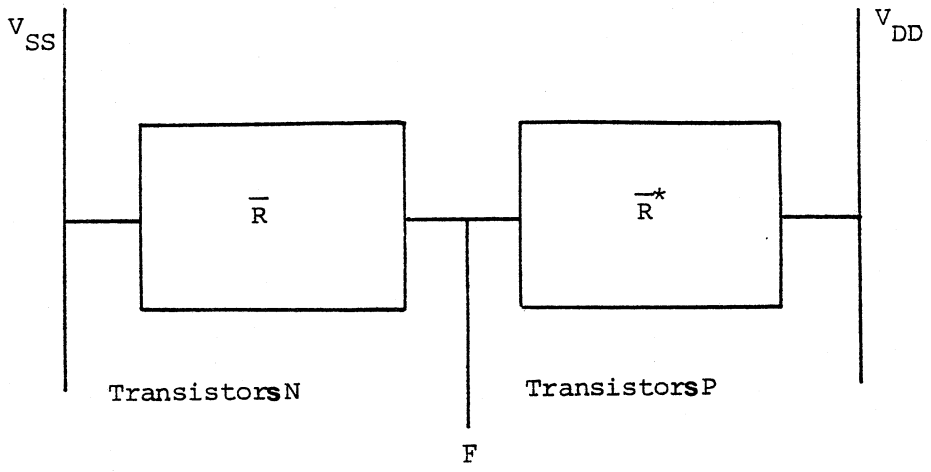


Figure 43a : 1ère organisation topologique pour une porte

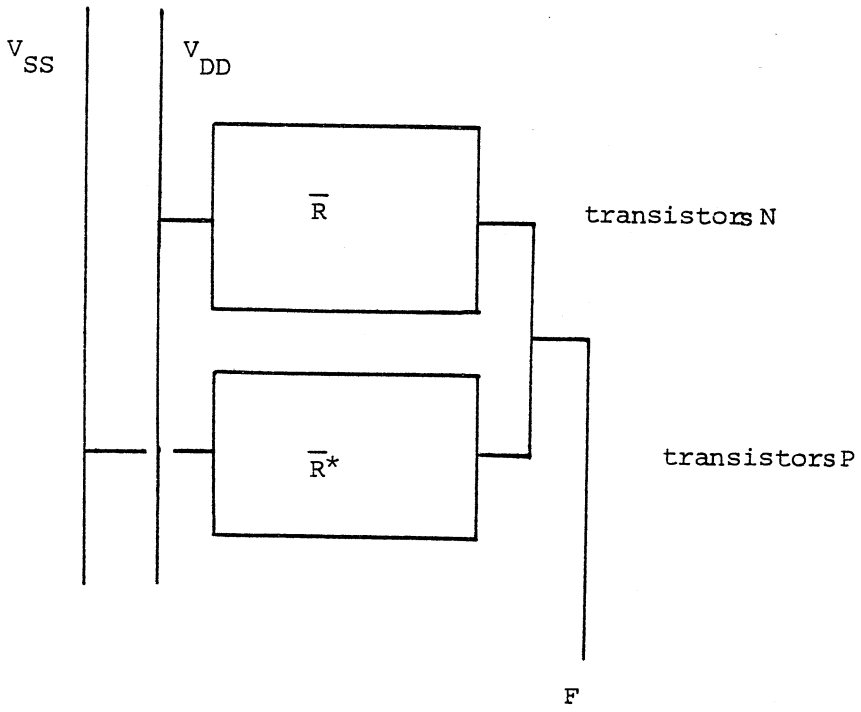


Figure 43b : 2ème organisation topologique pour une porte

L'implantation d'un réseau CMOS sera semblable à celle d'un réseau NMOS. Le réseau est un quadrillage : les variables d'entrée sur les lignes verticales (données), les chemins horizontaux seront en diffusion N ou P. Les variables traversent les réseaux (colonnes), elles peuvent arriver par le haut ou le bas du réseau suivant les connexions de celui-ci avec le reste du circuit.

On propose les deux mêmes types d'implantation (figure 44a, 44b) que pour les portes NMOS : le 2ème type d'implantation de la technologie NMOS ne peut exister en CMOS où il n'existe pas de transistors déplétés.
On traite ces deux types d'implantation sur l'exemple de porte CMOS donné sur la figure 41b.

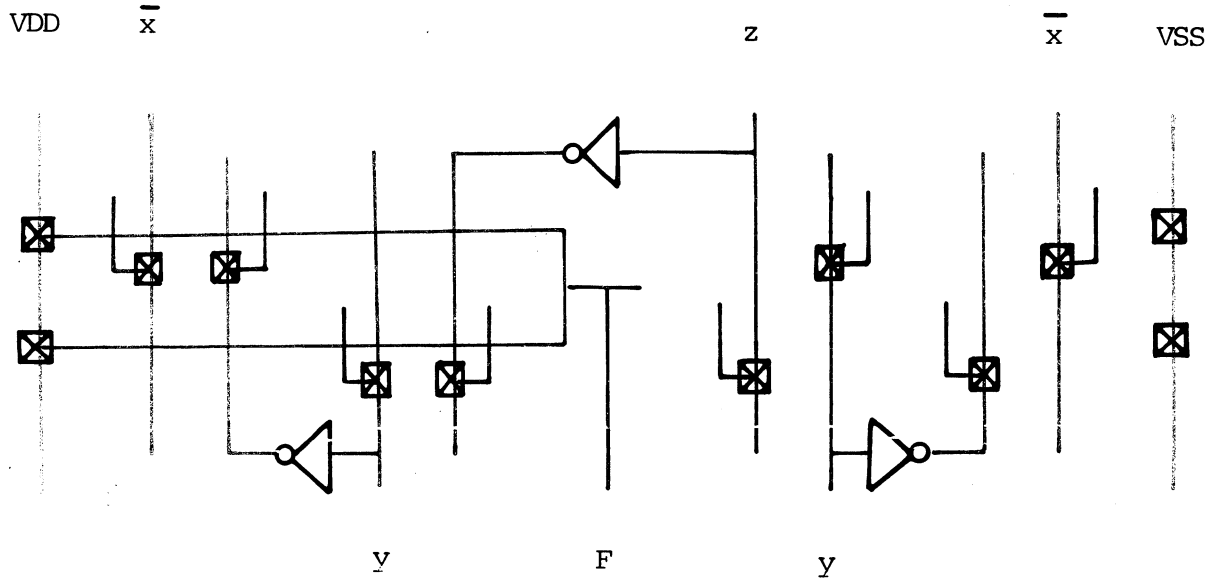


Figure 44a : 1er type d'implantation avec la 1ère organisation topologique (réseaux optimisés séparément)

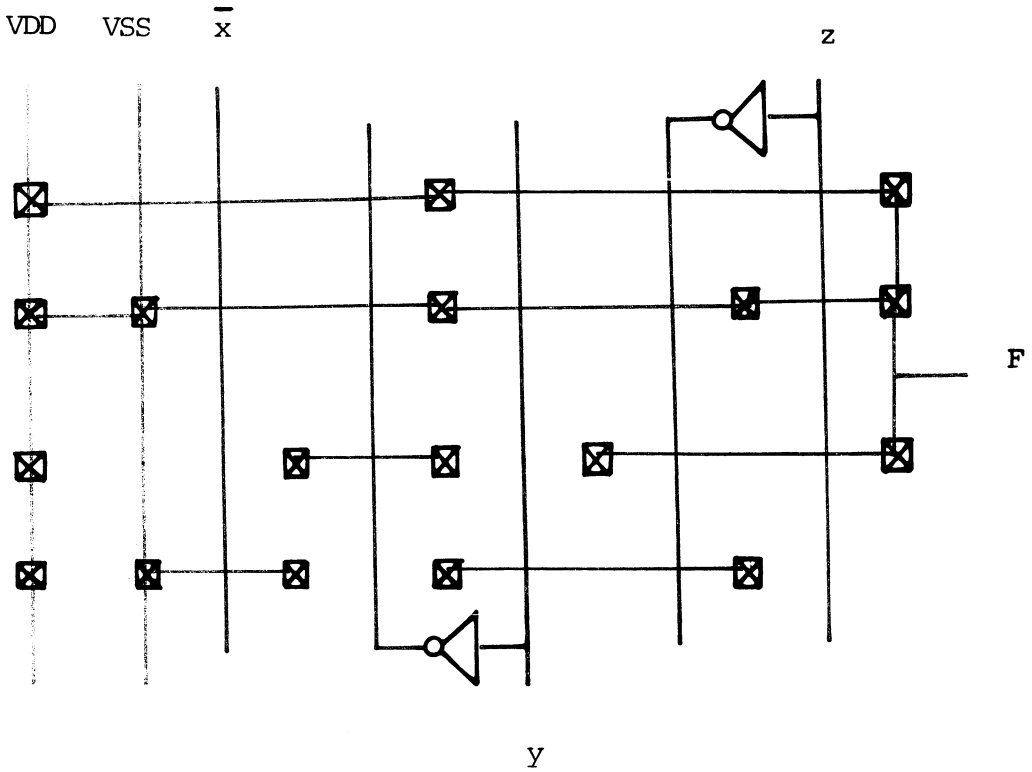


Figure 44b : 2ème type d'implantation avec la 2ème organisation topologique (réseaux optimisés ensemble)

De même que pour les portes NMOS, il faudra dimensionner les transistors (cf IV-3. dimensionnement), on pourra tordre les lignes de diffusion, imbriquer les portes, couper les lignes de variables.

Ces deux types d'implantation sont les mêmes que ceux proposés pour les portes NMOS, la différence réside dans le fait qu'on a des chemins en diffusion N dans les réseaux $\bar{R}$ et en diffusion P dans les réseaux $\bar{R}^*$.

IV - 3. Synthèse logique et topologique (cas d'une porte)

Dans le cas de la première organisation topologique (figure 43a), les réseaux $\bar{R}$ et $\bar{R}^*$ sont optimisés séparément. Les ordres lexicographiques sont distincts pour les deux réseaux et la méthode identique à celle utilisée pour le NMOS. Pour le réseau $\bar{R}^*$, il est plus simple de repartir de la forme algébrique minimisée en somme de produits. L'écriture en produit de monaux amenant difficilement à un réseau lexicographique non plié et le plus souvent à une solution pliée.

Dans la deuxième organisation topologique (figure 43b), nous sommes ramenés à une optimisation simultanée de deux réseaux, le problème d'optimisation de deux réseaux est le même que celui d'un réseau en considérant que l'ensemble des chemins initiaux est l'union des chemins. Il faut signaler l'intérêt exceptionnel des fonctions autoduales (porte majorité par exemple) en CMOS.

Exemple 1

L'exemple de la figure 41b est un cas sans difficulté (cf. figure 45)

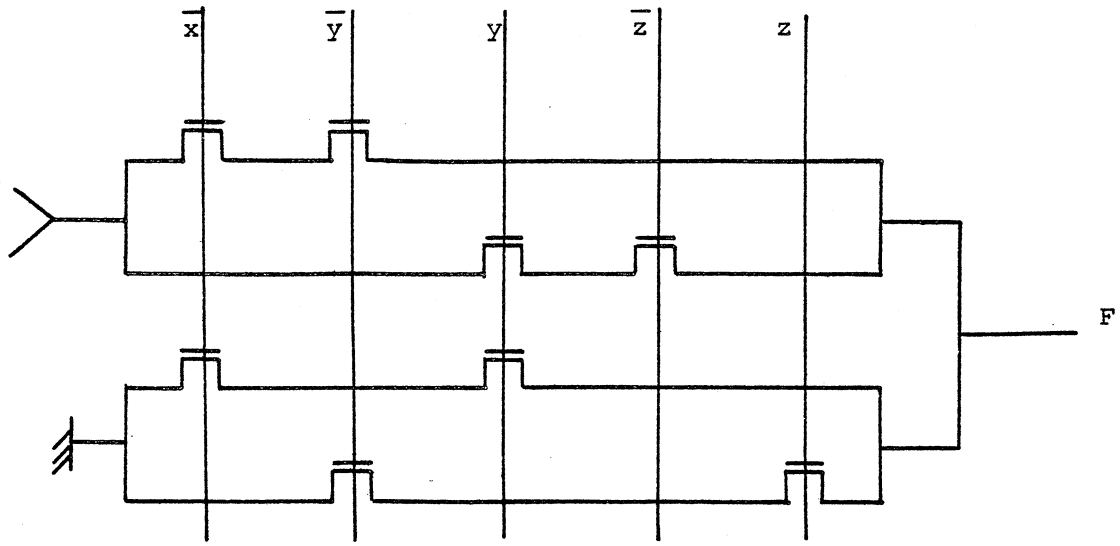


Figure 45 : Porte CMOS réalisant la fonction $F = xy + \bar{y}z$

Exemple 2

Soit le réseau $\bar{R}$ de la figure 14b du § 2.3.3. Un dualisme topologique déduit directement de la fonction donne un réseau avec pliage :

$$\bar{F} = x1\ (x2x4 + \bar{x}2\bar{x}3) + \bar{x}1x3\ (x4 + \bar{x}2)$$
$$\bar{F}^* = (x1 + (x2 + x4)\ (\bar{x}2 + \bar{x}3))(\bar{x}1 + x3 + x4\bar{x}2)$$

En effet, cette écriture de $\bar{F}^*$ sous forme de produit de monaux amène à une solution avec un pliage entre les deux monaux, en intervertissant $\bar{x}2$ et $x2$ (figure 46a)

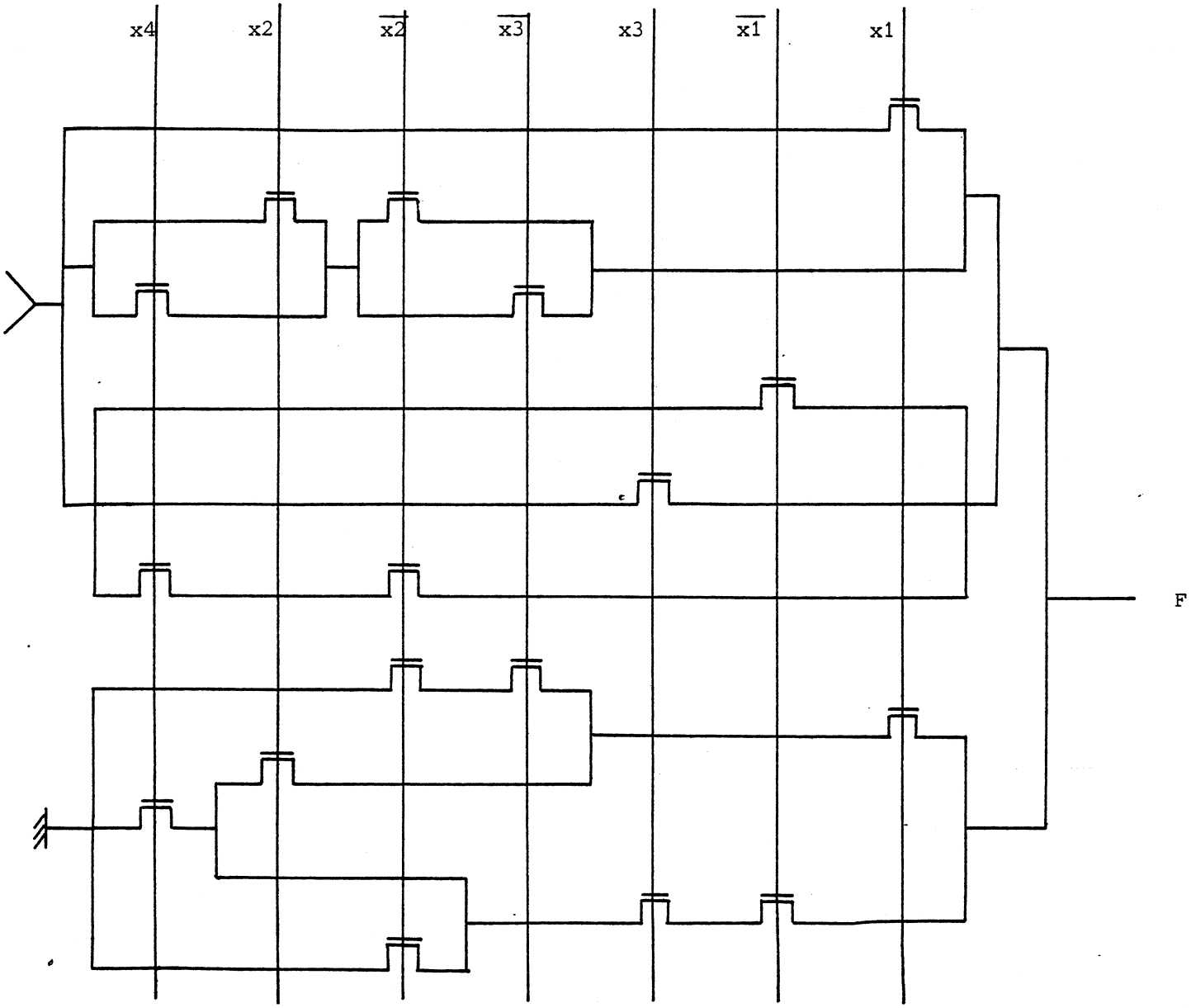


Figure 46a : Porte CMOS, $\bar{R}^*$ exprimée sous forme produit de monaux

Le calcul de la fonction duale donne :

$$\bar{F}^* = x1x3 + \bar{x}2x4 + \bar{x}1x2\bar{x}3 + \bar{x}1\bar{x}3x4$$

On obtient le tableau suivant :

variables	x1	x2	x3	x4
partition	(1)(34)	(3)(2)	(1)(34)	(24)
gain	1	0	1	1

L'ordre lexicographique intéressant sera x1x3...., le gain sera alors de 2 transistors.

On peut de plus fusionner les transistors commandés par la variables x4 dans une deuxième étape sans introduire de chemins parasites avec l'ordre : x1x3x2x4

Pour $\bar{R}$ et $\bar{R}^*$, l'ordre lexicographique x1x3x2x4 est intéressant en effet (figure 46b) :

- pour $\bar{R}$: fusionnement de x1,$\bar{x}$1 et x3 dans une première étape
  fusionnement de x4 dans une deuxième étape
- pour $\bar{R}^*$: x1,x3 dans la première étape
  x4 dans la deuxième étape.

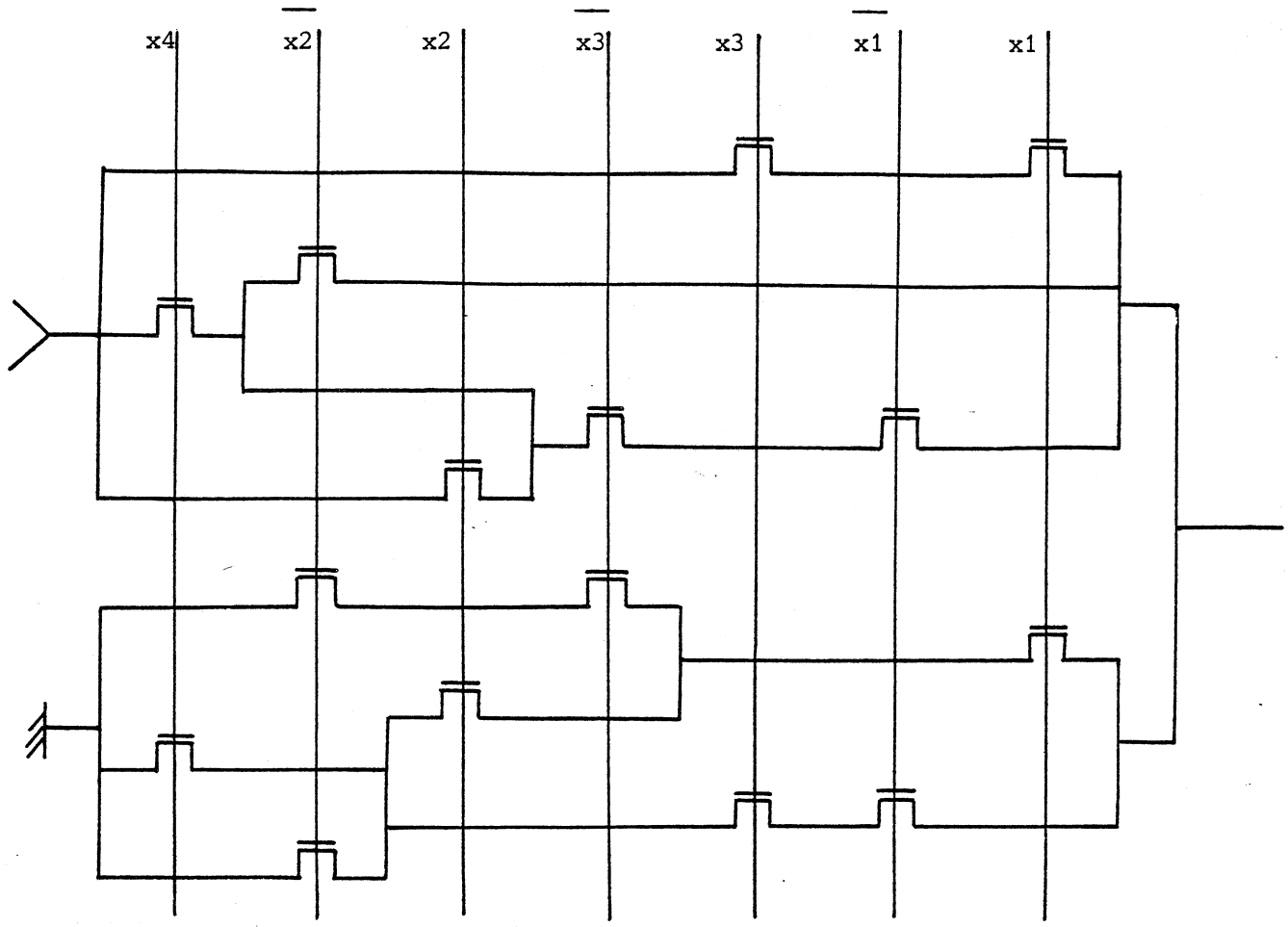


Figure 46b : Porte complexe CMOS optimisée

On voit donc l'intérêt de travailler sur les réseaux, et non sur les fonctions.

IV - 4. Dimensionnement d'une porte complexe CMOS

Comme pour les portes complexes NMOS (§ III-3.), on calcule d'abord les dimensions de l'inverseur répondant aux objectifs de fonctionnement, puis on en déduira les dimensions de la porte complexe équivalente.

On note les dimensions des transistors MOS comme suit (figure 47) :

- WN, largeur de la grille du transistor N
- LN, longueur de la grille du transistor N
- WP, largeur de la grille du transistor P
- LP, longueur de la grille du transistor P

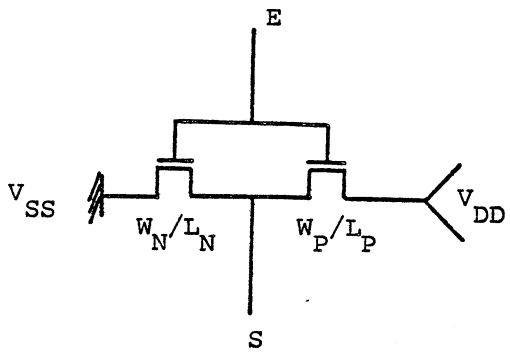


Figure 47 : Inverseur

On cherche à déterminer les quatre paramètres de dimensions d'un inverseur, à savoir :

- WN,LN pour le transistor NMOS
- WP,LP pour le transistor PMOS

de façon à atteindre les objectifs de fonctionnement fixés.

II n'est pas nécessaire de calculer ces quatre paramètres, il suffit d'évaluer γN et γP définis par :

- γN = WN/LN rapport du transistor NMOS

- γP = WP/LP rapport du transistor PMOS

On présente ici une étude simplifiée du calcul des dimensions. On trouvera dans (RAM 73), un calcul plus précis des dimensions d'un inverseur CMOS.

(i) Dimensions du transistor NMOS

Soit td, le temps de descente désiré pour cet inverseur. Le passage du niveau logique "1" au niveau "0" en sortie de l'inverseur se fait en déchargeant la capacité de la sortie C. On considère le schéma suivant (figure 48) pendant la décharge de la capacité.

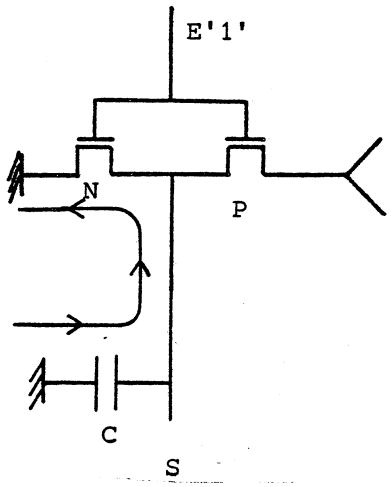


Figure 48 : Décharge de la capacité C en sortie d'un inverseur

C est la capacité équivalente en sortie de l'inverseur. Le transistor PMOS est bloqué en effet, sa tension de seuil VseuilP est négative et la tension appliquée VGS = E"1" - VDD = 0 ; ce transistor n'est conducteur que pour une tension négative, supérieure en valeur absolue à sa tension de seuil.

la décharge de la capacité C se fait par l'intermédiaire du transistor NMOS passant : la tension VGS appliquée est égale à VDD, donc supérieure à VseuilN

On a :

$$VDSsat = VGS - VseuilN = VDD - VseuilN$$
$$VDS = S - VSS = S$$

Le transistor fonctionne dans le mode saturé si :

$$VDS > VDSsat \Longrightarrow S > VDD - VseuilN$$

Soit Ssat = VDD - VseuilN

Pour Ssat < S < 1, le transistor fonctionne dans le mode saturé.

$$IDsat = 1/2\ \bar{u}n\ Cox\gamma N\ (VGS - VseuilN)^2$$
$$IDsat = 1/2\ \bar{u}n\ Cox\gamma N(VDD - VseuilN)^2 = 1/2\ \bar{u}n\ Cox\gamma N\ Ssat^2$$
$$IDsat = \text{constante} \times \gamma N$$

Pour S > Ssat le transistor fonctionne dans le mode conducteur.

$$IDcond = \bar{u}n\ Cox\gamma N((VGS - VseuilN)\ VDS - VDS^2/2))$$
$$IDcond = \bar{u}n\ Cox\gamma N((VDD - VseuilN)\ VDS - VDS^2/2)$$
$$IDcond = IDsat\ (2\ VDS/Ssat - VDS^2/Ssat^2)$$

On calcule le temps de descente de la façon suivante :

$$td = -C\int_{0,9VDD}^{0,1VDD} dS/ID$$
$$td = -C\left(\int_{0,9VDD}^{Ssat} dS/IDsat + \int_{Ssat}^{0,1VDD} dS/IDcond\right)$$

Après calcul, on trouve :

$$\int_{0,9VDD}^{Ssat} dS/IDsat = 1/IDsat\ (Ssat - 0,9VDD)$$
$$\int_{Ssat}^{0,1VDD} dS/IDcond = Ssat/2IDsat\ (\log\ (S/2Ssat-S))_{Ssat}^{0,1\ VDD}$$

$$= (Ssat/2IDsat)\ \text{Log}\ 0,1VDD/(2Ssat-0,1VDD)$$

D'où

$$td = -(Ssat - 0,9VDD + Ssat/(2\ \log\ 0,1VDD/(2Ssat-0,1VDD)))\ C/IDsat$$

$$td = \text{constante} \times C/\gamma N$$

A partir du temps de descente td donné et de la capacité C calculée, on trouve le rapport du transistor NMOS : $\gamma N = WN/LN$

La valeur de la capacité de la sortie se calcule en ajoutant les différentes capacités comme pour la technologie NMOS (§ III-3.).

(ii) Dimensions du transistor PMOS

On peut calculer le rapport du transistor PMOS $\gamma P = WP/LP$ à partir du temps de montée tm donné et de la capacité C calculée : le calcul est analogue à celui fait précédemment pour le transistor NMOS.

Pour une étude simplifiée, on prendra le rapport du transistor PMOS égal à 2 fois celui du transistor NMOS : $\gamma P = 2\ N$, c'est-à-dire WP = 2WN (la mobilité des trous étant inférieure à celle des électrons).

(iii) Application numérique

On fait les calculs avec les paramètres électriques de CMP pour la technologie CMOS (CMP 83) :

VDD = 5V
VseuilN = 0,9v
$\bar{u}n = 700 \times 10\text{-}4\ m^2/Vs$
$Cox = 4 \times 10\text{-}4 pF/\mu^2$

On calcule le rapport du transistor NMOS à partir de l'expression
td = constante x $C/\gamma N$.

On a :

Ssat = VDD - VseuilN = 4,1V
$\bar{u}n\ Cox = 2,8 \times 10\text{-}5\ F/Vs$
$IDsat = 1/\bar{u}nCox\ \gamma N(Ssat)^2 = 2,35 \times 10\text{-}4 \gamma N$

d'où $td(ns) \# 25,5\ C(pf)/\gamma N$

soit C = 0,125 pF d'où $td(ns) = 3,19/\gamma N$

Soit td = 18ns, le temps de descente désiré, on obtient $\gamma N = 1/5$

On prend $\gamma P = 2 \gamma N$

On a déterminé les quatre paramètres de dimensions d'un inverseur à savoir (pour CMP) :

- pour le transistor NMOS
  la longueur de la grille LN = 2λ
  la largeur de la grille WN = 10λ
- pour le transistor PMOS
  la longueur de la grille LP = 4λ
  la largeur de la grille WP = 10λ

(iv) Dimensions de la porte complexe équivalente

On s'intéresse ici aux dimensions des transistors composant les réseaux N et P comme pour les portes complexes NMOS :

- les transistors mis en parallèle auront la même taille que le transistor seul : W et L
- les transistors mis en série auront une largeur de grille égale à n W si n est le nombre de transistor en série, la longueur de grille L est la même.

On prendra une borne maximum pour les réseaux non série-parallèle.

Exemple

Soit la porte complexe CMOS dont les réseaux simplifiés sont donnés sur la figure 49 ; les dimensions des transistors sont notés sur cette figure, W et L étant les dimensions calculées pour le transistor N de l'inverseur.

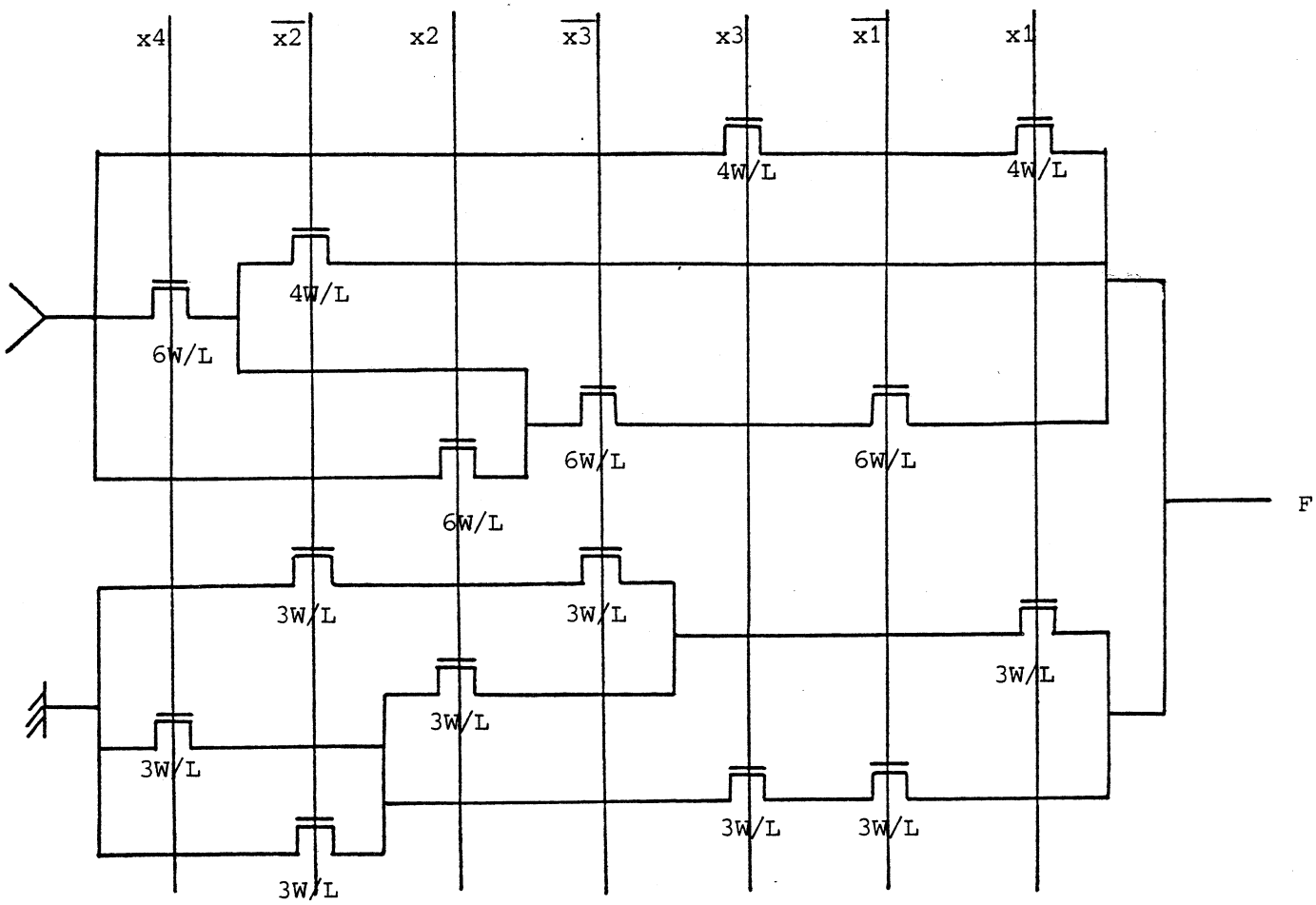


Figure 49 : Porte complexe et dimensions

Avec les valeurs calculées précédemment, on a :

- L = LN = 10 λ
- W = WN = 2λ, 3W = 6λ, 4W = 8λ, 6W = 12λ

IV - 5. Portes complexes CMOS à précharge

La précharge dans la technologie CMOS est intéressante pour le gain en surface, en effet on utilise alors un transistor au lieu d'un réseau $\overline{R}$ ou $\overline{R}^*$

On peut précharger les portes au niveau logique "1" ou "0" en supprimant respectivement les réseaux $\overline{R}$ ou $\overline{R}^*$. On étudie d'abord la précharge au niveau logique "1" de toutes les portes, puis la précharge au niveau logique "1" et "0" de façon alternée.

4.5.1. Précharge au niveau logique "1" de toutes les portes (KRA 82)

(i) Le principe de cette précharge pour une porte est le suivant (figures 50a et 50b).

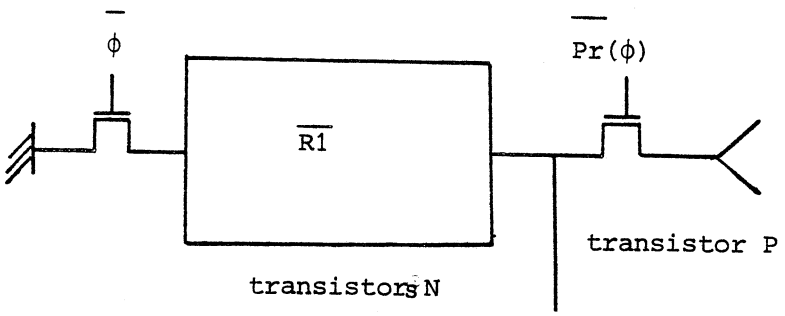


Figure 50a : Schéma de principe d'une porte à précharge à "1" en CMOS

$\overline{Pr}$ est le signal de précharge présent pendant ϕ. La validation des entrées du réseau $\overline{R}$ se fait pendant $\overline{\phi}$.

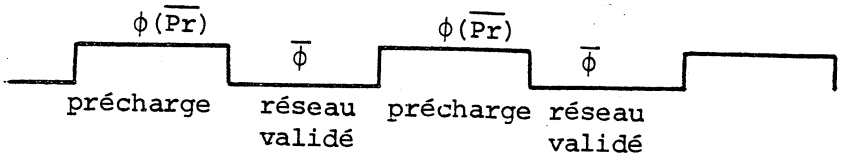


Figure 50b : Principe de fonctionnement d'une porte à précharge

(ii) La précharge au niveau logique "1" en CMOS est équivalente à la précharge en NMOS étudiée précédemment : la différence importante est qu'en CMOS la précharge porte la sortie à "5V", en effet le transistor PMOS transmet

correctement le niveau logique "1" (§ I-1.). On a les mêmes problèmes de fuite de précharge, de partage des charges et de précharge compatible, les schémas de précharge seront ceux des figures 38a,b,c, proposées pour la technologie NMOS au § III-4. à ceci près que le signal de précharge Pr sera appliqué à des transistors PMOS.

4.5.2. Précharge aux niveaux logiques "1" et "0" de façon alternée (GON83)

(i) Le principe de cette précharge est le suivant (figure 51a et 51b).

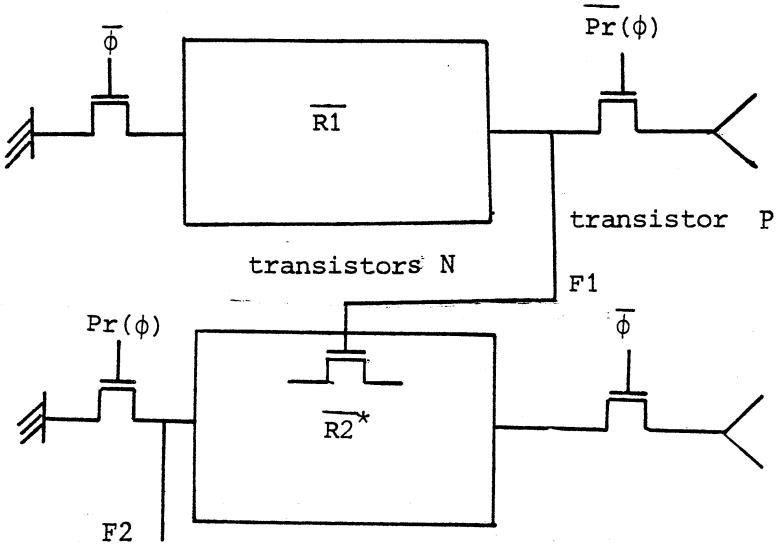


Figure 51a : Schéma de principe de la précharge à "1" et "0" en CMOS

$\overline{Pr}$ et Pr sont les signaux de précharge présents pendant ϕ La validation des entrées des réseaux $\overline{R}$ et $\overline{R^*}$ se fait pendant $\overline{\phi}$.

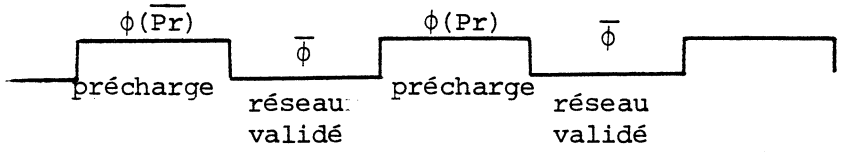


Figure 51b : Principe de fonctionnement de la précharge

(ii) La précharge au niveau logique "1", respectivement "0", porte la sortie à "5V", respectivement "0V",le transistor PMOS transmet correctement le niveau logique "1", le transistor NMOS le niveau logique "0" (§ I-1)

La précharge de la sortie à "1', respectivement "0", ne rend pas conducteurs les transistors P, respectivement N, attaqués. On peut donc appliqué $\overline{\phi}$ au même instant sur toutes les portes (précharge compatible).

On a aussi les problèmes de fuite de précharge, partage des charges.

Remarque : Si les réseaux $\overline{R}$ sont constitués de transistors en série et les réseaux $\overline{R}^*$ de transistors en parallèle : les portes réalisent des fonctions NAND (synthèse en NAND). Il est préférable de mettre des transistors N en série plutot que des transistors P : la taille d'un transistor P est plus importante qu'un transistor N (cf. § IV-3. Dimensionnement).

V - MACRO-CELLULES REALISANT DES FONCTIONS INDEPENDANTES

On s'intéresse ici à l'organisation topologique d'un ensemble de cellules NMOS ou CMOS réalisant des fonctions indépendantes (logique en bandes).

V - 1. Implantation de macro-cellules NMOS

Pour un ensemble de portes on aura l'organisation topologique de la figure 52.

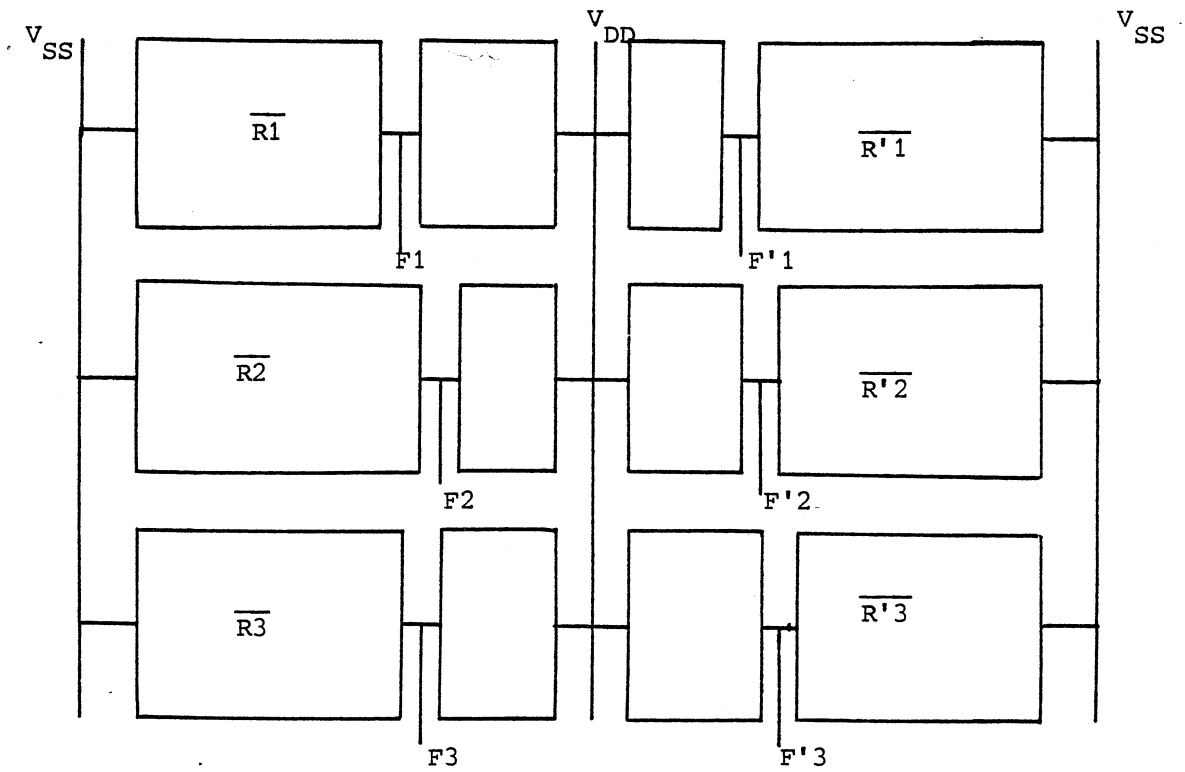


Figure 52 : Organisation topologique d'un ensemble de portes NMOS

L'optimisation d'un tel ensemble de portes suppose une partition des portes en deux sous- ensembles ; les portes appartenant à un même sous-ensemble doivent être optimisées suivant le même ordre lexicographique. Si cette partition est déjà réalisée, le problème d'optimisation de plusieurs portes est le même que celui d'une porte en considérant que l'ensemble des chemins initiaux est l'union des chemins.

Si la partition doit être déterminée, on peut étudier les ordonnancements optimaux de chaque porte et chercher à regrouper au mieux des chemins ayant même ordonnancement.

Exemple

Soient les quatre portes de partitions initiales données dans le tableau suivant on ne note pas le bloc où la variable n'apparaît sous aucune forme dans chaque partition.

variables / portes	x1	x2	x3	x4
porte 1 partition	(123)	()(235)	(12)	(12)(34)
gain	2	2	1	2
porte 2	(13)	(23)	()(345)	(15)(23)
	1	1	2	2
porte 3		(15)	(45)	(12)(45)
		1	1	2
porte 4	(135)		(35)	(34)
	2		1	1

Les ordonnancements optimaux pour chacune des portes sont :

porte 1	x1 x3 x4 ou x1 x4 x3		gain 4
porte 2	x4	x2	gain 3
porte 3	x4	x3	gain 3
porte 4	x1	x3	gain 3

Les portes 1 et 4 peuvent être regroupées avec l'ordonnancement x1x3 ...., leurs gains restent alors respectivement de 4 et 3.

Les portes 2 et 3 peuvent être regroupées avec l'ordonnnancement :

- soit x4x2, le gain de la porte reste de 3 tandis que celui de la porte 3 passe de 3 à 2
- soit x4x3 alors le gain de la porte 2 passe de 3 à 2.

Remarque : Si seule la première étape de l'optimisation est appliquée les portes auront une structure arborescente de forme lexicographique. Il sera alors intéressante de les placer tête-bêche (figure 53)

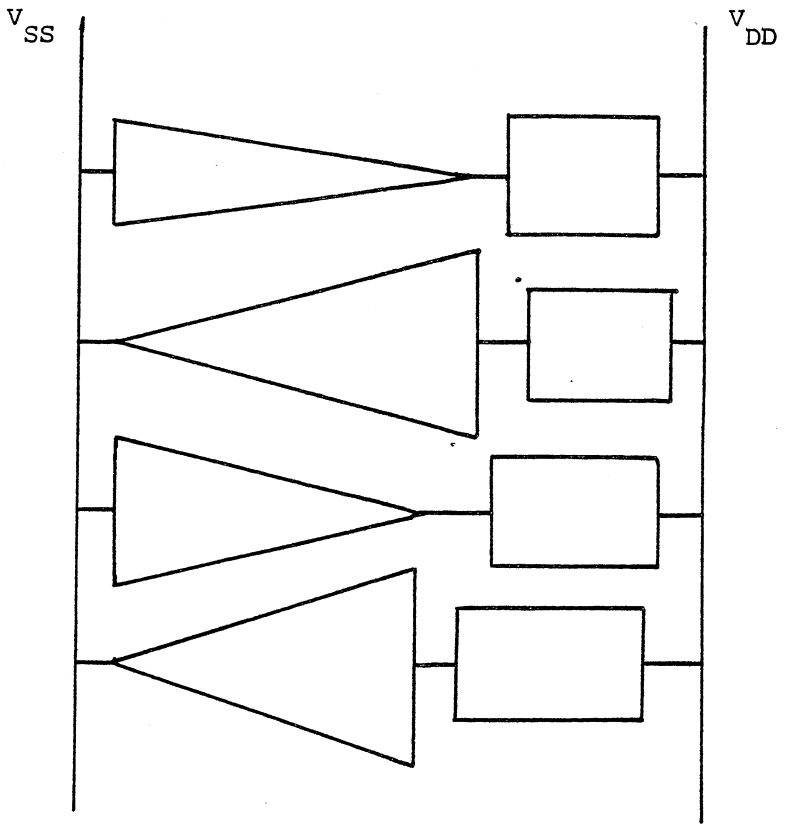


Figure 53 : Encastrement de portes de structure arborescente

Il s'agit alors de partitionner ces portes en deux sous-ensembles tel que l'ordre lexicographique d'un sous-ensemble soit l'inverse de l'ordre lexicographique du deuxième sous-ensemble. Dans un sous-ensemble, les fusionnements seront effectués à gauche et dans l'autre à droite.

Exemple

Si on reprend les quatre portes définies précédemment, on s'aperçoit que les portes 1,4 et les portes 2,3 peuvent être placées de façon tête bèche avec l'ordonnancement x1x3x2x4 (cf. figure 54)

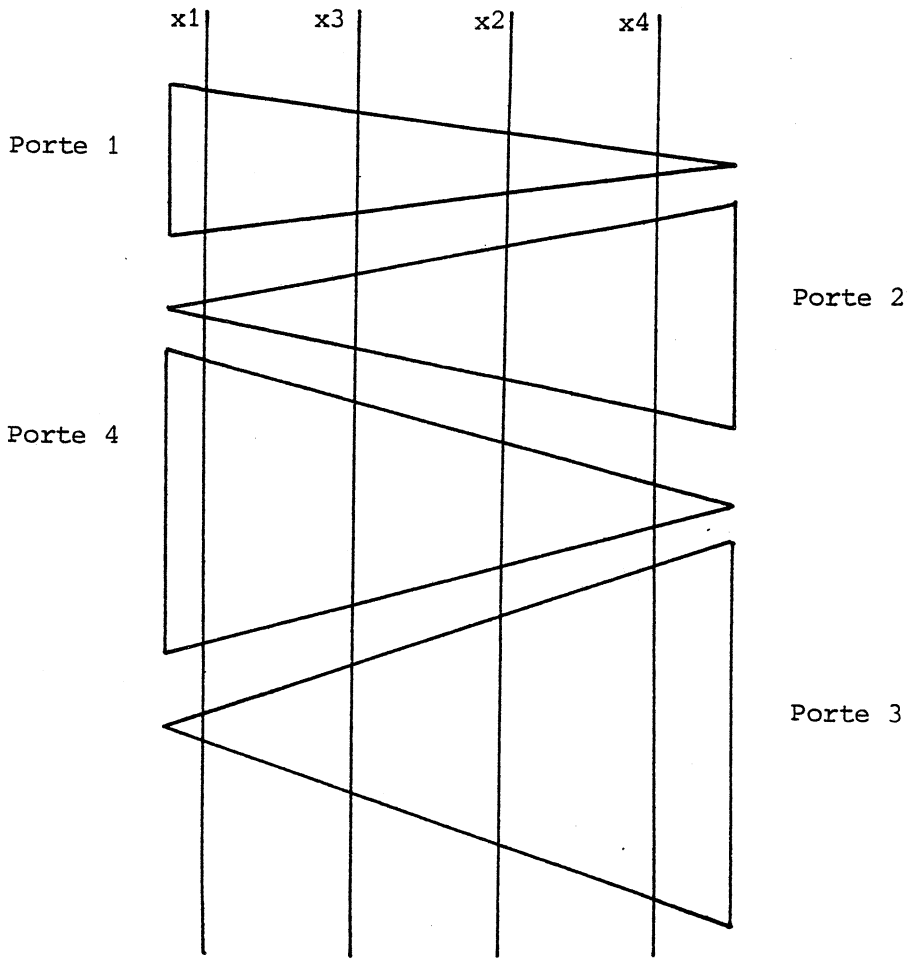


Figure 54 : Exemple d'encastrement de portes

V - 2. Implantation de macro-cellules CMOS

On peut distinguer deux organisations de base données dans les figures ci-dessous.

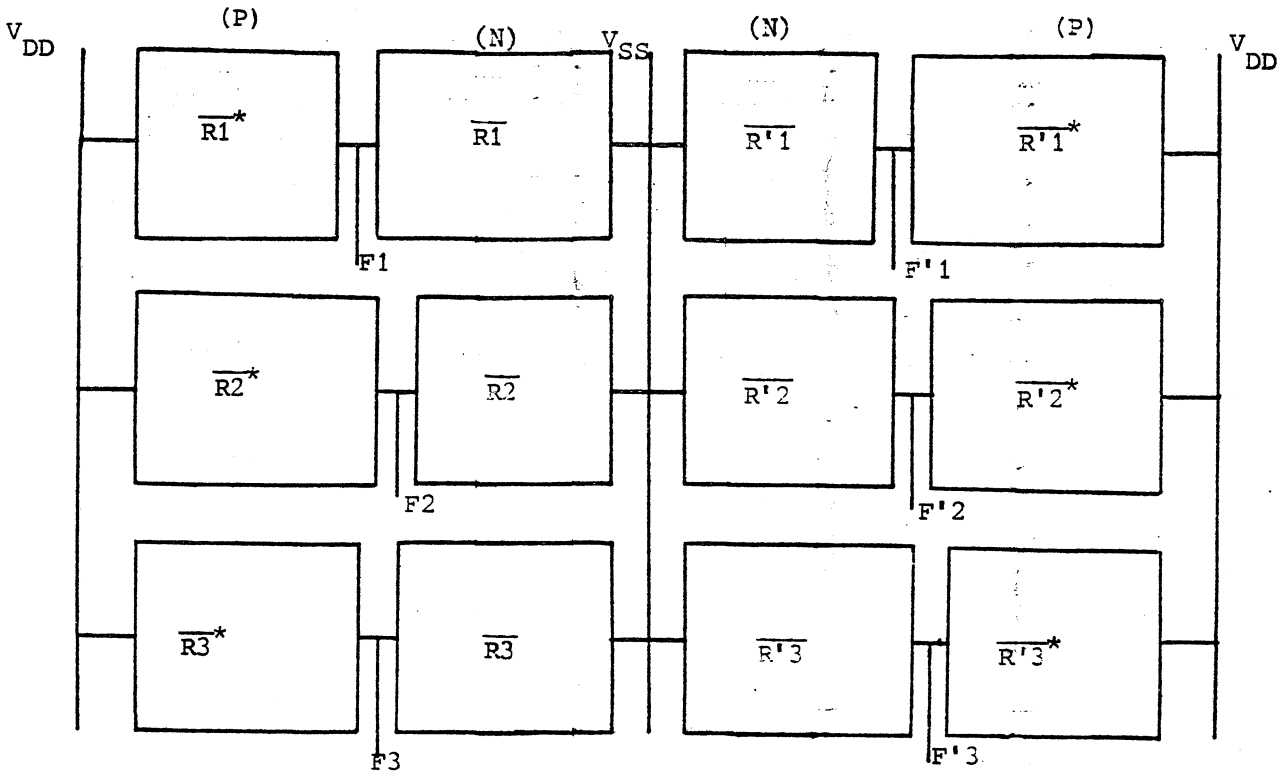


Figure 55a : 1ère organisation topologique pour un ensemble de portes CMOS

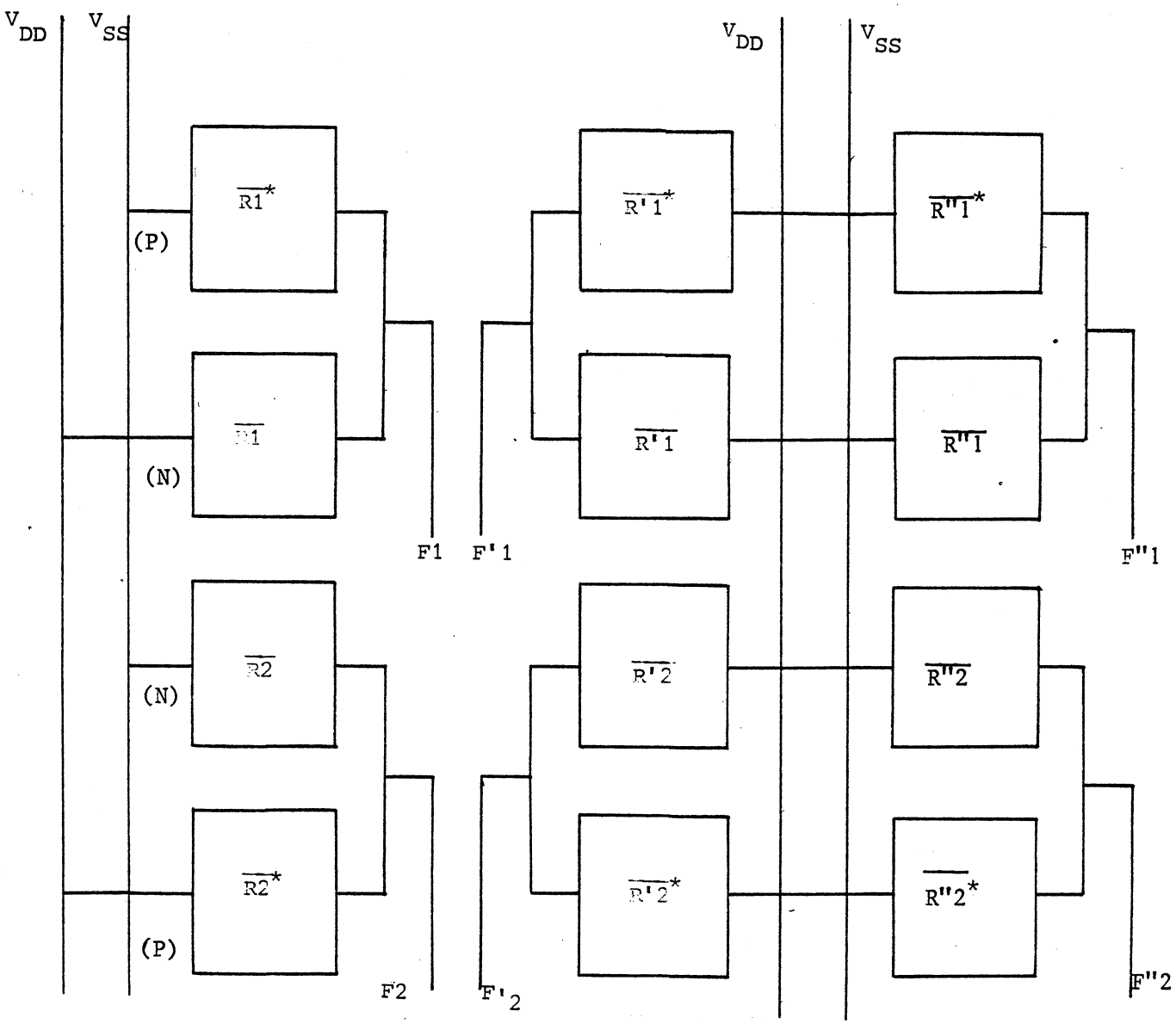


Figure 55b : 2ème organisation topologique pour un ensemble de portes CMOS

Dans le premier cas, le problème de synthèse consiste à trouver un ordre lexicographique optimisé pour les réseaux $\bar{R}1, \bar{R}2, \bar{R}3$ ... puis pour les réseaux $\bar{R}^*1, \bar{R}^*2, \bar{R}^*3$... Dans la deuxième solution, cet ordre optimisé doit être trouvé pour un ensemble ($\bar{R}1, \bar{R}^*1, \bar{R}2, \bar{R}^*2$....). Les réseaux duaux étant traités à partir d'une forme algébrique minimisée, il est difficile de définir laquelle des deux structures est la plus avantageuse. L'intérêt de la deuxième organisation réside dans le fait que les variables ne sont amenées qu'une seule fois pour les deux réseaux, les connexions sont donc moins nombreuses que dans la première organisation topologique.

VI - MACRO-CELLULES REALISANT DES FONCTIONS DEPENDANTES

Certaines fonctions seront réalisées à partir de plusieurs portes complexes. La décomposition d'une fonction booléenne est un problème très complexe. Dans (PER 67) la décomposition se fait à l'aide de matrices associées à ces fonctions. Ici, on étudiera le problème de la décomposition dans deux optiques différentes.

Le premier cas est celui d'une fonction dont la réalisation en une seule porte nécessite trop de transistors en série (§ III-1. et IV-1.). On décomposera alors la fonction en sous-fonctions, chacune réalisée par une porte.

Le deuxième cas est celui de la réalisation de plusieurs fonctions des mêmes variables à partir de portes complexes, on cherche alors des sous-fonctions communes menant à une bonne minimisation.

VI - 1. Composition de portes, générateurs de base

Soient deux portes complexes dont les réseaux associés ont comme fonction de transfert U et V, leur composition en série donne une porte de fonction $F = \overline{U\overline{V}}$, leur composition en parallèle donne une porte de fonction $F = \overline{U+\overline{V}}$.

Pour la technologie NMOS, on aura les schémas des figures 56a et 56b.

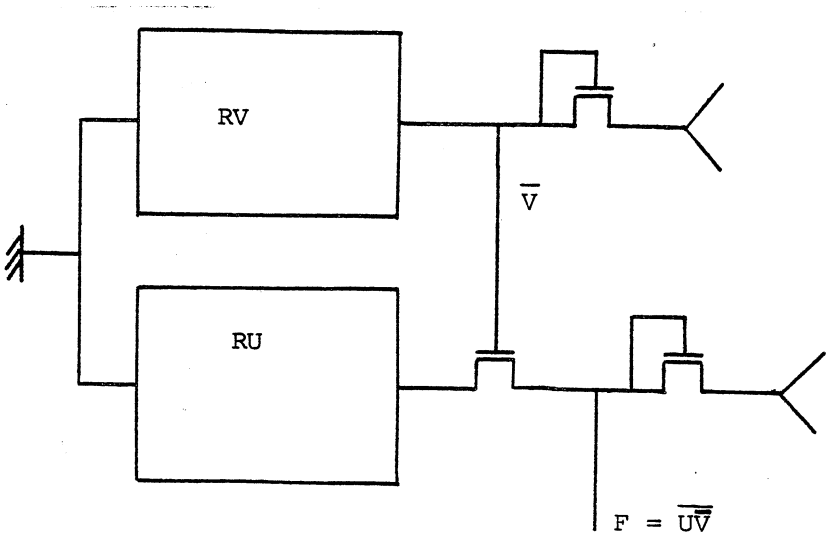


Figure 56a : $F = \overline{U\overline{V}}$

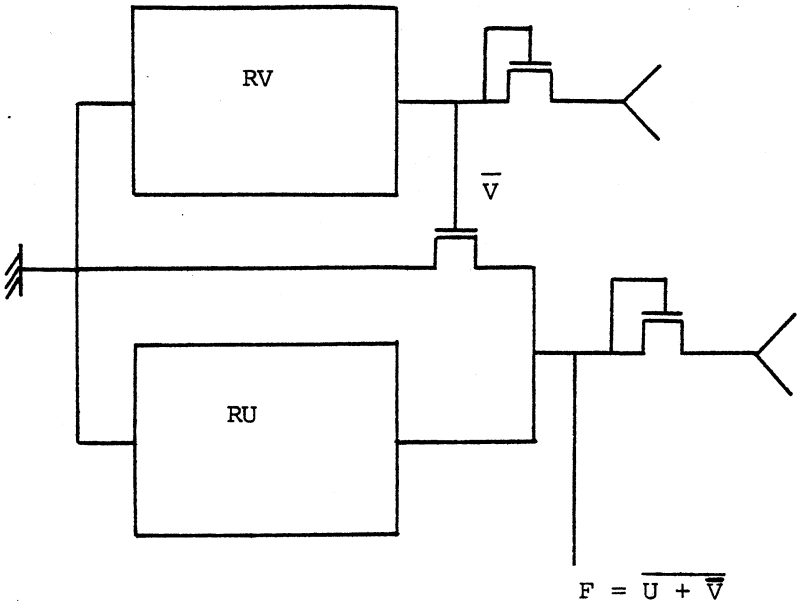


Figure 56b : $F = \overline{U+\overline{V}}$

Pour la technologie CMOS, on aura les schémas des figures 57a et 57b.

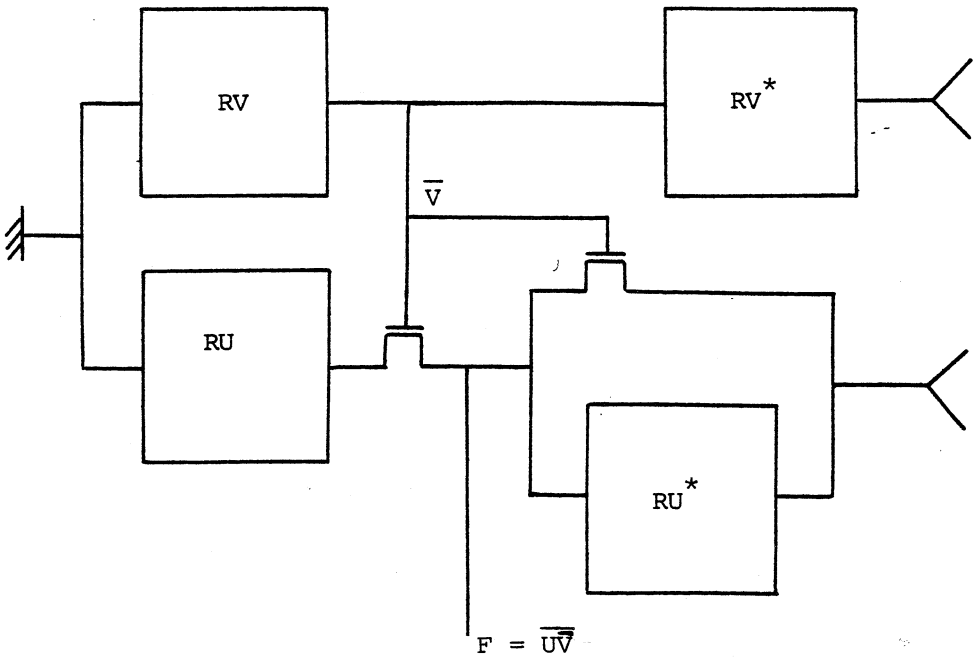


Figure 57a : $F = U\overline{V}$

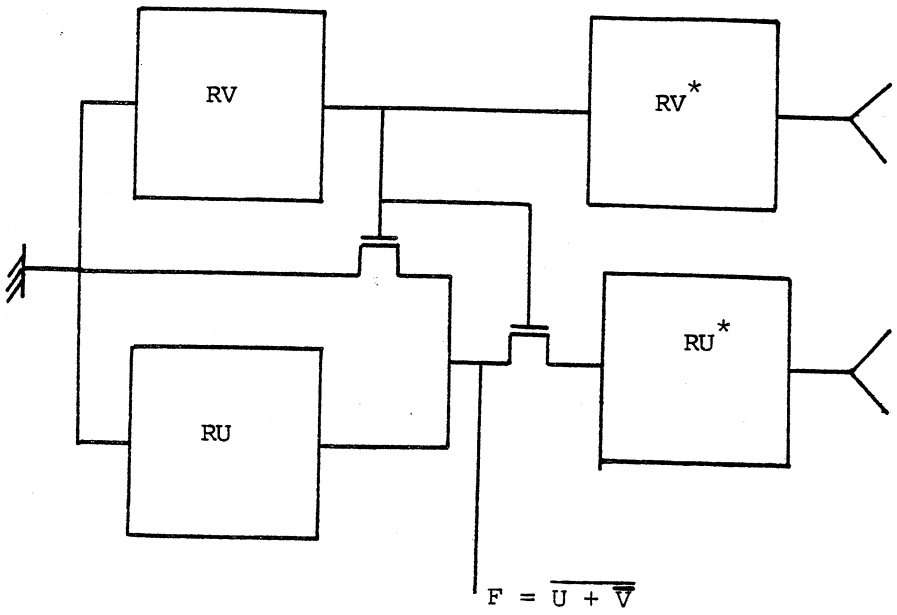


Figure 57b : $F = \overline{U+\overline{V}}$

De façon générale, la composition de plusieurs portes pourra se faire de façon série et parallèle avec l'expression suivante de la fonction :

$$F = \overline{U1\overline{V1} + U2\overline{V2} + ... + Un\overline{Vn}}$$

Si n est le nombre de fonctions U,V ; la réalisation de la fonction F se fera avec (2n + 1) portes.

Nous proposons de faire une étude topologique à partir du réseau $\overline{R}$ de la fonction F pour les technologies NMOS et CMOS. En CMOS on décomposera le réseau $\overline{R}$, puis on calculera le réseau dual $\overline{R}^*$.

VI - 2. Décomposition d'une porte trop complexe

Le nombre de transistors en série étant limité à quatre ou cinq une porte mettant en jeu des chemins de longueur supérieure sera décomposée en deux ou plusieurs portes.

Exemple

$\overline{F} = xyztu$

Ce produit pourra être décomposé en deux termes de la facon suivante :

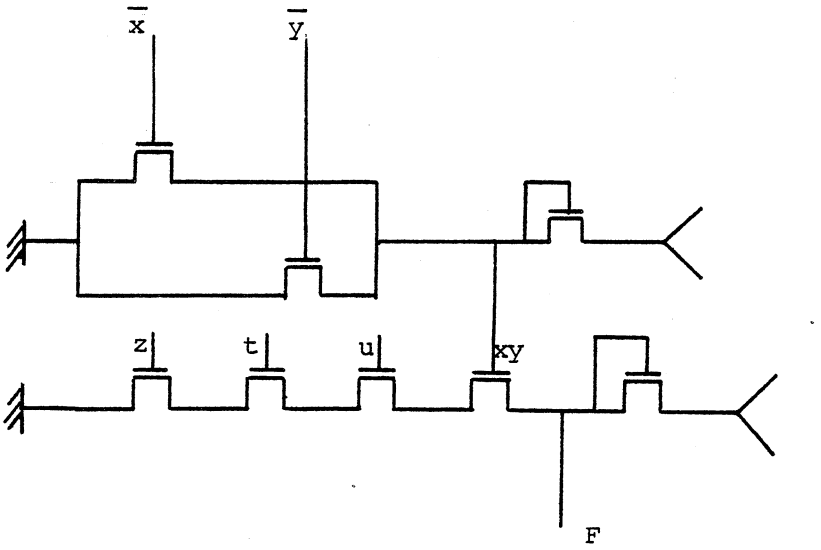


Figure 58 : Décomposition de $\overline{F} = xyz\ tu$

De façon générale, pour une fonction écrite sous forme de somme de produits on recherchera des mises en facteurs réduisant la longueur des chemins; la méthode est similaire à la méthode d'optimisation des portes complexes. On recherche les partitions initiales associées aux variables, puis on cherche les mises en facteur de termes maximaux.

Exemple

$\overline{F} = x\bar{y}\bar{z}tu + \bar{x}yztu + xyztv + xyztuv$

Le tableau suivant est associé à cette fonction :

variables	x	y	z	t	u	v
partitions	(14)(23)	(24)(13)	(34)(12)	(1234)	(124)	(34)

On en tire les mises en facteur de termes maximaux : xtu sur les chemins 14 ; ytu sur 24 ; $\bar{z}$tu sur 12 ; ztv sur 34.

En choisissant $\bar{z}$tu et ztv, la fonction F sera réalisée de la façon suivante (figure 59a) composition de portes sous la forme $F = U1\bar{V}1 + U2\bar{V}2$ avec $\bar{V}1$ = ztv et $\bar{V}2 = \bar{z}$tu

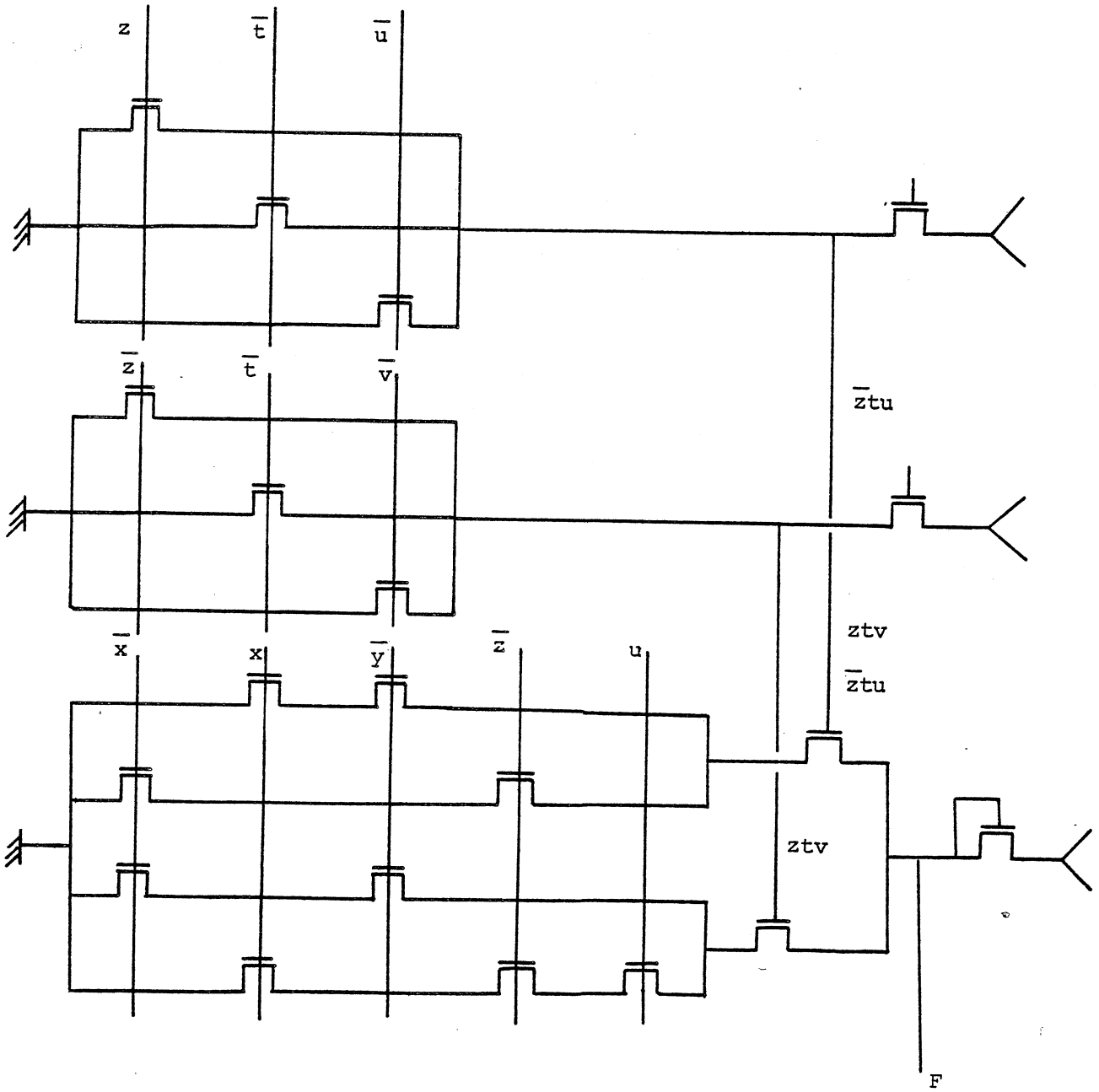


Figure 59a : Décomposition d'une fonction trop complexe

En décomposant une porte trop complexe de cette façon, on obtient la fonction sous la forme suivante $\overline{F} = U1\overline{V1} + U2\overline{V2} + ... UN\overline{VN}$ où toutes les fonctions $\overline{Vi}$ sont des produits de variables (Vi : sommes de variables) et les fonctions Ui sont des sommes de produits de variables.

On recherche alors un ordre lexicographique optimal pour l'ensemble des réseaux Ui ; les transistors des réseaux Vi respectant cet ordre.

Sur l'exemple précédent, on n'a pas de fusionnements possibles sur les réseaux $\overline{V1}$ et $\overline{V2}$ séparément, on pourra par contre fusionner $\overline{x}$ par exemple : les chemins introduits sont nuls (cf. figure 59b).

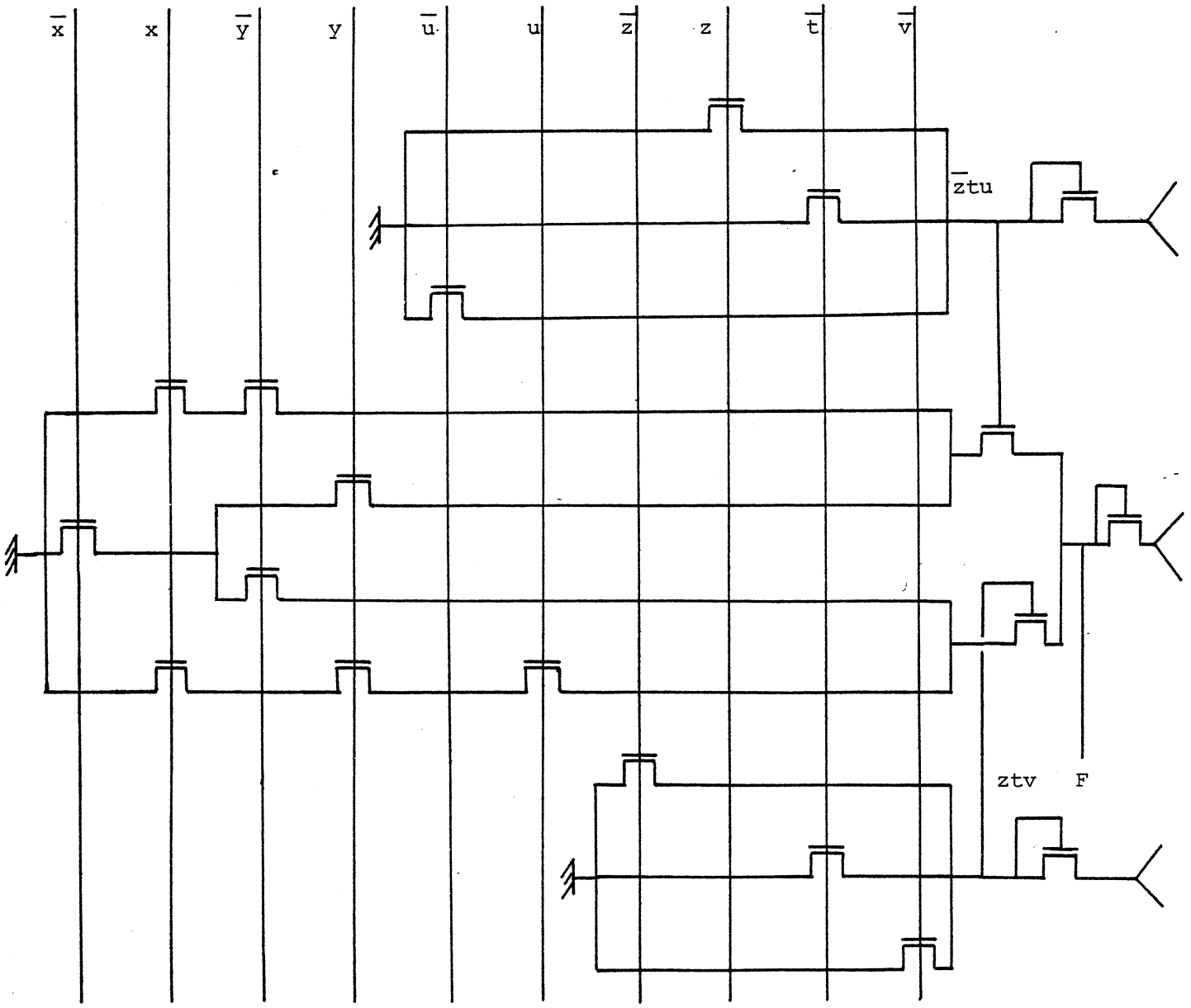


Figure 59b : Réseau final

Cette façon d'implanter amène à nouveau à une forme triangulaire propre à l'encastrement tête-bêche.

VI - 3. Réalisation de plusieurs fonctions des mêmes variables

6.3.1. Recherche de monômes communs à plusieurs fonctions

Pour réaliser plusieurs fonctions des mêmes variables, il peut être intéressant de rechercher des monômes communs. Ces monômes peuvent soit apparaître dans l'expression initiale des fonctions, soit sont inclus dans les fonctions.

Exemple 1

Soient les fonctions :

$\overline{F1} = ab + cd$
$\overline{F2} = ab + \bar{a}\bar{c}\bar{d}$

Le monôme ab est commun aux deux fonctions. On réalisera F1 et F2 avec une porte commune réalisant ab (figure 60)

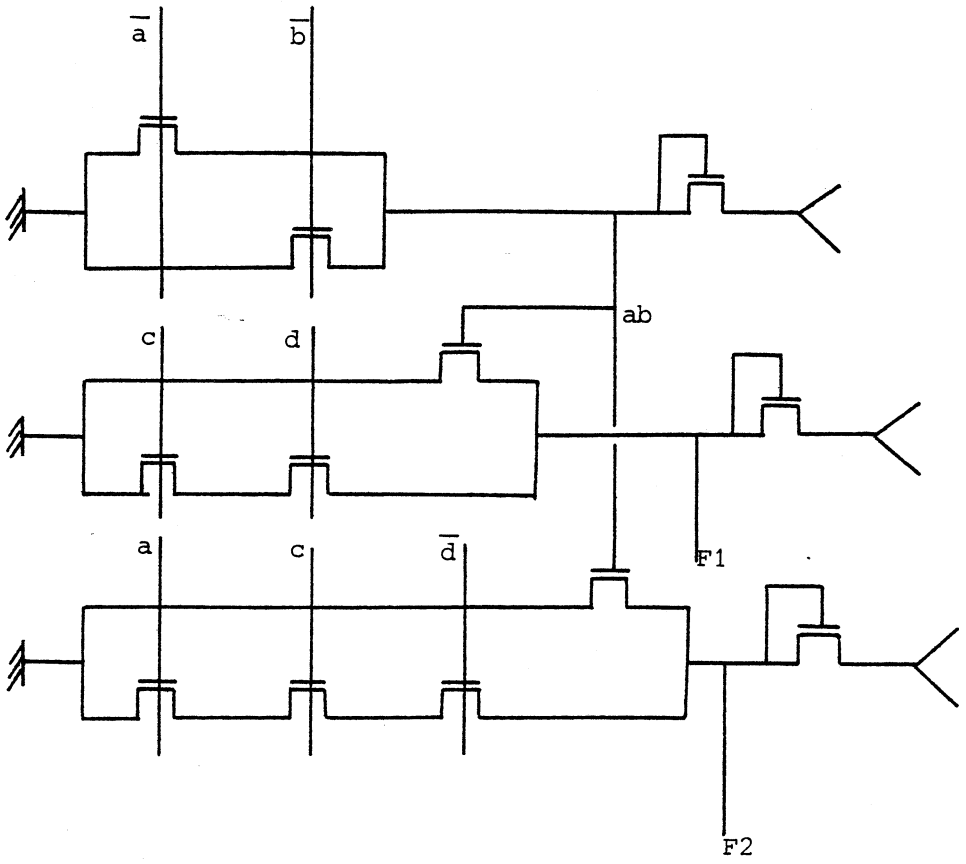


Figure 60 : Deux fonctions ayant un monôme commun

On obtient les fonctions Fi sous la forme parallèle : $\overline{U + \overline{V}}$. Dans l'exemple traité : $\overline{V}$ = ab

Exemple 2

Soient les fonctions $\overline{F1}$ = abc + d et $\overline{F2}$ = abd + e. Le produit de variables ab est commun aux deux fonctions. On réalisera F1 et F2 avec une porte commune réalisant ab (figure 61)

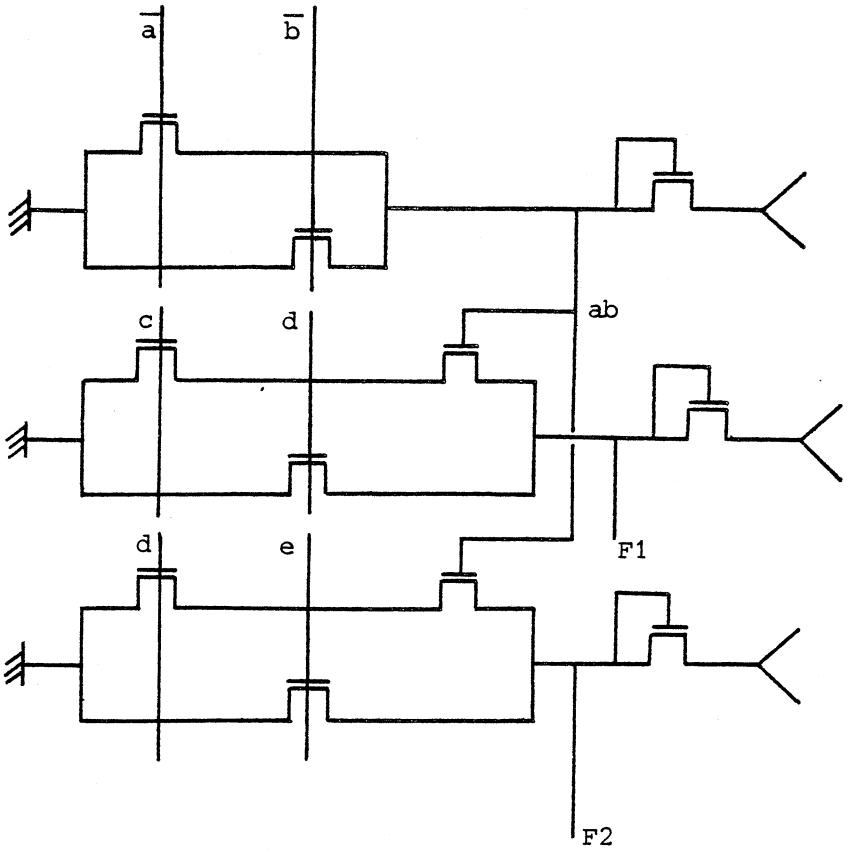


Figure 61 : Deux fonctions ayant un produit de variables commun

Ici, on obtient les fonctions Fi sous la forme $\overline{U1\overline{V1} + U2}$. Dans l'exemple traité $\overline{V1}$ = ab.

L'extraction des parties de monômes communes est triviale, il suffit de réaliser les intersections monôme par monôme entre les fonctions.

Cette recherche de monômes ou de produits de variables communs n'est pas toujours intéressante. Il convient de se demander quand il y aura gain de transistors. Soit un produit de n variables commun à m fonctions. Si la réalisation des m fonctions par décomposition en produit de variables commun nécessite plus de transistors que la réalisation de ces m fonctions de façon indépendante, elle n'aura pas d'intérêt.

La réalisation des m fonctions de façon indépendante nécessite :

(K + m.n) transistors

La réalisation des m fonctions par décomposition de fonctions nécessite :

(n + 1) transistors pour fabriquer le produit de variables commun
(K + m) transistors pour les fonctions.

La décomposition est intéressante si :

$$K + m + n + 1 < K + n\,m$$

$$\text{d'où } n > \frac{m + 1}{m - 1}$$

ce qui donne le tableau suivant :

nombre de fonctions m	nombre minimum de variables dans le produit commun n
2	4
3	3
4	2
5	2

Remarque : On peut réaliser un monôme de n variables d'une fonction par la somme de m monômes d'autres foncions (minimisation effectuée habituellement sur les PLAs) :

Exemple

Soient les monômes : abc, $ab\bar{c}$, ab

Le monôme ab = abc + $ab\bar{c}$

La réalisation des m monômes (avec leur intervention dans les fonctions) nécessite :

K transistors de façon indépendante
(K + 2m) transistors de façon globale

Celle du monôme de n variables :

n transistors de façon indépendante
m transistors de façon globale.

Cette décomposition est intéressante si :

$K + 2m + m < K + n$

d'où $n \geqslant 3m$

Ce qui donne n = 7 variables pour m = 2 monômes, ce n'est donc jamais intéressant puisqu'on se limite à 4 ou 5 variables.

Dans ce cas, on réaliserait le monôme de n variables à part, ce produit de variables est commun aux différentes fonctions (cf. précédemment).

Dans l'exemple précédent, on réaliserait le produit de variables commun aux trois monômes : ab, abc, $ab\bar{c}$.

6.3.2. Sous fonctions communes à plusieurs foncions

Soient les fonctions a réaliser : F1,F2,...,Fn. On exprime ces foncions à partir de foncions communes. Les fonctions communes peuvent être des fonctions simples telles que NAND, NOR ou des foncions plus complexes.

Exemple

Soit un additionneur un bit, les équations de la sortie Si et de la retenue sortante Cout fonction des deux entiers ai, bi et de la retenue entranteCin peuvent s'écrire sous la forme :

$$\overline{Si} = (\overline{ai} \oplus \overline{bi}) \oplus \overline{Cin}$$

$$\overline{Cout} = \overline{ai}\ \overline{bi} + \overline{Cin}\ (\overline{ai} \oplus \overline{bi})$$

La fonction $si = \overline{ai} \oplus \overline{bi}$ est commune aux deux fonctions à réaliser. On réalisera $\overline{Si}$ et $\overline{Cout}$ avec une porte commune réalisant $\overline{ai} \oplus \overline{bi}$ (figure 62)

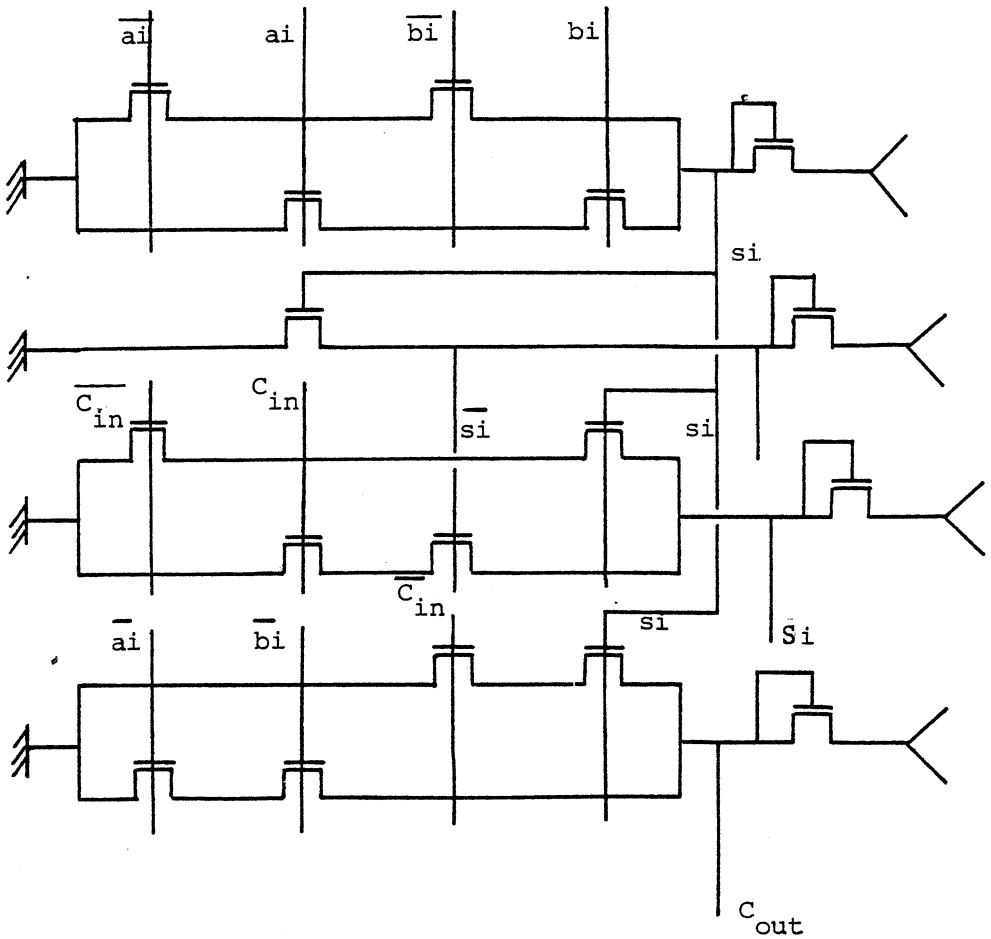


Figure 62 : Deux fonctions ayant une fonctions commune

Cas particulier : plusieurs fonctions réalisées à partir de l'une d'entre elles.

Soient les fonctions à réaliser F1, F2, ..., Fn. Dans certains cas, on peut exprimer ces fonctions à partir de l'une d'entre elles.

Exemple

Soit l'additionneur un bit de l'exemple précédent, les équations de Si et Cout fonction de ai, bi, Cin peuvent s'écrire sous la forme :

$Cout = ai\ bi + ai\ Cin + bi\ Cin$

$Si = ai\ \overline{Cout} + bi\ \overline{Cout} + Cin\ \overline{Cout} + ai\ bi\ Cin$ (composition de portes sous la forme $\overline{Fi} = U1\overline{V1} + U2$ avec $V1 = Cout$)

On réalisera Si et Cout avec une porte commune réalisant $\overline{Cout} = s1$ et une porte complexe réalisant $\overline{Si} = s2$ à partir de $\overline{Cout}$ (figure 63a). On trouve cette même réalisation de l'additionneur dans (MAN 80) avec une technologie à transistors de charge enrichis.

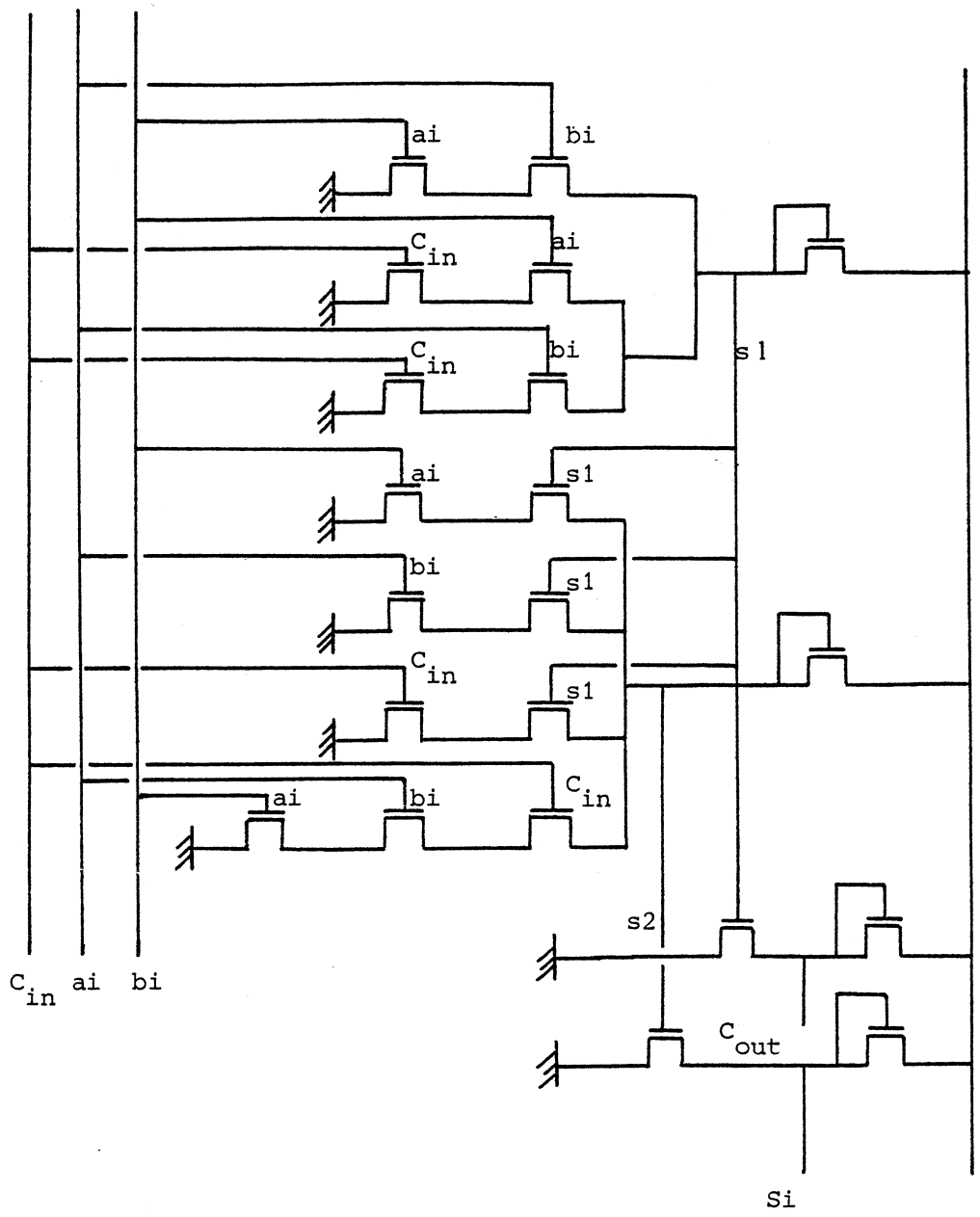


Figure 63a : Une fonction exprimée à poartir d'une autre fonction

Nous continuons à nous intéresser ici à une réalisation lexicographique. L'étude conjuguée des deux fonctions Cout = ai bi + ai Cin + bi Cin

$$Si = \underset{4}{ai\ \overline{Cout}} + \underset{5}{bi\ \overline{Cout}} + \underset{6}{Cin\ \overline{Cout}} + \underset{7}{ai\ bi\ Cin}$$

donne le tableau initial :

		ai	bi	Cin	Cout
Cout	TTi	(12)	(13)	(23)	
	gain	1	1	1	
	TTi	(47)	(57)	(67)	()(467)
Si	gain	1	1	1	2

On en déduit trois ordonnancements possibles :

Cout --- ai
ou Cout --- bi
ou Cout --- Cin

La 2ème phase de l'optimisation introduit des chemins parasites. Aucune autre simplification n'est donc possible.

On obtient donc le réseau optimal (figure 63b)

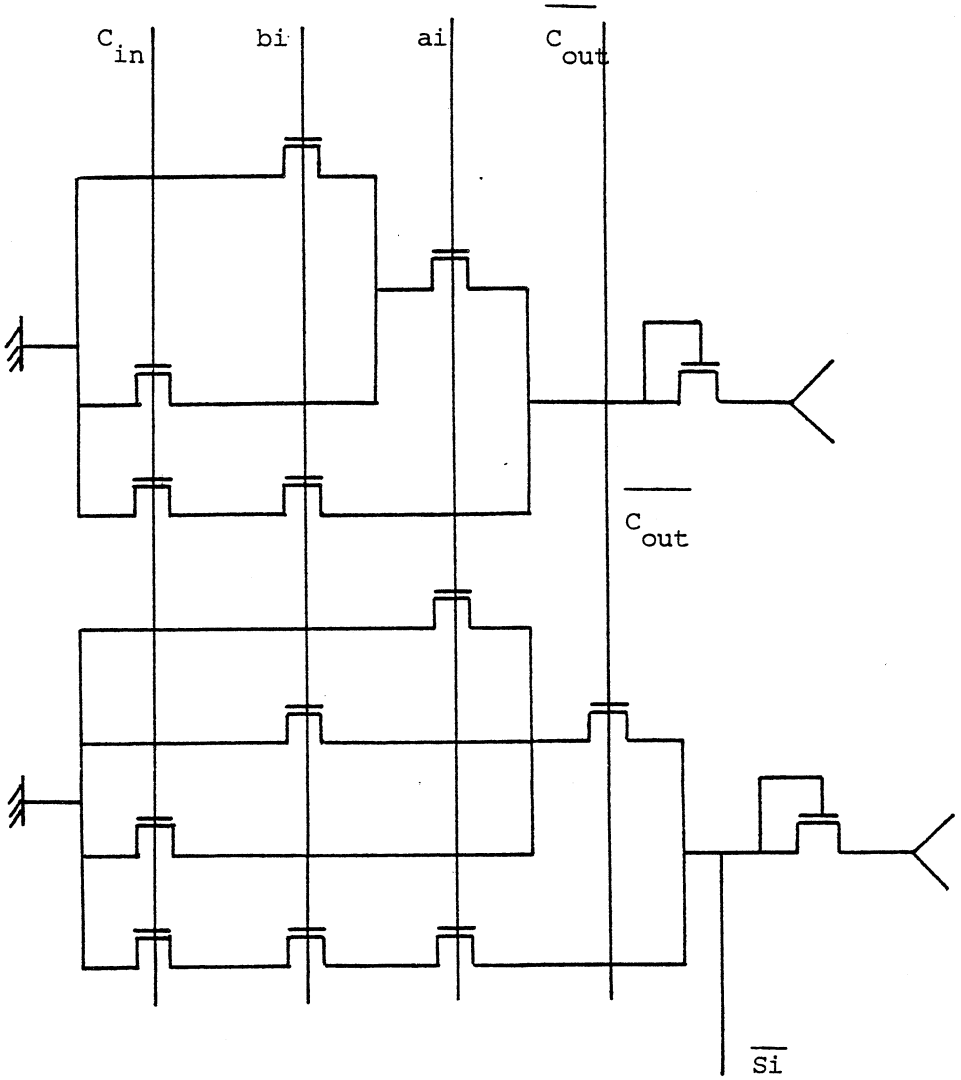


Figure 63b : Portes complexes réalisant les fonctions Cout et Si avec des réseaux optimaux

VI - 4. Implantation de macro-cellules réalisant des fonctions dépendantes

Pour les macro-cellules réalisées précédemment, la structure en bande obtenue est faite d'une ensemble de cellules et d'un "canal" d'interconnexions (figure 64).

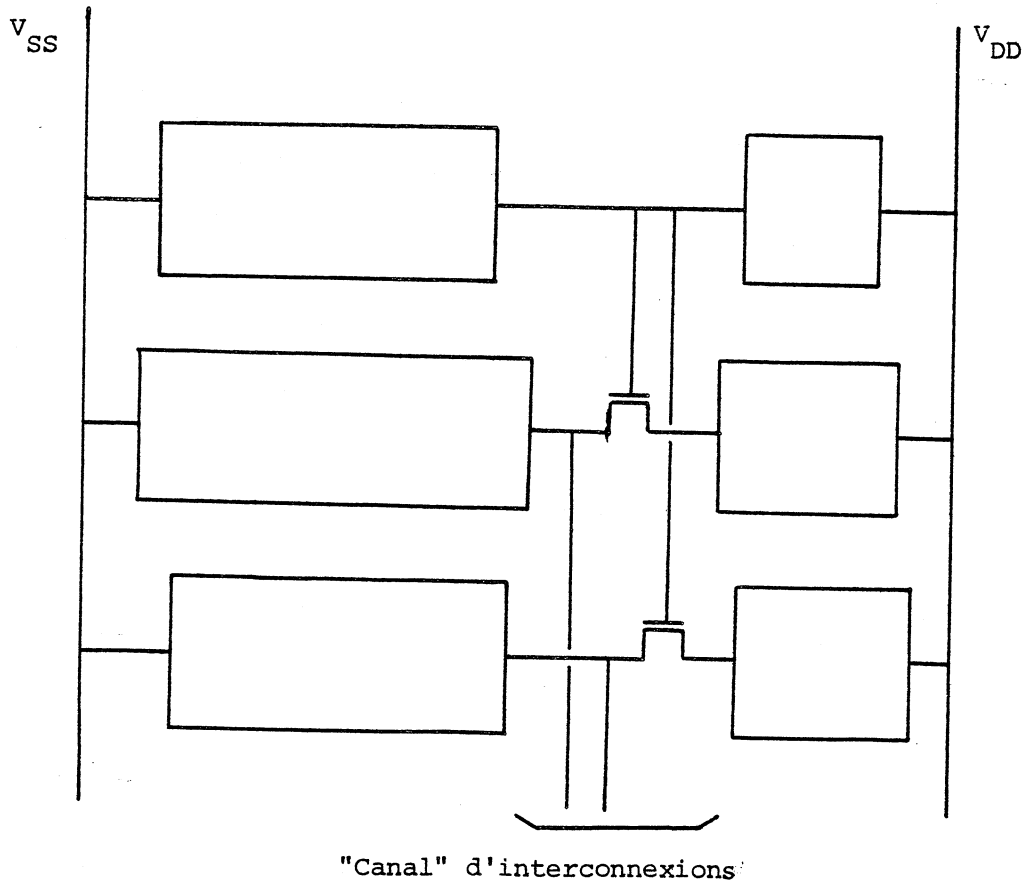


Figure 64

Pour minimiser la zone d'interconnexions, on utilisera les méthodes classiques d'échange : échange par paires par exemple dans OASIS (MAL 81).

CONCLUSION

Les cellules MOS bien structurées et optimisées et les macro-cellules présentées ici sont caractérisées par une optimisation logique puissante. L'implantation sur une grille virtuelle suivi d'un compactage conduit à des macro-cellules à structure très régulière. Cette solution évite la synthèse à deux couches NOR/NOR utilisée dans les PLAs et semble être une excellente alternative.

CHAPITRE II

LOGIQUE DE TRANSFERT

INTRODUCTION

Dans ce chapitre, on étudie la logique de transfert dans la technologie NMOS. Les problèmes de synthèse logique et topologique sont considérés simultanément, en utilisant une spécification initiale sous forme d'arbre de décision binaire.

La représentation d'une fonction booléenne sous forme d'arbre de décision binaire nous permet d'effectuer une miminisation globale, d'obtenir une implantation facile de façon automatique.

I - DEFINITION DE LA LOGIQUE DE TRANSFERT

I - 1. Principe de base

On appelle logique de transfert, la réalisation d'une fonction booléenne, suivant le schéma de la figure 1. Un exemple est donné sur la figure 2.

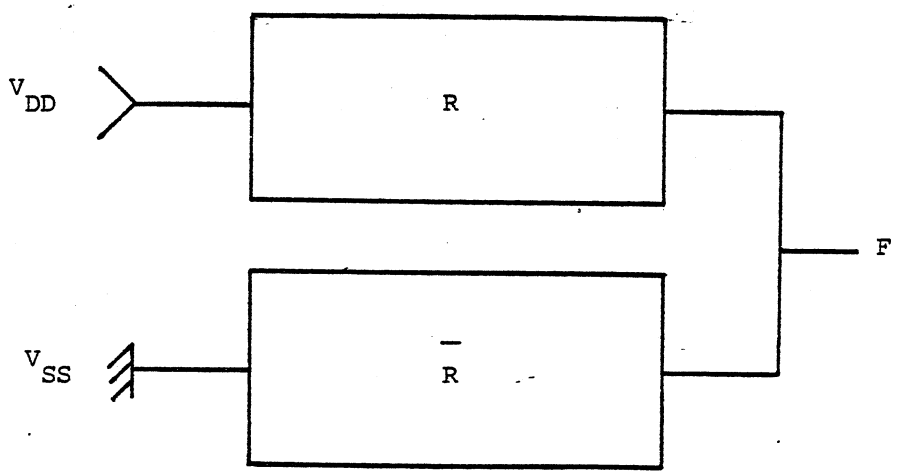


Figure 1 : Principe de réalisation d'une fonction en logique de transfert

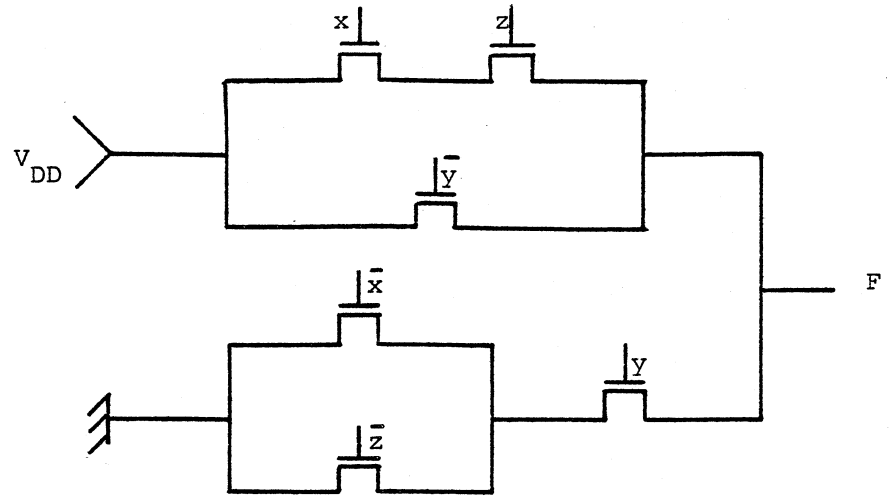


Figure 2 : Réalisation de $F = xz + \bar{y}$ en logique de transfert

Une fonction est réalisée en reliant la sortie F à 1 (alimentation VDD) par le réseau $\bar{R}$ et à 0 (masse VSS) par le réseau R. Les réseaux R et $\bar{R}$ sont des réseaux d'interrupteurs où l'interrupteur est le transistor NMOS, cet élément de base a été défini au § I.1 du chapitre I

L'étude de la logique de transfert se ramène à l'étude de réseaux d'interrupteurs et des fonctions de transfert associées comme on les a définis au § I.2 du chapitre I. On étudiera aussi une forme arborescente de la logique de transfert où les réseaux R et $\bar{R}$ sont imbriqués.

Deux conditions sont nécessaires et suffisantes pour réaliser une fonction F en logique de transfert (de même que pour réaliser une porte CMOS § IV-1. du chapitre I).

Condition C1 : On ne doit jamais avoir un chemin direct entre VDD et VSS (il suffit que sur chaque chemin reliant VDD à VSS, la même variable apparaisse sous forme directe dans un des réseaux et sous forme complémentée sur l'autre).

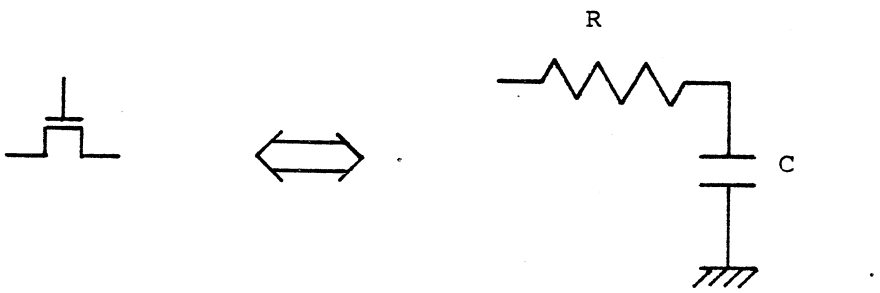
Condition C2 : On doit toujours avoir un chemin reliant la sortie soit à VDD, soit à VSS (il faut et il suffit que la somme logique des chemins reliant la sortie F à VDD ou VSS soit égale à 1).

De part leurs définitions, les réseaux R et $\overline{R}$ vérifent ces deux conditions.

I - 2. Caractéristiques électriques et topologiques

On étudie des réseaux d'interrupteurs où l'interrupteur est le transistor NMOS. Les conventions d'ouverture et de fermeture d'un tel transistor sont données au § I.1 du chapitre I.

Le transistor MOS utilisé en interrupteur (pass transitor) est l'équivalent d'une cellule RC, au niveau temps de propagation :



Lorsqu'on a des transistors en série, on doit couper la ligne RC pour minimiser les retards. On mettra quatre à cinq transistors en série suivis de un ou deux inverseurs.

Le transistor NMOS utilisé en interrupteur dégrade le niveau logique "1" d'une tension de seuil, le niveau logique "0" est transmis correctement. De plus, la transmission du "1" est plus lente que celle du "0" (la résistance RON est plus grande). La dégradation du niveau logique "1" est la même pour un ou plusieurs transistors en série (cf. § I.1 du chapitre I).

Il est donc nécessaire de régénérer le signal par une porte lorsqu'on réalise les deux réseaux R et $\overline{R}$ en technologie NMOS (figure 3) quel que soit le nombre de transistors en série dans les réseaux (même pour un transistor). Les inverseurs précédents qui servent à couper la ligne RC permettent aussi de récupérer un niveau logique correct.

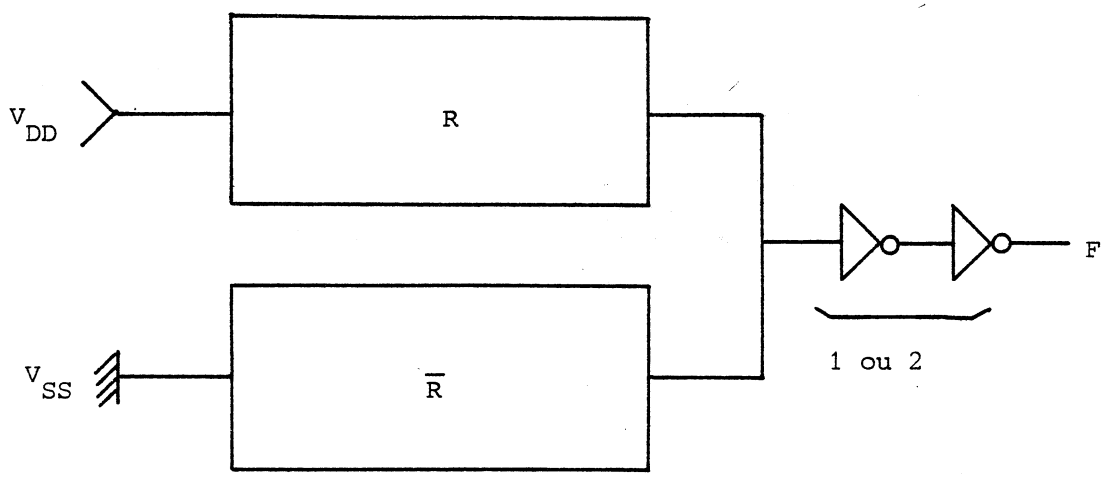
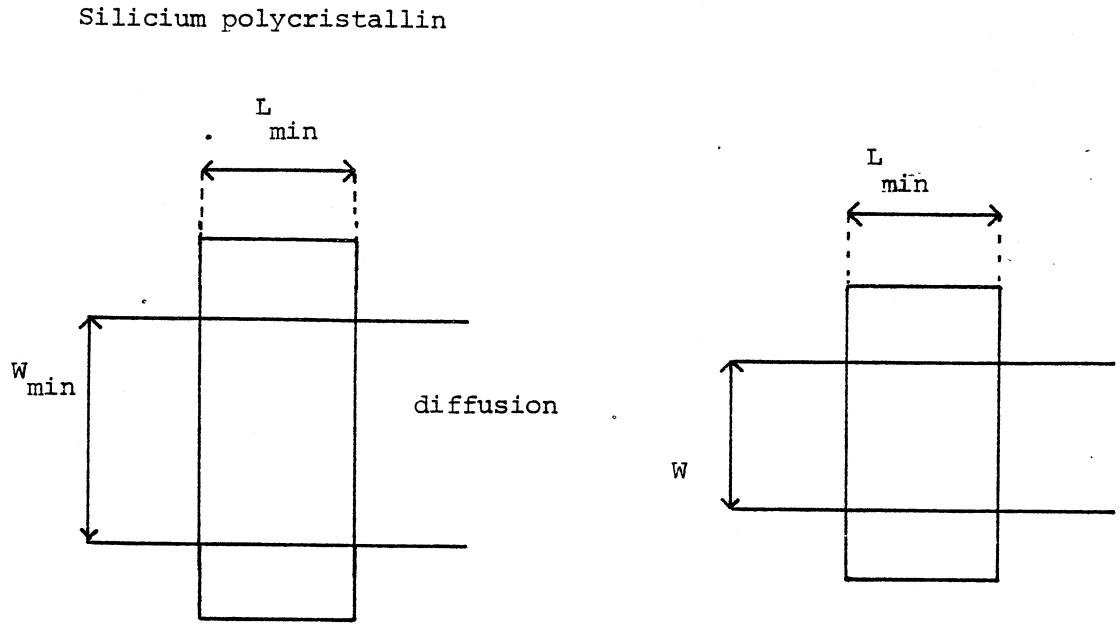


Figure 3: Fonction réalisée en logique de transfert

Le dimensionnement des transistors se fera par rapport au temps de traversée (temps de charge et décharge) qui dépend du nombre de transistors traversés. On cherche à minimiser les capacités de grille et les résistances de diffusion. Si la largeur de la diffusion augmente (W), avec une longueur de grille minimum (Lmin), la résistance diminue alors que la capacité augmente dans le même rapport :



Les transistors seront des transistors minimaux : Wmin = Lmin (MEA 80)

- L'avantage de la logique de transfert par rapport aux portes complexes au niveau électrique et topologique est d'avoir un dimensionnement simplifié : transistors de taille fixe.

II - FORME CANONIQUE DE LA LOGIQUE DE TRANSFERT

II - 1. Définition

La forme canonique de la logique de transfert consiste à implémenter les réseaux R et $\overline{R}$ à partir des expressions canoniques de F et $\overline{F}$ sous forme de somme de monômes. C'est la solution la plus simple, mais non optimisée (nombre de transistors non minimisé).

Exemple

Soit une fonction à 3 variables x, y, et z définie par la table de vérité de la figure 4.

x	y	z	F
0	0	0	F(0,0,0) = 0
0	0	1	F(0,0,1) = 1
0	1	0	F(0,1,0) = 0
0	1	1	F(0,1,1) = 0
1	0	0	F(1,0,0) = 0
1	0	1	F(1,0,1) = 1
1	1	0	F(1,1,0) = 1
1	1	1	F(1,1,1) = 1

Figure 4 : Table de vérité d'une fonction à 3 variables

Les expressions canoniques de F et $\bar{F}$ sont les suivantes :

$$F = \bar{x}\bar{y}z + \bar{x}y\bar{z} + x\bar{y}\bar{z} + xyz$$
$$\bar{F} = \bar{x}\bar{y}\bar{z} + \bar{x}\bar{y}z + \bar{x}y\bar{z} + x\bar{y}z$$

II - 2. Réalisation systématique de la forme canonique

La forme canonique de la logique de transfert conduit à un circuit universel et programmable. On a directement un réseau lexicographique (défini au § I.7 du chapitre I) en immergeant le tableau de valeurs sur un quadrillage fictif où les variables d'entrée déterminent les lignes verticales (colonnes), les lignes horizontales (chemins) correspondent aux différents monômes. Ce réseau est facilement implantable : il n'y a pas de croisements entre deux lignes de même type ; on a un ordonnancement lexicographique de variables, une variable et sa variable complémentée étant adjacentes. Si n est le nombre de variables, le circuit a 2n entrées (colonnes) et 2^n chemins. Une telle solution est donnée en (MEA 80) page 77.

Exemple

Soit l'exemple de la figure 4, sa réalisation donne le schéma électrique de la figure 5, réseau lexicographique avec par exemple l'ordonnancement des variables x, y, z.

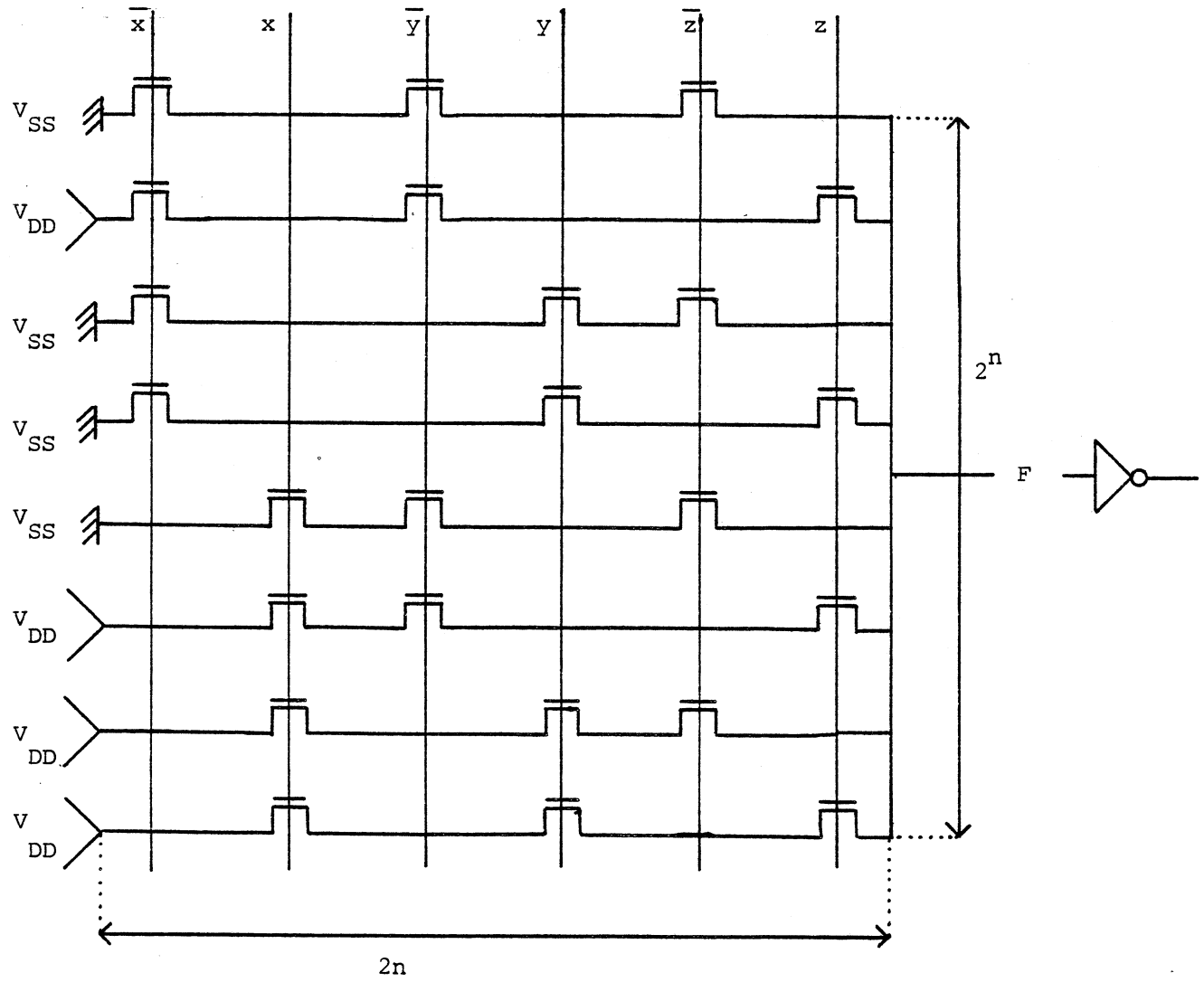


Figure 5 : Schéma d'une fonction à 3 variables sous forme canonique

Remarque : Les variables traversent le réseau (colonnes), elles peuvent donc arriver par le haut ou le bas de celui-ci. L'ordre des variables ne modifie pas le nombre de transistors du réseau pour la forme canonique de la logique de transfert. On choisira donc l'ordonnancement et le point d'entrée des variables en fonction des connexions du réseau avec le reste du circuit.

La réalisation précédente permet d'obtenir une réalisation dans un rectangle. Si n est le nombre de variables, on a 2^n chemins, 2 n colonnes et n x 2^n transistors.

Les variables d'entrée peuvent arriver des deux côtés (en haut ou en bas), de plus leur ordonnancement peut être modifié à volonté, ceci permet de tenir compte de la connectique.

III - FORME ARBORESCENTE DE LA LOGIQUE DE TRANSFERT

III - 1. Intérêt de la forme arborescente

3.1.1. Position du problème

La forme canonique de la logique de transfert conduit à un réseau lexicographique non minimisé. On recherche une minimisation globale de R et $\bar{R}$ ainsi qu'un ordonnancement des variables d'entrée pour avoir un réseau facilement implantable (réseau lexicographique). Pour cela, on part de l'arbre de décision binaire comme représentation booléenne de F.

3.1.2. Bibliographie

L'arbre de décision binaire a largement été étudié, ses domaines d'application sont très vastes.

On trouvera dans (KUN 68), une théorie générale des arbres lexicographiques englobant l'arbre communément appelé arbre de décision binaire.

S.B AKERS s'est intéressé essentiellement au problème du test de dispositifs digitaux (AKE 78). L'arbre de décision binaire est utilisé pour modéliser les fonctions des circuits et pour générer des ensembles de valeurs de test pour ces circuits.

La formalisation et les propriétés des arbres de décision binaires ont été étudiées en profondeur dans (DAV 83). Les arbres sont utilisés ici pour construire des programmes de décision binaires optimaux d'évaluation de fonctions booléennes. Cette étude est orientée vers des algorithmes permettant d'obtenir des arbres de décisions binaires optimaux avec les critères suivants:

nombre de noeuds minimal (lié à l'occupation mémoire), nombre d'arcs entre le noeud initial et final minimal lié au temps de calcul) pour une fonction booléenne donnée (ou un ensemble de fonctions booléennes).

Dans (MAN 78), les arbres de décisions binaires sont utilisés pour réaliser des systèmes logiques câblés ou programmés. Un arbre de décision binaire peut représenté un système combinatoire : réseau de multiplexeurs à 2 variables d'entrées, 1 variable de sélection (SAN 78), réseau de relais (câblage en allant des feuilles vers la racine de l'arbre) ou réseau de démultiplexeurs à 2 sorties, 1 variable de sélection (câblage en allant de la racine vers les feuilles de l'arbre) (CER 79). Un système séquentiel est également modélisé comme une machine de décision binaire (MAN 78).

Exemple extrait de (CER 79)

Soit l'arbre de décision binaire (figure 6)

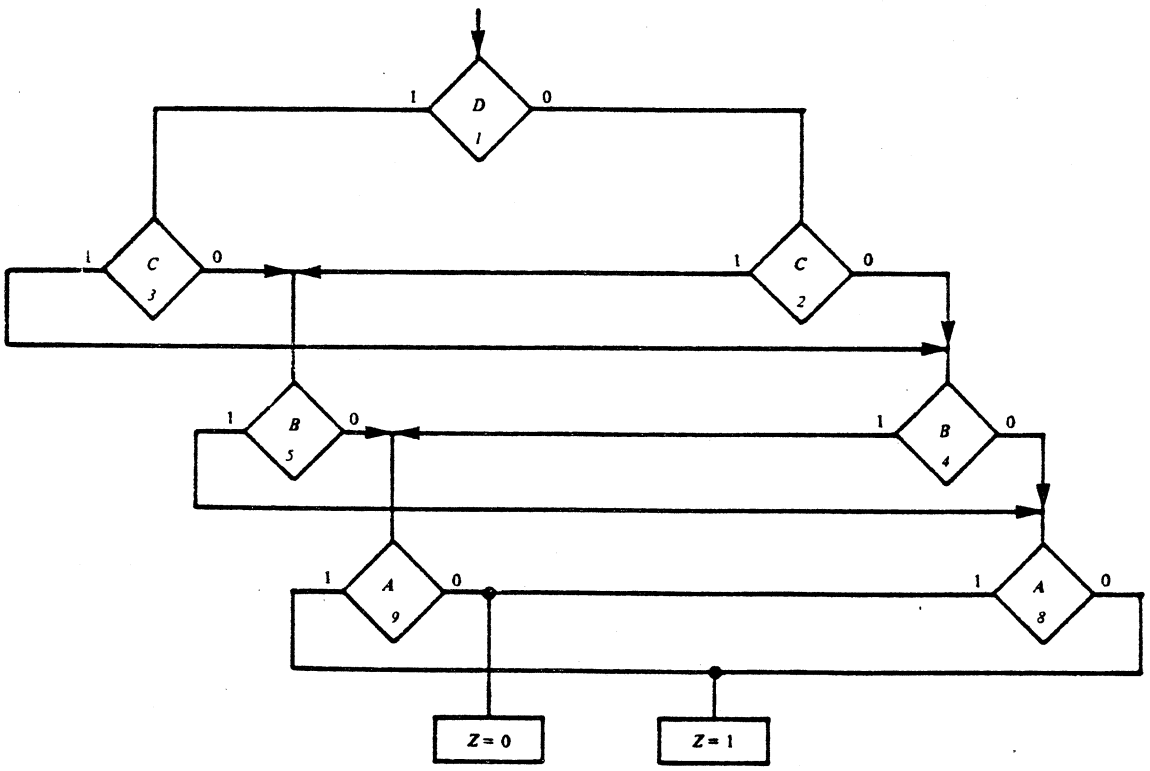


Figure 6 : Arbre de décision binaire

Il peut représenter les réseaux suivants (figure 7 a,b,c)

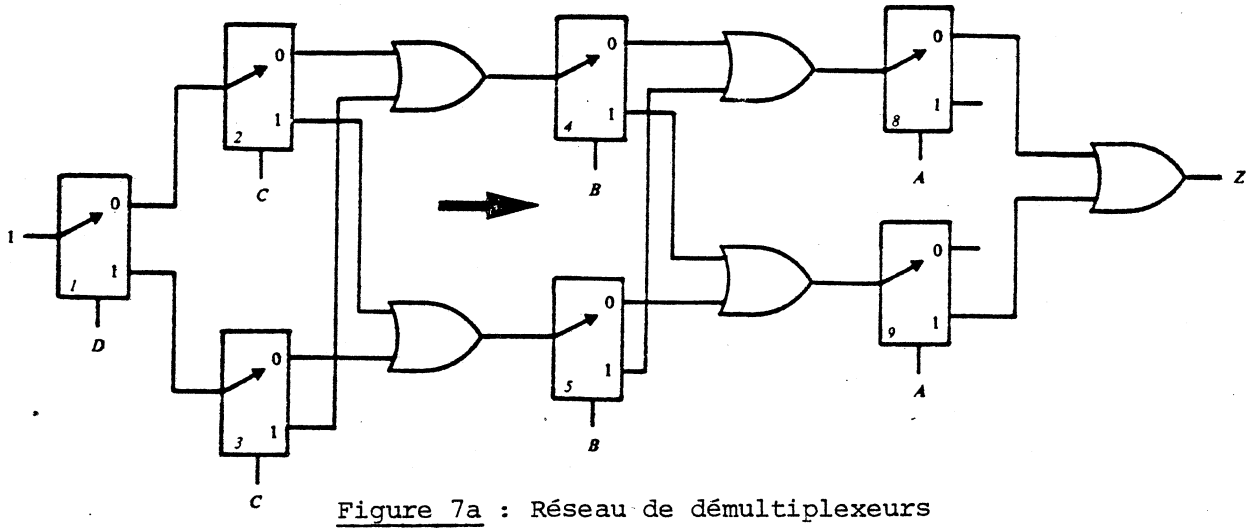


Figure 7a : Réseau de démultiplexeurs

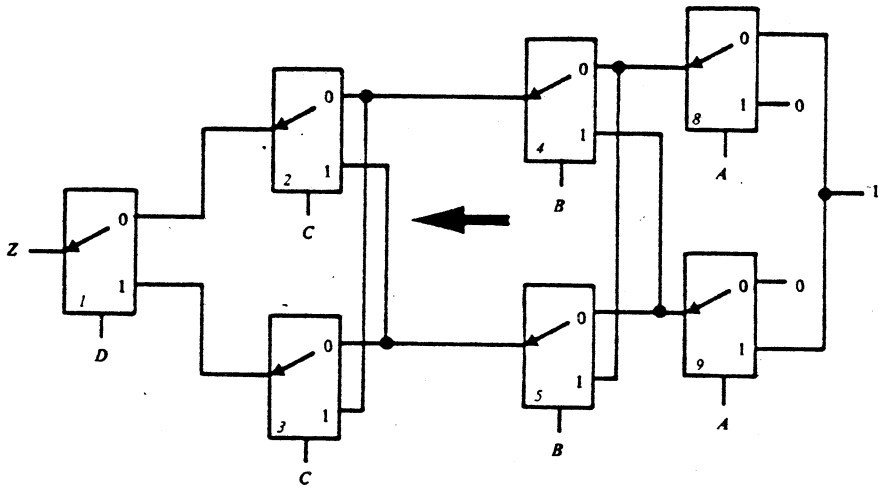


Figure 7b : Réseau de multiplexeurs

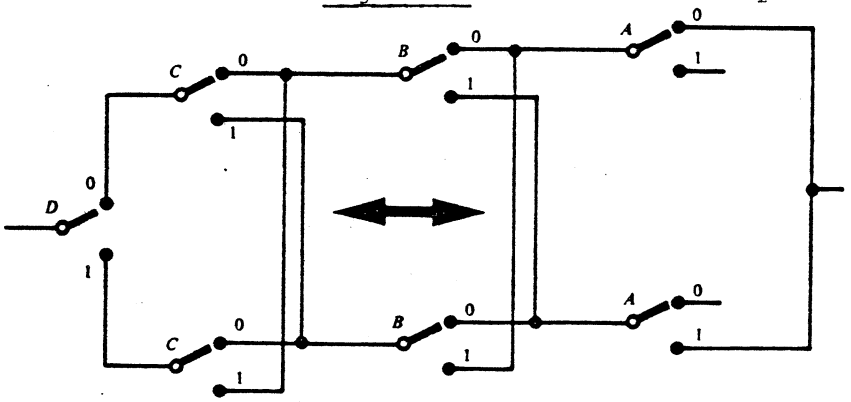


Figure 7c : Réseau de relais

Dans (MOR 82), on trouvera une formalisation des arbres de décision binaire, des applications dans différents domaines ainsi que des résultats récents. Le problème de l'optimisation est abordé en cherchant à minimiser la hauteur de l'arbre et le nombre de noeuds. L'optimisation d'un arbre est un problème très complexe et de nombreuses heuristiques sont données.

Enfin, dans (MAT 83) et (MAT), le passage d'un tableau de vérité ou d'une équation logique à un arbre de décision binaire en vue d'une implantation systématique par un réseau à transistors est étudié.

3.1.3. Position par rapport à la bibliographie

On étudie ici les arbres de décision binaire pour implanter des fonctions booléennes sous forme de réseaux d'interrupteurs, l'interrupteur étant le transistor MOS.

On cherche à obtenir :

- un réseau à nombre minimum de transistors réalisant l'arbre qu'on définira comme minimal,
- un réseau facilement implantable (réseau lexicographique).

Les arcs ne sont pas orientés à la différence des arbres représentant des réseaux de multiplexeurs et démultiplexeurs. En effet, à chaque arc est associé une variable représentant un transistor MOS qui est un élément bidirectionnel.

Exemple

Considérons la partie d'un arbre donnée sur la figure 8a :

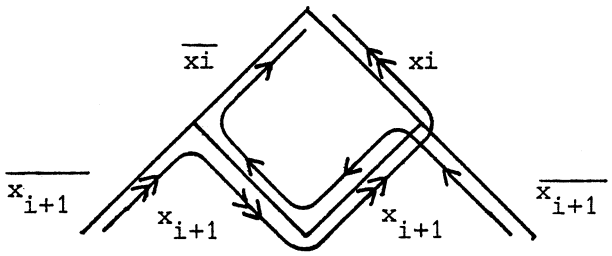


Figure 8a : Partie d'un arbre

Le réseau de transistors associé est le suivant (figure 8b) :

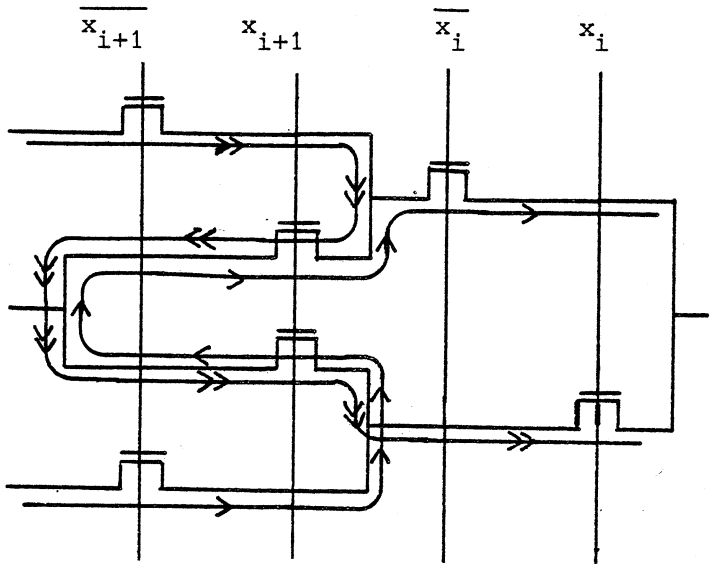


Figure 8b : Réseau de transistors associé

Les chemins supplémentaires dûs aux arcs non orientés sont notés sur l'arbre et le réseau associé (figures 8a et 8b). Les chemins s'annulent ici par la suite de la présence de xi+1 et $\bar{x}$i+1 sur chacun d'eux.

La présente étude sera donc propre aux problèmes de la technologie MOS et se préoccupera des différentes solutions d'implantation.

III - 2. Construction de l'arbre de décision binaire à partir de la forme canonique

On considère l'arbre complet représentant tous les monômes canoniques d'une fonction booléenne. Cet arbre est construit comme suit (AKE 78).

Partant d'un noeud initial, désignant la fonction, une variable de la fonction est introduite à chaque niveau sur les arcs. Le couple d'arcs partant d'un noeud correspond aux deux valeurs : 0 et 1 de la variable introduite à ce niveau. Les feuilles terminales sont étiquetées par la valeur de la fonction pour le chemin joignant cette feuille à la racine.

Exemple

Soit la fonction à 3 variables définie par la table de vérité de la figure 4. L'arbre complet de la figure 9 représente ce tableau, avec l'ordonnancement: x, y, z.

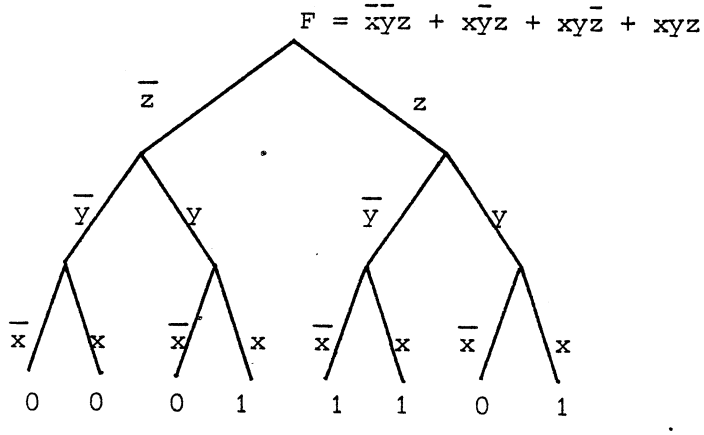


Figure 9 : Arbre canonique représentant une fonction à 3 variables

III - 3. Réalisation systématique de l'arbre canonique

L'arbre canonique conduit à un circuit universel et programmable comme pour la forme canonique de la logique de transfert. On a directement un réseau lexicographique en immergeant l'arbre sur un quadrillage fictif où les variables d'entrées déterminent les lignes verticales (colonnes), les lignes horizontales aux coudages près (chemins) correspondent aux différents chemins de l'arbre. Ce réseau est facilement implantable : il n'y a pas de croisements entre deux lignes de même type ; on a un ordonnancement lexicographique de variables, une variable et sa variable complémentaire étant adjacentes. Si n est le nombre de variables, le circuit a 2^n entrées (colonnes), le nombre de chemins passe de 2^n à 2 (diminution d'un facteur 2 à chaque variable traversée: 2^n, 2^{n-1}, ..., 4, 2).

Exemple

Soit l'arbre canonique de la figure 9, sa réalisation donne le schéma électrique de la figure 10, réseau lexicographique avec l'ordonnancement des variables x, y, z, donné par l'arbre initial.

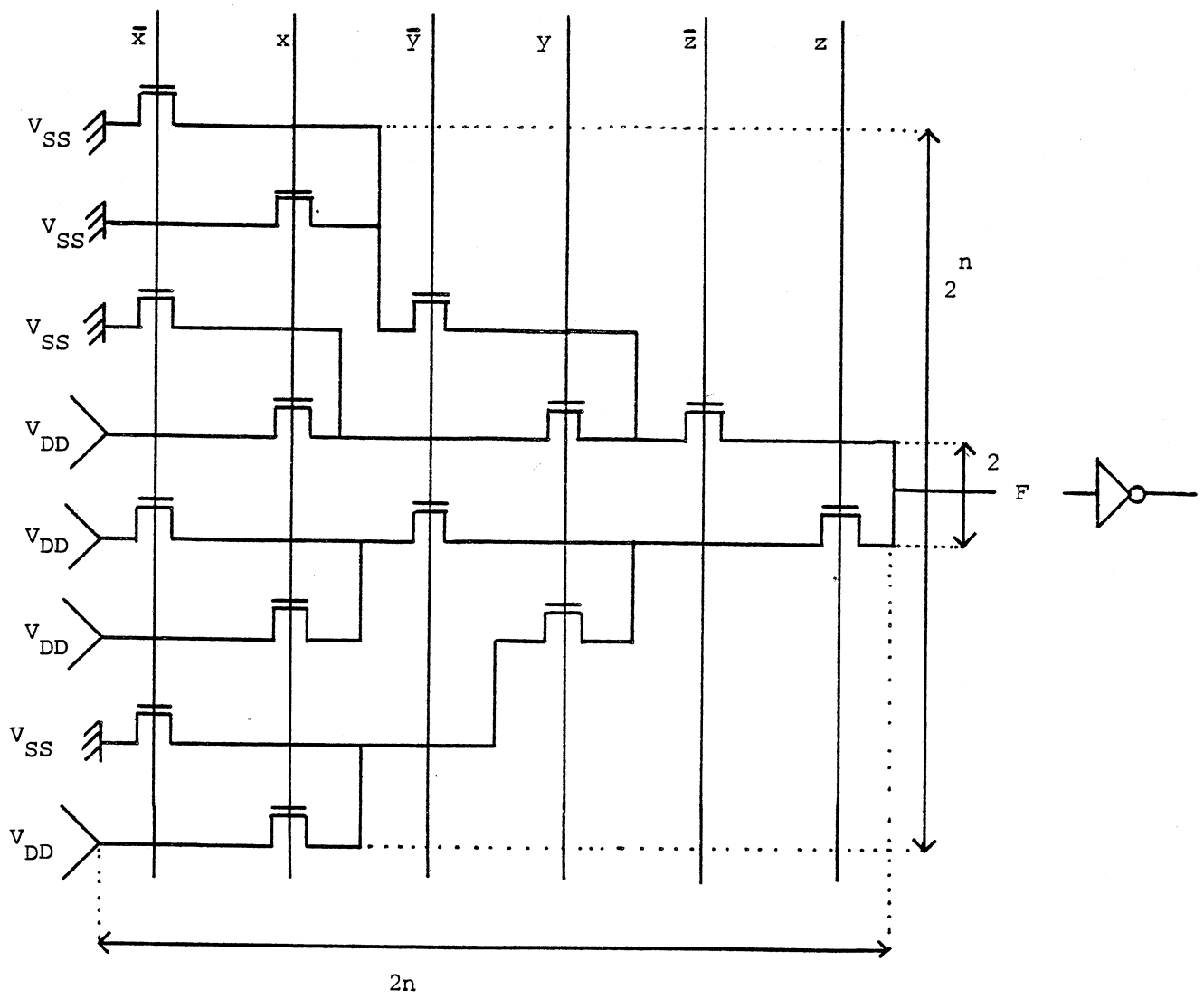


Figure 10 : Schéma électrique d'une fonction à 3 variables à partir de l'arbre canonique

La forme canonique de la logique de transfert et l'arbre canonique conduisent à des réalisations systématiques.

La réalisation sous forme d'arbre canonique permet un gain en nombre de transistors, donc en surface, important par rapport à la forme canonique. Pour une fonction à n variables, le nombre de transistors passe de $n \times 2^n$, à $2 + 2^2 + 2^3 + \ldots 2^n$, si n = 3 de 24 à 14.

Si n est le nombre de variables, la réalisation de la forme canonique se fera dans un rectangle de dimensions 2^n x 2n, tandis que l'arbre canonique conduira à une réalisation dans un triangle isocèle inclus dans le rectangle précédent de dimensions : 2^n pour la base, 2n pour la hauteur.

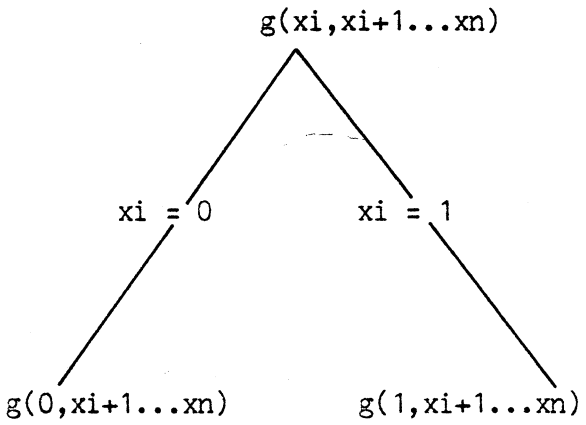
Remarque : Comme pour la forme canonique de la logique de transfert, les variables peuvent arriver par le haut ou le bas du réseau, l'ordre des variables ne modifie pas le nombre de transistors du réseau pour l'arbre de décision binaire canonique. On choisira aussi l'ordonnancement et le point d'entrée des variables en fonction des connexions du réseau avec le reste du circuit.

III - 4. Construction de l'arbre de décision binaire à partir de la forme non canonique

Soit une fonction booléenne simplifiée : toutes les variables n'apparaissent pas dans chaque monôme. L'arbre de décision binaire est construit de la façon suivante (AKE 78) :

- partant d'un noeud initial désignant la fonction F(x1,x2,...xi...xn), n niveaux introduiront successivement les variables x1, x2....xi...xn

- on considère le ième niveau introduisant les variables directes et complémentées respectivement xi et $\bar{x}i$: l'extrémité d'un arc sera étiquetée par la fonction dans laquelle la variable xi a été remplacée par la valeur 0 ($\bar{x}i$) ou 1 (xi) :



On applique donc la formule de SHANON :

$$g(xi,xi+1...xn) = \bar{x}i\ g(0,xi+1...xn) + xi\ g(1,xi+1...xn)$$

Exemple

Soit la fonction simplifiée à 3 variables $F = xy + \bar{y}z$. L'arbre de décision binaire sera celui de la figure 11 avec l'ordonnancement x,y,z.

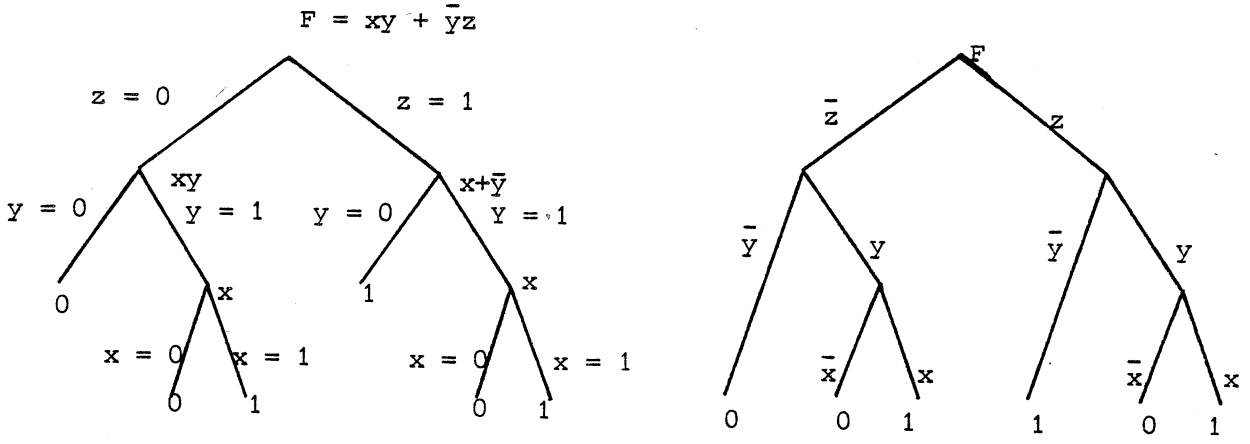


Figure 11 : Arbre construit à partir de la forme simplifiée de F

L'arbre de décision binaire construit à partir d'une fonction simplifiée n'est pas complet par rapport à l'arbre canonique : des simplifications apparaissent déjà. Le nombre de variables associées aux arcs (correspondant au nombre de transistors du réseau) sera différent suivant l'ordre lexicographique choisi à la différence de la forme canonique ou de l'arbre canonique.

III - 5. Optimisation de l'arbre de décision binaire

L'optimisation d'un arbre de décision binaire est un problème complexe qui peut être abordé de deux façons :

(i) on considère que l'ordonnancement des variables est figé et a été choisi en fonction des connexions externes du réseau implémentant l'arbre.

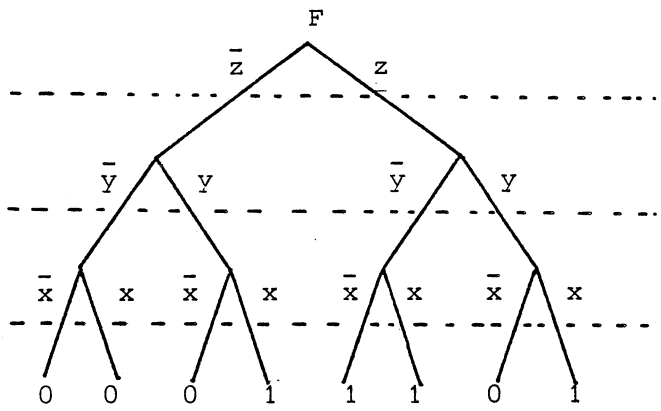
(ii) l'ordonnancement des variables n'est pas fixé et cet ordre est un paramètre de l'optimisation.

Nous considérerons dans ce chapitre la première situation.

La comparaison entre deux arbres de décision binaire porte habituellement sur le nombre de transistors ; on peut également faire intervenir un facteur de forme ou d'encombrement du circuit. Pour ceci on peut faire intervenir la valeur de la fonction de pic (SER 82). Pour une variable d'entrée, la fonction de pic est le nombre d'arcs étiquetés par cette variable (directe ou complémentée).

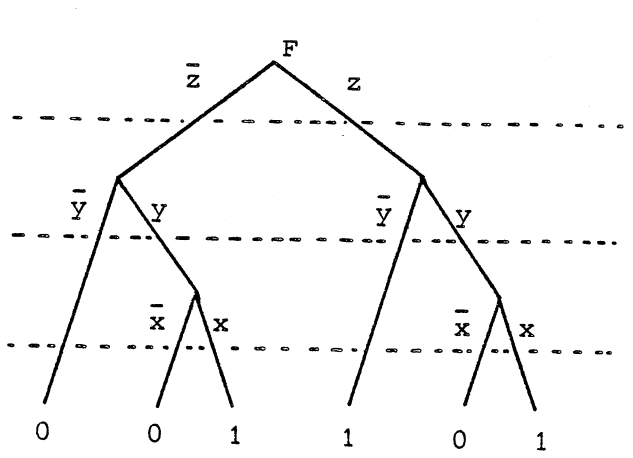
Exemple

Soit l'arbre canonique de la figure 9 (§ III.2) et l'arbre non canonique de la figure 11 (§ III.4), on obtient les fonctions de pic et le nombre de transistors sur ces deux arbres respectivement sur les figures 12a et 12b.



	Fonction de pic	Nombre de transistors
	2	2
	4	4
	8	8
		14

Figure 12a : Fonction de pic et nombre de transistors d'un arbre canonique



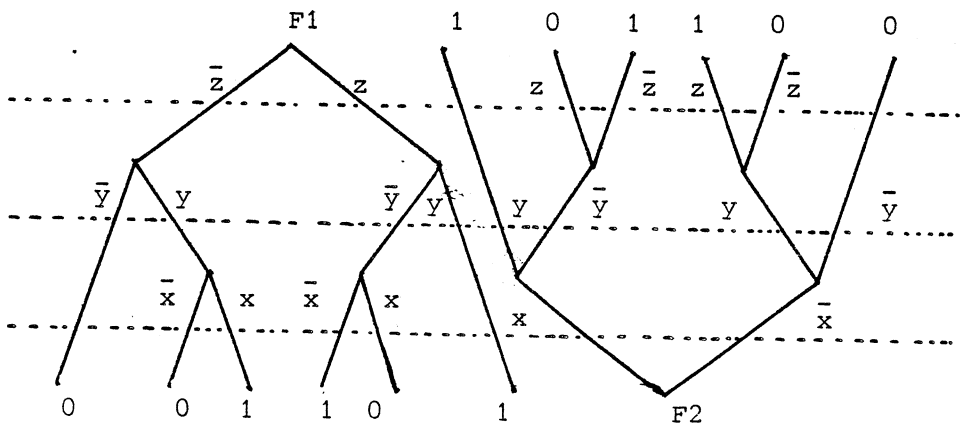
Fonction de pic	Nombre de transistors
2	2
4	4
6	4
	10

Figure 12b : Fonction de pic et nombre de transistors d'un arbre non canonique

A nombre de transistors égal, deux solutions peuvent être différenciées par le facteur de forme donné par la suite des valeurs de la fonction de pic. Suivant la forme désirée, une solution peut en effet être préférable. Considérons par exemple l'encastrement de deux réseaux triangulaires (de façon analogue au § III.3 du chapitre I). Dans ce cas, la somme des fonctions de pic pour les deux réseaux, sur une même horizontale doit être constante pour un encastrement idéal.

Exemple

Soient les deux arbres de la figure 13 :



Fonctions de pic	
F1	F2
2	6
4	4
6	2

Figure 13 : Deux fonctions encastrées

Contraintes à respecter au cours de l'optimisation

- pas de croisements entre les arcs (pas de croisements entre les chemins du réseau),
- pas de chemins parasites introduits sur l'arbre au cours des simplifications de celui-ci (pas de chemins parasites sur le réseau).

Simplifications dans le cas où l'ordonnancement est fixé

On part d'une fonction exprimée sous forme canonique ou simplifiée comportant au maximum quatre ou cinq variables (§ I.2). On construit l'arbre de décision binaire associé (l'ordonnancement des variables est choisi en fonction de la connectique du circuit). On simplifie l'arbre à l'aide des trois règles suivantes :

1ère règle : suppression d'arcs

Deux arcs étiquetés xi et $\overline{xi}$ ayant même origine et racine de deux sous-arborescences identiques peuvent être supprimés. La sous-arborescence est reportée à l'arc origine (cf figure 14a).

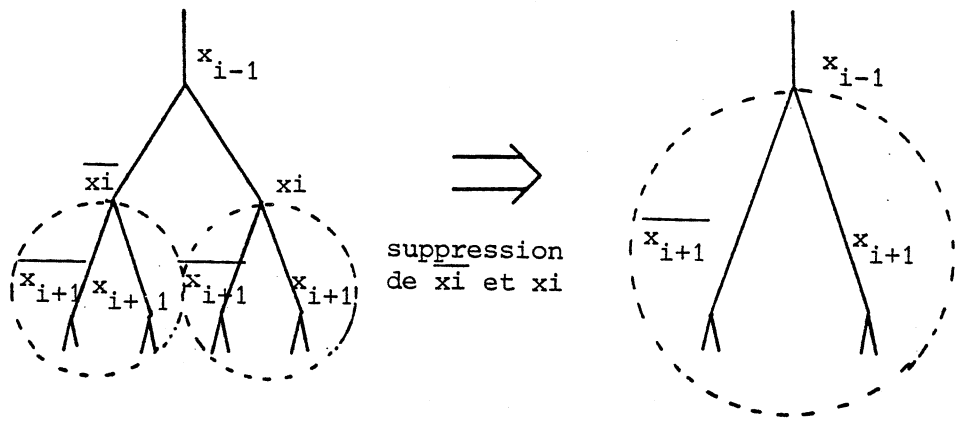


Figure 14a : Suppression d'arcs

Cas particulier : feuilles terminales (figure 14b)

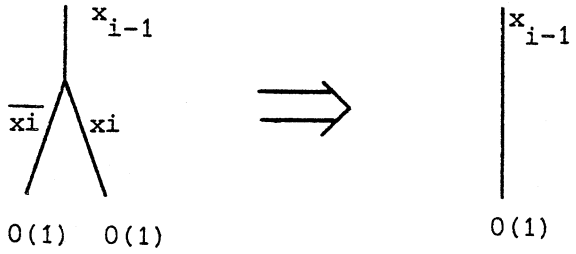


Figure 14b : Valeur commune

Exemple

Soit l'arbre canonique de la figure 9, § III.2. On peut appliquer la 1ère règle à deux couples d'arcs dont les extrémités sont les feuilles terminales. On obtient l'arbre de la figure 15.

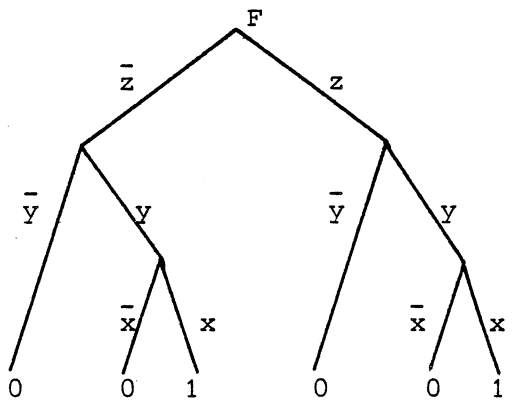


Figure 15 : Arbre canonique de la figure 9 après application de la 1ère règle

L'arbre obtenu est celui de la figure 11, § III.4 : arbre construit à partir de la forme simplifiée de F. Pour une fonction plus complexe, on peut obtenir deux arbres différents suivant la forme initiale et les simplifications effectuées sur ces deux arbres.

2ème règle : "convergence"

Deux chemins de deux arcs :

- ayant même origine
- étiquetés respectivement xi xi+1 et $\overline{xi}$ xi+1
- et dont les extrémités sont racines de deux arboresences identiques

peuvent être remplacés par un seul arc :

- ayant même origine
- étiqueté xi+1 (cf figure 16a)

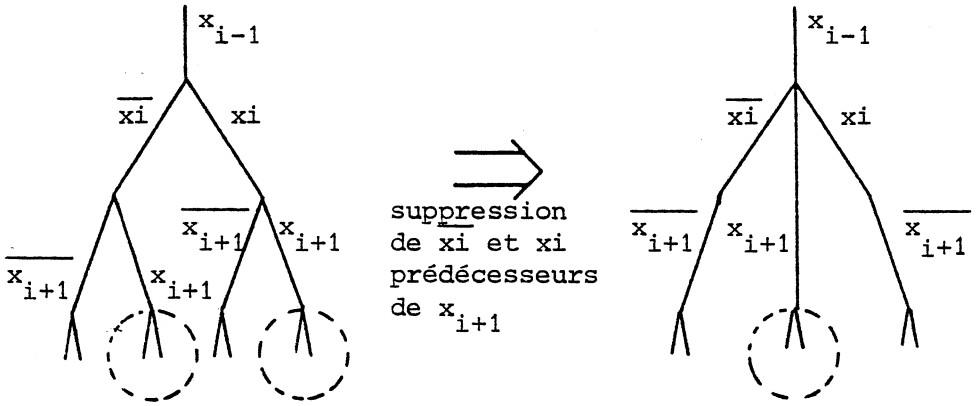


Figure 16a : "Convergence"

Cas particulier : feuilles terminales (figure 16b)

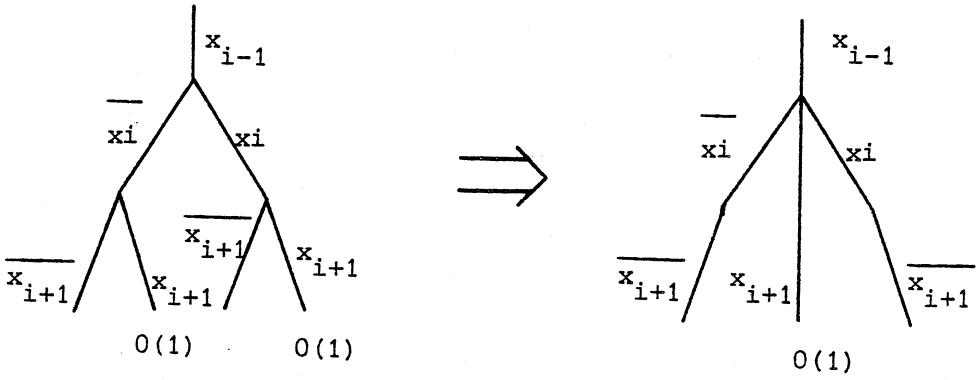


Figure 16b : "Convergence" sur les feuilles terminales .

Exemple

Soit l'arbre précédent (figure 15) si on applique la 2ème règle, on obtient l'arbre de la figure 17 :

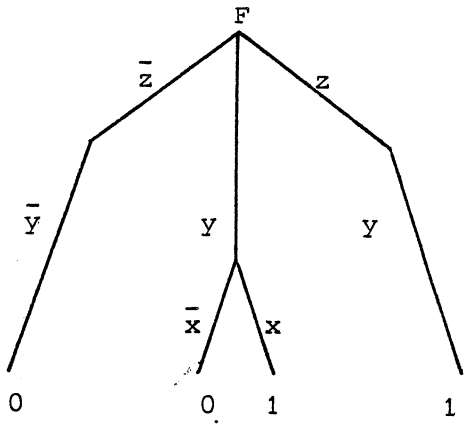


Figure 17 : Arbre de la figure 9 simplifié

3ème règle : "convergence - divergence"

Soient deux chemins de deux arcs de l'arbre :

- ayant même origine
- étiquetés respectivement $\overline{xi}$ xi+1 et xi $\overline{xi+1}$
- racine de deux arborescences identiques

alors les extrémités de ces deux chemins peuvent être confondues (cf. la figure 18a)

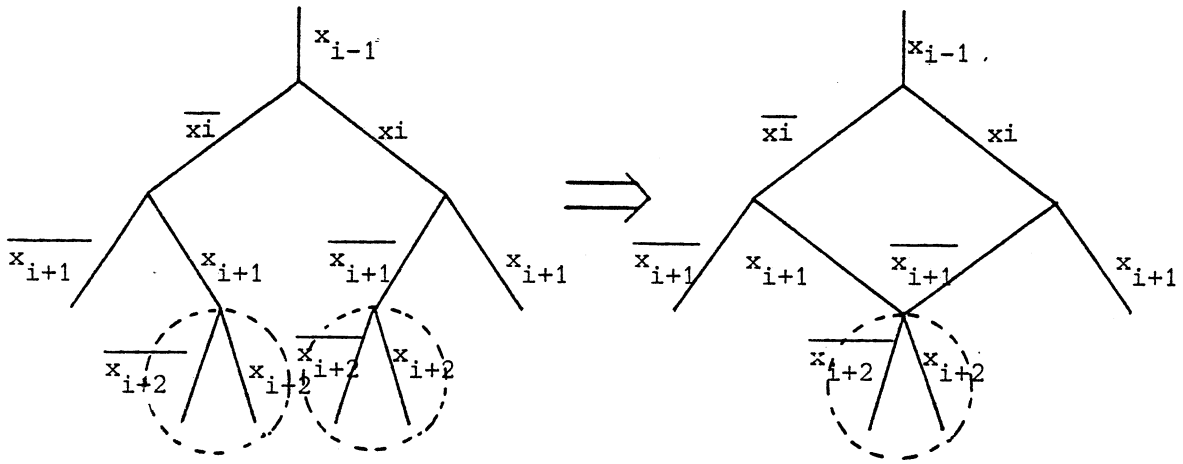


Figure 18a : "Convergence - divergence"

Remarque : Les chemins parasites créés sont annulés (présence de xi+1 et $\overline{xi+1}$ sur ces chemins).

Cas particulier : feuilles terminales (figure 18b)

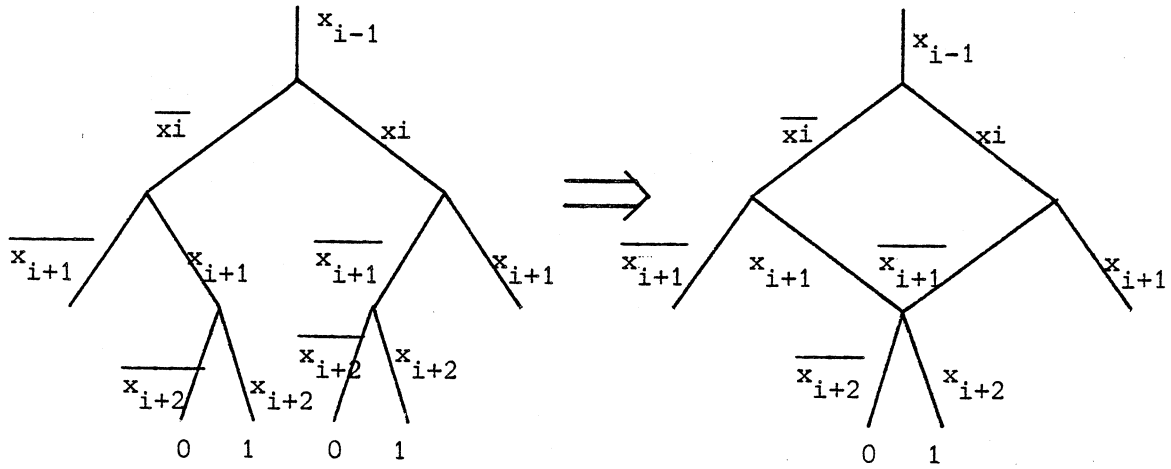


Figure 18b : "Convergence - divergence" sur les feuilles terminales

Exemple

Soit l'arbre canonique suivant (figure 19a) représentant une fonction à 4 variables w, x, y, z.

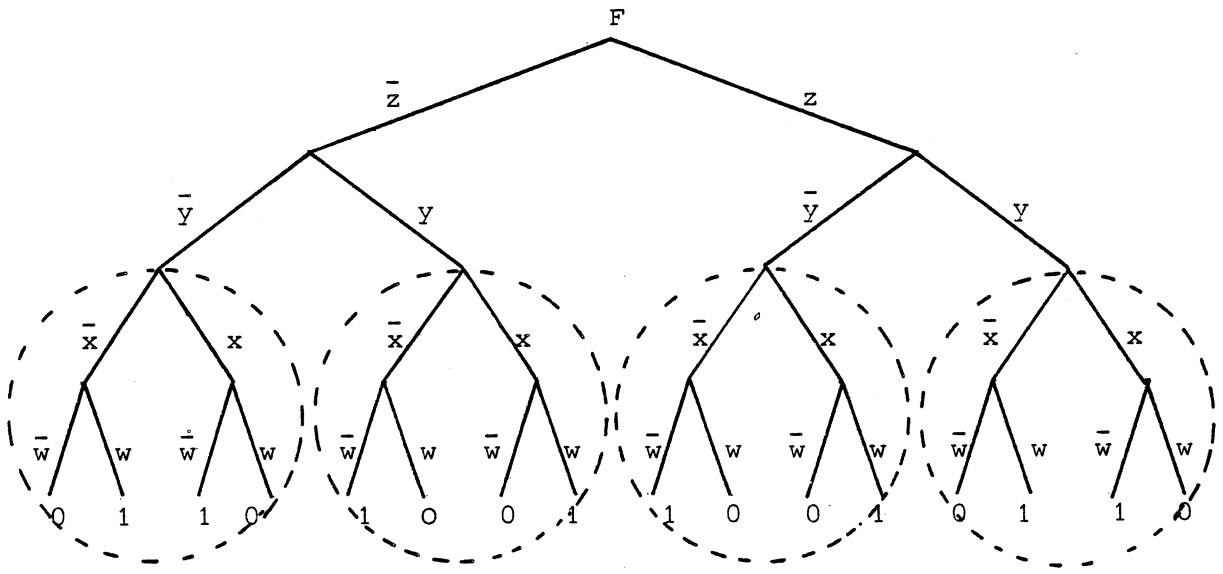


Figure 19a : Arbre canonique représentant une fonction à 4 variables

Les 1ère et 2ème règles ne sont pas applicables. Si on applique la 3ème règle à deux reprises, on obtient des croisements entre les arcs, on choisit donc un seul fusionnement. L'arbre obtenu est l'arbre simplifié (figure 19b)

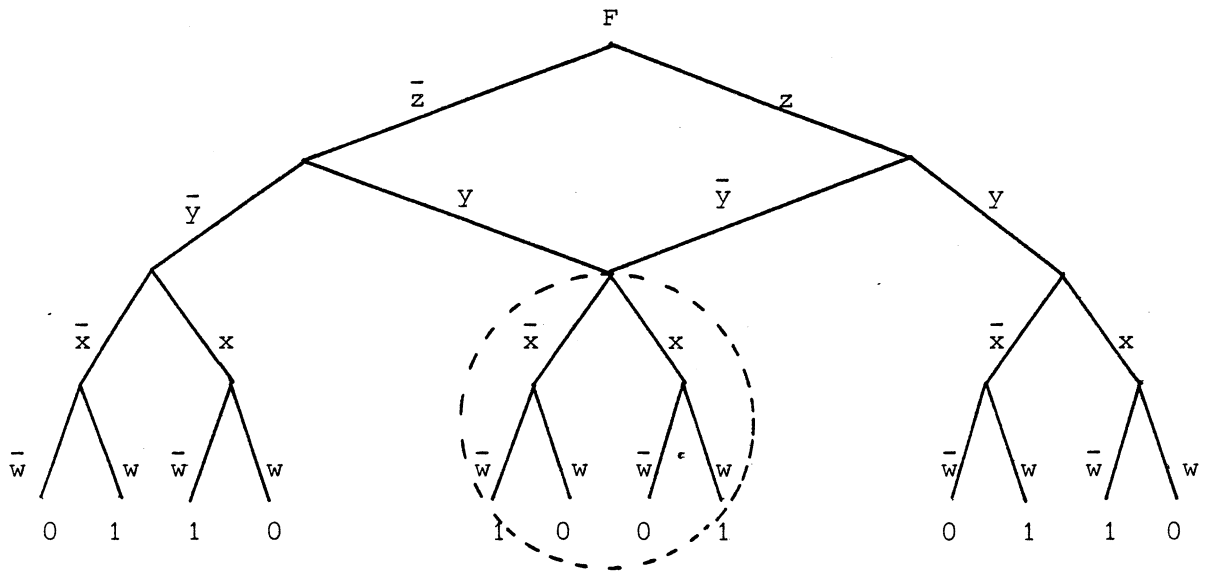


Figure 19b : Arbre canonique de la figure 19a simplifié

IV - LOGIQUE DE TRANSFERT A VARIABLE PRIVILEGIEE

IV - 1. Principe général

Précédemment on s'est intéressé à la réalisation de fonctions en logique de transfert définies de la façon suivante (figure 20) :

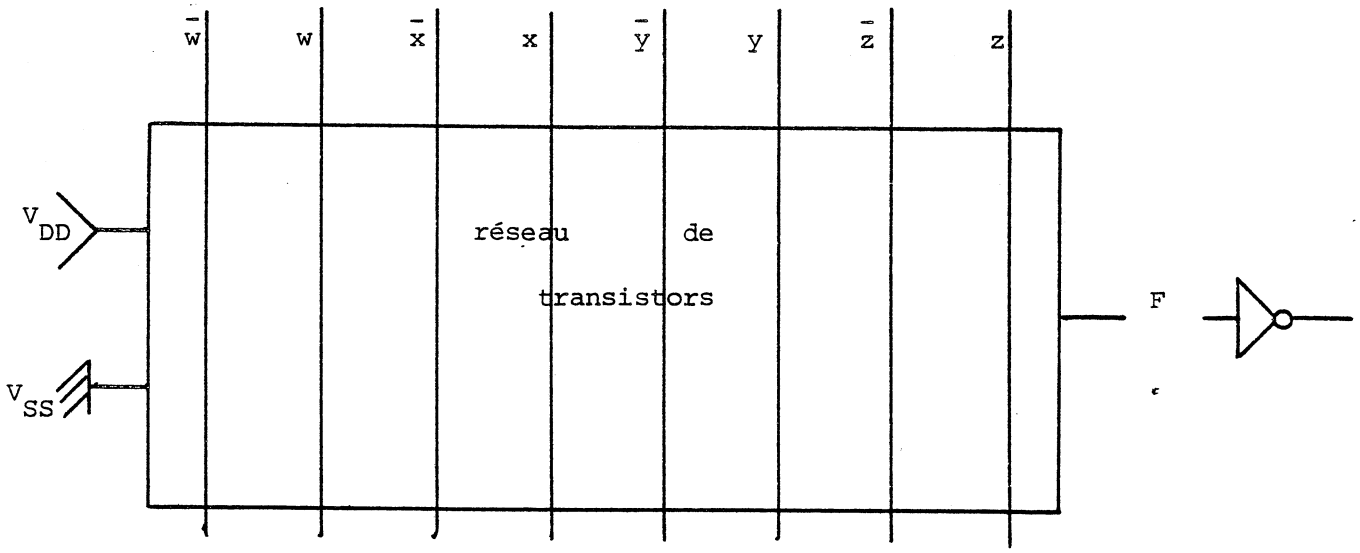


Figure 20 : Fonction F(w,x,y,z) réalisée en logique de transfert

Le réseau de transistors est obtenu à partir de :

- la forme canonique de la fonction
- ou de l'arbre canonique ou simplifié de la fonction.

Les variables d'entrée déterminent les colonnes, les chemins de la fonction sont reliés à l'alimentation VDD ou à la masse VSS (réseau R et $\bar{R}$ imbriqués ou non).

On réalise ainsi des fonctions de quatre à cinq variables (§ I.2).

Ici, on étudie la logique de transfert à variable privilégiée, dans laquelle une variable d'entrée et les constantes 0 et 1 apparaissent à la place des alimentations VDD et VSS selon le schéma de la figure 21 :

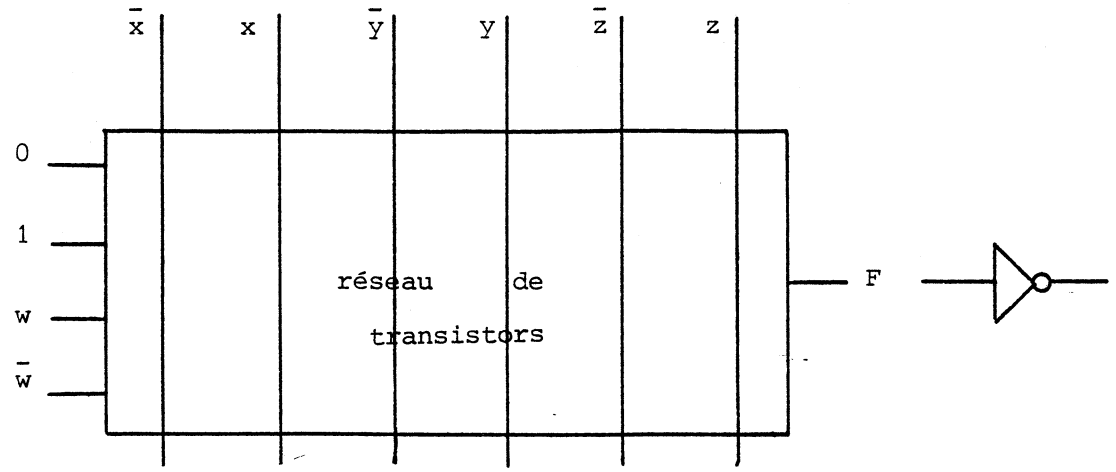


Figure 21 : Fonction F(w,x,y,z) réalisée en logique de transfert à variable privilégiée : w

Le réseau de transistors est obtenu à partir de :

- soit la forme canonique de la fonction
- soit l'arbre canonique ou simplifié de la fonction

dans lesquels une variable est privilégiée.
La variable privilégiée est choisie en fonction des connexions avec les autres éléments du circuit.

On réalise ainsi des fonctions de cinq ou six variables, en réduisant à quatre ou à cinq le nombre de transistors en série dans le réseau (§ I.2), mais les inverseurs attaquant les entrées de chemin seront plus puissants.

Les variables d'entrée déterminent les colonnes, les chemins de la fonction sont reliés aux valeurs de la variable privilégiée (variable directe, complémentée,0 ou 1).

La variable d'entrée privilégiée doit être stabilisée avant les variables d'entrée qui sont en fait des commandes pour le réseau : elles arrivent sur les

grilles des transistors (figure 22). Ceci élimine les possibilités de réalisation où la variable privilégiée arriverait sur les chemins et sur les colonnes.

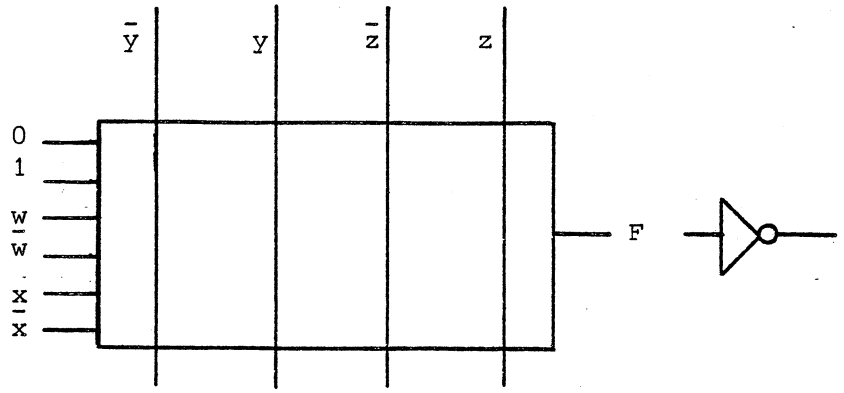


Figure 22 : Allure des signaux d'entrée

Extension : logique de transfert à deux ou plusieurs variables privilégiées

Considérons la réalisation où deux ou plusieurs variables apparaissent à la place des alimentations VDD et VSS selon le schéma de la figure 23 :

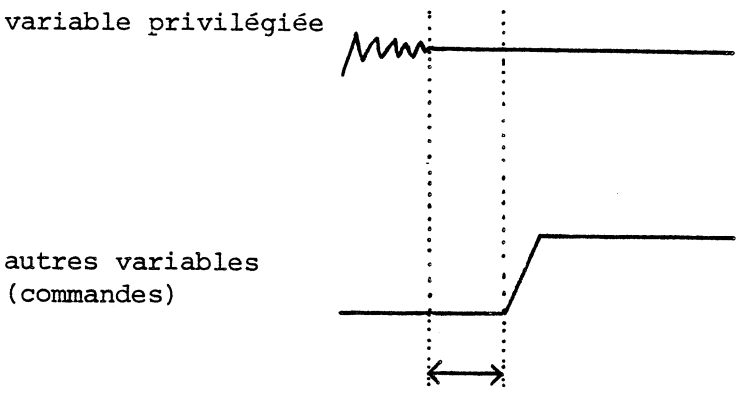


Figure 23 : Fonction F(w,x,y,z) réalisée en logique de transfert à deux variables privilégiées : w,x

Cette extension est en fait un cas particulier de réalisation de fonctions à l'aide de multiplexeurs. Dans (BLA 79), il est montré qu'une fonction à n variables peut être réalisée par un multiplexeur à n variables de sélection avec les 2^n valeurs binaires à l'entrée, soit par un multiplexeur à (n-1) variables de sélection ayant à l'entrée une fonction de la nème variable etc... (cf figures 24 et 25)

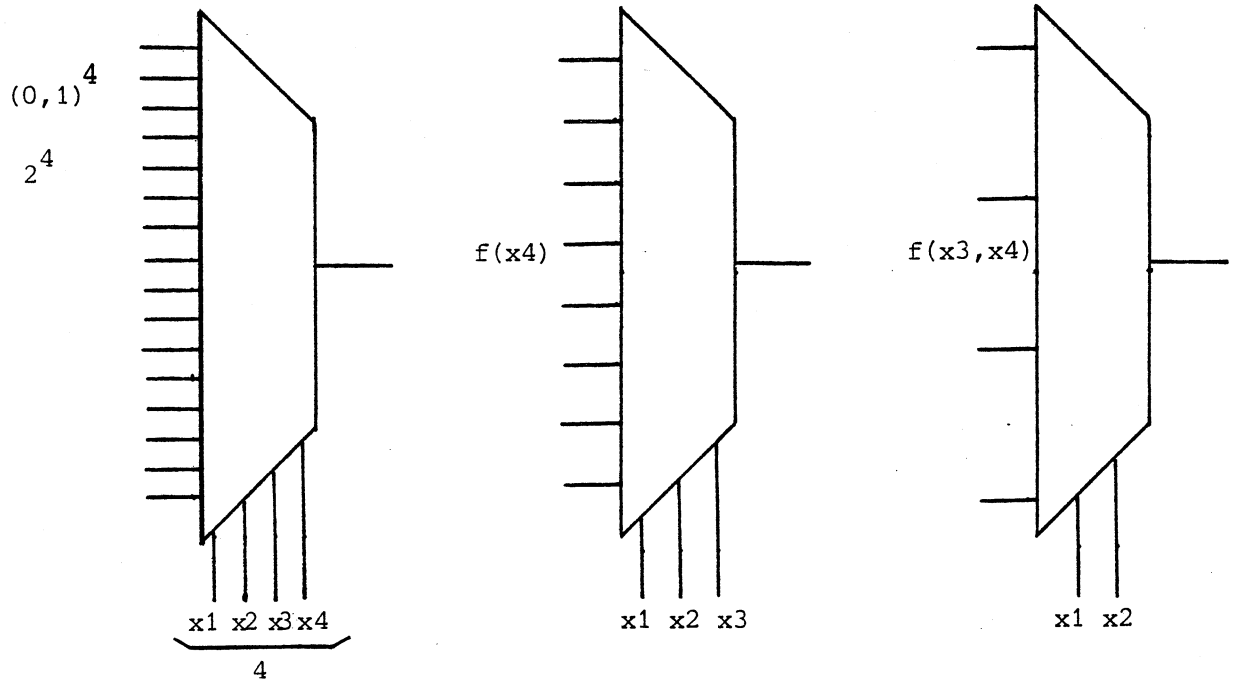


Figure 24 : Réalisation d'une fonction à 4 variables à l'aide de multiplexeurs

Variables de sélection	Variables d'entrée
n	$(0,1)^n$
n - 1	f(n ième variable)
n - 2	f((n-1) ième et n ième variables)

Figure 25 : Différentes façons de réaliser une fonction de n variables avec un multiplexeur

De la même façon, une fonction de n variables pourra être réalisée soit avec les n variables d'entrée en commandes et les alimentations VDD et VSS à l'entrée des chemins, soit avec (n-i) variables en commandes et les ième variables à l'entrée des chemins, mais pour des raisons de compatibilité technologique on se restreint aux cas très particuliers où les fonctions aux entrées se ramènent aux fonctions triviales xj, $\bar{x}j$, 0, 1 (pour les différentes variables ramenées aux entrées).

IV - 2. Forme canonique et logique de transfert à variable privilégiée

Soit une fonction définie par sa table de vérité. Une variable sera choisie pour arriver à l'entrée des chemins à la place des alimentations VDD et VSS (0 et 1 sur la table de vérité).

Exemple

Soit la fonction à 3 variables F(x,y,z) définie par sa table de vérité (figure 4, § II.1) :

$$F = \bar{x}\bar{y}z + \bar{x}y\bar{z} + xy\bar{z} + xyz$$
$$\bar{F} = \bar{x}\bar{y}\bar{z} + \bar{x}\bar{y}z + \bar{x}y\bar{z} + x\bar{y}\bar{z}$$

L'implantation systématique de la forme canonique est donnée sur la figure 5 du § II.2.

Soit x, la variable privilégiée. L'implantation systématique de la forme canonique avec la variable privilégiée x est donnée sur la figure 26. On l'obtient à partir de la table de vérité : pour la combinaison $\bar{y}\bar{z}$ des variables y et z, F = 0, pour la combinaison $y\bar{z}$, F = x.....

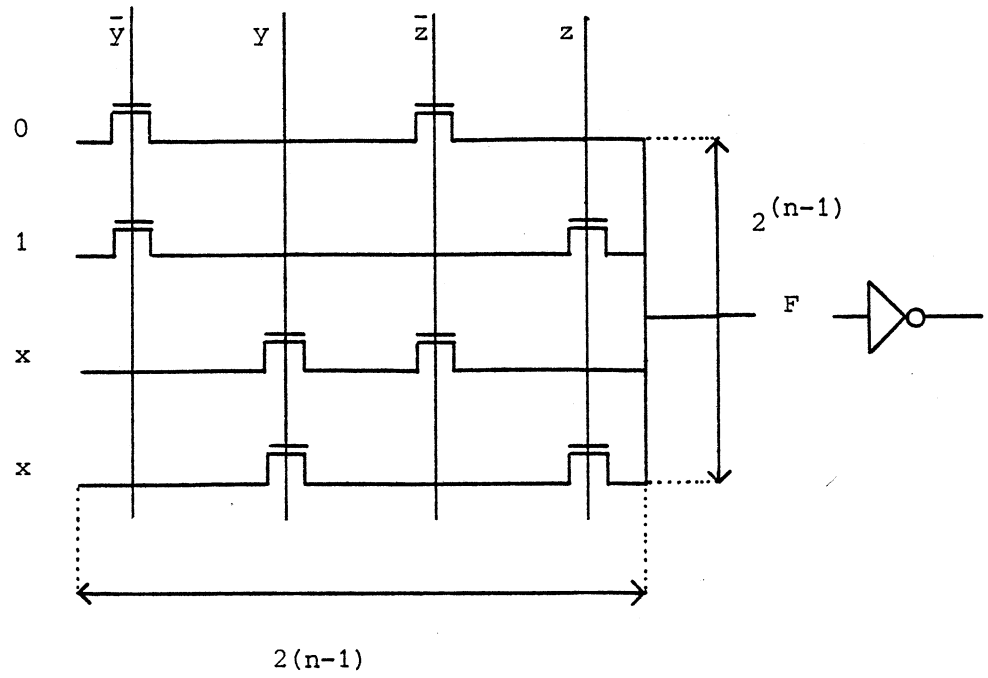


Figure 26 : Réalisation d'une fonction à 3 variables sous forme canonique avec une variable privilégiée : x

Si n est le nombre de variables, on a $2^{(n-1)}$ chemins, 2(n-1) colonnes et $(n-1)x2^{(n-1)}$ transistors. Précédemment, on avait 2^n chemins, 2n colonnes et $nx2^n$ transistors pour la forme canonique (0, 1 à l'entrée des chemins). Pour une fonction à 3 variables, le nombre de transistors passe de 24 à 8.

Cas particulier : logique de transfert à plusieurs variables privilégiées

Avec la forme canonique, les variables d'entrée arriveront soit sur les colonnes, soit à l'entrée des chemins si elles sont choisies comme variables privilégiées.

Exemple

Soit la fonction F(x,y,z) précédente définie par sa table de vérité (figure 4, § II.1)

$$F = \bar{x}\bar{y}z + \bar{x}y\bar{z} + x\bar{y}\bar{z} + xyz$$
$$\bar{F} = \bar{x}\bar{y}\bar{z} + \bar{x}\bar{y}z + \bar{x}y\bar{z} + x\bar{y}\bar{z}$$

En choisissant x,z comme variables privilégiées, on n'a pas une fonction f(x,z) à l'entrée des chemins mais la valeur x pour y et z pour $\bar{y}$, cf la table de vérité. L'implantation systématique de la forme canonique avec les variables privilégiées x,z est donnée sur la figure 27.

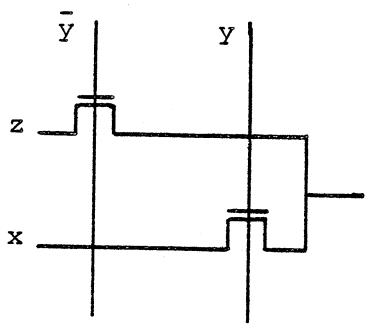


Figure 27 : Réalisation d'une fonction particulière à 3 variables privilégiées sous forme canonique avec deux variables privilégiées : x et z.

IV - 3. Forme arborescente et logique de transfert à variable privilégiée

Soit une fonction définie par son arbre canonique ou simplifié, l'ordonnancement des variables sera choisie en fonction de la variable privilégiée : elle apparaîtra sur les feuilles terminales de l'arbre.

Exemples

Soit la fonction à 3 variables précédente F(x,y,z) sur laquelle on a étudié la forme canonique normale et les formes canoniques à une ou deux variables privilégiées. L'arbre canonique est celui de la figure 28a.

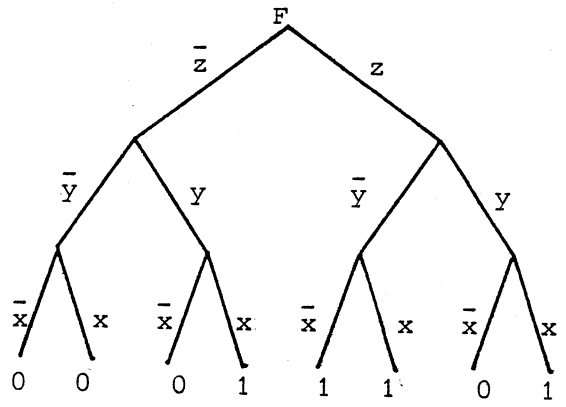
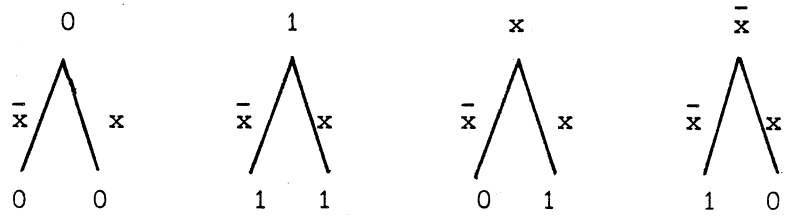


Figure 28a : Arbre canonique d'une fonction à 3 variables

La variable privilégiée sera x, l'arbre canonique avec la variable privilégiée x aura la variable y sur les arcs terminaux. Les valeurs 0, 1, x, $\bar{x}$ pour la variable privilégiée x seront calculées de la façon suivante :



On obtient alors l'arbre de la figure 28b :

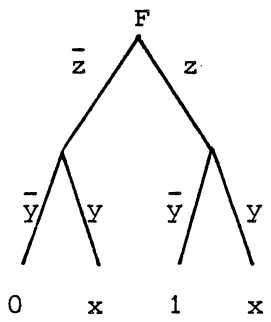


Figure 28b : Arbre canonique représentant une fonction à 3 variables avec une variable privilégiée : x

Si n est le nombre de variables, on passe de $2 + 2^2 + 2^3 + \ldots 2^n$ transistors (nombre d'arcs de l'arbre) pour l'arbre canonique à $2 + 2^2 + 2^3 + \ldots 2^{(n-1)}$ transistors pour l'arbre canonique à variable privilégiée, si n = 3 de 14 à 6.

L'arbre simplifié de cette même fonction est celui de la figure 29a :

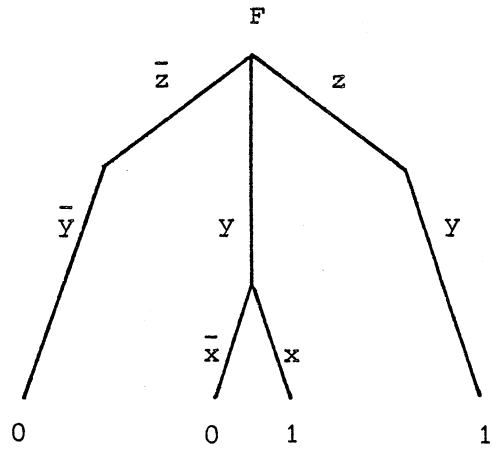


Figure 29a : Arbre simplifié d'une fonction à 3 variables

De même, x sera la variable privilégiée , on obtient l'arbre de la figure 29b :

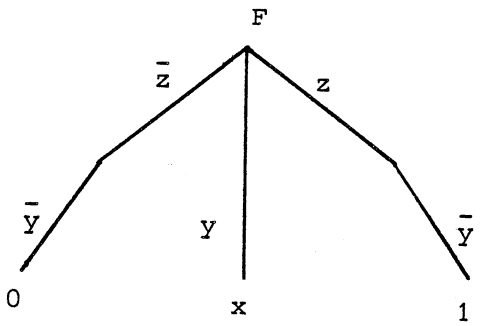
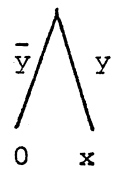


Figure 29b : Arbre simplifié avec une variable privilégée : x

Cas particulier : logique de transfert à plusieurs variables privilégiées

Avec l'arbre canonique, on ne peut avoir qu'une variable privilégiée. En effet, en choisissant deux variables privilégiées, on obtient une fonction des deux variables sur les feuilles terminales.

Sur l'exemple de la figure 28b, on aurait la fonction xy au lieu de



Avec l'arbre simplifié, les variables privilégiées peuvent ne pas faire de fonctions sur les feuilles terminales.

Exemple

Soit l'arbre simplifié suivant (figure 30a)

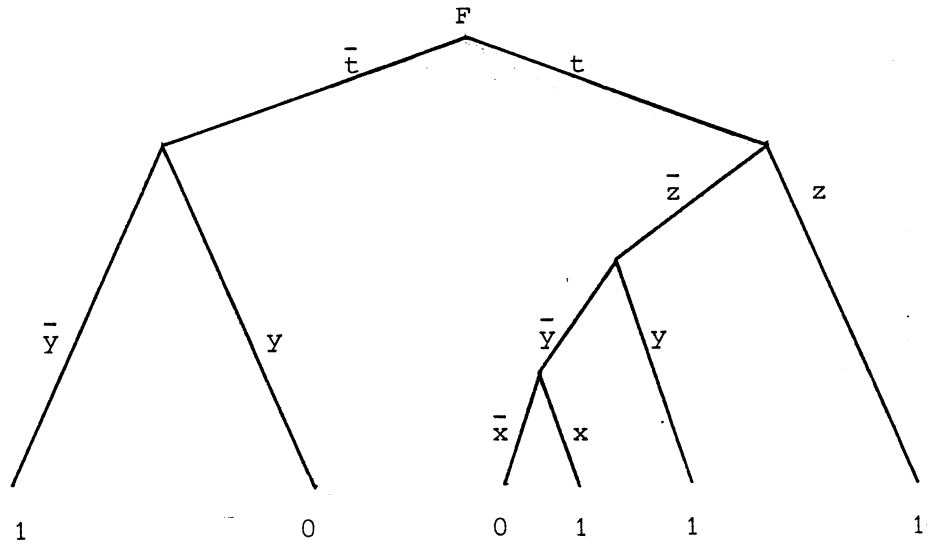


Figure 30a : Arbre simplifié représentant une fonction à 4 variables t,z,y,x

On peut obtenir l'arbre de la figure 30b avec les variables privilégies x et y:

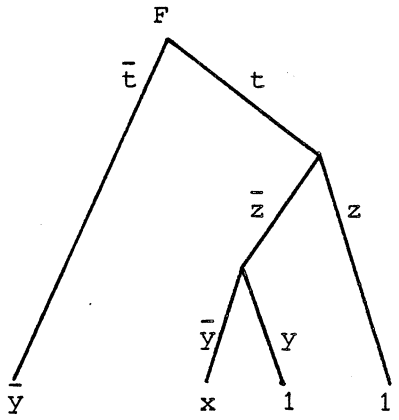


Figure 30b : Arbre simplifié avec deux variables privilégiées x,y créant des fonctions triviales aux feuilles

Sur cet arbre, on a la variable y qui apparaît à la fois sur les arcs et les feuilles terminales. On élimine ce cas, en effet les variables arrivant sur les feuilles terminales doivent être stabilisées avant les variables arrivant sur les arcs (commandes).

On ne peut donc avoir plusieurs variables privilégiées avec la forme arborescente que pour des arbres simplifiés particuliers.

Exemple

Soit l'arbre simplifié suivant (figure 31a)

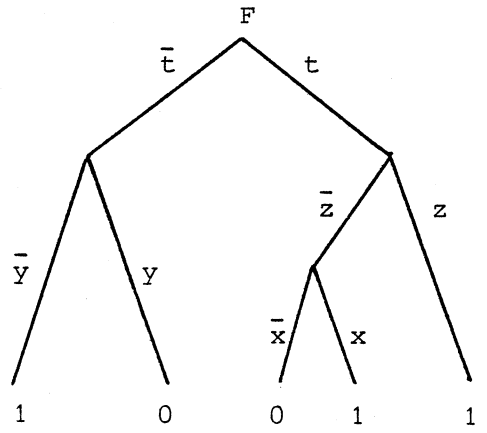


Figure 31a : Arbre simplifié représentant une fonction à 4 variables t,z,y,x

On obtient l'arbre de la figure 31b avec les variables privilégiées x et y.

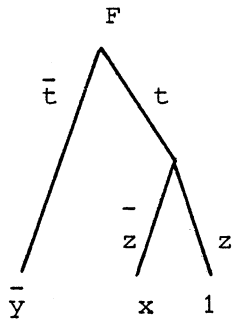


Figure 31b : Arbre simplifié avec deux variables privilégiées x,y

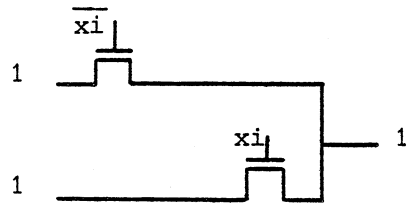
Les valeurs des fonctions apparaissant aux feuilles sont des fonctions triviales.

De plus, les variables privilégiées n'apparaissent que sur les feuilles.

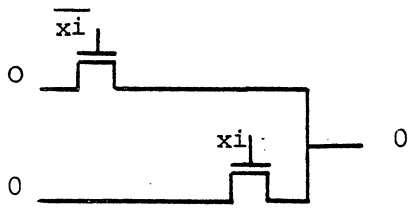
IV - 4. Remarque : application à la technologie CMOS

En appliquant le principe de base de la logique de transfert (§ I.1) à la technologie CMOS, on obtient le schéma d'une porte complexe CMOS donné au § IV.1 du chapitre I. On a un schéma implanté de la porte à deux zones : diffusion N (réseau $\overline{R}$) et diffusion P (réseau $\overline{R}^*$). L'optimisation de ces réseaux doit donc se faire de façon séparée comme nous l'avons vu au chapitre I.

La logique de transfert à variable privilégiée ne peut pas s'appliquer. En effet, pour une variable xi on n'aura que les simplificaions suivantes (réseaux $\overline{R}$ et $\overline{R}^*$ optimisés séparément) :



réseau $\overline{R}^*$ (transistors P)



réseau $\overline{R}$ (transistors N)

Par contre, on peut réaliser une fonction particulière en remplaçant l'alimentation VDD et la masse VSS par une variable et sa variable complémentaire. On obtient à la sortie s d'une porte CMOS une fonction F de la façon suivante : $s = F \times 1 + \overline{F} \times 0$. Avec une variable a en entrée à la place des alimentations, on obtient : $s = F \times a + \overline{F} \times \overline{a} = \overline{F} \oplus a$: OU exclusif disjonction.

Exemple

Soit la fonction $\overline{F} = bc$, on a $F = \overline{bc} = \overline{b} + \overline{c}$ et $\overline{F} = b + c$. Son schéma électrique est donné sur la figue 32 :

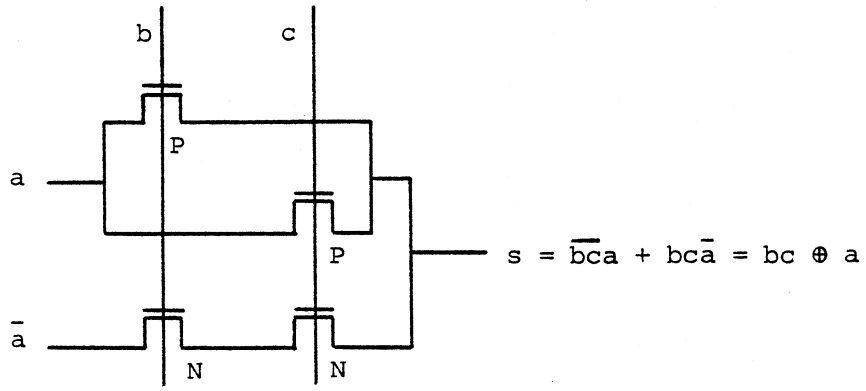


Figure 32 : Fonction particulière réalisée en technologie CMOS avec des variables en entrée à la place des alimentations

L'inconvénient est la consommation statique de la porte suivante. En effet, le niveau logique "1" (respectivement "0") est mal transmis par le réseau $\overline{R}$ (transistors N) (respectivement $\overline{R}^*$ (transistors P)). Le transistor P (respectivement N) attaqué n'est pas bloqué. Dans ces deux cas, un léger courant statique circule dans la porte suivante. On peut :

- soit accepter cette consommation
- soit y remédier en dimensionnant la porte suivante de façon à minimiser ce courant (L >> W)

V - IMPLANTATION DE LA LOGIQUE DE TRANSFERT

On s'intéresse ici à l'implantation sur Silicium d'un réseau réalisé en logique de transfert.

On étudiera :

- la réalisation systématique d'un arbre de décision binaire
- différents types d'implantation.

Enfin, on parlera du passage automatique d'un arbre de décision binaire jusqu'au schéma implanté.

V - 1. Réalisation systématique

On a vu les réalisations systématiques pour la forme canonique (§ II.2) et l'arbre canonique (§ III.3) ; dans ces deux cas, on a un circuit universel et programmable.

A partir d'un arbre de décision binaire simplifié, on obtient aussi directement un réseau lexicographique en immergeant l'arbre sur un quadriallage fictif où les variables d'entrée déterminent les lignes verticales (colonnes), les lignes horizontales aux coudages près (chemins) correspondent aux différents chemins de l'arbre. Dans ce cas, on n'a pas un circuit universel et programmable : chaque arbre simplifié correspond à une fonction particulière.

Exemple

Soit l'arbre simplifié de la figure 31a § IV-3, sa réalisation donne le schéma électrique de la figure 33,réseau lexicographique avec l'ordonnancement des variables x, y, z, t donné par l'arbre.

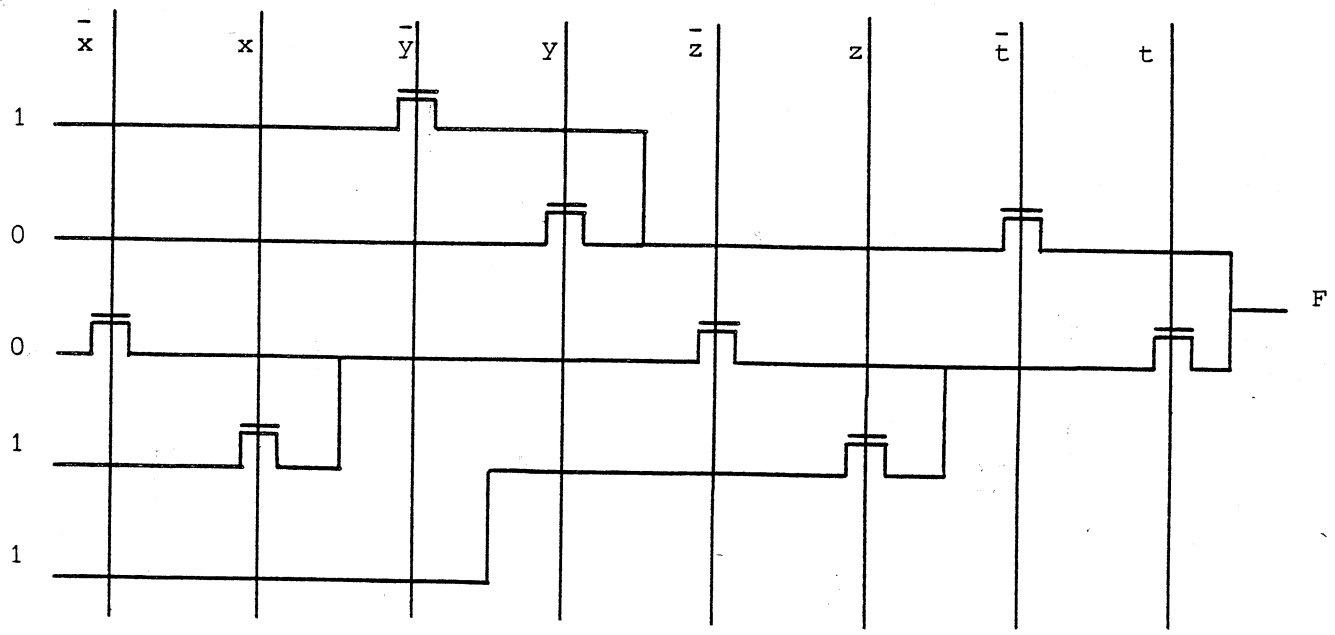


Figure 33 : Schéma électrique obtenu à partir de l'arbre simplifié de la figure 31a

Remarque : Les variables traversent le réseau (colonnes), elles peuvent arriver par le haut ou le bas du réseau suivant les connexions de celui-ci avec le reste du circuit.

V - 2. Différents types d'implantation

On utilise une technologie NMOS à déplétion en charge (un niveau de Silicium polycristallin, pré-contact pour le transistor de charge). Les matériaux employés ont été définis au § III.2 du chapitre I.

L'implantation pourra se faire facilement de façon automatique. Effectivement, l'avantage de la logique de transfert par rapport aux portes complexes au niveau topologique est d'avoir un dimensionnement simplifié : les pass transistors sont des transistors de taille fixe : transistors minimaux (cf. § I.2).

On propose trois types d'implantation (figure 34 a,b,c) : on les traite sur l'arbre simplifié de la figure 31a dont le schéma électrique est donné sur la figure 33.

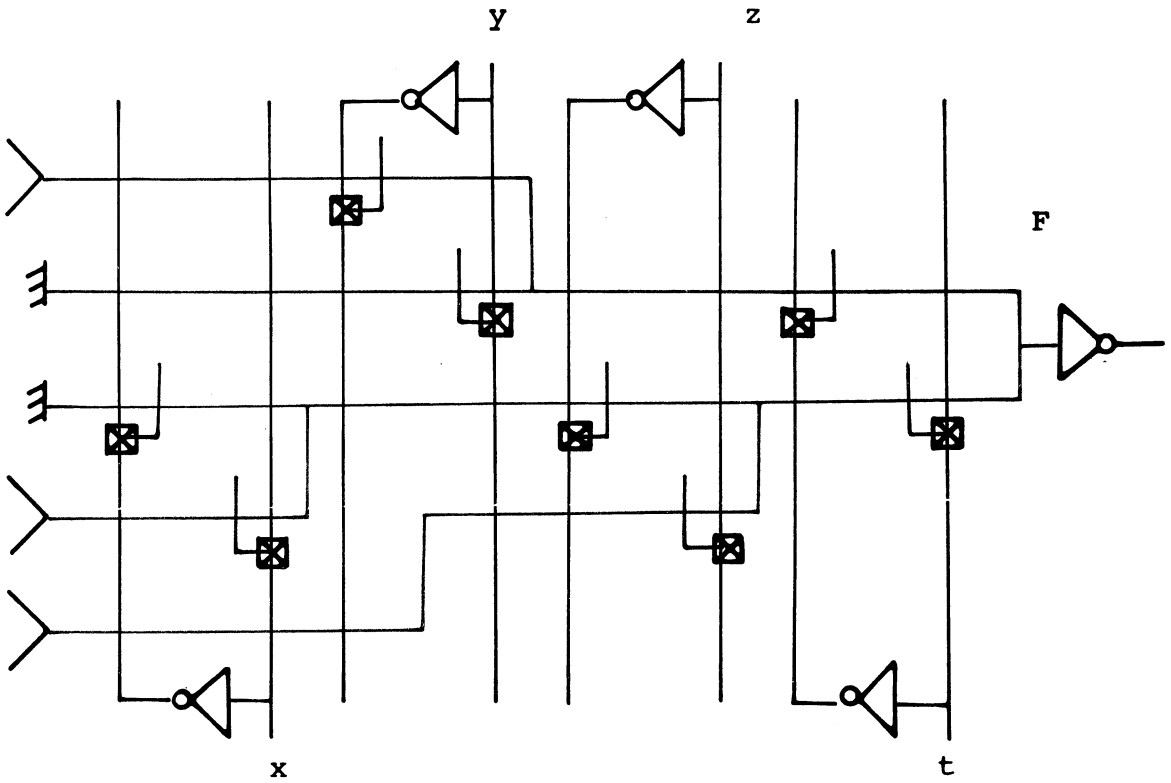


Figure 34 a : 1er type d'implantation

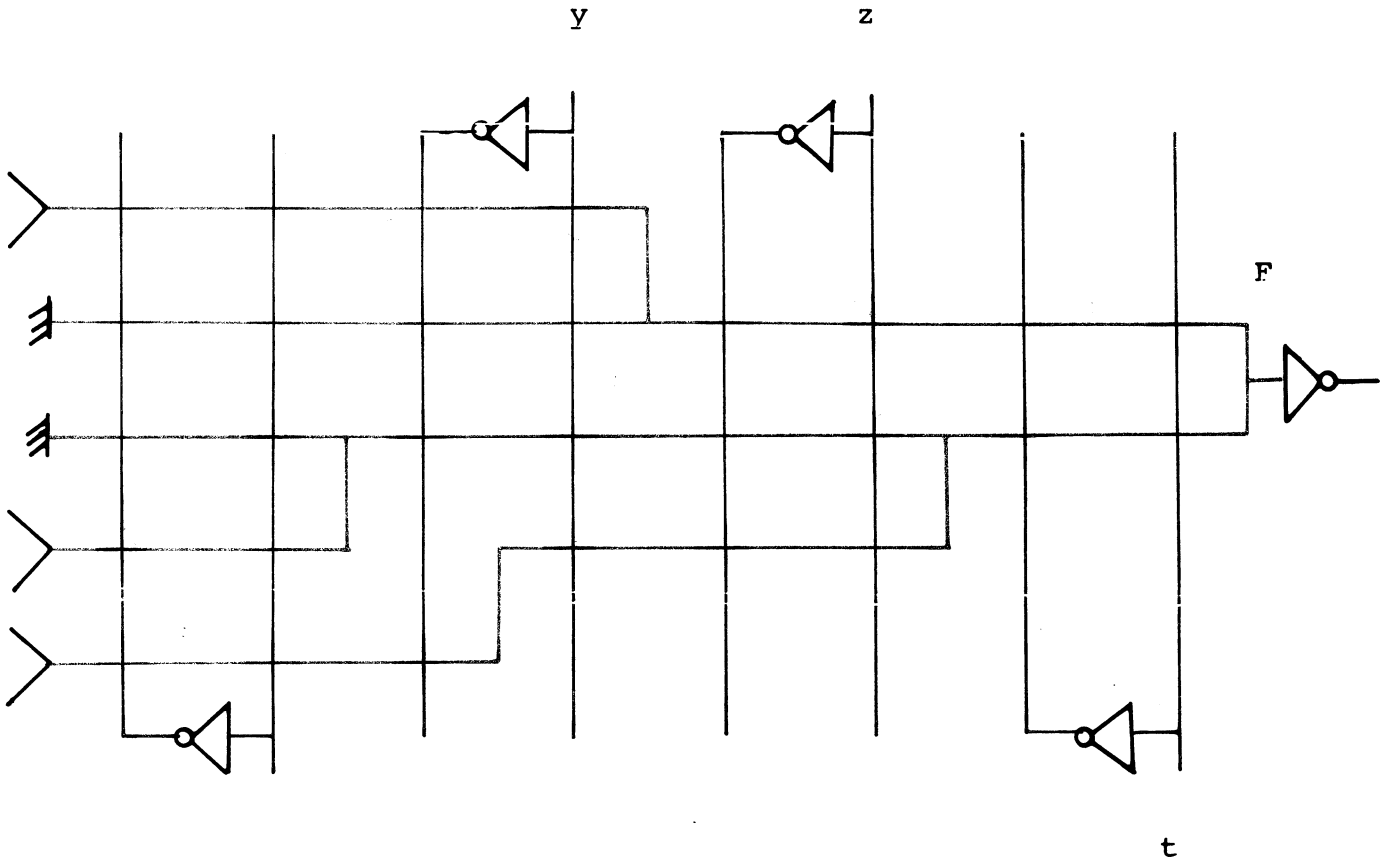


Figure 34b : 2ème type d'implantation

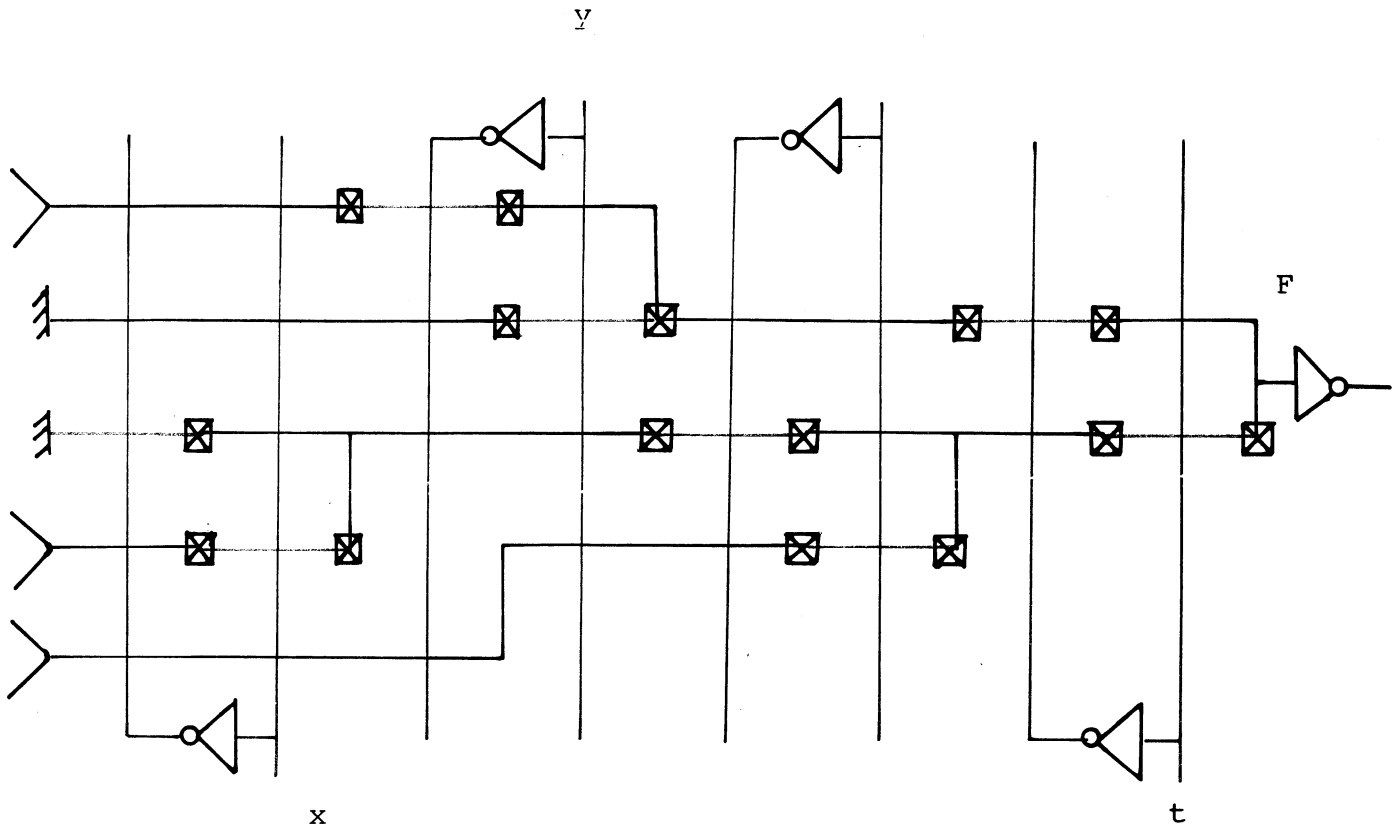


Figure 34c : 3ème type d'implantation

Dans ces trois types d'implantation, les lignes horizontales sont en diffusion N (chemins).

Pour le 1er type, les lignes verticales sont en aluminium (variables d'entrée). S'il existe un transistor, le silicium polycristallin en contact avec l'aluminium (variable d'entrée) viendra au-dessus de la diffusion (figure 34a). Une telle solution est donnée pour les fonctions symétriques de la page 79 du (MEA 80) planche 7a.

Pour les 2ème et 3ème types, les lignes verticales sont en silicium polycristallin (variables d'entrée). On a donc un transistor à chaque croisement silicium polycristallin-diffusion.

S'il n'existe pas de transistor, on peut :

- soit fabriquer un transistor déplété (implantaion ionique, ce transistor est toujours passant (commande "0" ou "1" sur sa grille) : le chemin en diffusion est toujours connecté (figure 34b)). Une telle solution est donné pour la forme canonique à deux variables de la page 77 du (MEA 80) planche 7b.

- soit court-circuiter celui-ci avec l'aluminium (figure 34c).

Suivant les cas, on choisira un schéma implanté parmi ces trois schémas proposés. On voudra avoir les lignes de variables en aluminium ou en silicium polycristallin.

Le 2ème type est intéressant pour le passage de connexions au-dessus du réseau, effectivement on n'a pas utiliser le niveau aluminium. Mais les transistors déplétés introduisent des résistances sur les chemins, la rapidité sera moins grande.

Le 3ème type est très intéressant dans le cas où l'on travaille avec des matrices toutes prêtes : lignes de diffusion et de silicium polycristallin, pour implanter des réseaux à partir de la forme canonique ou de l'arbre canonique de la fonction. En effet, on utilise le niveau aluminium pour :

- court-circuiter les transistors
- relier entre elles les différentes cellules à quatre ou cinq variables.

Comme pour les réseaux pré-diffusés, seul le niveau aluminium est à ajouter.

Remarque : La logique de transfert est très intéressante d'un point de vue topologique :

- l'implantation est facile, cf les trois schémas implantés précédemment (pas de dimensionnement des transistors)
- les blocs sont déformables : suivant la place dont on dispose, les réseaux peuvent être implantés différemment : on peut tordre les lignes de diffusion, imbriquer les réseaux.

V - 3. Implantation automatisée

Les étapes qui font passer d'un arbre de décision binaire canonique ou non au schéma implanté de son arbre simplifié sont programmables, l'implantation peut être obtenue de façon automatique (pas de dimmensionnement des transistors). Une telle synthèse est complètement sûre (pas d'introduction d'erreurs de conception). On obtient une structure régulière, compacte dont l'intérêt est connu dans les circuits V.L.S.I., avec les PLA, RAM, ROM.

VI - EXTENSION A DES SYNTHESES A PLUSIEURS COUCHES

Certaines fonctions seront réalisées à partir de plusieurs fonctions réalisées en logique de transfert :

- si la réalisation d'une fonction unique nécessite plus de quatre à cinq transistors en série dans son réseau,
- si on a plusieurs fonctions des même variables, on cherchera une minimisation multi-fonctions.

La décomposition d'une fonction en plusieurs fonctions est un problème très complexe lorsqu'il s'agit de réaliser une fonction par une porte (cf § VI du chapitre I). L'étude effectuée ici se fait sur des fonctions réalisées en logique de transfert, donc sur l'arbre de décision binaire : ce problème est donc différent.

VI - 1. Composition de fonctions réalisées en logique de transfert

Précédemment, on a utilisé l'arbre de décision binaire comme représentation d'une fonction booléenne. Ici, on propose de faire une étude logique sur l'arbre de décision binaire : pour étudier la composition de fonctions réalisées en logique de transfert, on étudiera la composition d'arbres de décision binaire.

L'arbre de décision binaire associé à une fonction F ne représente pas que cette fonction, mais des tas de fonctions fi. En effet, le noeud initial désigne la fonction F, chaque noeud intermédiaire introduit entre le noeud initial et les feuilles terminales désigne une fonction fi.

Exemple

Soit l'arbre canonique de la figure 10, § III.2, les fonctions associées à chaque noeud sont notées sur l'arbre de la figure 35.

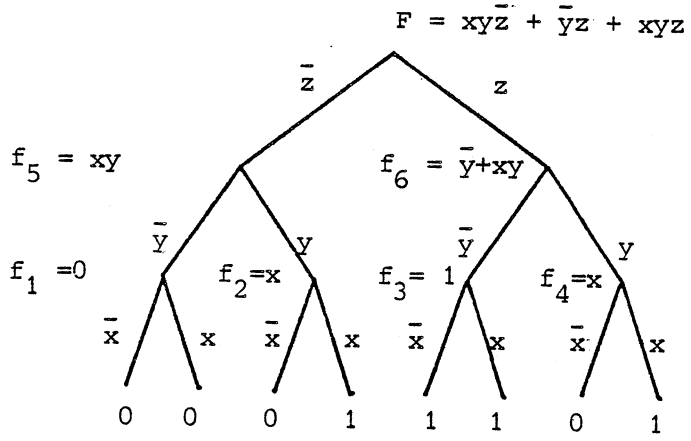


Figure 35 : Fonctions représentées par un arbre canonique

VI - 2. Décomposition d'une fonction trop complexe

Ici, on cherche à minimiser le nombre de transitors et à avoir une implantation facile pour le ou les réseaux réalisés en logique de transfert représentant la fonction, tout en respectant la contrainte suivante : le nombre maximum de transistors en série sera de quatre ou cinq.

Soit une fonction à plus de quatre ou cinq variables exprimée sous forme canonique ou simplifiée, représentée par son arbre de décision binaire canonique ou non. Cette fonction n'est pas réalisable en logique de transfert par un seul réseau. On cherchera une décomposition de cette fonction en plusieurs foncions en travaillant sur l'arbre de décision binaire.

Exemple

Soit la fonction F à 6 variables F(x1,x2,x3,x4,x5,x6) dont l'arbre canonique est donné sur la figure 36.

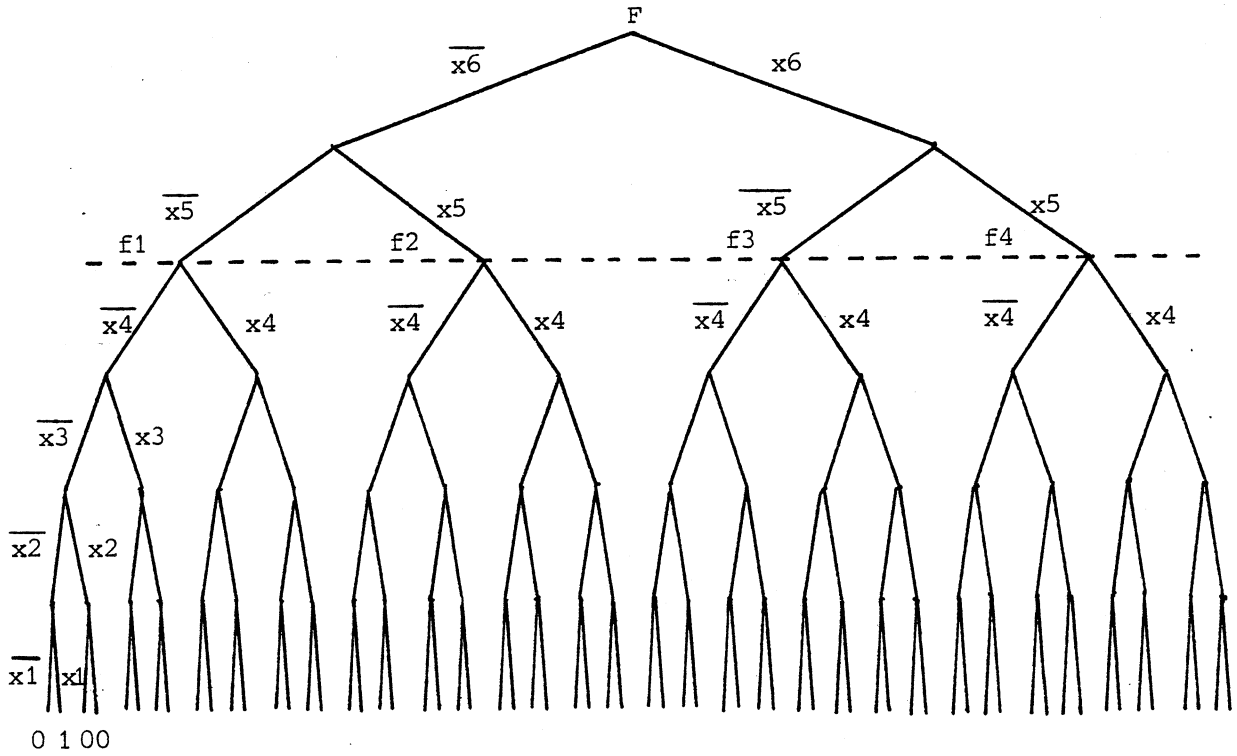


Figure 36 : Arbre canonique représentant une fonction à 6 variables

On désire avoir quatre transistors en série au maximum. Pour cela, on coupera l'arbre de décision binaire à un certain niveau. Sur l'exemple, on réalisera des fonctions de x1,x2,x3,x4 puis la fonction F de x5,x6.

On propose la méthode suivante :

- on construit l'arbre de décision binaire à partir de la forme canonique ou simplifiée de la fonction,
- on coupe cet arbre à un certain niveau pour n'avoir que quatre transistors en série dans les réseaux réalisés en logique de transfert,
- on simplifie alors les arbres obtenus.

Remarque : On travaille sur l'arbre canonique ou non, mais sur lequel on n'a pas appliqué les règles de simplification. En effet, l'arbre simplifié est difficile à couper (en particulier lorsque la 3ème règle est appliquée). On effectue les simplifications sur les arbres obtenus.

Sur l'exemple précédent, on coupe l'arbre entre les variables x4 et x5. On simplifie ensuite les cinq arbres obtenus. La réalisation de la fonction F se fera à l'aide d'une synthèse à deux couches (figure 37).

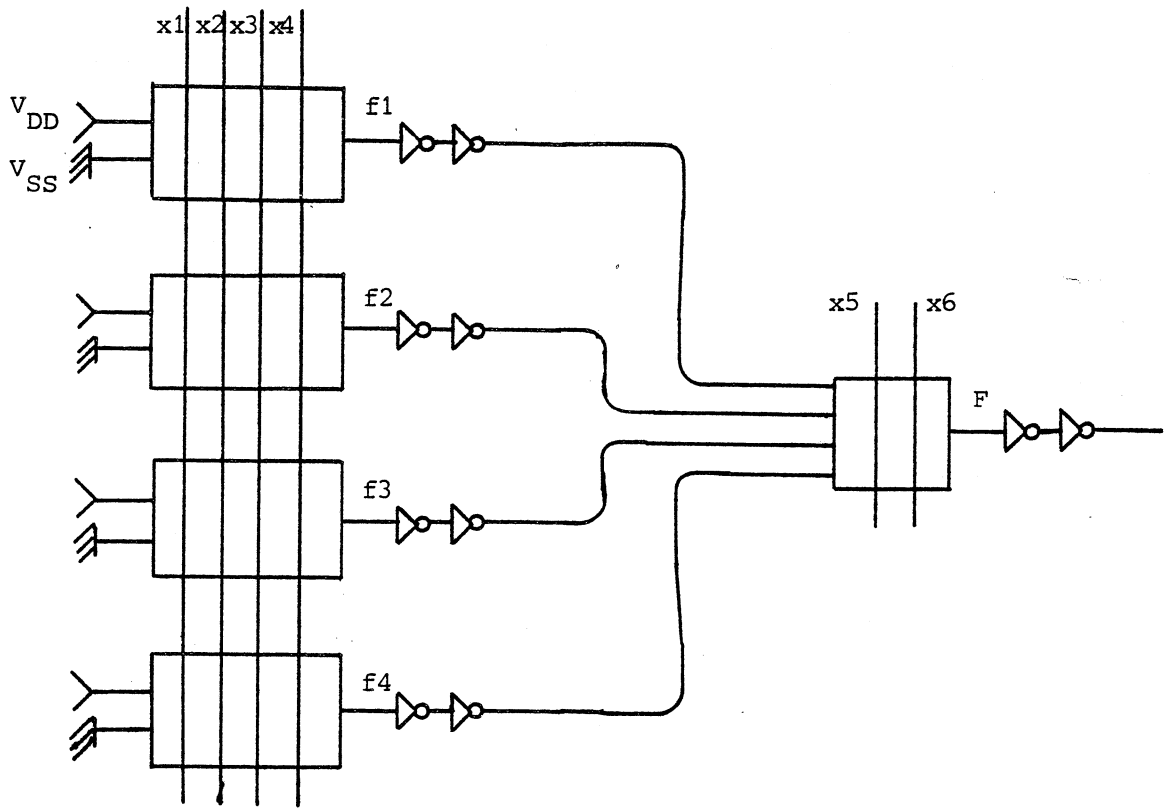


Figure 37 : Réalisation d'une fonction F à 6 variables

Les blocs de la figure 37 sont des réseaux de transistors associés aux arbres de décision binaire simplifiés représentant :

- les fonctions f1, f2, f3, f4 des variables x1, x2, x3, x4 avec les valeurs "0" et "1" à l'entrée des chemins,
- la fonction F des variables x5, x6 avec les fonctions f1, f2, f3, f4 à l'entrée des chemins.

On peut réaliser la 1ère couche de fonctions avec de la logique de transfert à variable privilégié.

Soit quatre : le nombre maximum de transistors en série accepté, ce qui correspond à quatre variables sur l'arbre de décision binaire. Avec une synthèse à deux couches, on peut réaliser des fonctions de :

- 8 variables avec "0" et "1" à lentrée des fonctions fi
- 9 variables avec une variable privilégiée à l'entrée des fonctions fi.

Il est important de couper la ligne capacitive (présence des inverseurs entre les blocs), ce qui permet aussi de régénérer le signal (§ I.2).

Remarque : Comme au § IV.1, on peut faire l'analogie entre la réalisation de fonctions logiques avec de la logique de transfert et celle à base de multiplexeurs dans des synthèses à plusieurs couches (BLA 79) (LOT 79)

VI - 3. Réalisation de plusieurs fonctions des mêmes variables

Il s'agit de minimiser le nombre de transistors et à avoir une implantation facile pour un réseau à plusieurs sorties, réalisé en logique de transfert, représentant plusieurs fonctions des mêmes variables.

Soient plusieurs fonctions des mêmes variables exprimées sous formes canoniques ou simplifiées, représentées par leurs arbres de décisions binaires canoniques ou non. On cherchera des sous-fonctions communes à ces fonctions à réaliser, ce qui correspond à chercher des sous-arborescences communes sur les arbres de décisions binaires associés : on obtiendra des sous-réseaux communs aux différents réseaux de transistors réalisant ces fonctions. Effectivement un arbre de décision binaire représente des tas de fonctions (§ VI.1), les sous-arborescences associées peuvent être utilisées pour d'autres fonctions. AKERS a utilisé cette technique : en particulier, il a étudié les différentes fonctions d'une unité arithmétique et logique (U.A.L. à 14 entrées, 8 sorties (AKE 79).

On propose la méthode suivante :

- on construit les arbres de décision binaire des fonctions à partir de leurs formes canoniques ou simplifiées, on simplifie ces arbres.
- on cherche les sous-arborescences communes à ces différents arbres simplifiés.

- on construit alors l'arbre complet représentant ces différentes fonctions avec les sous-arborescences communes.

Exemple

Soient les trois fonctions F0, F1, F2 à deux variables x1, x2 représentées par les arbres simplifiés de la figure 38.

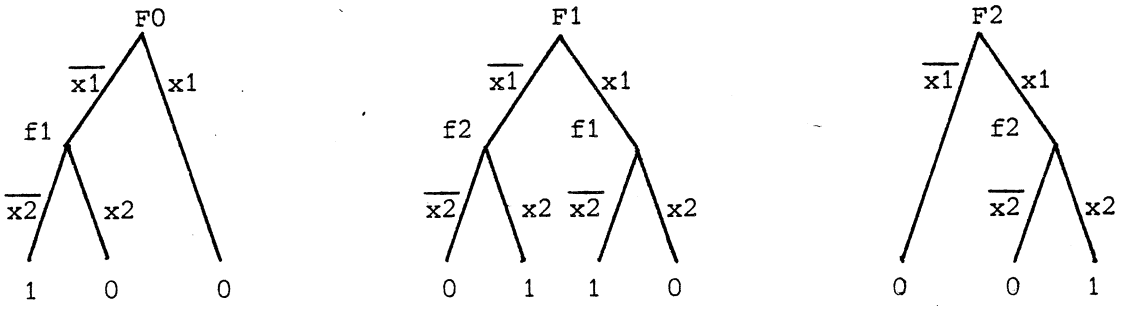


Figure 38 : Arbres simplifiés représentant les trois fonctions F0, F1, F2

On a deux sous-arborescences communes à ces différents arbres que l'on associe aux fonctions f1 et f2.
On construit l'arbre complet représentant ces différentes fonctions avec les sous-arborescences communes (figure 39).

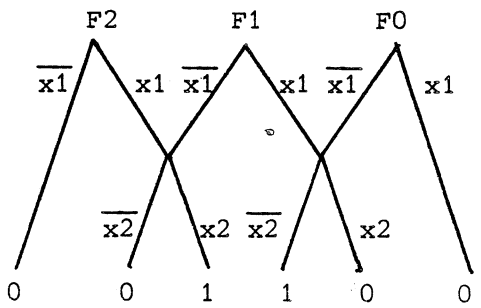


Figure 39 : Arbre complet représentant les trois fonctions F0, F1, F2

On obtient le schéma électrique de la figure 40.

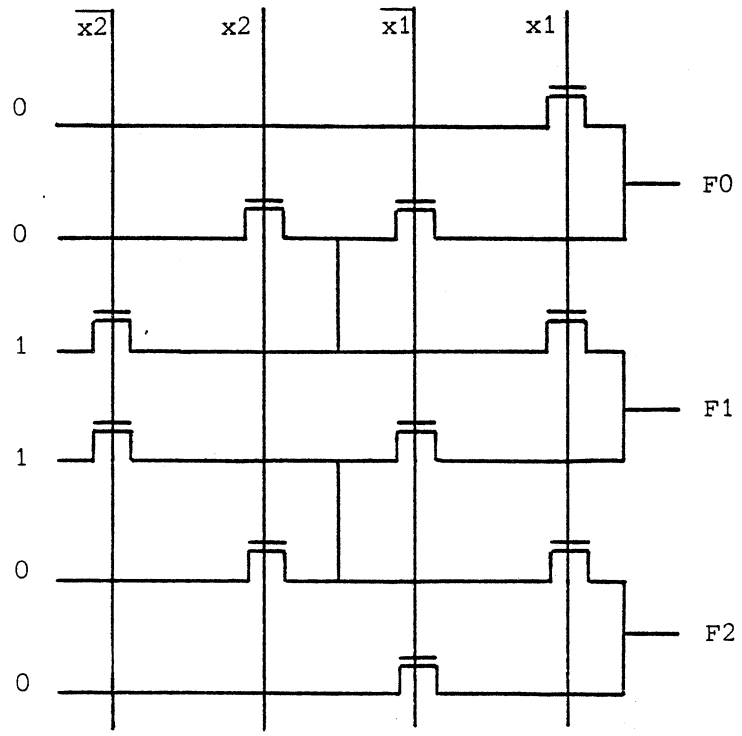


Figure 40 : Schéma électrique obtenu à partir de l'arbre complet de la figure 39

En fait, la réalisation imbriquée de ces trois fonctions n'est pas le fruit du hasard. Il s'agit de trois fonctions booléennes symétriques. Un exemple de réalisation imbriquée est donné pour quatre fonctions symétriques de trois variables en (MEA 80) page 79.

Chapitre III

CONCEPTION DE CIRCUITS COMPLEXES

INTRODUCTION

Une méthode dite de conception progressive par affinements successifs de spécifications a été définie dans (SAU 82). Les spécifications du circuit deviennent de plus en plus précises, par suite des choix du concepteur jusqu'à arriver au produit final. Il s'agit au cours de cette démarche de "prouver" qu'aucune erreur de conception n'a été introduite, c'est à dire que l'on ne s'est pas écarté de l'objectif initial.

Cette démarche s'applique à tout circuit ou partie de circuit exécutant un certain enchaînement de micro-opérations décrit par un algorithme. On dit que ces circuits interprètent un certain algorithme. Ceci est vrai pour tous les circuits du type microprocesseur (spécialisé ou d'usage général), les circuits de calcul, les circuits de contrôle. Cette démarche ne s'applique pas aux circuits dont la complexité est uniquement due à la répétitivité (PLA, mémoire) et dont les fonctions sont soit combinatoires, soit trop simples (fonctions lire ou écrire).

Nous nous proposons de reprendre cette démarche de conception, de rappeler ses étapes et de l'appliquer pour la conception de deux circuits (extracteur de racine carrée, multiplieur) réalisés en technologie NMOS et d'étudier spécialement les problèmes temporels de ces circuits. L'un de ces circuits sera étudié en logique statique, l'autre en logique biphasée.

I - LES SPECIFICATIONS INITIALES OU CAHIER DES CHARGES DU CIRCUIT

Les spécifications initiales d'un circuit intégré se décomposent en spécifications fonctionnelles (fonctions que le circuit doit réaliser) et un ensemble de contraintes à respecter (SAU 82).

I - 1. Spécifications fonctionnelles initiales

Il s'agit d'une spécification du type comportemental du circuit donnant la ou les fonctions usagers que le circuit devra offrir. Cette description est bien sûr indépendante de la structure interne qui est encore non définie. Ce type de spécification, outre les données des fonctions, précise en général les formats des données et des sorties. Suivant le type de circuits, ces fonctions sont

spécifiées, soit en langage parlé, soit mathématiquement, soit par des techniques de descriptions graphiques etc.

Exemple I

On désire réaliser un circuit calculant la partie entière de la racine carrée d'un nombre entier positif en base 2 sur 8 bits (THU 82).

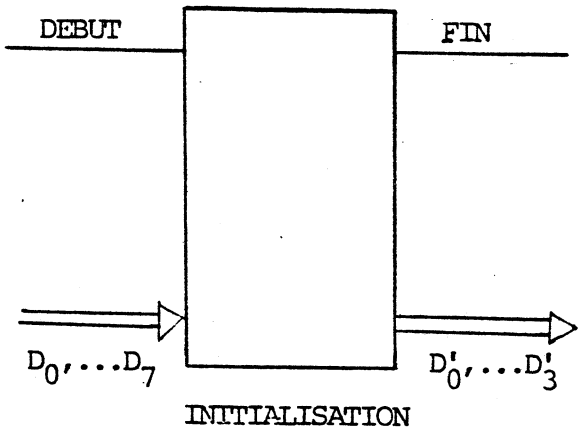


Figure 1 : Schéma général de l'extracteur de racine carrée

Nous avons 8 bits d'entrée DO...D7, ainsi qu'un signal DEBUT
4 fils de sorties D'O...D'3 ainsi que le signal FIN.

Un signal INIT permettra d'initialiser les bascules de contrôle de ce circuit.

Exemple II

On désire réaliser un multiplieur effectuant la multiplication de deux nombres entiers positifs de 4 bits.

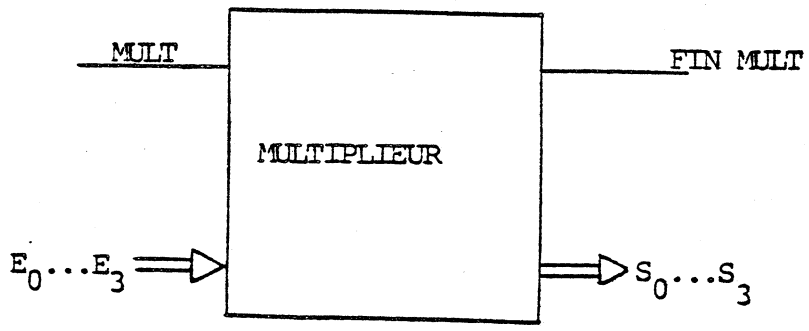


Figure 2 : Schéma général du multiplieur

On charge les 4 bits du multiplicande E0...E3, puis les 4 bits du multiplieur E'0...E'3.
On sortira d'abord les 4 bits de poids faibles du résultat S0...S3, puis les 4 bits de poids forts S'0...S'3.

multiplicande multiplicateur

On a donc 2 x 4 bits d'entrée (E0.....E3 E'0.....E'3), 2 x 4 bits de sortie (S0...S3), un signal d'initialisation MULT et le signal de fin de multiplication FIN MULT.

I - 2. Les contraintes

Les contraintes auxquelles le circuit intégré devra répondre sont variées et dépendent du type de circuit ainsi que de l'application envisagée. Les contraintes les plus courantes sont :

- les performances exprimées, en général, par les bornes maximales des temps d'exécution des fonctions du circuit,
- la surface et l'encombrement,
- le coût comprenant le coût de la conception,
- la fiabilité de la conception liée au nombre de phases de conception automatisables pouvant en conséquence être classées comme complètement sûres,
- la testabilité ; ce mot traduit en fait d'une part la possibilité d'identifier rapidement une erreur de conception (non conformité aux spécifications initiales), d'autre part, la possibilité de tester et de

localiser rapidement une défaillance physique dans un circuit. Le premier point concerne la validation de prototype, le deuxième le test en fin de fabrication,

- la consommation et la dissipation d'énergie.

II - PRINCIPE D'UNE METHODE DE CONCEPTION DESCENDANTE

II - 1. Principe général

Les méthodes de conception descendante largement utilisées en informatique (logiciel et matériel) ainsi qu'en automatique consistent à concevoir un système en s'appuyant sur une succession de spécifications obtenues par affinements successifs à partir d'une spécification initiale.

On passe d'une spécification d'un niveau donné à une spécification de niveau inférieur en explicitant (ou interprétant) une ou plusieurs fonctions en termes de primitives plus simples jusqu'à arriver aux primitives matérielles considérées comme des entités de base. Une telle méthode est dite une méthode de conception sûre si le passage d'un niveau à un niveau plus fin est soit "prouvé" (méthode complètement sûre), soit partiellement validé. Il s'agit de démontrer qu'au cours de ce passage aucune erreur ou non conformité aux spécifications initiales n'a été introduite.
Un passage entre deux niveaux implique, en général, un choix de conception, c'est à dire la décision du concepteur de choisir parmi un ensemble de solutions qu'il est capable d'énumérer. Dans ce cas, on cherche à contrôler ce choix et à l'optimiser par rapport aux contraintes énumérées lors des spécifications initiales (SAU 82).

II - 2. Cas de la conception d'un circuit

Dans la démarche que nous proposons, trois niveaux de spécifications sont considérés, traduisant la prise en compte de trois choix du concepteur ; ces choix sont :

- le choix d'algorithmes de calcul, cette étape peut ne pas exister pour certains circuits,
- le choix du chemin de données,
- le choix de la structure de contrôle.

A chaque niveau, la spécification du circuit permet de contrôler et d'optimiser le choix du concepteur. Ces étapes sont représentées dans l'organigramme de la figure 3. Il est clair qu'en pratique, cette démarche n'est pas linéaire. Il faut également remarquer que la décomposition en partie contrôle et partie opérative est récursive, en particulier la partie contrôle sera elle-même décomposée en partie contrôle et partie opérative (microséquenceur complexe). Pour simplifier l'exposé de la méthode nous ne traiterons qu'un niveau de décomposition.

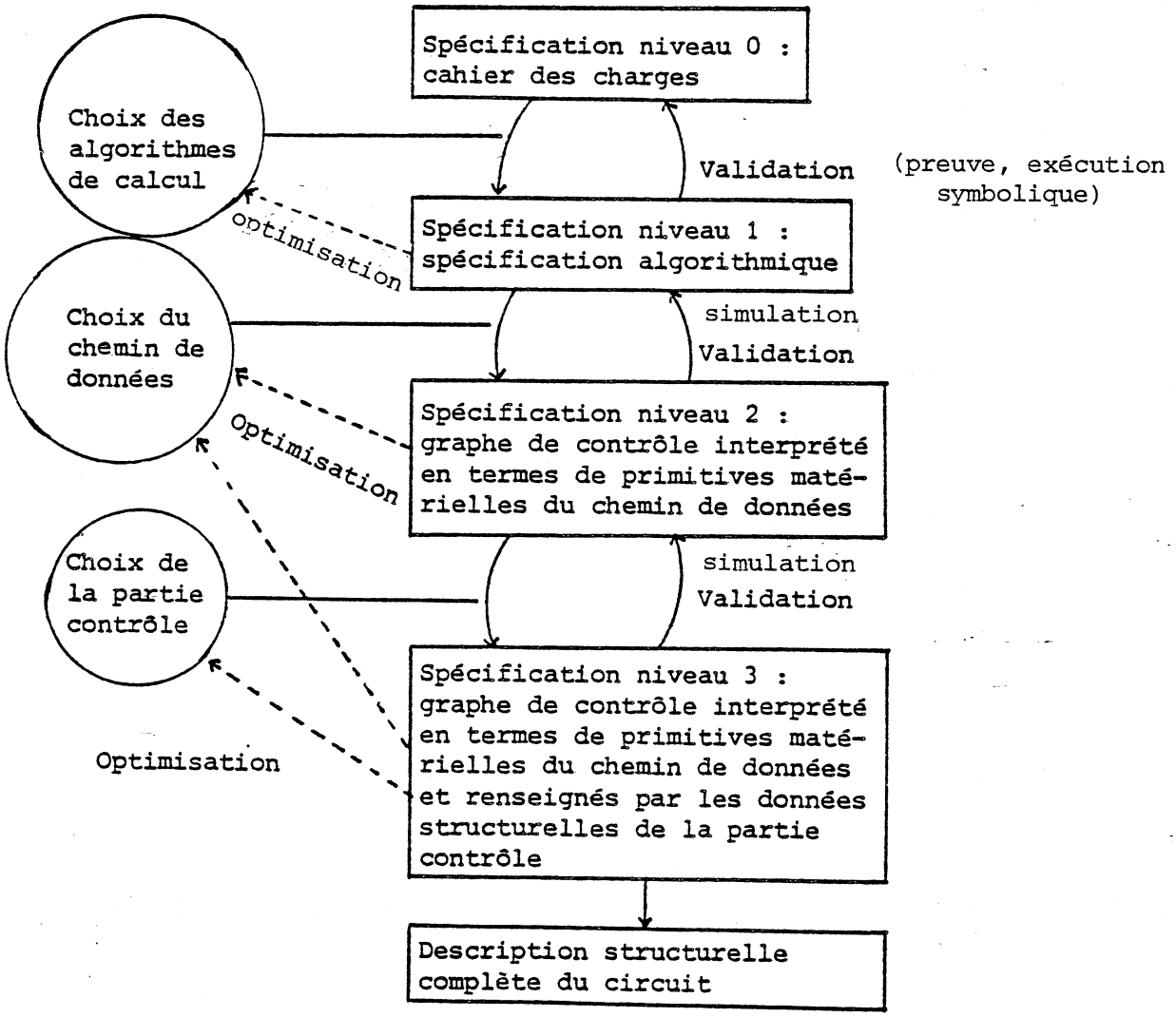


Figure 3 : Etapes de la conception

III - SPECIFICATIONS ALGORITHMIQUES

Ce premier niveau de spécification consiste à décrire les fonctions d'entrée-sortie d'un circuit sous forme d'un algorithme de calcul en termes de variables internes et de fonctions intermédiaires. Il s'agit bien d'expliciter ou d'interpréter la fonction globale en terme de fonctions plus simples. A ce niveau, les fonctions ne sont pas forcément des primitives matérielles. Ce type de description peut être représenté graphiquement. Il met en évidence le parallélisme maximal (compatible avec l'ordre logique du calcul) (SAU 82).

Un grand choix d'algorithmes peut alors se présenter au concepteur. Un algorithme sera caractérisé par :

- les opérations utilisées (opérations effectuées sur les variables) ; ces opérations devront se traduire par des primitives matérielles (opérateurs), donc une certaine complexité :

  - difficulté de conception
  - temps de traversée
  - surface

- le nombre de pas de l'algorithme (nombre de passages globaux et locaux) donc une certaine performance globale.

Exemple I

Dans le circuit d'extraction de racine carrée, on peut utiliser les algorithmes de calcul donnés en pseudo-algo :

Algorithme 1 : $\sqrt{y}$

```
début
     entiers i, y ;
     lire (y)// i:=0 ;
     tant que i * i ≤ y faire i := i+1 ;
     i := i-1 ;
     écrire (i) ;
fin.
```

Algorithme 2 : $\sqrt{y}$

```
début
    entiers u, x, y, z ;
    lire (y) ;
    x := y// u:= 1 ;
    tant que u ≤ x faire u := u mult 4 ;
    z := u ;
    tant que u > 1 faire
      début
        z := z div 2 - u div 4 // u := u div 4
        si z ≤ x alors
           début
             x := x - z // z := z + u mult 2
           fin
      fin
    écrire (z div 2) ;
fin.
```

	Opérateurs	Algorithme en	Nombre maximal de pas avec le parallélisme maximal y sur n bits
Algorithme 1	Multiplication Comparaison Incrémentation Décrémentation	$\sqrt{y}$	n = 4 : 11 n = 8 : 35 n = 16 : 515
Algorithme 2	Multiplication par 2 ou 4 Division par 2 ou 4 Addition Soustraction Comparaison	$\log_2 \sqrt{y}$	n = 4 : 22 n = 8 : 34 n = 16 : 108

L'algorithme choisi ici est l'algorithme 2. En effet, les multiplications et divisions par 2 seront réalisées par un décalage droite ou gauche de 1 ou 2 bits. Nous sommes donc ramenés à des opérateurs très simples (addition, soustraction logique à décalage). On constate que l'on introduit les variables internes u x ...

Un multiplieur est nécessaire à la réalisation de l'algorithme 1. L'algorithme choisi est représenté sous forme de graphe : à chaque noeud du graphe est associé le graphe de données explicitant le type d'opérateurs utilisé et les variables sources et puits (figure 4).

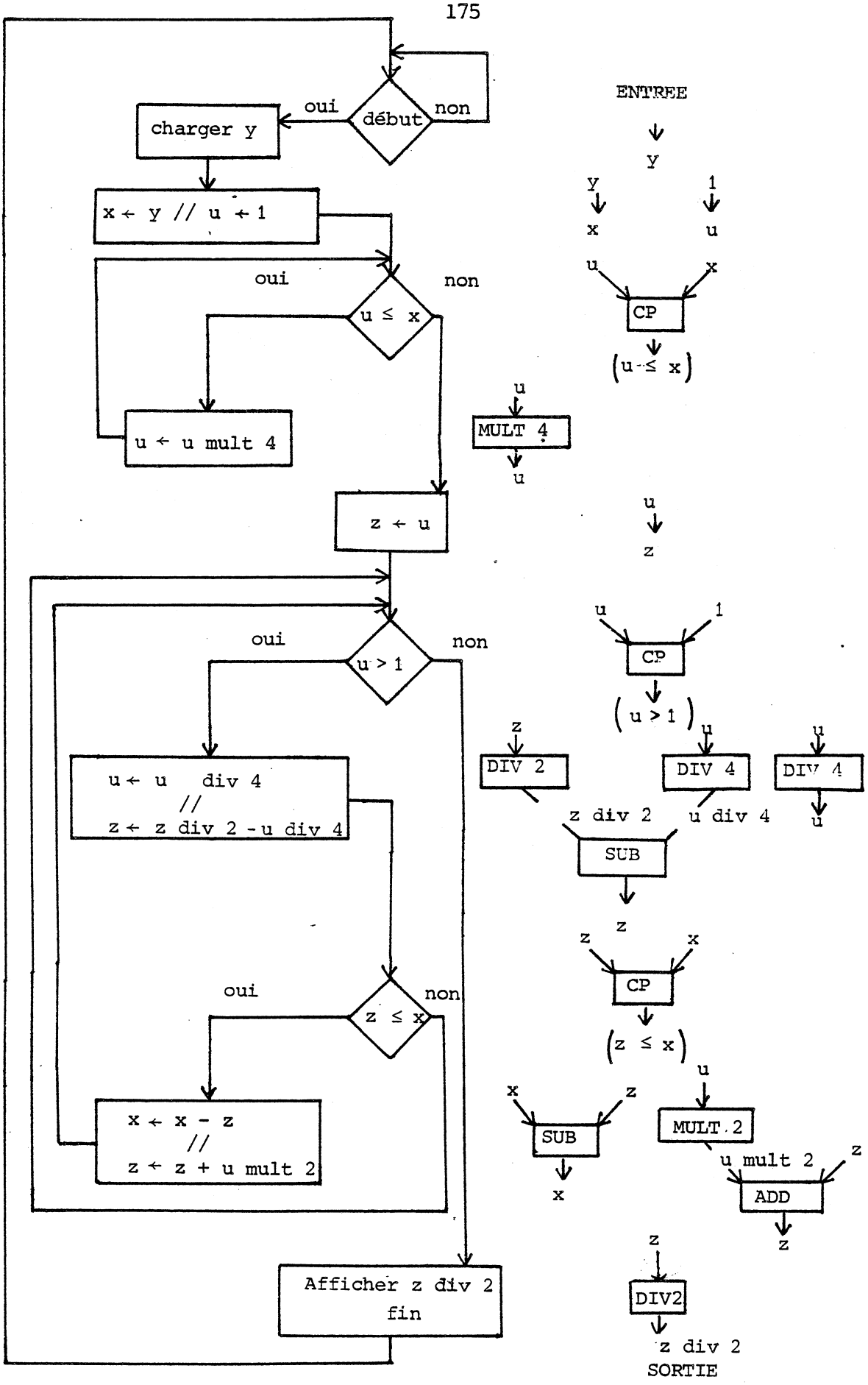


Figure 4 : Graphe représentant l'algorithme

Exemple II

Pour le circuit de multiplication, on peut utiliser l'algorithme de calcul suivant :

```
x = A ; y = B ; p := 0

tant que x = 0 faire
     si impair (x) alors p := p + y ;
     y := y * 2 ; x = x div 2
fin
```

Cet algorithme est constitué d'une boucle "tant que" exécutée n fois ; à l'intérieur de cette boucle, doit être effectuée une opération conditionnelle suivie d'une deuxième opération.

Remarque : Ce premier niveau de spécifications peut ne pas exister pour certains ciruits soit qu'il ne présente pas d'intérêt (circuit trop simple) soit que ce niveau est fusionné avec le niveau suivant (circuit à très haut parallélisme par exemple). La conception débutera donc immédiatement au niveau suivant.

IV - CHOIX DE LA STRUCTURE DU CHEMIN DE DONNEES ; SPECIFICATION DU CIRCUIT SOUS FORME DE GRAPHE DE CONTROLE INTERPRETE

IV - 1. Choix du chemin de données

Le concepteur définit la structure du chemin de données à savoir les éléments de mémorisation, les opérateurs qui seront les primitives fonctionnelles de base ou microopérations élémentaires, les structures de communication ainsi que la technologie employée. Le concepteur cherche une solution respectant les spécifications fonctionnelles de plus haut niveau (validation) ainsi que les contraintes du cahier des charges par rapport auxquelles il recherchera une optimisation (SAU 82). L'ensemble des solutions considérées pour le chemin de données ou partie opérative (P.O) peut être obtenu :

- soit relativement intuitivement. L'expérience préalable du concepteur est en jeu et on parlera de styles de partie opérative (S.E 81). Ces styles sont déduits des architectures d'ordinateur classique.

Les structures mentionnées dans (S.E 81) sont les structures à communcations privées, les structures à bus, les structures pipe-line ;

- soit de façon plus structurée en partant des spécifications de niveau supérieur et en privilégiant une des contraintes. On peut par exemple privilégier les performances du circuit et analyser les différentes solutions dans un ordre de performances décroissant (EL 79). Ceci est particulièrement intéressant si l'on dispose de spécifications algorithmiques. On recherche alors d'abord une solution respectant le parallélisme maximal. Pour cela des ressources suffisantes (élément de mémorisation, bus, module fonctionnel) sont affectées matériellement à des variables ou fonctions de façon à respecter ce parallélisme maximal (EL 79). Les solutions sont alors examinées par rapport aux autres contraintes (coût, place ...) en diminuant progressivement les ressources matérielles (restrictions du parallélisme) jusqu'à obtenir une solution satisfaisante par rapport aux autres contraintes. Dans ce cas, les styles énumérés précédemment (S.E 81) seront balayés dans un ordre bien défini (pipe-line, structure distribuée avec communication privée, structure à bus ...). A un niveau beaucoup plus fin, les propriétés de symétrie des fonctions sont soigneusement analysées et conduisent souvent à des structures répétitives par tranches de bit (structure bit slice). Ceci facilitera les procédures de vérification.

IV - 2. Outil de spécification permettant le choix du chemin de données : le graphe de contrôle interprété

Principe général

Une structure du chemin de données étant définie, l'algorithme de calcul ou les fonctions usagers sont décrits ou "interprétés" en terme de primitives de ce chemin de données. Le fonctionnement du circuit se traduit par un graphe dans lequel :

- chaque fonction usager est décrite par un chemin décomposé en une séquence de phases,
- à chaque noeud ou phase est associé l'ensemble des microopérations effectuées dans la partie opérative décrit à un niveau transfert de registre ainsi que l'ensemble des chronogrammes, des signaux de contrôle émis par la partie contrôle pour déclencher et contrôler ces microopérations,

- les arcs représentent le séquencement entre les phases et portent comme renseignement les variables testées au cours de branchement (compte-rendu de partie opérative, variable externe ...).

De telles représentations ont été largement commentées dans (EL 81)(AMB 81) et se prêtent bien à des descriptions multiniveaux (graphe de macrophases, chaque macrophase étant elle-même un réseau de phases).

Exemple I

Pour le ciruit d'extraction de racine carrée, on distinguera :

(i) Les architectures sans minimisation de ressources matérielles, respectant des possibilités maximales de parallélisme et de pipe-lining. La solution extrême consiste à affecter à chaque opération un opérateur muni de ses entrées-sorties et à implémenter une succession d'opérations élémentaires (à l'intérieur d'une boucle) par une réalisation itérative. L'architecture est alors une copie de l'algorithme de calcul (figure 4). La réalisation peut être soit entièrement combinatoire (figure 5, 6) soit partitionnée en modules séparés par des points de mémorisation (figure 7), qui constituent des barrières temporelles et nécessitent donc l'introduction d'un niveau de contrôle. Le contrôle spécifiera l'écoulement des données, un pipe-line maximal est possible.
Cette solution apparemment coûteuse (en surface) pourra être souhaitable pour les circuits très rapides (dynamiques) et à très fort pipe-lining.

(ii) Les architectures à ressources matérielles minimisées respectant le parallélisme maximal, l'algorithme précédent étant exprimé avec un parallélisme maximal, des opérations identiques pouvant être exécutées en parallèle se traduiront par des opérateurs distincts et des voies de communication distinctes. On pourra donc rechercher une architecture minimisant les ressources matérielles et respectant cette contrainte.

(iii) Les architectures à ressources matérielles minimisées restreignant le parallélisme fonctionnel maximal. Les opérateurs ne sont, non seulement pas répliqués, mais des opérateurs distincts peuvent être regroupés sur une ressource matérielle unique. Le cas extrême est le cas où tous les

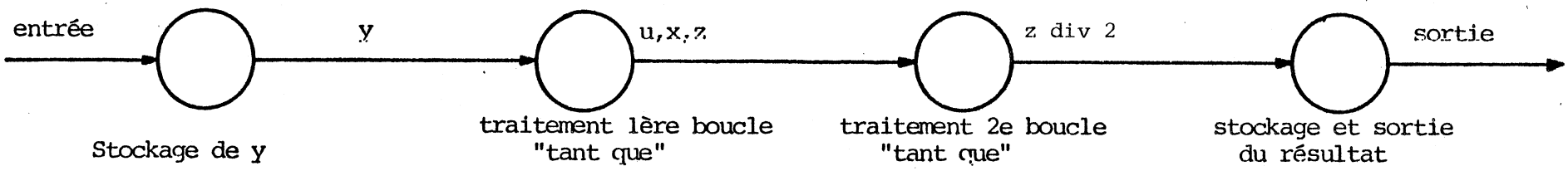


Figure 5. Schéma de principe de la réalisation itérative et de la réalisation pipe-line

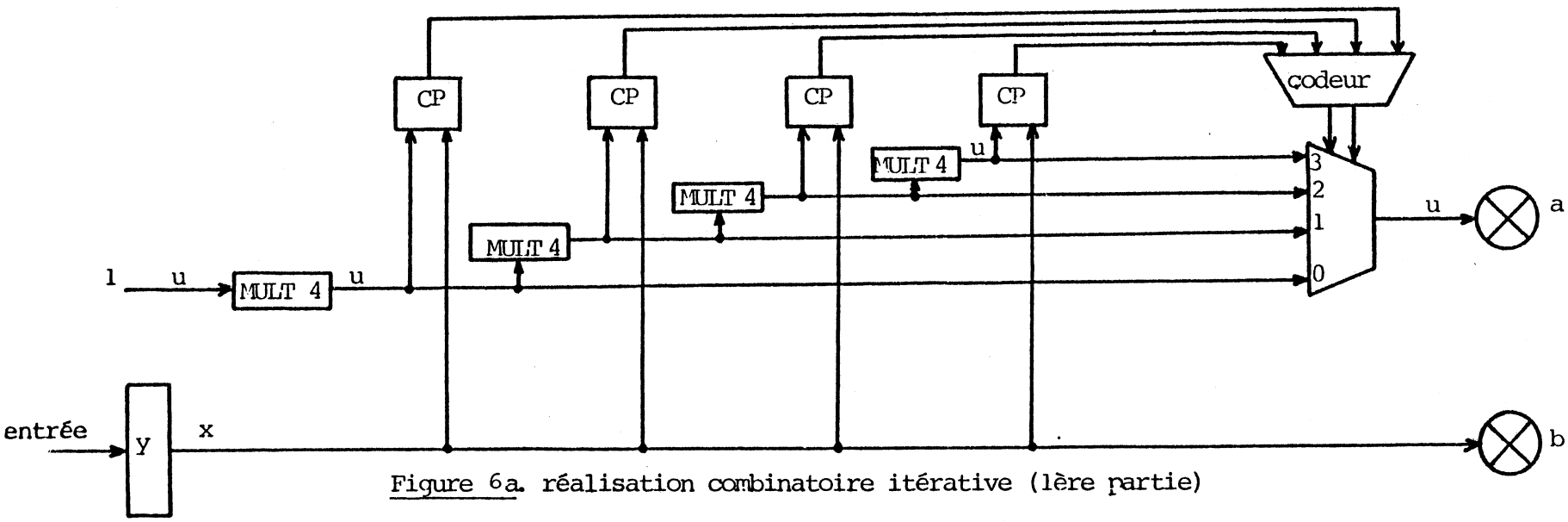
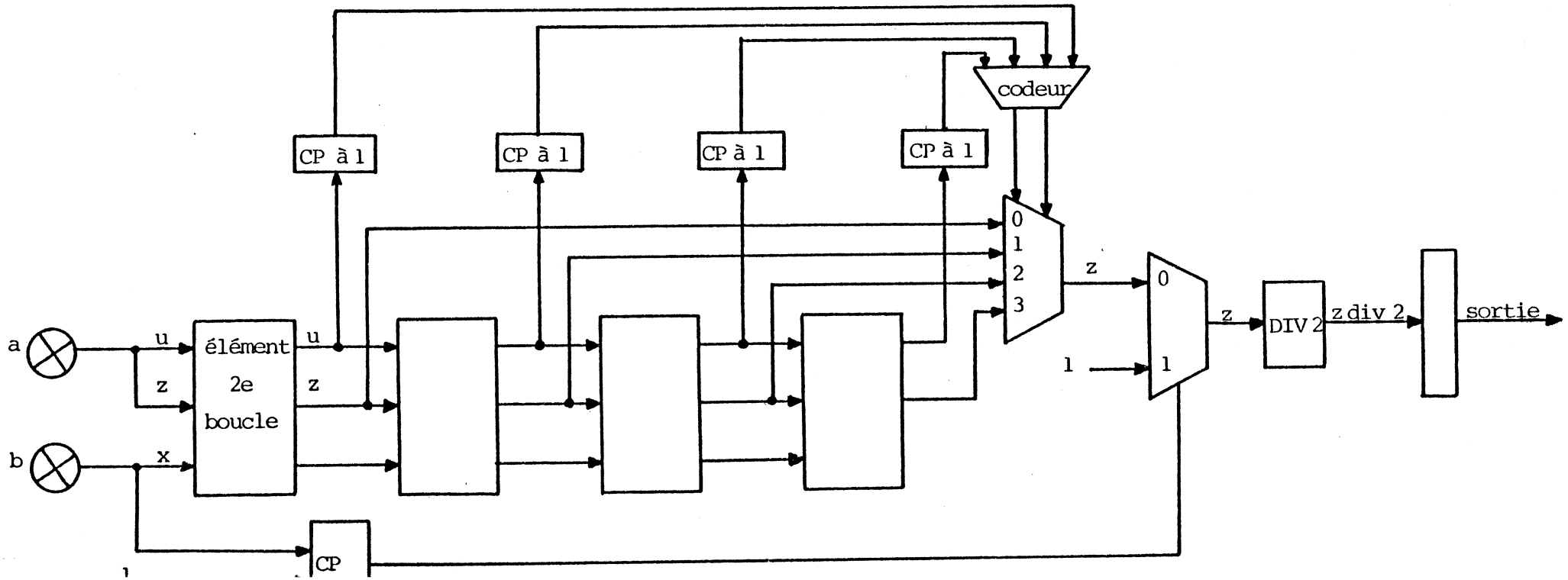


Figure 6a. réalisation combinatoire itérative (1ère partie)



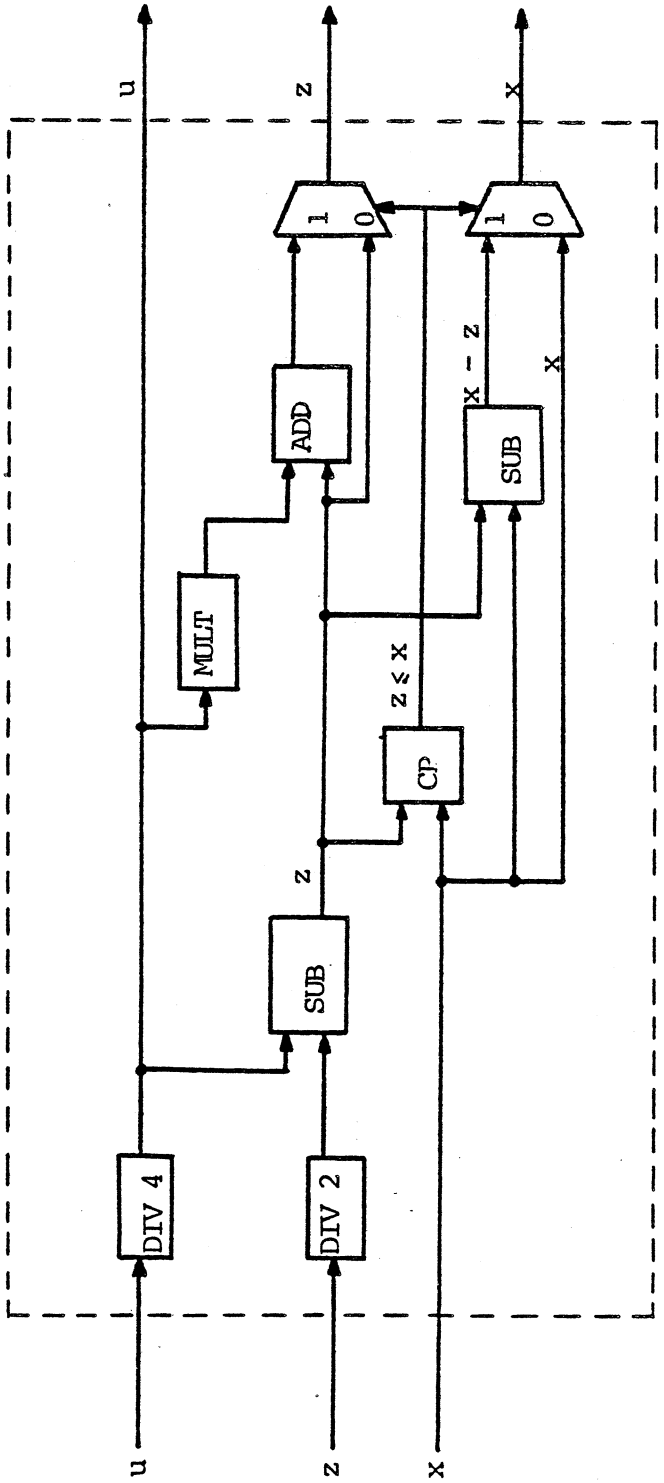


Figure 6 c: élément de la 2e boucle pour la réalisation combinatoire (Fig.6c)

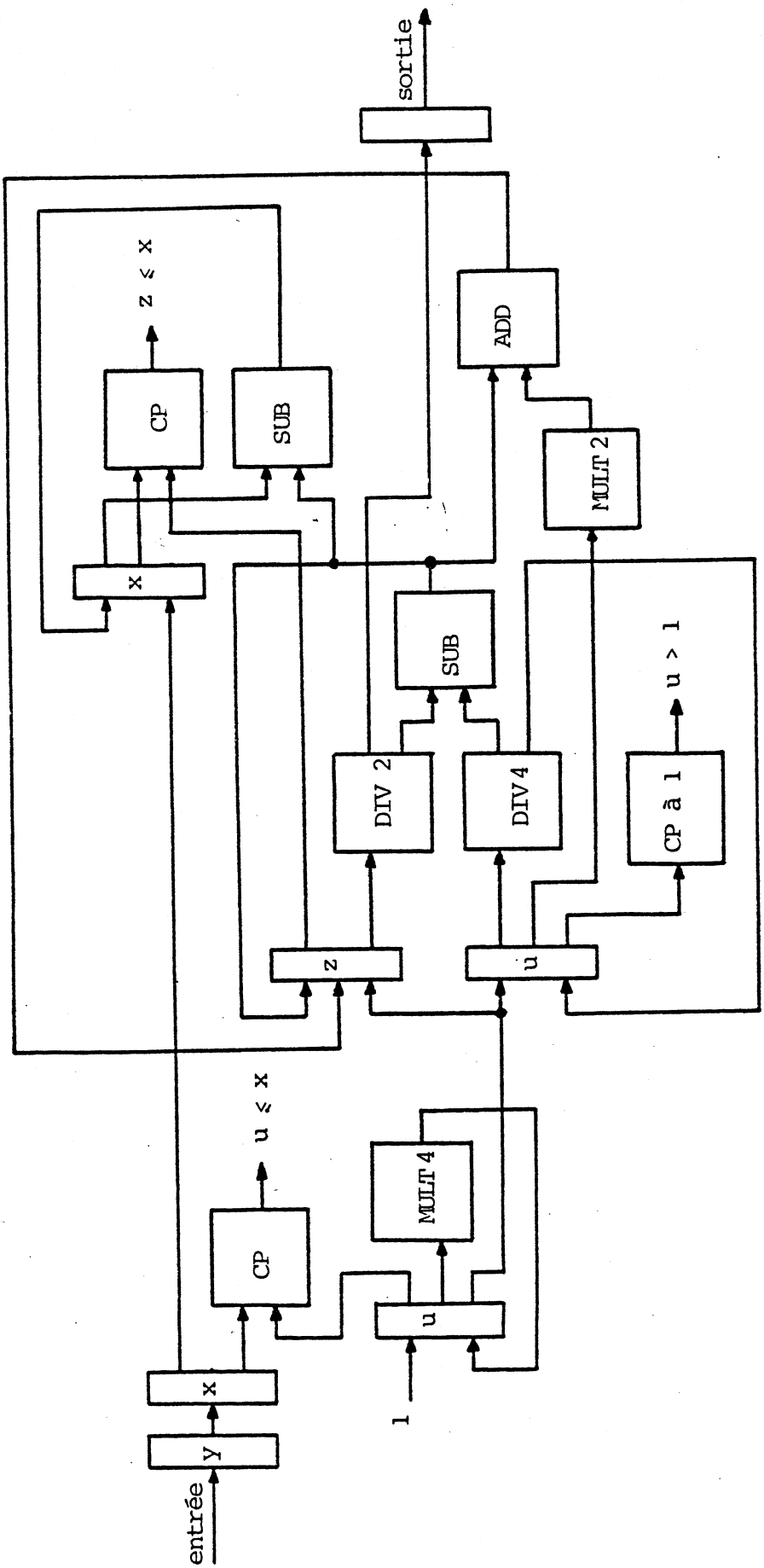


Figure 7 réalisation pipe-line (boucles réalisées de façon séquentielle)

opérateurs arithmétiques et logiques sont regroupés dans une unité de traitement, les éléments de mémorisation étant constitués en une mémoire locale de registres banalisés (figure 8).

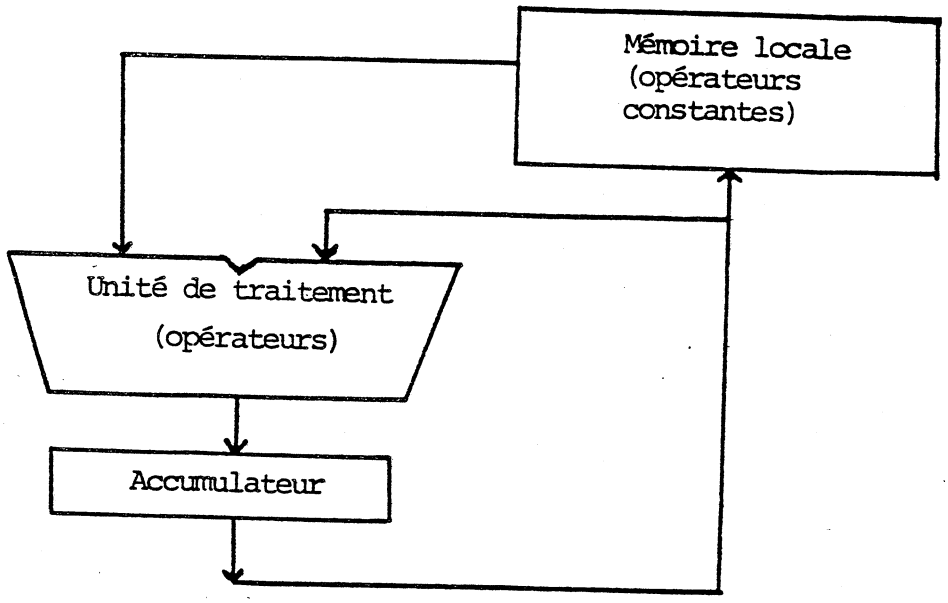


Figure 8 : Architecture à ressources matérielles minimisées

La solution adoptée (figure 9) :

- minimise les ressources matérielles
- ne respecte pas le parallélisme possible
- distribue les opérateurs en
  - . additionneur-soustracteur
  - . registres munis d'une possibilité de décalage réalisant la multiplication et la division par 2 et 4.

On trouvera donc dans la partie opérative de la figure 9 :

- un additionneur-soutracteur (U.A.L.) avec ses deux multiplexeurs d'accès 5 et 6
- quatre registres R0, R1, R2 et R3 avec leurs multiplexeurs d'accès 1,2,3, et 4....
  - . R0 est le registre d'entrée/sortie
  - . R1, R2, R3 sont les registres associées aux variables internes : x,z et u.

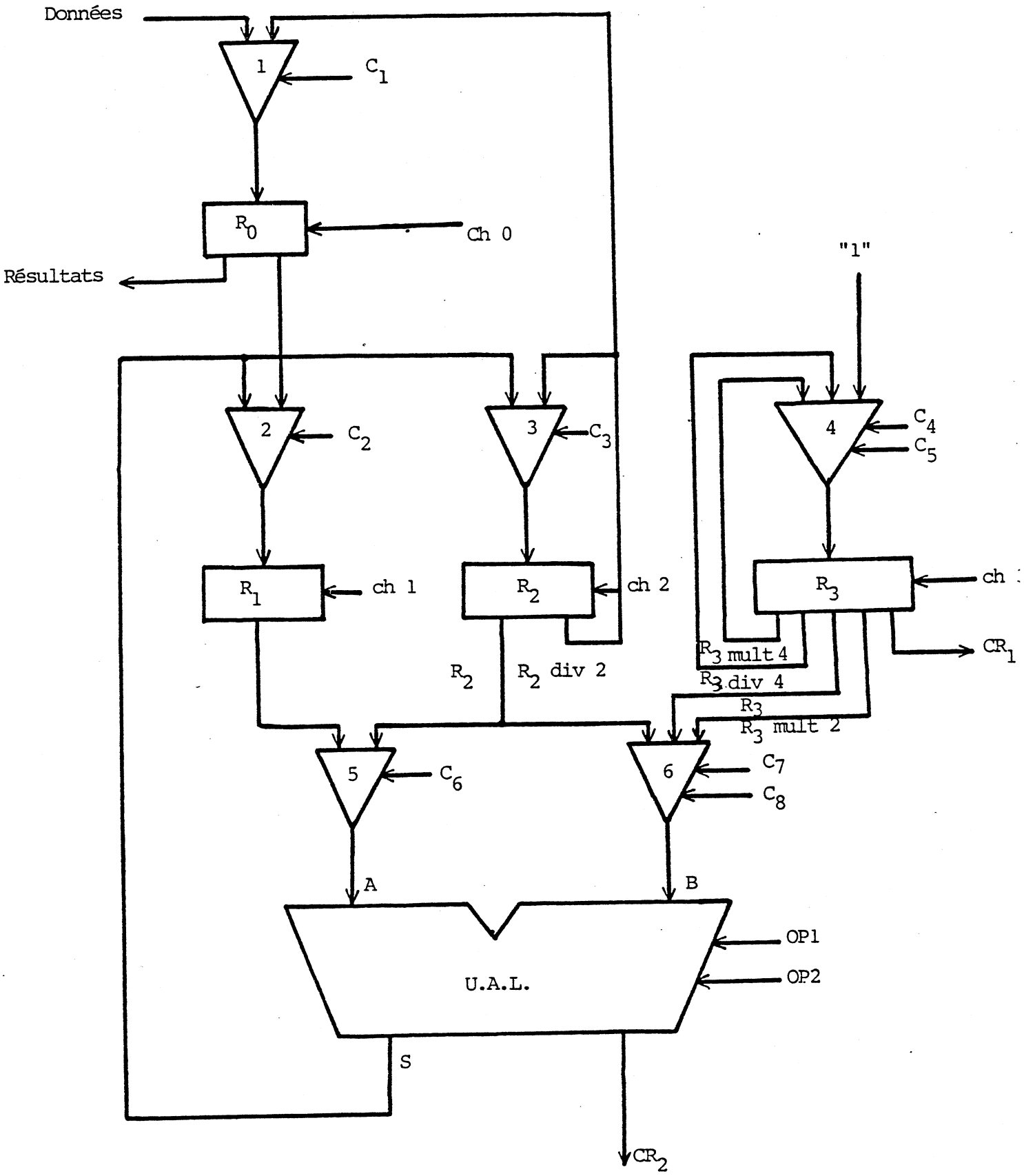


Figure 9. Architecture de la partie opérative

Les commandes C1..C8, ch0...ch3, OP1, OP2 sont envoyées par la partie contrôle suivant le schéma de la figure 10. Les compte-rendus sont CR1 et CR2.

CR1 : valeur de $u \leqslant 1$,

CR2 : résultat de la soustraction $\geqslant 0$

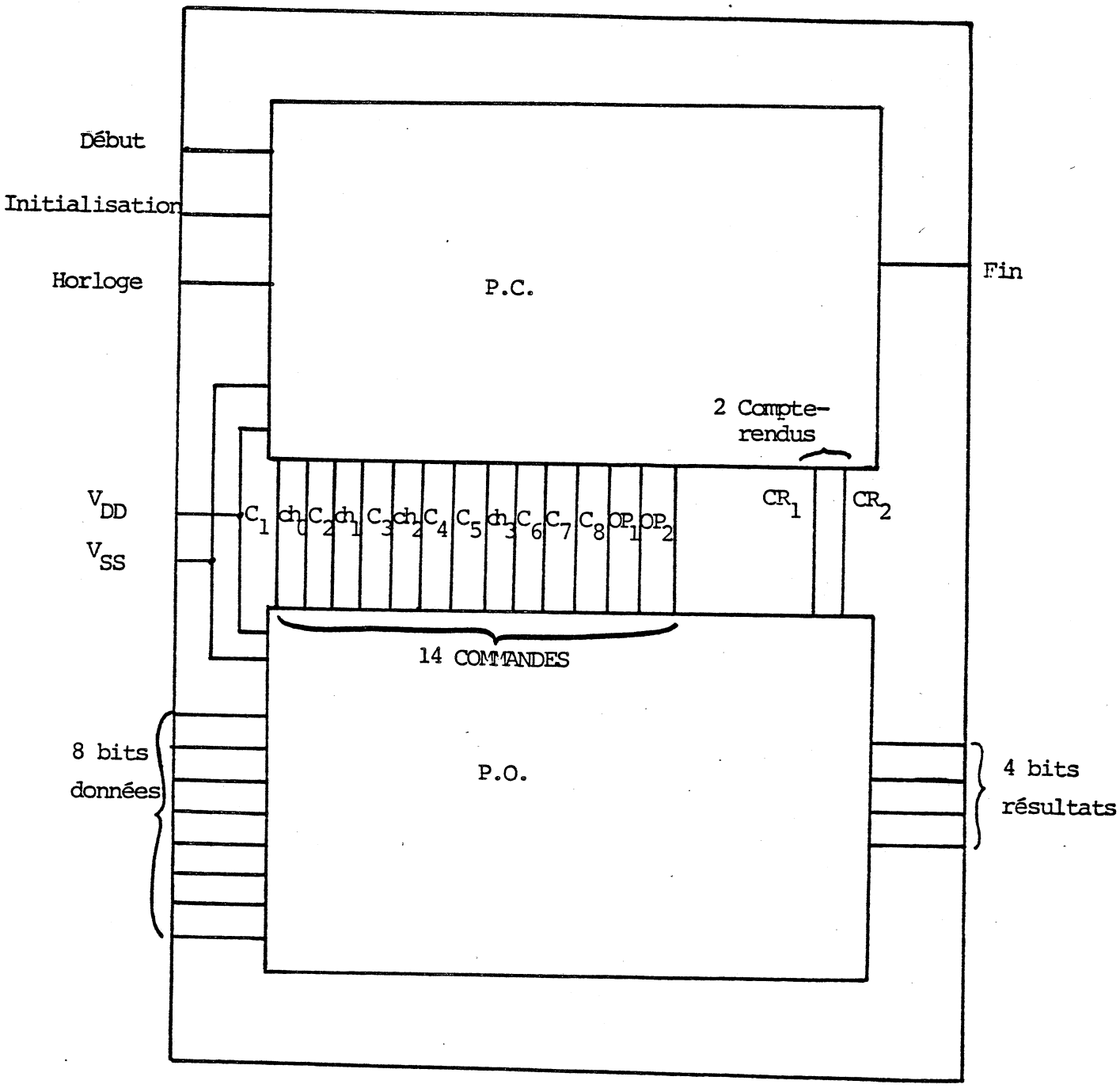


Figure 10 : Schéma général du ciruit

Les spécifications de ce niveau se traduisent par le graphe de contrôle interprété (figure 11). Les transistions entre noeuds portent les prédicats ou compte-rendus de la partie opérative. Il s'agit donc d'une transcription de l'algorithme initial à l'aide des ressources de la partie opérative.

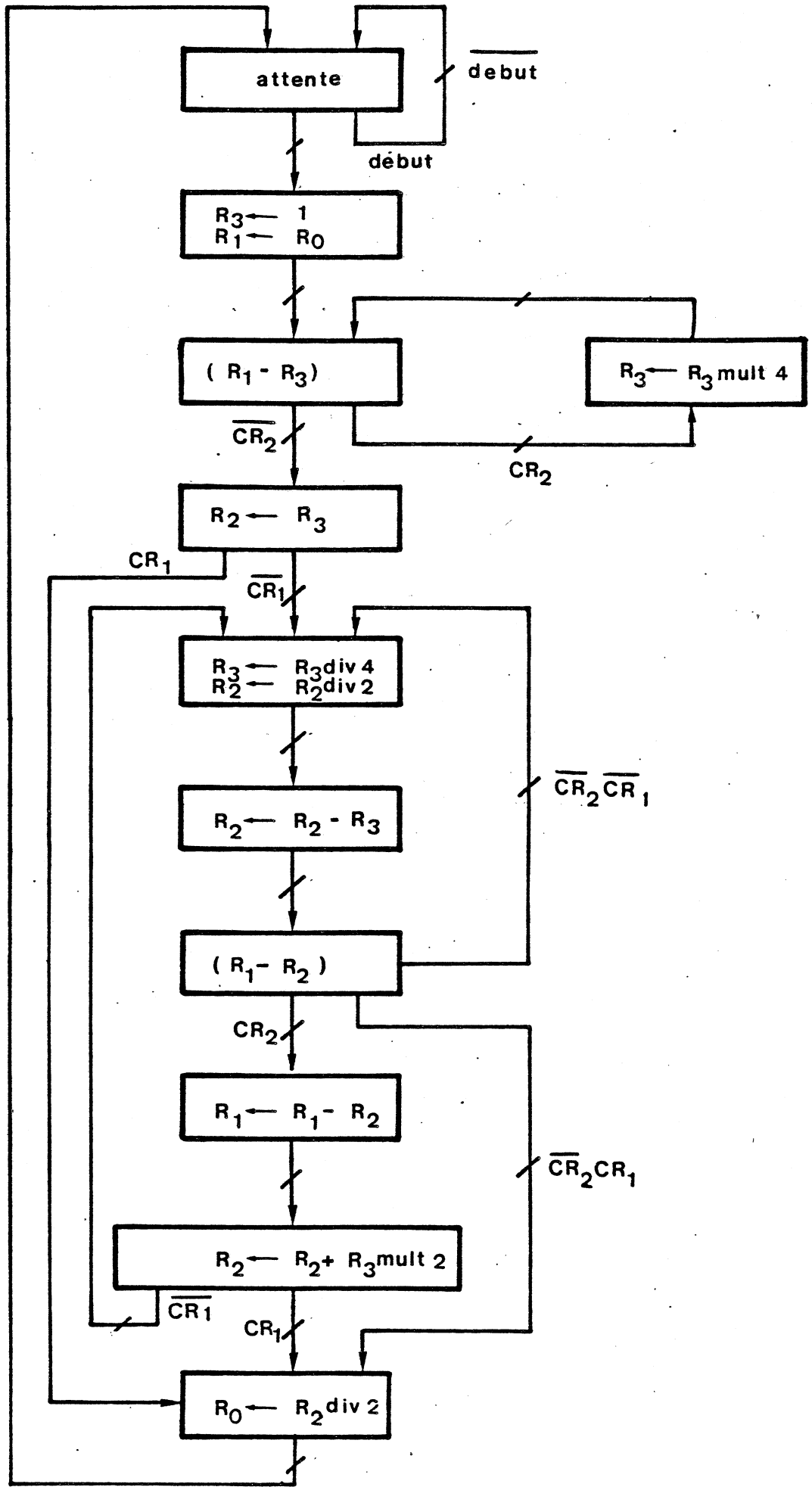


Figure 11 : Graphe de contrôle

Exemple II

Dans le circuit de multiplication, on opte également pour une solution séquentielle à ressources matérielles minimales.

La multiplication s'effectue de la façon suivante (cf. figure 12) :

- on charge le multiplicande dans le registre B(4 bits), le multiplieur dans le registre MQ (4 bits)
- la multiplication se fait par addition et décalage à droite :
  - . on additionne les registres B et ACC, le résultat de l'addition est chargé dans le registre ACC
  - . on effectue un décalage à droite sur les registres B, ACC, MQ (les premiers bits de MQ ont été utilisés)
  - . le résultat final se trouve dans les registres ACC-MQ.
- l'addition et le décalage à droite se font en passant quatre fois dans une boucle (multiplieur 4 bits), d'où la nécessité du compteur CPTR pour compter le nombre de passage dans la boucle.

On trouve aussi dans la partie opérative du multiplieur (figure 12) :

- un bus d'entrée/sortie 4 bits
- des amplificateurs tri-state TRIS1 et TRIS2 pour amplifier le résultat.

Cette arhitecture est à ressources matérielles minimisées. En effet :

- le registre MQ sert au changement du multiplieur et au stockage des quatre premiers bits du résultat de la multiplication,
- le registre ACC sert à la fois de registre d'entrée/sortie pour l'additionneur et au stockage des quatre dernier bits du résultat,
- le circuit ne possède qu'un bus d'entrée/sortie.

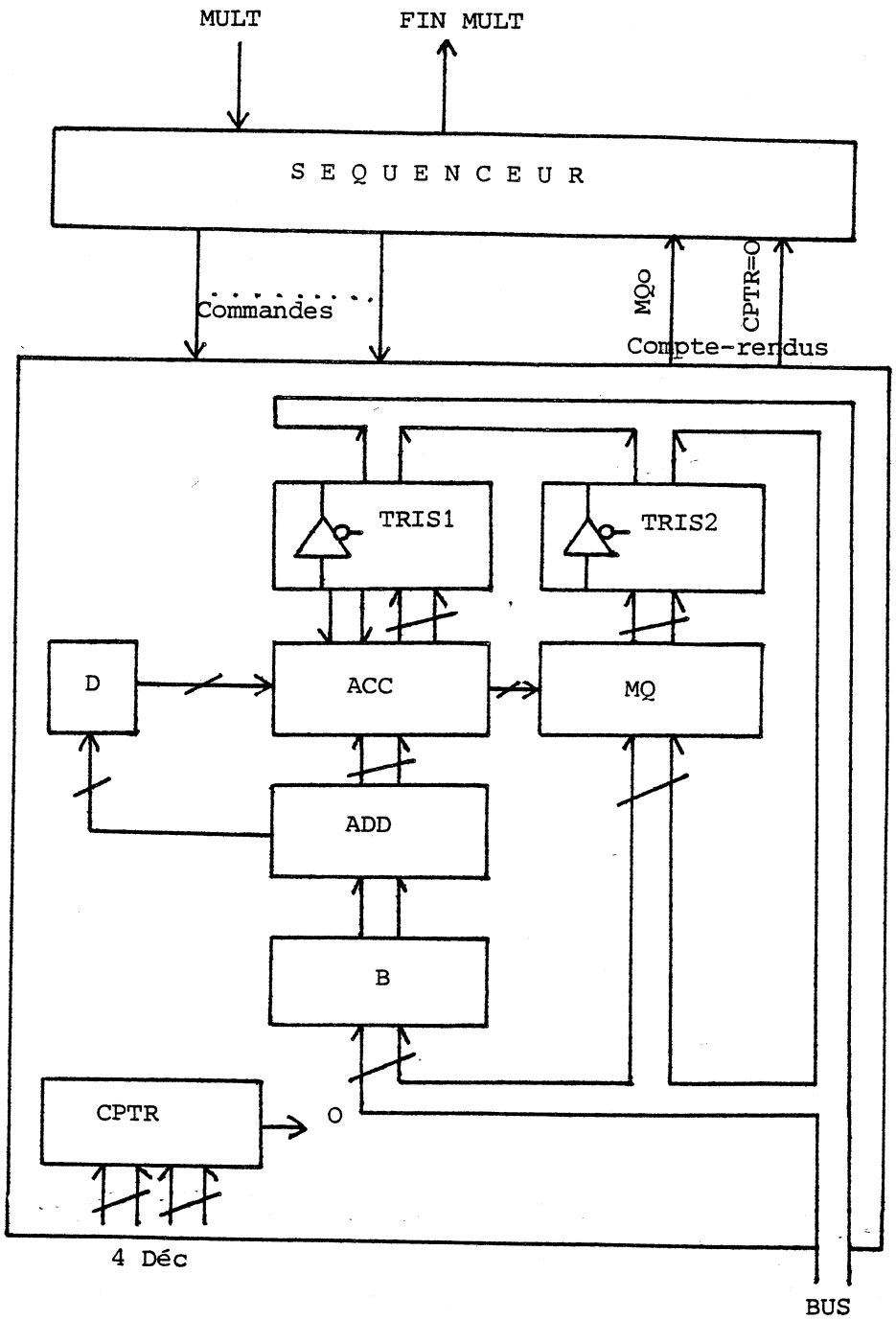


Figure 12 : Architecture de la partie opérative

L'architecture de la partie opérative étant choisie, on passe de l'algorithme ou du graphe de fonctionnement au graphe de contrôle interprété en respectant les compatibilités matérielles de la partie opérative. Dans le cas du multiplieur, on ne pourra charger le multiplieur et le multiplicande dans les

registres MQ et B en même temps à cause de l'unique bus d'entrée 4 bits ; pour la même raison on enverra d'abord les quatre premiers bits du résultat vers l'extérieur, puis les quatre derniers bits (bus unique 4 bits).

Les spécifications de ce niveau se traduisent donc par le graphe de ontrôle interprété de la figure 13.

On trouvera dans (ŒL 79), l'exemple simplifié d'un microprocesseur.

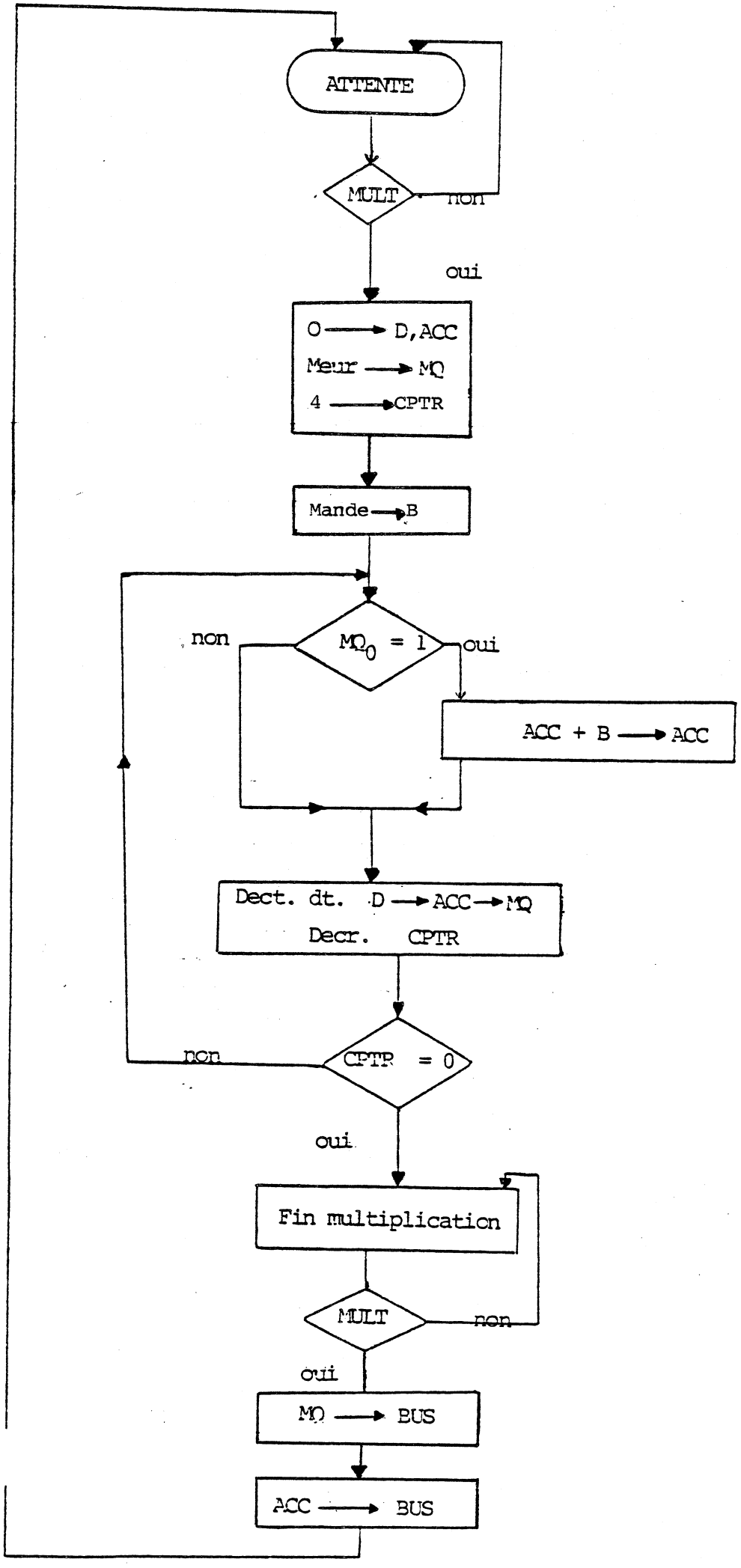


Figure 13 : graphe de contrôle interprété

V - PROBLEMES TEMPOREELS, LOGIQUE STATIQUE ET DYNAMIQUE : GRAPHE DE CONTROLE INTERPRETE ET TEMPORISE

V - 1. Logique statique

Le changement d'état de la partie contrôle et le chargement des résultats dans la partie opérative se feront à des instants différents de la période de l'horloge du circuit (figure 14).

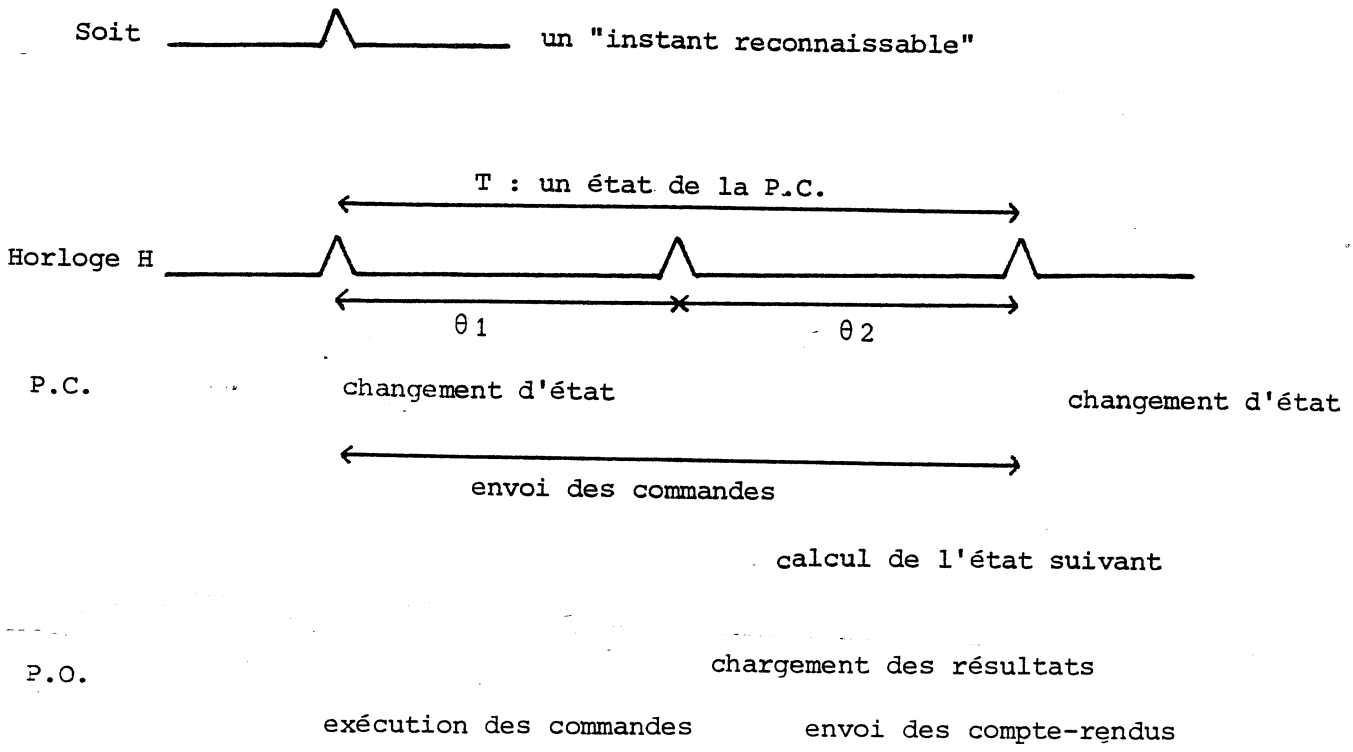


Figure 14 : Temporisation statique

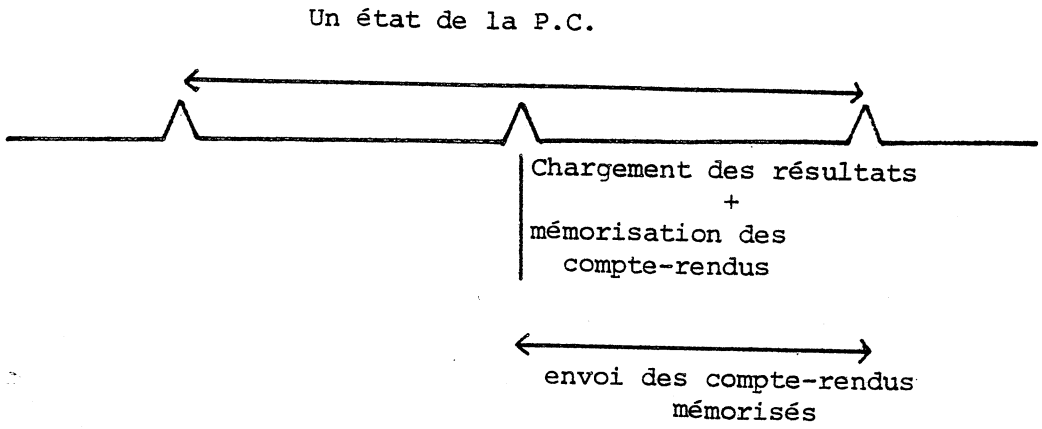
La durée d'un état de l'automate de contrôle délimitée par deux instants "changement d'état de la partie contrôle". Les commandes émises pendant cette phase auront cette durée T (machine de Moore).

Ce temps T est partitionnée en deux tranche θ1 et θ2. La partie opérative effectue les microopérations pendant la durée θ1, échantillonne ses résultats (changement d'état) à la fin de θ1.

Les compte-rendus sont envoyés à la partie contrôle pendant θ2, la partie contrôle calcule son état suivant pendant cette durée θ2, change d'état à la fin de θ2.

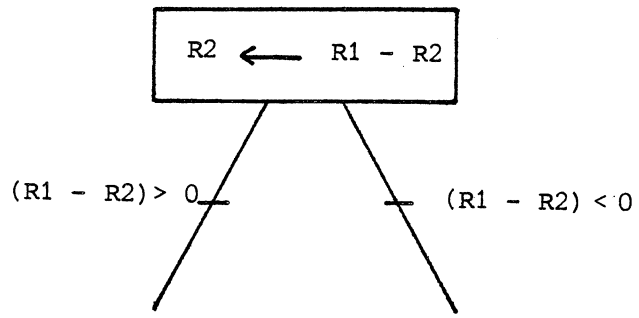
Pour la prise en compte des compte-rendus par la partie contrôle, il faut distinguer les trois cas suivants :

1er cas : Les compte-rendus sont mémorisés en même temps que le chargement des résutats dans les registres de la partie opérative.



Les compte-rendus se rapportent donc aux "anciennes valeurs" mémorisées dans les registres.

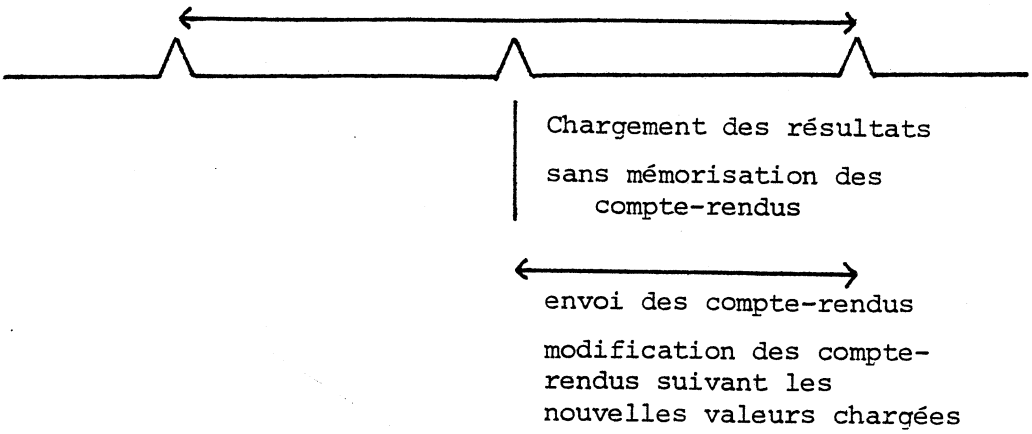
Exemple



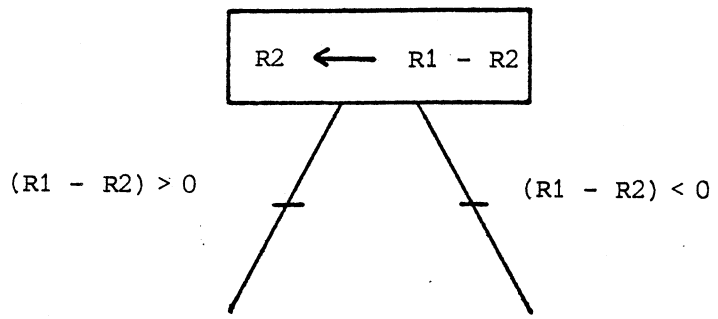
Le compte rendu (R1 - R2) est mémorisé en même temps que le résultat (R1 - R2) est chargé dans le registre R2 : ce compte-rendu porte donc sur l'ancienne valeur de R2, donc de (R1 - R2).

2ème cas : Les compte-rendus ne sont pas mémorisés et ils se rapportent aux "nouvelles valeurs" mémorisées dans les registres :

Un état de la P.C.

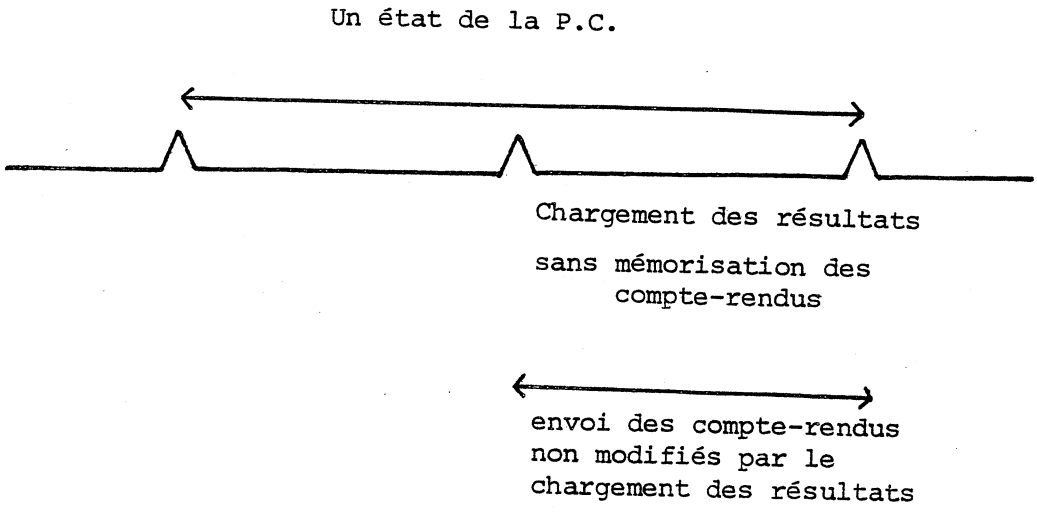


Exemple



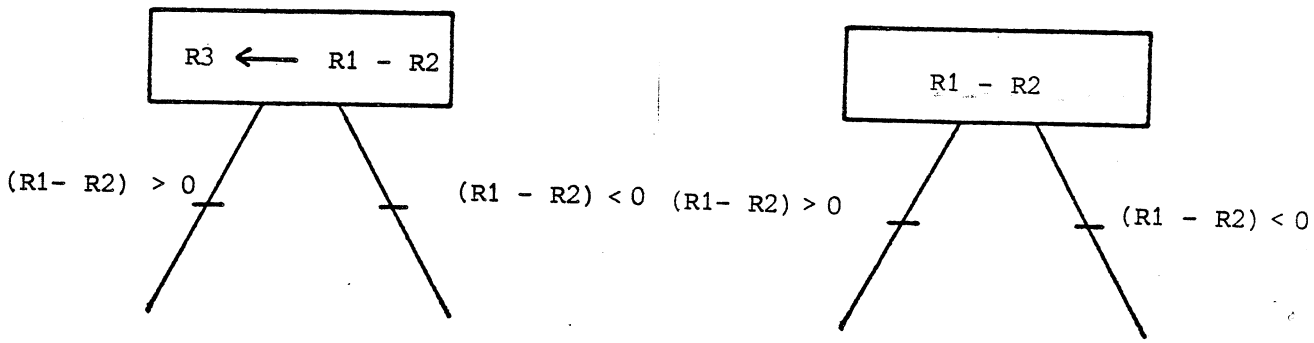
Le résultat (R1 - R2) est chargé dans le registre R2, cette nouvelle valeur de R2 implique une nouvelle valeur pour (R1 - R2) : le compte rendu envoyé au moment du changement d'état de la partie contrôle (fin de θ2) portera sur cette nouvelle valeur de (R1 - R2).

3ème cas : Les compte-rendus ne sont pas mémorisés et ils se rapportent aux "anciennes valeurs" mémorisées dans les registres :



Ceci n'est possible que si le chargement des résultats n'affectent pas les registres sur lesquels portent les compte-rendus de cet état.

Exemples



Le résultat (R1 - R2) est soit chargé dans le registre R3, soit calculé uniquement pour le compte-rendu (R1 - R2) : ce compte-rendu porte donc sur les anciennes valeurs de R1 et R2 donc de (R1 - R2).

Exemple I

Dans le circuit d'extraction de racine carrée, la temporisation et la synchronisation sont faites sur ce mode avec des compte-rendus non mémorisés dans les registres (3ème cas) : en effet, les registres sur lesquels portent les compte-rendus CR1 (R3) et CR2 (R1 - R2) ou(R1 - R3) ne sont pas affectés par le chargement des résultats (cf. le graphe de contrôle interprété de la figure 11).

La durée d'un état de l'automate de contrôle délimitée par deux fronts descendants successifs de l'horloge, c'est à dire que les changements d'état de la partie contrôle se feront au front descendant de l'horloge (H↓). La partie opérative échantillonne ses résultats sur le front montant de l'horloge (H↑). Ceci peut se résumer par les schémas suivants (figures 15a et 15b).

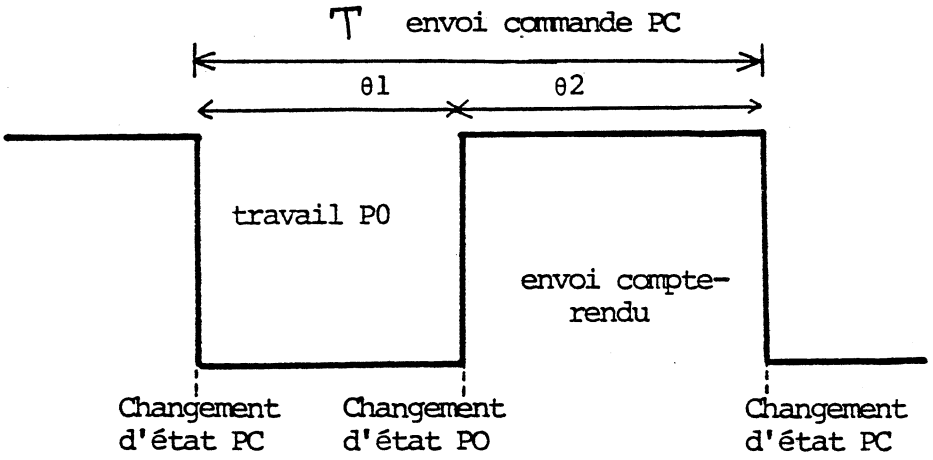


Figure 15a : Temporisation de l'extracteur

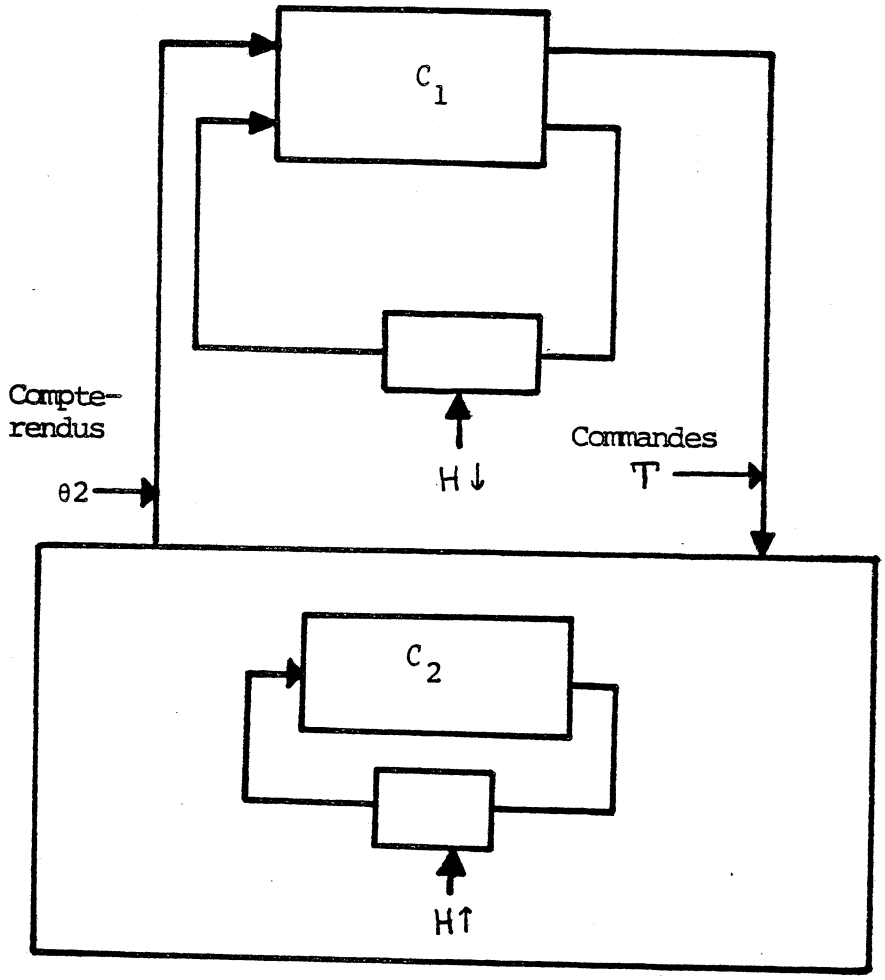


Figure 15b : Fonctionnement temporel de l'extracteur

Le choix de la temporisation se traduit au niveau du graphe de contrôle par :

- Un partitionnement définitif. A un niveau de contrôle, il y a bijection entre l'ensemble des noeuds et l'ensemble des valeurs des éléments de mémorisation. Un état est associé à une valeur des éléments de mémorisation (bascules).

- Des renseignements temporels ; ceux-ci concernent
  - . le contenu des états ; précision ou modification des chronogrammes
  - . les transistions entre états de la partie contrôle (occurrence sur front ou niveau, durée).
  - . l'évolution dynamique respective de la partie contrôle et de la partie opérative.

De tels renseignements sont portés sur le graphe de contrôle sous forme d'étiquettes associées aux arcs.

Le choix de la temporisation amène au graphe de contrôle temporisé ; à côté de chaque signal émis est indiqué sa durée (T, θ2) ou l'instant d'échantillonnage (figure 16).

A partir d'une telle description, on peut obtenir les chronogrammes des signaux de contrôle d'un chemin du circuit (un chemin est défini dans le graphe d'exécution symbolique).

Remarque : Le signal "début" est présent pendant une période d'horloge (il arrive sur la partie opérative et la partie contrôle). Le signal "initialisation" est synchronisé avec l'horloge H.

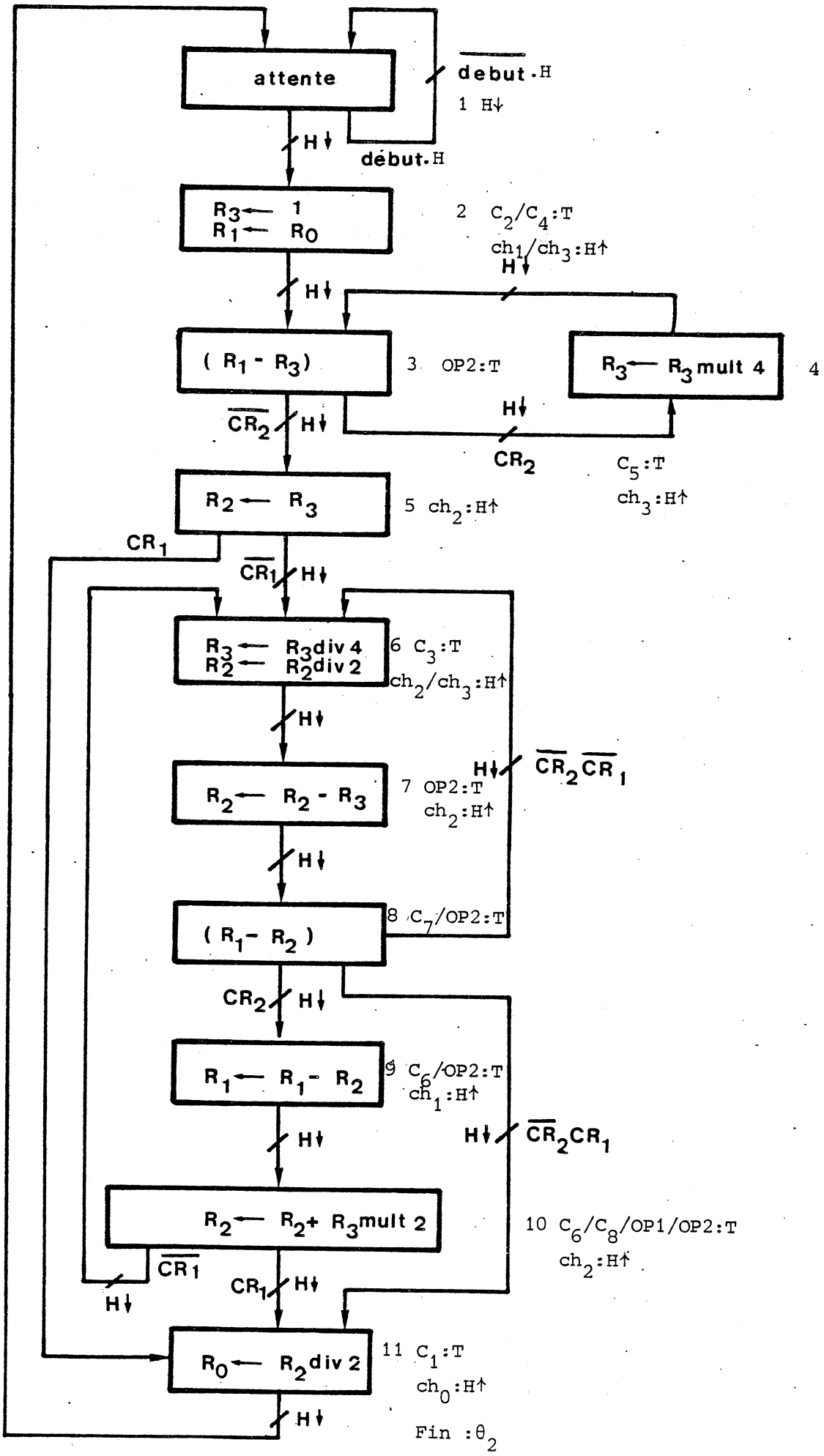
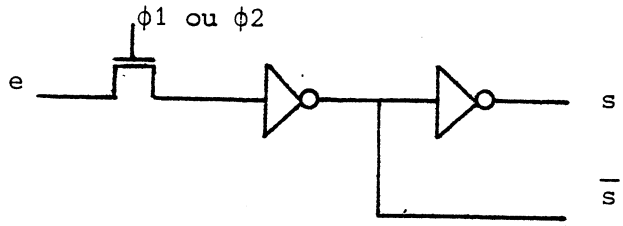


Figure 16 : Graphe de contrôle temporisé

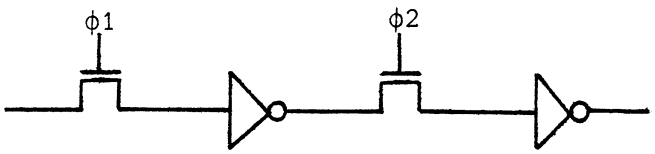
V - 2. Logique biphasée

5.2.1. Principe général

Le point de mémorisation élémentaire, pour le séquenceur et les registres de la partie opérative sera le suivant :



Les problèmes de temporisation peuvent être facilement résolus en utilisant une horloge à 2 phases $\phi 1$, $\phi 2$, sans recouvrement et des points de mémorisation comparables aux points de mémorisation statiques du type maître-esclave.



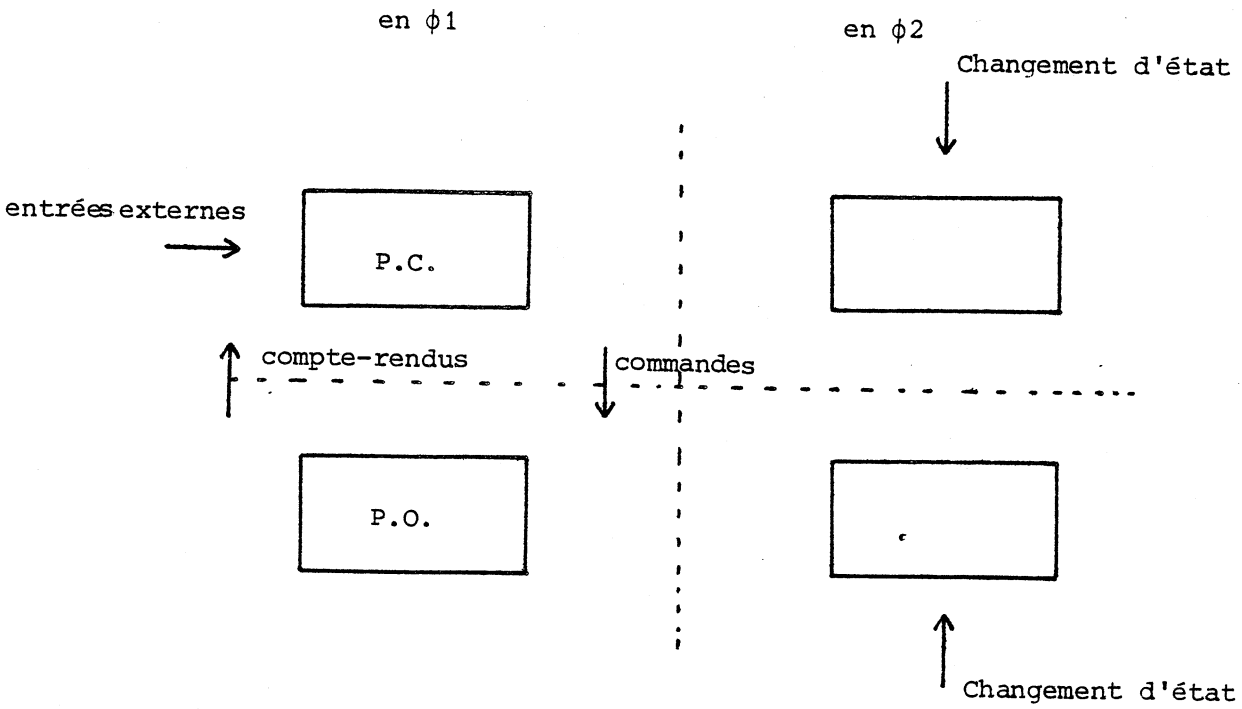
Le principe bien connu est le suivant : en $\phi 1$: ce point de mémorisation charge la valeur mais présente au monde extérieur son ancien état ; en $\phi 2$, la nouvelle valeur s'établit en sortie.

Dialogue entre un contôleur et une partie opérative

A partir des considérations précédentes, le dialogue PC-PO peut se faire de la façon suivante :

en $\phi 1$: les informations de dialogue entre les deux partenaires et le monde extérieur sont échangées ; il s'agit donc d'échanger des compte rendus (PO vers PC) et des commandes (PC vers PO). L'élaboration des signaux de ce dialogue ne peut se faire que sur l'ancien état des partenaires puisque c'est celui-ci qu'ils présentent au monde extérieur.

en $\phi 2$: Les deux partenaires changent d'état au vu des informations recueillies en $\phi 1$ et sont coupés de toute information extérieure.



5.2.2. Modes fondamentaux

On appelle mode fondamental, le cas où une phase ou un état du contrôleur déclenche des opérations dont la durée est délimitée par deux phases successives de l'horloge $\phi 1$, un état de l'automate étant délimité par deux phases successives de l'horloge $\phi 1$.

Pendant la phase $\phi 1$:

- la partie contrôle calcule son état suivant à l'aide des compte rendus envoyés au $\phi 2$ précédent par la partie opérative.
- la partie opérative exécute les commandes de l'ancien état établi au $\phi 2$ précédent dans la partie contrôle.

Ceci peut se résumer par le schéma suivant (figure 17)

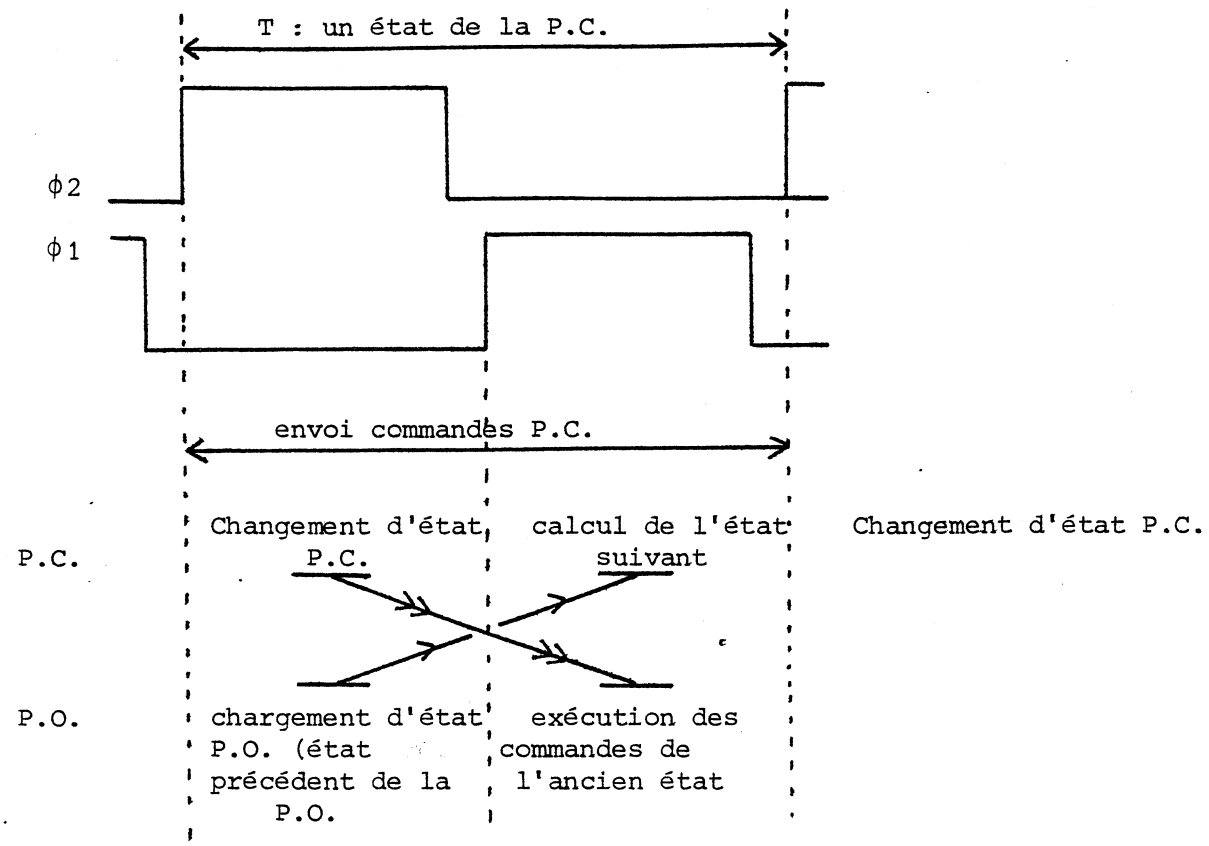
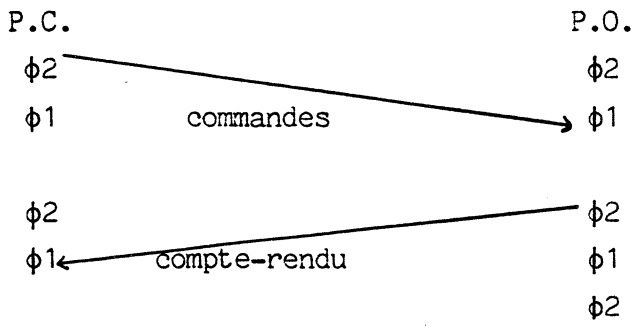


Figure 17 : Temporisation biphasée

Dans le diagramme suivant les flèches représentent les échanges entre partie contrôle et partie opérative; leur origine indique la phase d'émission, leur extrémité la phase de prise en compte :



Une commande envoyée pendant une phase $\phi 2$ par la partie contrôle est reçue pendant la phase $\phi 1$ suivante par la partie opérative, de même le compte-rendu envoyé pendant la phase $\phi 2$ suivante par la partie opérative sera reçue par la par la partie contrôle pendant la phase $\phi 1$ suivante : il existe un décalage d'une phase entre partie contrôle et partie opérative.

Remarque importante

La temporisation d'un circuit dynamique nécessite un cadrage correct des commandes avec les deux phases $\phi 1$, $\phi 2$. Il faut tenir compte du retard des commandes dû à l'interface partie contrôle-partie opérative (figure 18)

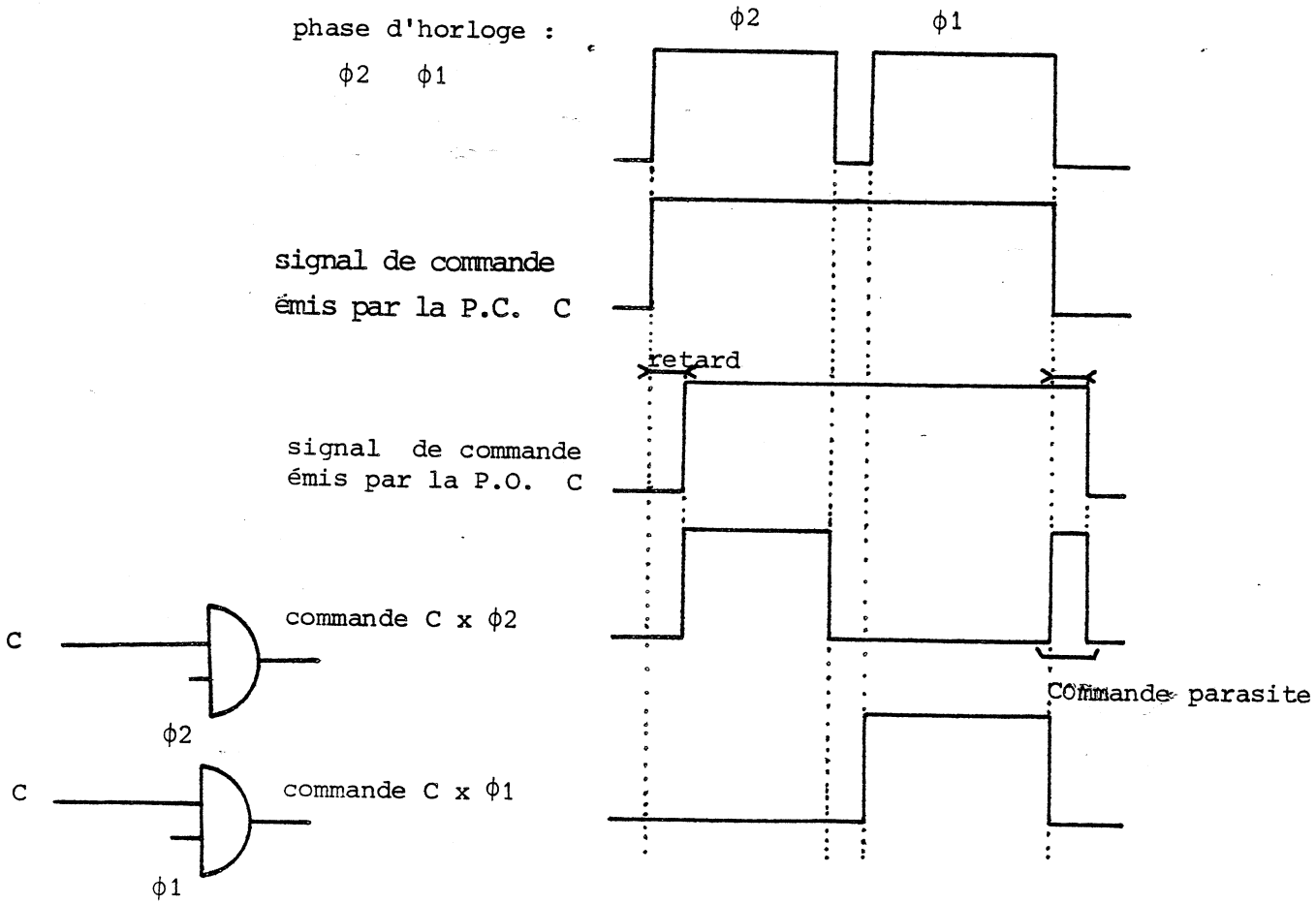


Figure 18 : Cadrage des commandes

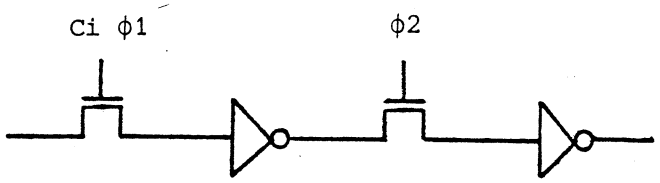
Tout signal de commande émis en $\phi 2$ par la partie contrôle ne pourra être reçu qu'au o1 suivant par la partie opérative (Pour avoir un circuit plus rapide, on aurait pu l'émettre et l'envoyer en $\phi 2$, on aurait alors une commande parasite).

La commande CXϕ1, sera générée localement dans la partie opérative.

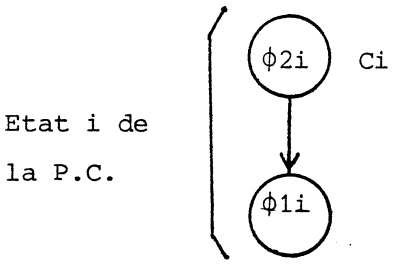
Dans la présentation précédente, nous ne nous sommes intéressés qu'à l'évolution des états du contrôleur et de la partie opérative.

Nous devons cette fois "insérer" dans ce schéma temporel la traversée des parties combinatoires.

(i) Il n'y a pas de traversée de partie combinatoire, il s'agit du chargement d'un registre à décalage. Le schéma du registre à décalage sera le suivant :

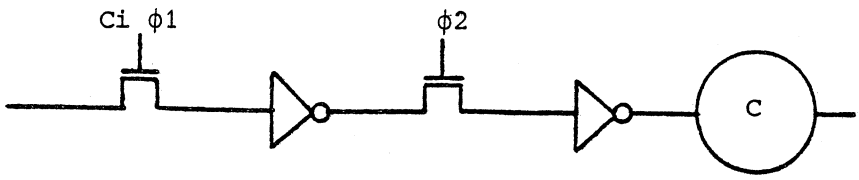


Cette opération se fera de la façon suivante dans le graphe de contrôle :



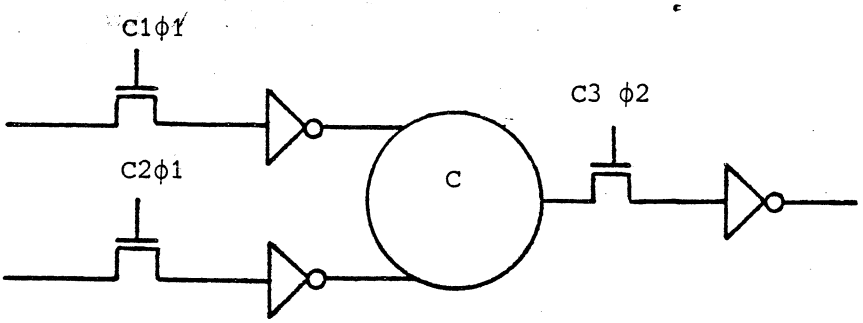
P.C.	P.O.
2 envoi de la commande Ci	
1	exécution de Ci chargement du maître
2	chargement de l'esclave

(ii) Il s'agit ici du chargement d'un registre à décalage suivi d'une partie combinatoire : C suivant le schéma de la partie opérative :

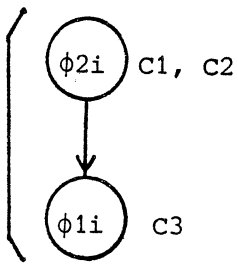


Cette opération se fera de la même façon que précédemment au niveau du graphe de contrôle.

(iii) On s'intéresse maintenant aux chargements de registres simples spérarés par une partie combinatoire C, suivant le schéma de la partie opérative:



Cette opération se fera de la façon suivante dans le graphe de contrôle.



Etat i de la P.C.

	P.C.	P.O.
	2 envoi des commandes C1, C2	
	1 envoi de la commande C3	exécution de C1, C2
	2	exécution de C3

5.2.3. Modes non fondamentaux

On appelle mode non fondamental, le cas où une phase ou un état du contrôleur déclenche des opérations dont la durée est supérieure à la phase d'horloge $\phi 1$ ou $\phi 2$. C'est le cas où la traversée de la partie combinatoire C est plus grande que la phase $\phi 1$ ou $\phi 2$, elle nécessite l'introduction de deux ou plusieurs phases vides.

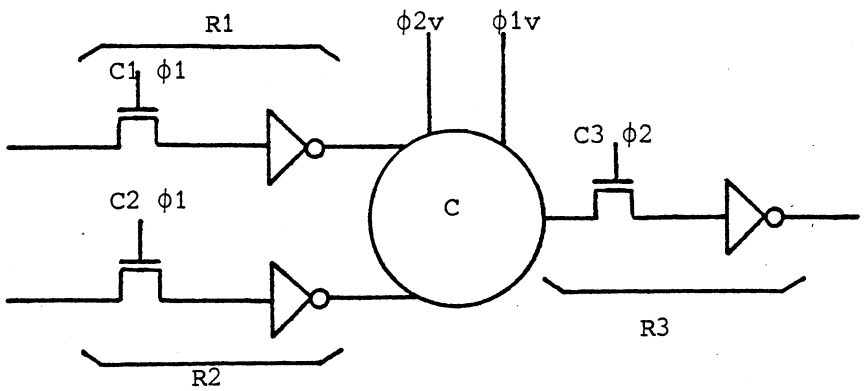
(i) Extension sans maintien

C'est le cas où la traversée de la partie combinatoire est assez rapide et nécessite peu de phase ϕi vides.

En effet, l'information est présente pendant un certain temps en sortie des registres de la partie opérative (1µs environ dans une technologie NMOS 6u, si T = 200 ns (f = 5MHz) l'information est présente pendant 5 périodes).

Exemple 1

Soit une partie combinatoire nécessitant deux phases supplémentaires $\phi 2v$ et $\phi 1v$:



L'information est toujours présente en sortie des registres R1 et R2 pendant la traversée de la partie combinatoire C et le chargement de R3.

Exemple 2

L'attente d'un compte rendu se rapportant à une commande envoyée par la partie contrôle nécessitera deux phases vides à cause du décalage partie contrôle-partie opérative.

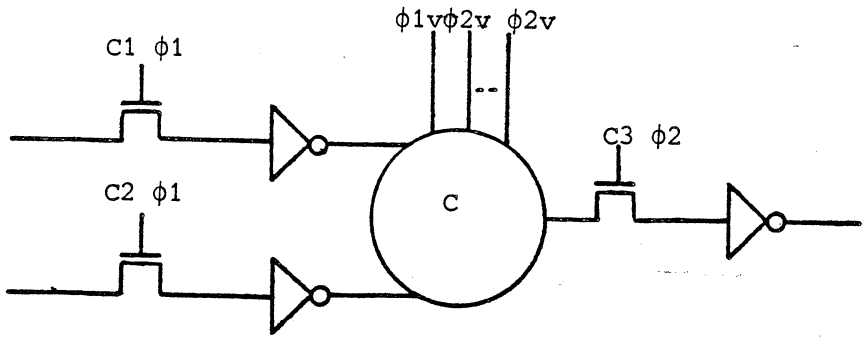
(ii) Extension avec maintien

C'est le cas où la traversée de la partie combinatoire C nécessiste beaucoup de phases ϕi vides. L'information n'est donc plus présente sur les registres d'entrée de la partie combinatoire. Pour y remédier, on pourra :

- soit maintenir la valeur sur les registres de la partie opérative précédents la partie combinatoire (cf. les schémas des points de mémorisation avec maintien donnés ci-après).

- soit maintenir les commandes arrivant sur les registres de la partie opérative précédents la partie combinatoire.

Exemple



Les commandes C1, C2 seront maintenues de la façon suivante (figure 19).

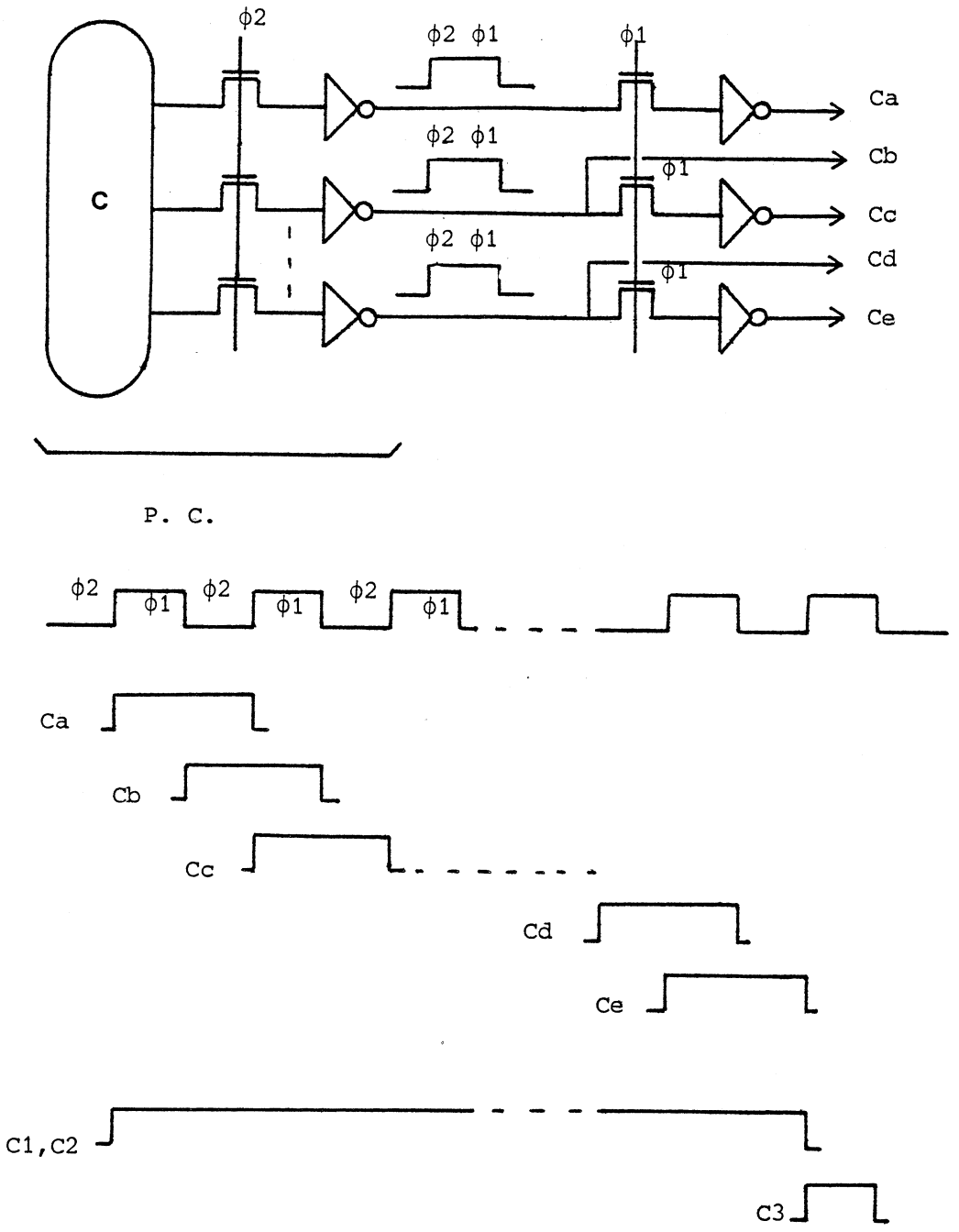
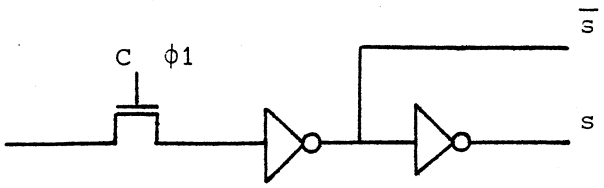


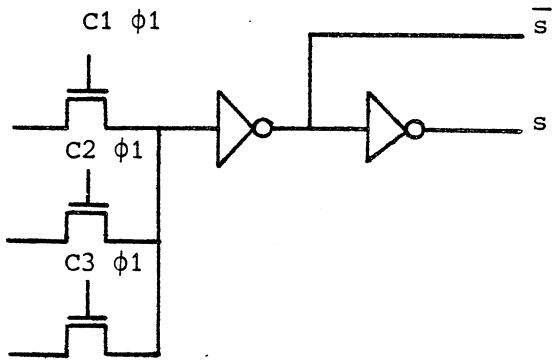
Figure 19 : Principe du maintien des commandes

On utilisera les points de mémorisation suivants :

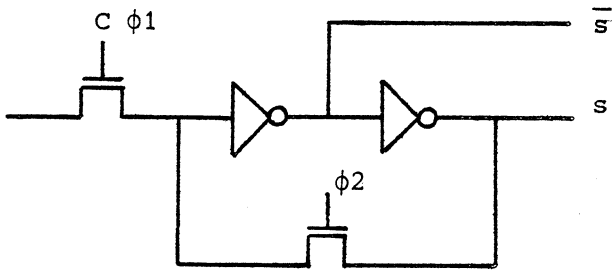
- point de mémorisation simple sans maintien de la valeur



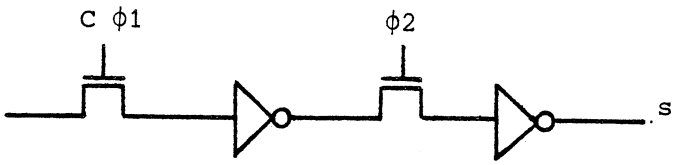
- point de mémorisation simple chargeable sélectivement



- point de mémorisation simple avec maintien de la valeur

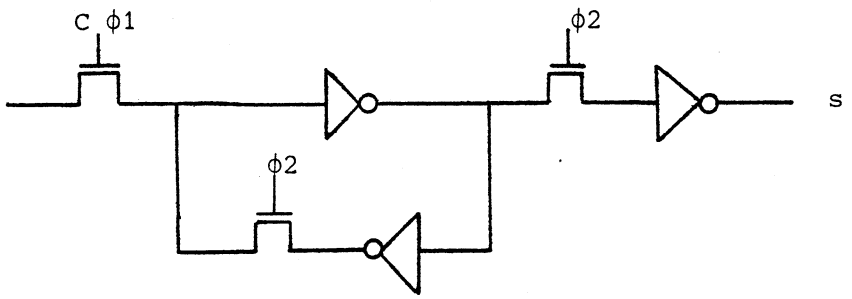


- point de mémorisation maitre-esclave sans maintien

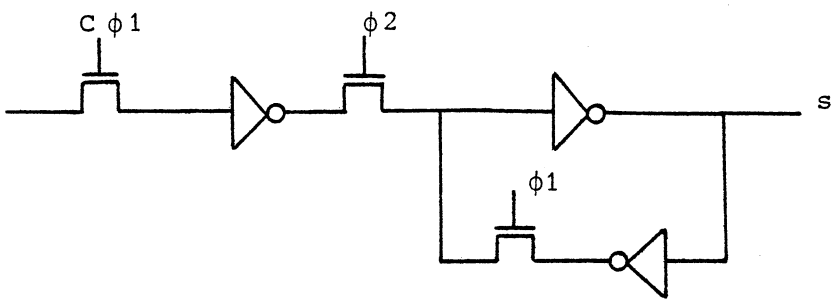


- point de mémorisation maitre-esclave avec maintien :

. sur le maitre, la valeur est maintenue en sortie du maitre et de l'esclave



. sur l'esclave, la valeur est maintenue en sortie de l'esclave.



On utilisera des points de mémorisation avec maintien lorsque l'information en sortie des registres est nécessaire pendant un certain temps (la valeur est présente pendant 1µs), c'est-à-dire dans les cas suivants :

- pour maintenir la valeur sur les registres précédents une partie combinatoire longue à traversée (cf. précédemment (ii) extension avec maintien).

- pour maintenir la valeur sur les registres non modifiés assez souvent : on utilisera des points de mémorisation à maintien pour le stockage des résultats pour qu'ils soient intacts jusqu'à l'instant où ils seront lus par l'extérieur (registre ACC et MQ dans le cas du multiplieur, le multiplicande

(registre B) doit rester intact pendant toute la multiplication, le registre B sera donc à maintien.
Un maintien sera nécessaire aussi sur le contrôleur dans les états d'attente des signaux début et fin envoyés par l'extérieur ; dans le cas du circuit de multiplication, ce maintien n'est pas nécessaire du fait de la présence des signaux MULT et $\overline{\text{MULT}}$ dans ces états.

Exemple II

Dans le circuit de multiplication, la temporisation et la synchronisation se fera suivant le principe exposé précédemment.

les registres MQ, ACC, D ainsi que le compteur CPTR seront des registres à décalage travaillant sur les phases $\phi 1$ et $\phi 2$. Le registre B sera un registre simple chargé en $\phi 1$, le chargement des données par l'extérieur se fera en $\phi 1$ pour le registre B comme pour le registre MQ. Les résultats seront envoyés sur la phase $\phi 1$ à l'extérieur, c'est à dire que la commande de blocage des amplificateurs tri-state sera inactive durant la phase $\phi 1$;

Le temps de traversée des registres et des amplificateurs tri-state est faible le temps d'établissement de la valeur sur le bus est plus important, la durée de ces opérations sera inférieure à une phase ϕi.

Par contre, le temps de traversée de l'additionneur 4 bits sans anticipation de retenue sera pénalisant si la durée de cette opération doit se faire durant une phase ϕi ; en effet, il faudrait prendre une phase ϕi de durée importante, on obtiendrait alors un circuit lent. On préfère introduire deux phases vides $\phi 1v$ et $\phi 2v$: l'addition se fera en trois phases.

Le fonctionnement temporel du circuit (partie opérative et partie contrôle) sera celui de la figure 20.

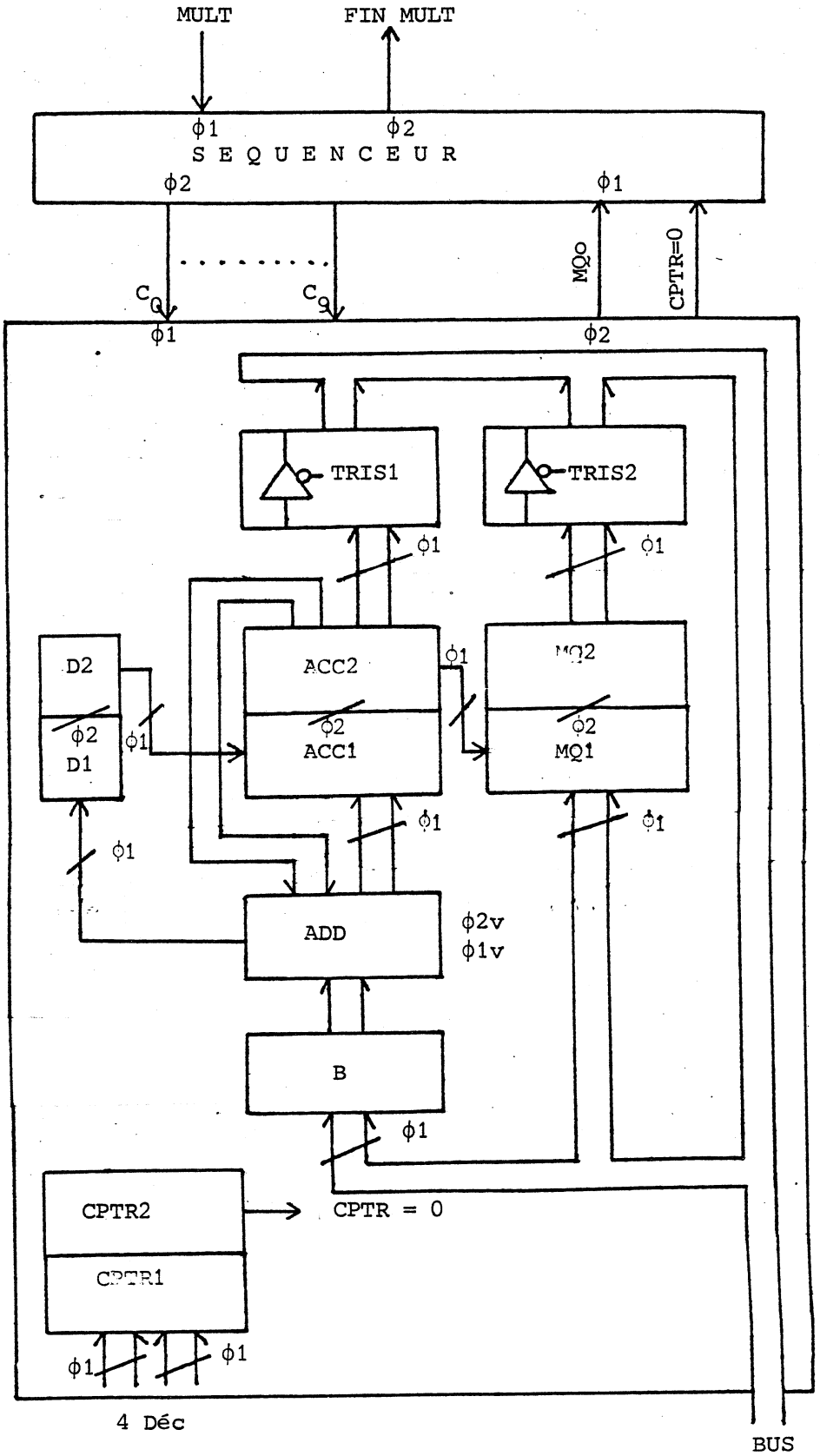


Figure 20 : Fonctionnement temporel du multiplieur

Le choix de la temporisation permet de passer du graphe de contrôle interprété au graphe de contrôle interprété et temporisé. Le contrôleur travaille surout dans des modes fondamentaux. Celui-ci travaillera dans des modes non fondammentaux, c'est-à-dire qu'on a alors introduction de phases vides sur le graphe, aux endroits suivants :

- pour la traversée de la partie combinatoire du circuit additionneur
- pour attendre un compte rendu de la partie opérative.

Ces différents cas seront des extensions sans maintien. On obtient alors le graphe temporisé de la figure 21. On arrive dans un état du graphe en $\phi 2$ chaque état correspondant à deux phases $\phi 2$, $\phi 1$. La partie opérative travaille sur les commandes envoyées durant un état du séquenceur pendant les phases $\phi 1$, $\phi 2$. Ceci est explicité sur le graphe de la figure 22.

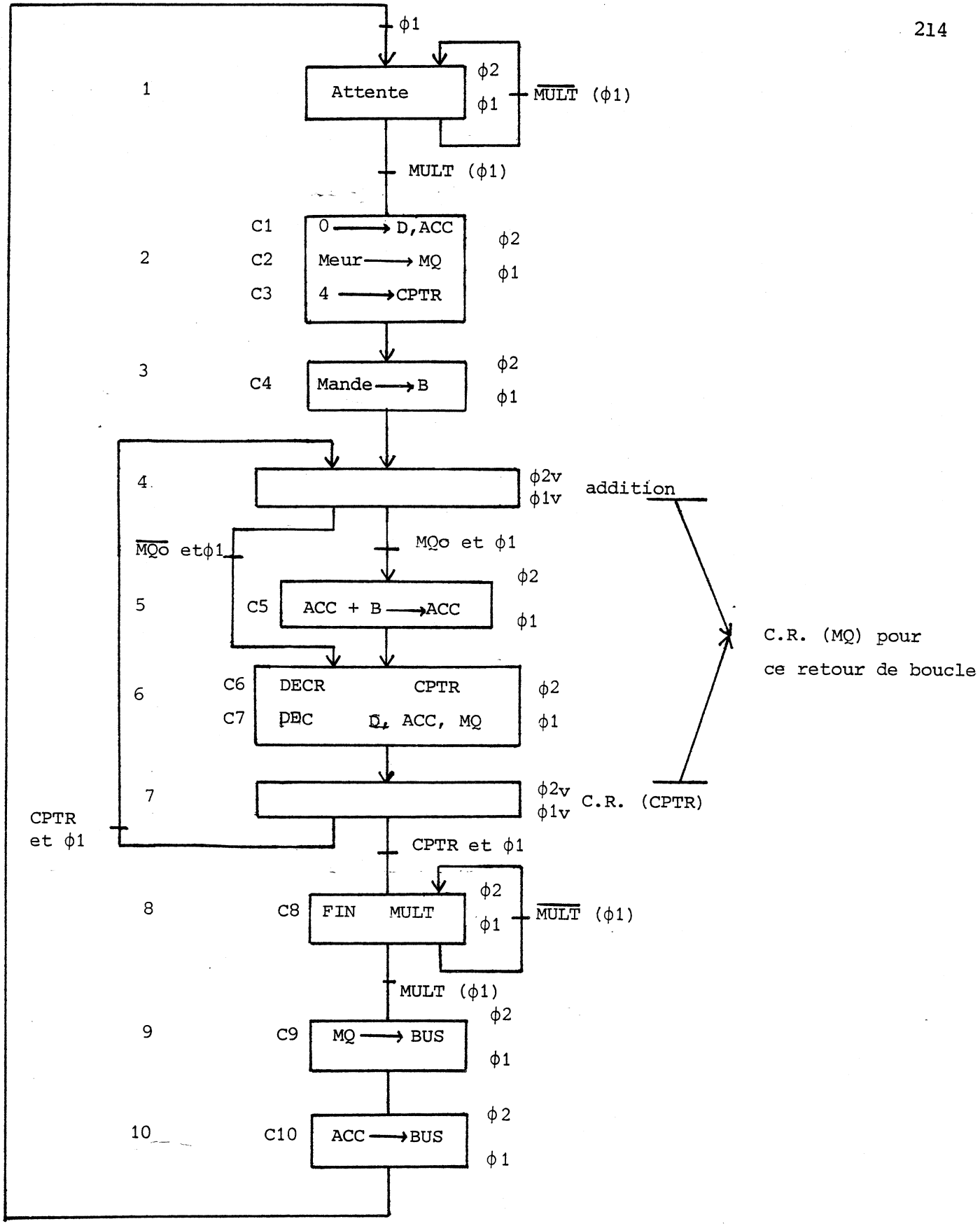
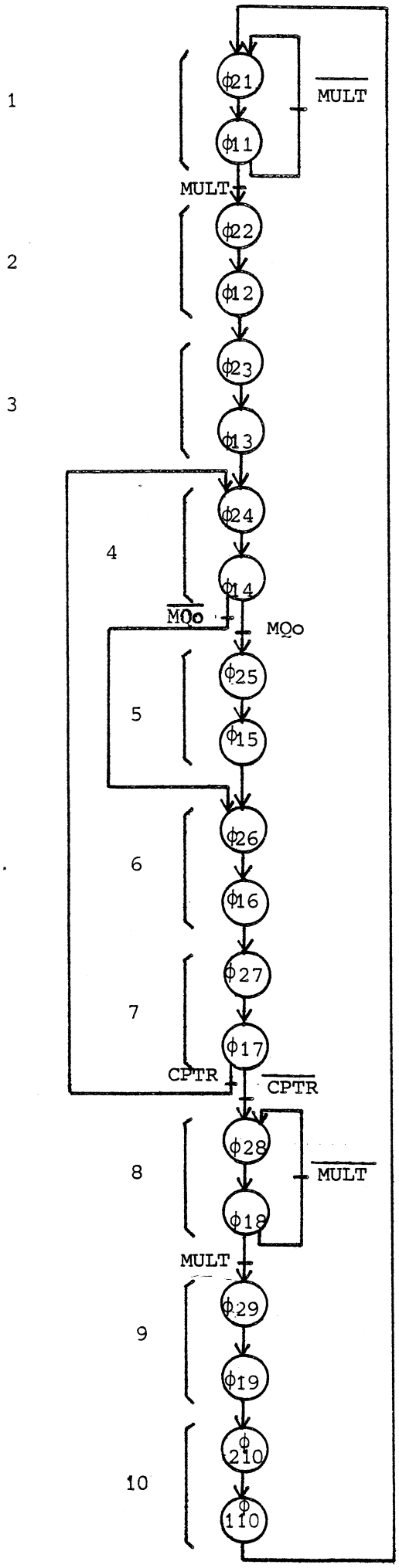


Figure 21 : Graphe de contrôle interprété et temporisé



	P.C.	P.O.
1	état 1	C10 ϕ2 ou rien
	échantillonnage de MULT	rien
2	2	rien
	rien	C1 C2 C3 ϕ1
3	3	C1 C2 C3 ϕ2
	rien	C4 ϕ1
4	4	C4 ϕ2 ou C7 ϕ2
	MQo	rien
5	5	rien
	rien	C5 ϕ1
6	6	C5 ϕ2 ou rien
	rien	C6 ϕ1
7	7	C6 ϕ2
	CPTR	C7 ϕ1
8	8	C7 ϕ2 C8 ϕ2
	MULT	C8 ϕ1
9	9	C8 ϕ2
	rien	C9 ϕ1
10	10	C9 ϕ2
	rien	C10 ϕ1

Figure 22 : Décomposition en phases du graphe

Les intérêts de la logique biphasée par rapport à la logique statique sont :

- un gain en rapidité ave les deux horloges $\phi 1, \phi 2$
- un gain en surface : les transistors sont moins nombreux, les problèmes de connexions sont simples, en effet il n'existe pas de bouclage dans les points de mémorisation à l'opposé des points de mémorisation statique (cf. la bascule D utilisée dans le circuit extracteur en VII).

Par contre, la logique biphasée nécessite deux horloges $\phi 1$ et $\phi 2$. De plus, la conception est plus complexe : les pass transistors s'emploient avec précaution, il est plus facile d'assembler des portes uniquement.

V - 3. Les solutions dans le choix de la partie contrôle

Pour les solutions câblées, on distingue essentiellement les solutions à codage compact et la solution décodée (AMB 81).

La solution entièrement décodée

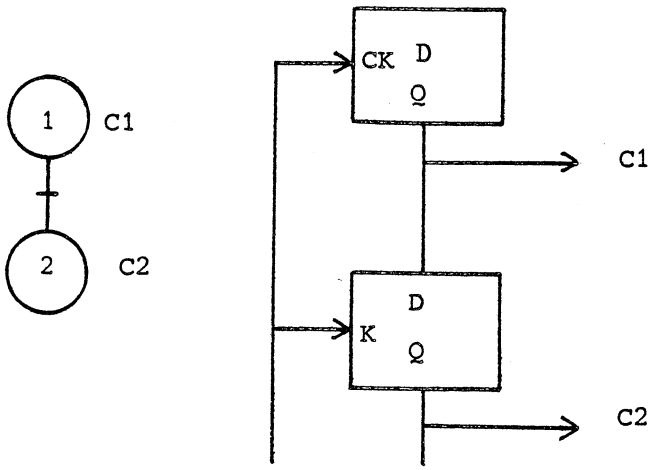
Dans cette solution appelée solution à organigramme câblé, un élément de mémorisation est affecté à chaque noeud du graphe de contrôle (on travaille à un niveau donné), des portes ET réalisant les aiguillages de test, des portes OU les jonctions du graphe de contrôle. Le schéma en logique aléatoire est donc obtenu de façon automatique à partir du graphe de contrôle (figures 23 a et b)

La solution à codage compact

Les noeuds du graphe sont codés par un nombre minimal de variables internes. Les états sont donc implémentés sur un nombre minimal de bascules. $\log_2 n$ points de mémorisation si n est le nombre d'états.

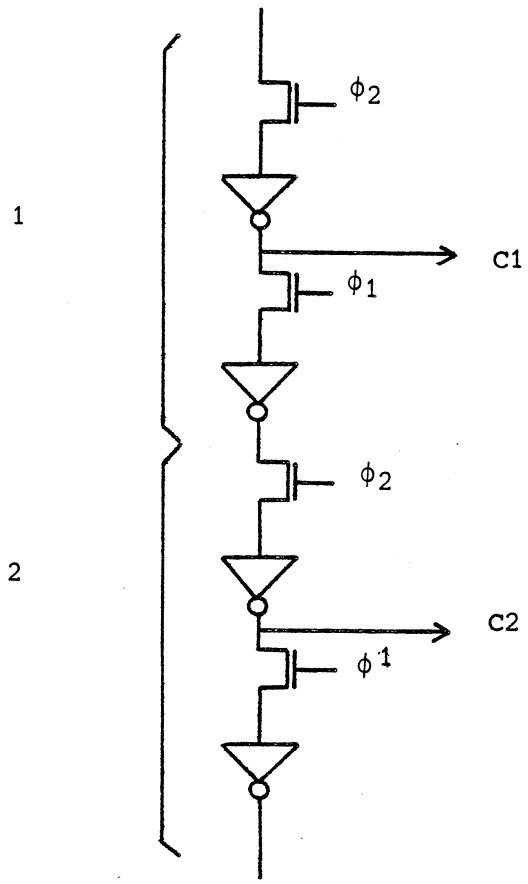
Les fonctions de transition entre états, ainsi que les fonctions de sortie sont implémentées en général sur un PLA. La programmation du PLA est obtenue automatiquement par lecture du graphe de contrôle. La méthode est explicite dans (AMB 81) (figure 24 a,b)

Pour les exemples traités : extracteur et multiplieur, la partie contrôle pourra être réalisée avec l'une ou l'autre de ces solutions.



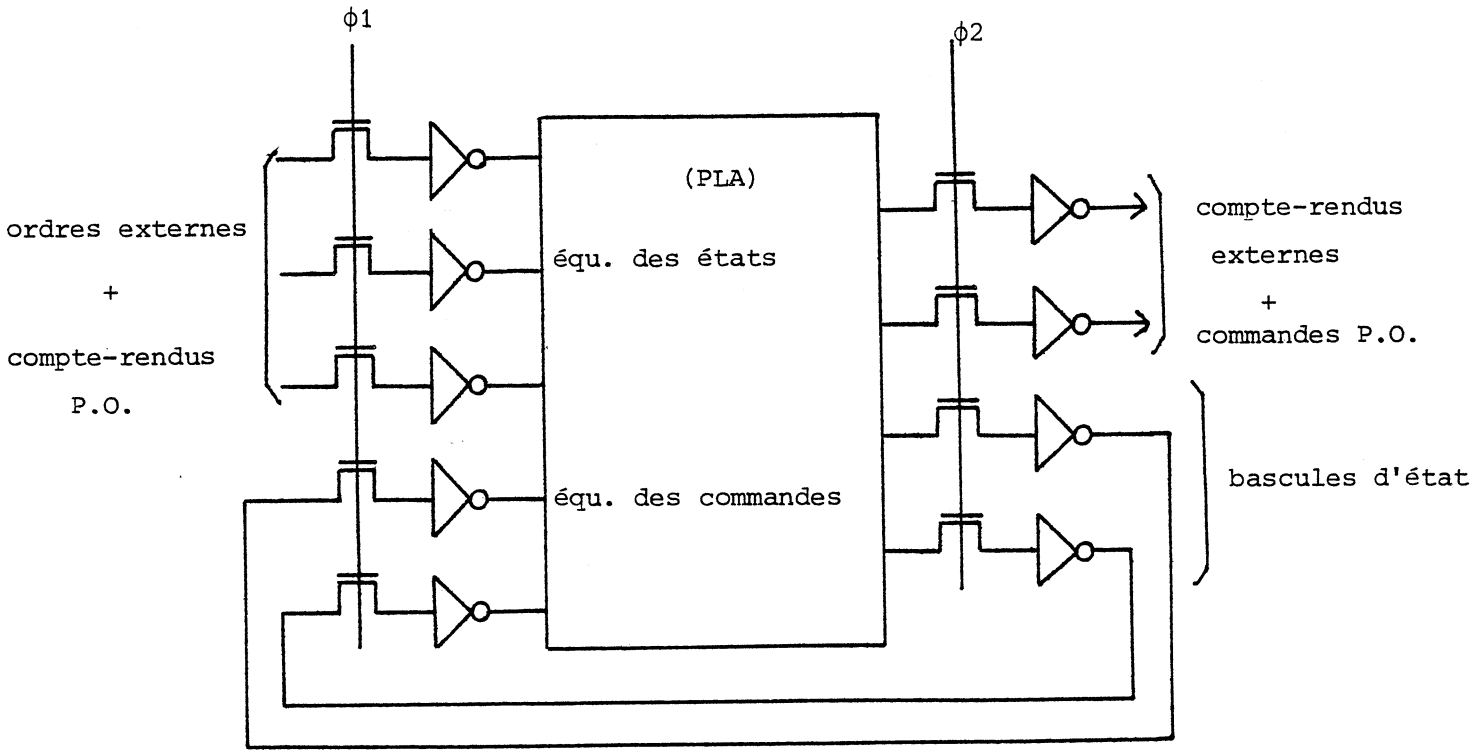
Horloge

a) logique statique

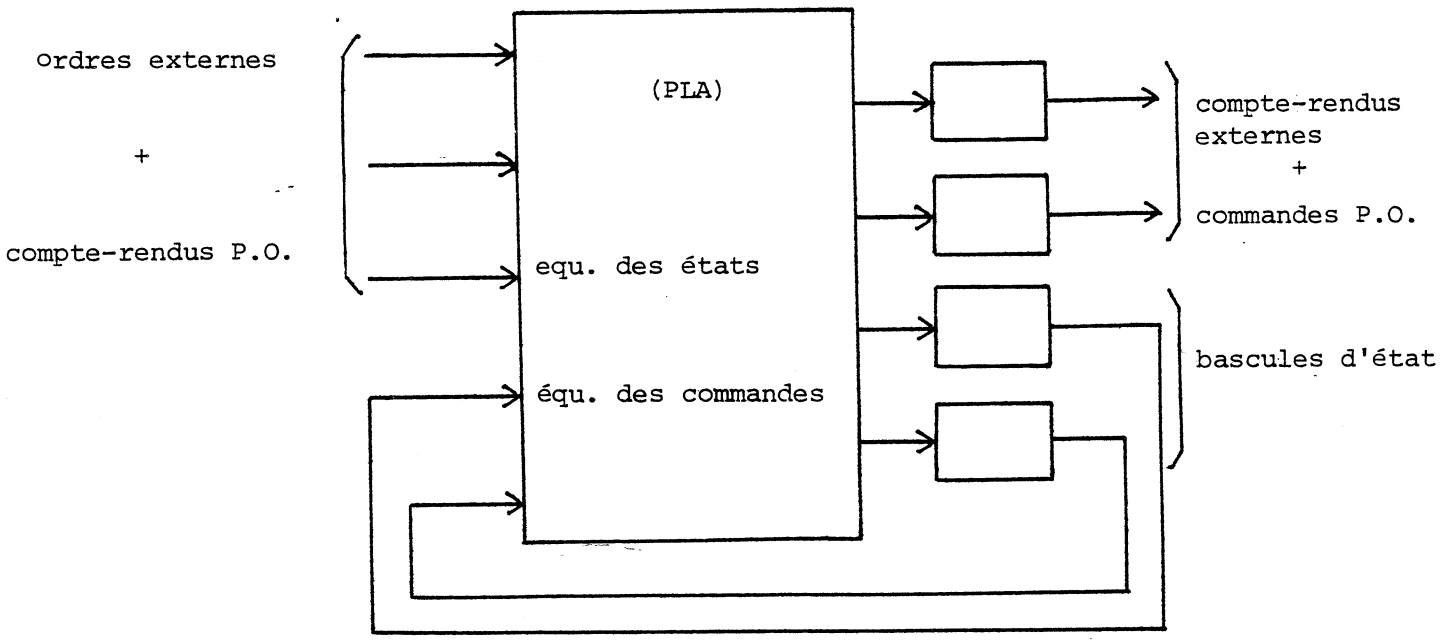


b) logique biphasée

Figure 23 : Solution entièrement décodée, passage en séquence



b) logique biphasée



a) logique statique

Figure 24 : Solution à codage compact, PLA

VI - EVALUATION PREDICTIVE DE SURFACE

VI - 1. Evaluation de surface

Dans cette étape, une tehcnologie NMOS 6µ à déplétion en charge est prise comme exemple. La même méthode peut être appliquée à d'autres technologies.

Pour définir la surface des éléments du circuit, deux types d'éléments sont considérés :

i) les éléments ayant une structure répétitive : PLA,ROM, RAM, registres à déalages statique ou dynamique. Pour ce type d'éléments, il existe des formules pour évaluer la surface dans les différentes technologies. Par exemple, pour un PLA, nous avons (figure 25) :

$$S = (2nE + nS + 4 \text{ à } 5) \times 20\,\mu \times (nM + 6 \text{ à } 10) \times 20\,\mu$$

où nE : nombre d'entrées
nS : nombre de sorties
nM : nombre de monômes

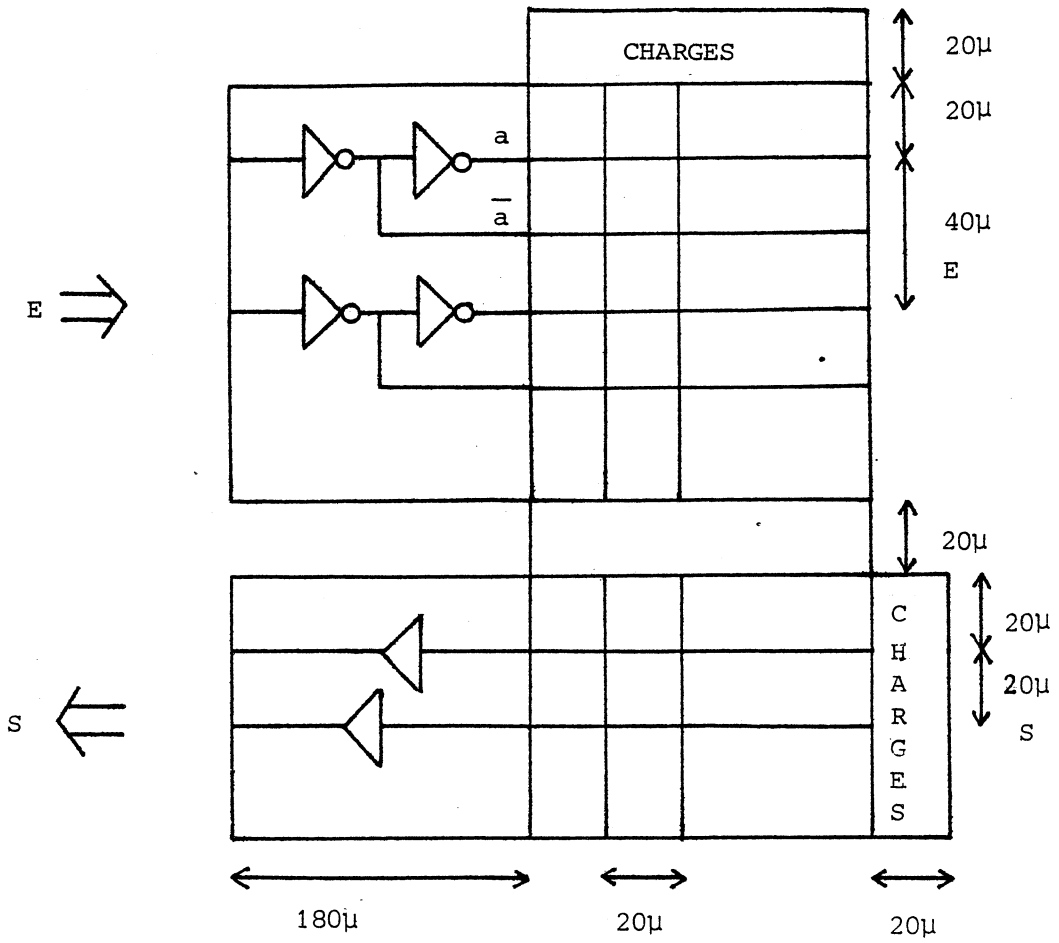


Figure 25 : Surface d'un PLA

ii) Les éléments contenant de la logique aléatoire. Nous considérons deux types d'implantation :

- l'implantation au micron
- l'implantation dans un symbolisme fonctionnel (par exemple le symbolisme MDMOS (LEB 80) où une représentation symbolique, permet le placement et l'interconnexion automatiques.

La surface est ici évaluée en prenant la densité d'intégration calculée comme le nombre de transistors par mm2. Par exemple :

D = 1000 transistors/mm2 pour une implantation au micron
D = 800 transistors/mm2 pour une implantation symbolique MDMOS

L'estimation de la surface totale de la puce, avec une précision de ± 20 %, se fait comme suit :

S1 surface active (transistors) = surface de la logique répétitive + surface de la logique aléatoire

S2 surface occupée par les interconnexions = 0,8 S1

S3 surface occupée par les plots d'entrées/sorties = N/10 + n/8 en mm2

avec N : nombre de plots d'entrées/sorties
n : nombre de buffers basse impédance (équivalents TTL)

Surface totale de la puce = S1 + S2 + S3

Pour le circuit d'extraction de racine carrée une évaluation prédictive de surface est faite sur les différents modules de la partie opérative et de la contrôle (cf. §VII-4).

VI - 2. Pré-implantation

A cette étude de surface, doit être associée une étude de prépositionnement (first floor).

Cette étape concerne le placement des éléments ou blocs et leurs connexions. Nous décomposons la pré-implantation en deux étapes :

- définition des blocs
- assemblage des blocs

i) définition des blocs

La forme, la surface, les entrées-sorties de chaque bloc doivent être connues. Ceci est facile pour les éléments ayant une structure répétitive et pour les éléments implantés en symbolique. Les éléments implantés au micron sont essentiellement déformables, leurs formes et leurs entrées-sorties sont définies en tenant déjà compte des problèmes d'interconnexions.

ii) Assemblage des blocs

Tous les blocs Bi étant définis, pour faire le schéma de pré-implantation, nous considérons le "graphe de connexions" défini comme suit (figure 26) :

- à chaque bloc Bi est associée sa surface Si
- les arcs représentent le nombre de connexions reliant les blocs.

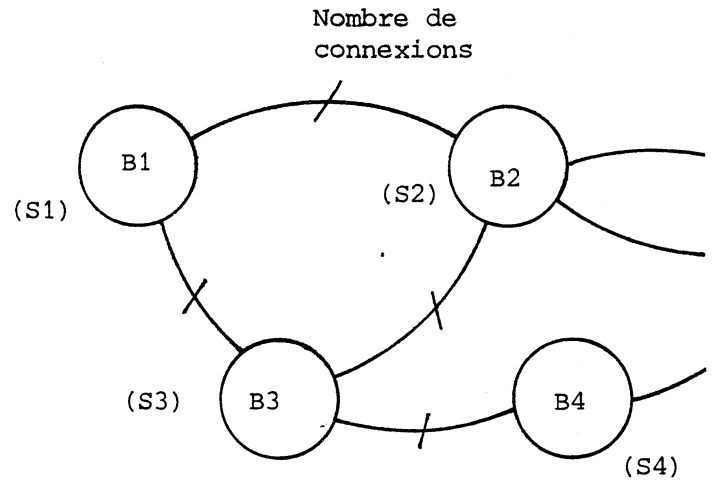


Figure 26 : Graphe de connexions

Nous plaçons d'abord :

- soit les blocs les plus reliés avec l'extérieur autour du circuit, c'est à dire près des plots d'entrées-sorties
- soit un noyau initial (pouvant être constitué d'un ou plusieurs blocs) si:

- il existe une priorité dans le cahier des charges : rapidité, consommation..., ce qui implique qu'il y a une partie importante et délicate du circuit à implanter : forme et surface imposées.

- il existe un élément qui prend de la place : logique répétitive importante : RAM par exemple.

Le reste du circuit va s'implanter autour.

Le placement des autres blocs peut se faire avec une méthode constructive, utilisant la notion de barycentre (HAN 72)(WIL 74).

Pour ceci, on associe à chaque bloc :

- sa surface
- son nombre de connexions avec les éléments déjà placés
- son poids : nombre de connexions de cet élément avec tous les autres éléments

L'élément le plus connecté aux éléments déjà positionnés est choisi (à nombre de connexions égal : celui de poids maixum). L'élément sélecté est alors placé au barycentre des éléments auxquels il est connecté ou le plus près possible de celui-ci.

Très souvent, le préplacement respectera la séparation entre partie contrôle et partie opérative, les canaux de communication étant organisés autour du flux de données traitées par la partie opérative et le flux des siganux de contrôle échangés entre la partie régulière (tranche de bits), la partie contrôle est, elle, constituée d'un ou plusieurs modules de taille importante ROM ou PLA connecté à un ensemble important de modules secondaires. On pourra, à ce niveau, utiliser, à profit, une méthode de placement automatique organisé autour de ce noyau central, ainsi que des méthodes de tracé automatique de connexions telles que celles définies en (14) (niveau des blocs).

Pour le circuit d'extraction de racine carrée, l'implantation de la partie opérative se fera de façon répétitive en décomposant par bits; pour définir le placement des modules pour un bit, on considère le graphe de connexions défini précédemment, on obtient facilement un schéma de pré-implantaiton (cf. § VII-1.1).

VII - SCHEMA LOGIQUE AU DESSIN DES MASQUES POUR LE CIRCUIT D'EXTRACTION DE RACINE CARREE

Les différentes étapes pour passer du schéma logique au dessin des masques du circuit extracteur sont représentées sur l'organigramme de la figure 27.

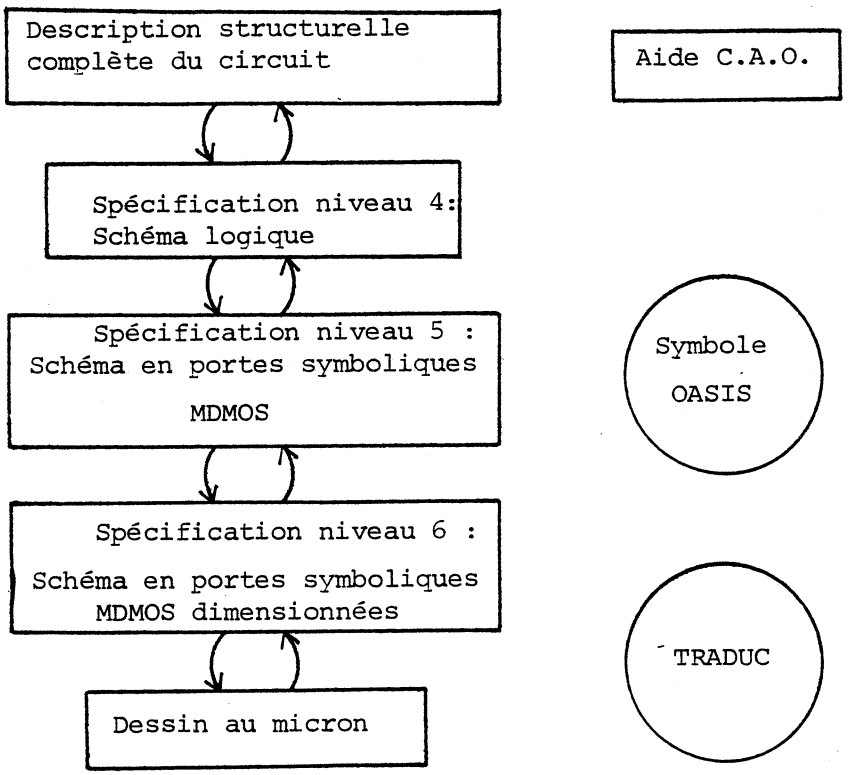


Figure 27 : Etapes finales de la conception

Remarque : Ce circuit sera réalisé en technologie NMOS ; les règles de dessin sont celles de la technologie à 6µ et à titre pédagogique (enseignement), on utilisera les règles simplifiées du service CMP (CMP 82). De façon plus précise, il sera entièrement réalisé avec le symbolisme fonctionnel MDMOS (MAJ 78).

VII - 1. Conception logique : spécification de niveau 4

On donne d'abord le codage des commandes qui a été choisi :

1) Multiplexeurs à deux entrées

Multiplexeurs	1		2		3		5	
Commandes	C1	Sortie	C2	Sortie	C3	Sortie	C6	Sortie
	0	Extérieur	0	Sortie UAL	0	Sortie UAL	0	R1
	1	R2 div 2	1	Sortie R0	1	R2 div 2	1	R2

2) Multiplexeurs à trois entrées

Multiplexeurs	4			6		
Commandes	C4	C5	Sortie	C7	C8	Sortie
	0	0	R3 div 4	0	0	R3
	0	1	R3 mult 4	0	1	R3 mult 2
	1	0	l"arithmétique"	1	0	R2
	1	1	l"arithmétique"	1	1	R2

3) Commandes UAL

Commandes op1	op2	Sortie
0	0	passage de l'entrée B
0	1	soustraction et résultat
1	0	passage de l'entrée B
1	1	addition et résultat

4) Registres

Commandes ch	Sortie
0	valeur précédente
1	Chargement de la nouvelle valeur

L'opérande "y" est chargé dans le registre R0 sur le signal "début".
Le résultat est disponible dans le registre R0 lorsque le signal "fin" a été mis à 1 par la partie contrôle.

L'ensemble des signaux de contrôle nécessaireau déclenchement et au déroulement des microopérations est associé aux phases suivant le tableau de la figure 28

Commandes / Phases	C_1	ch_0	C_2	ch_1	C_3	ch_2	C_4	C_5	ch_3	C_6	C_7	C_8	op_1	op_2
1	0	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ
2	ϕ	0	1	1	ϕ	0	1	ϕ	1	ϕ	ϕ	ϕ	ϕ	ϕ
3	ϕ	0	ϕ	0	ϕ	0	ϕ	ϕ	0	0	0	0	0	1
4	ϕ	0	ϕ	0	ϕ	0	0	1	1	ϕ	ϕ	ϕ	ϕ	ϕ
5	ϕ	0	ϕ	0	0	1	ϕ	ϕ	0	ϕ	0	0	ϕ	0
6	ϕ	0	ϕ	0	1	1	0	0	1	ϕ	ϕ	ϕ	ϕ	ϕ
7	ϕ	0	ϕ	0	0	1	ϕ	ϕ	0	1	0	0	0	1
8	ϕ	0	ϕ	0	ϕ	0	ϕ	ϕ	0	0	1	ϕ	0	1
9	ϕ	0	0	1	ϕ	0	ϕ	ϕ	0	0	1	ϕ	0	1
10	ϕ	0	ϕ	0	0	1	ϕ	ϕ	0	1	0	1	1	1
11	1	1	ϕ	0	ϕ	0	ϕ	ϕ	0	ϕ	ϕ	ϕ	ϕ	ϕ

Figure 28. Signaux de contrôle

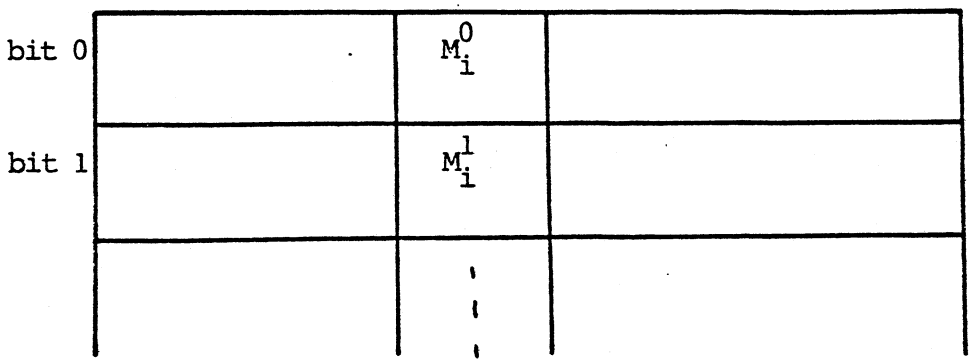
7.1.1. Conception de la partie opérative

a) Structure générale

La partie opérative est partitionnée en 5 modules : M1, M2, M3, M4 et M5 donnés sur la figure 29. L'algorithme choisi implique le nombre de bits suivant pour les modules :

M1 et M2 : 8 bits b0 à b7
M3, M4 et M5 : 9 bits b0 à b8

L'implantation de la partie opérative se fera de façon répétitive, en découpant par bit :



Pour définir le placement des modules pour un bit, on considère le graphe de connexions dans lequel un noeud représente un module ; l'arrête entre 2 noeuds est étiqueté par le nombre maximal de connexions entre les 2 modules correspondants (sans tenir compte des particularités qui existent pour certains bits).

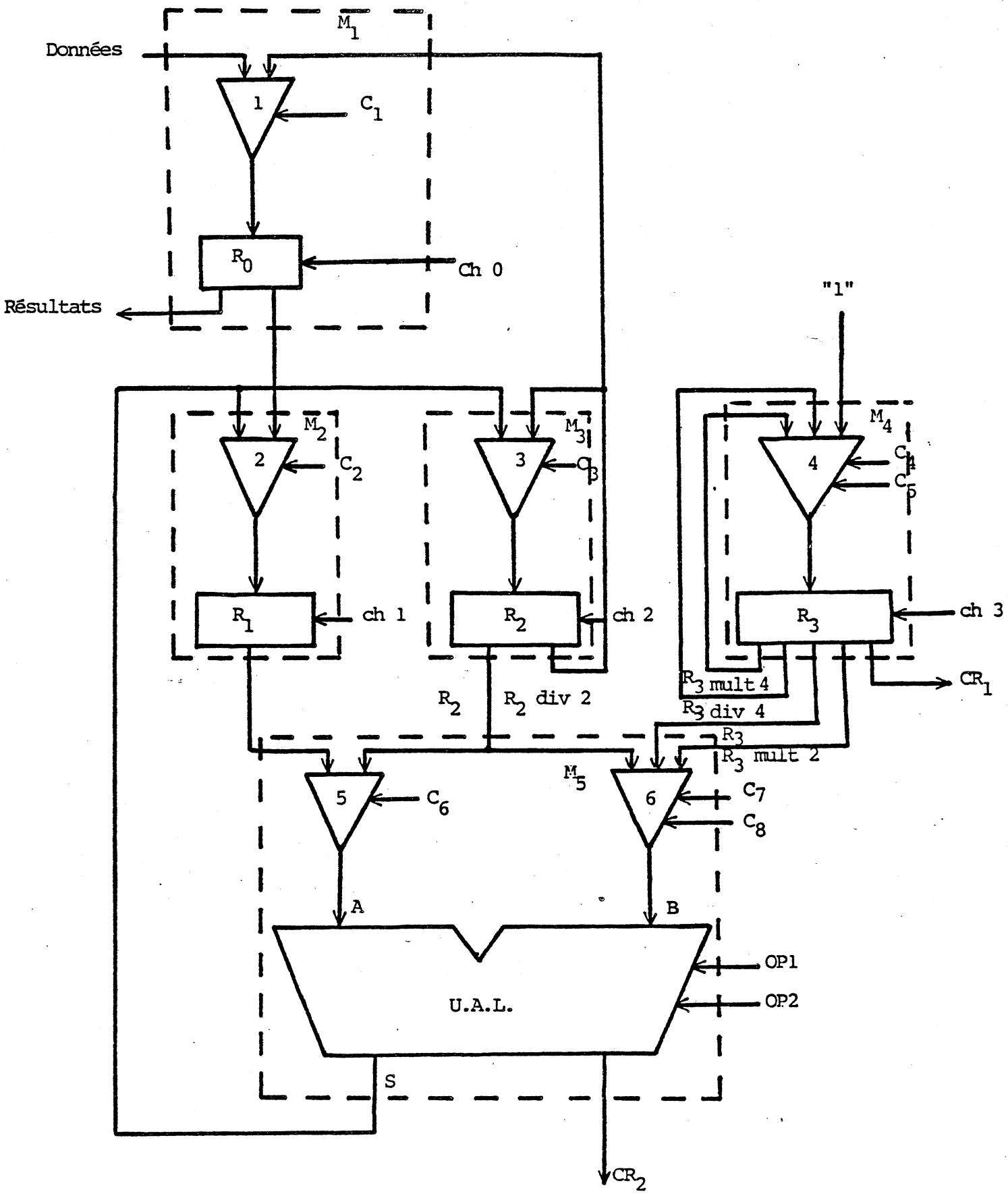
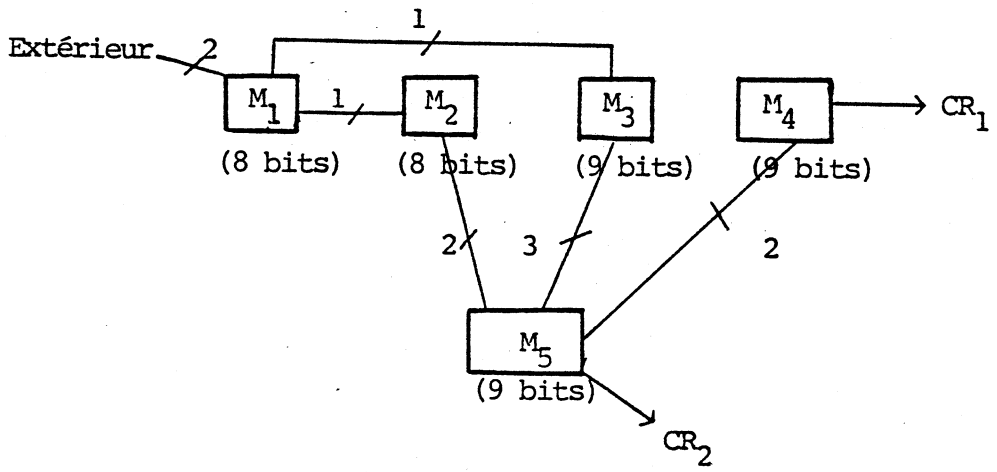


Figure 29 : Modules de la partie opérative

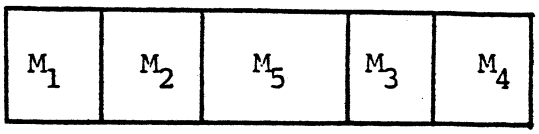
Ce graphe permet de voir facilement quels sont les éléments les plus reliés :



A l'aide du graphe, on aura :

- M1 et M2 voisins (sur 8 bits)
- M1 près des plots (extérieur)
- M3 et M5 voisins (liaisons importantes)
- M4 à une extrémité (CR1)

Avec ces remarques, on choisit :



On a alors le schéma de pré-implantation de la partie opérative (figure 30).

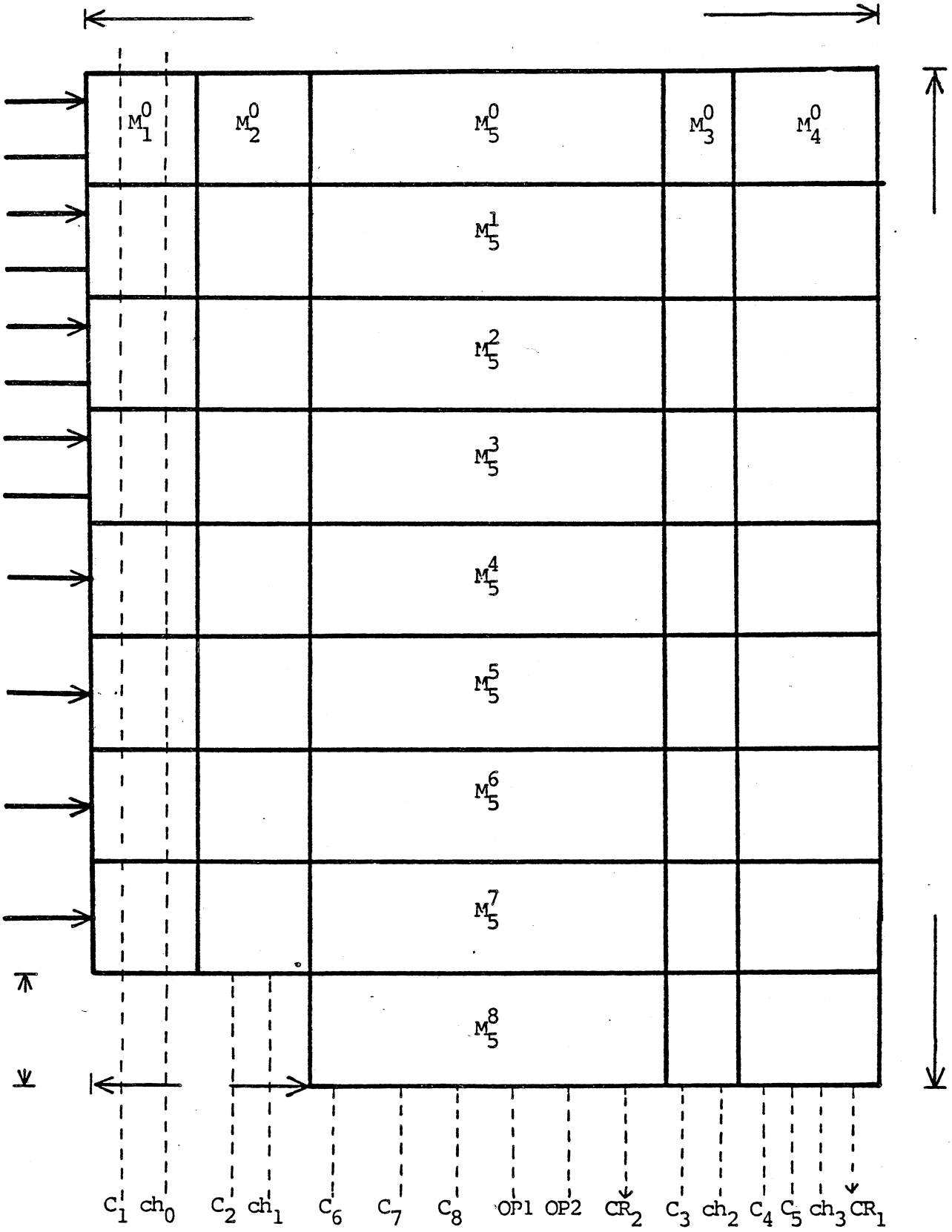


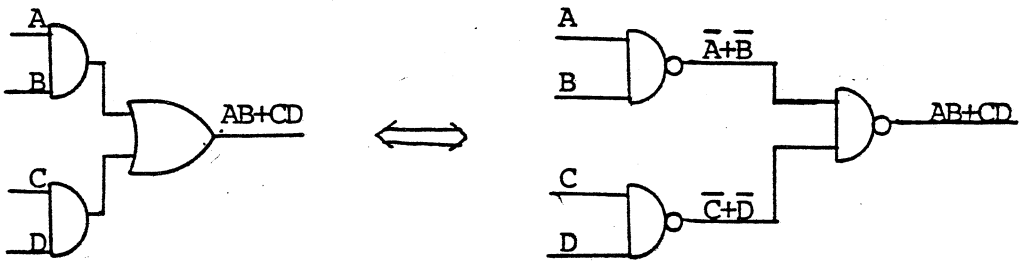
Figure 30. Pré-implantation de la partie opérative

b) Synthèse des différents blocs

La synthèse logique se fera à base de portes NAND. Rappelons qu'une synthèse à 2 couches de portes NAND est "calquée" sur une synthèse à 2 couches ET/OU.

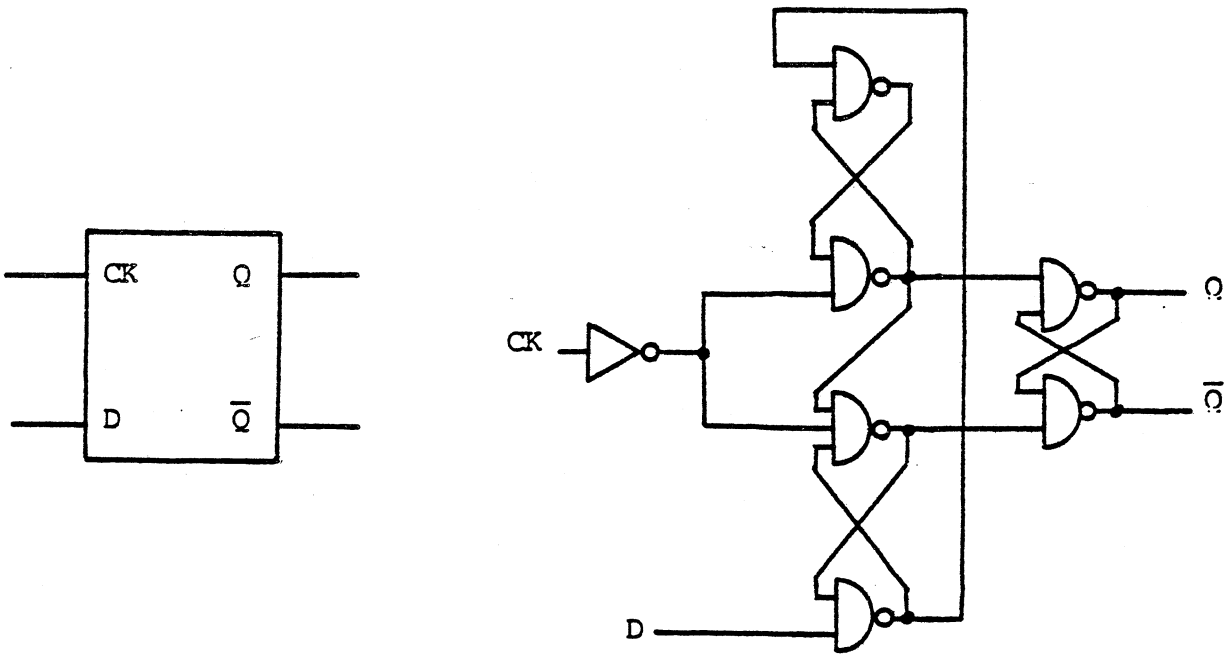
Exemple

AB + CD

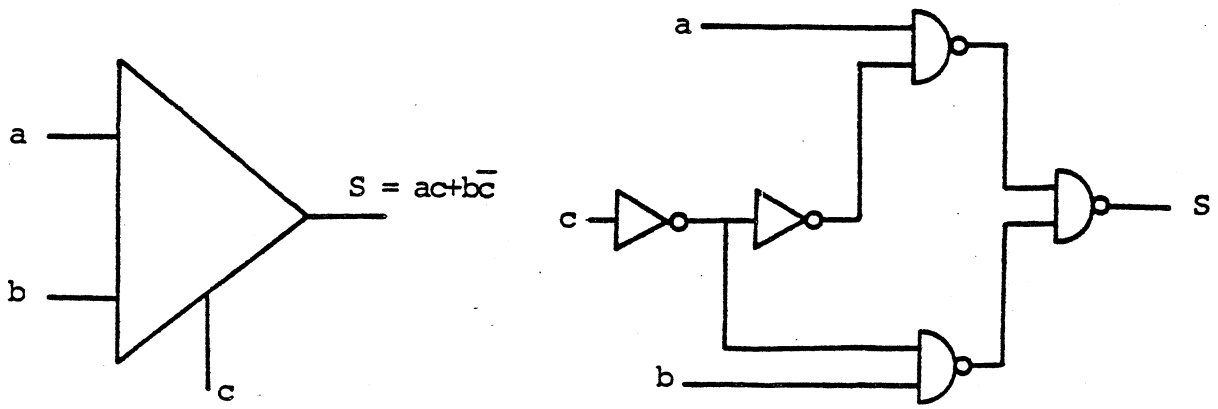


On énumèrera les principaux blocs de la partie opérative du circuit et le schéma logique utilisé :

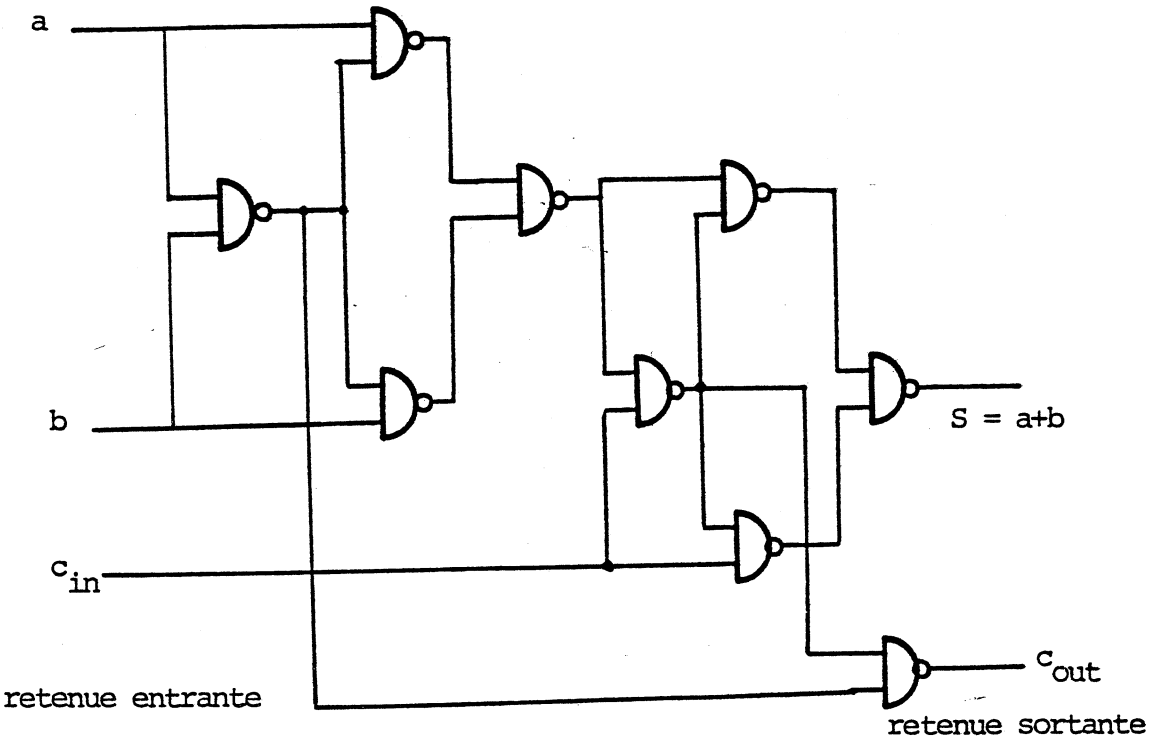
- Point de mémorisation : bascule D à front descendant



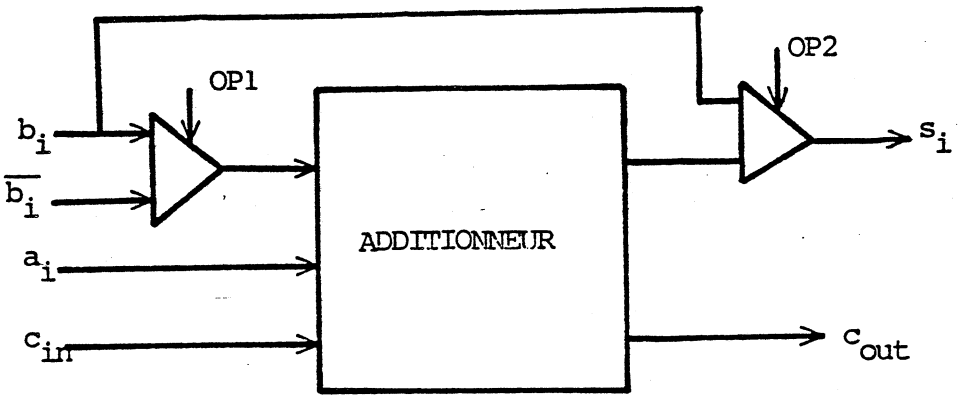
- multiplexeur à deux entrées



- additionneur sur un bit



- U.A.L sur un bit : additionneur - soustracteur

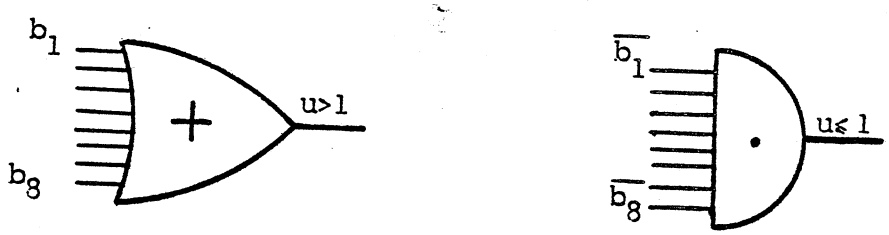


$$c_{in\ bit\ 0} = \begin{matrix} 0 \text{ si addition} \\ 1 \text{ si soustraction} \end{matrix}$$

$$c_{in\ bit(i\neq 0)} = c_{out\ bit(i-1)}$$

- Réalisation des compte-rendus CR1 et CR2

i) CR1 = 1 : la valeur de u est $\leqslant 1$



ii) CR2 = 1 : le résultat de la soustraction est ≥ 0 : ceci est vrai si la retenue du dernier étage de l'UAL (bit 8) = 1.

7.1.2. Réalisation de la partie contrôle

On réalise la partie contrôle par la méthode de "l'organigramme" câblé ; à chaque état est affectée une bascule, cette bascule est active quand l'automate est dans l'état correspondant.

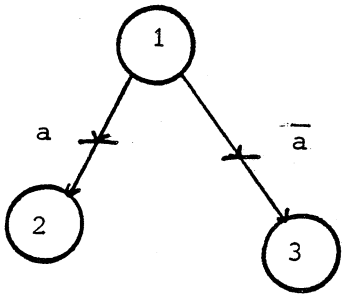
Initialisation : Le signal initialisation active la bascule 1 (attente) et met dans l'état "inactif" les autres bascules ; chaque bascule aura des entrées d'initialisation :

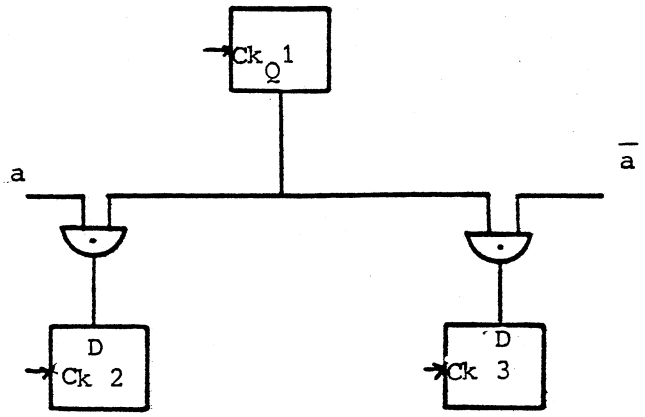
RAZ : mise à l'état inactif
RAU : mise à l'état actif.

Schéma général : Le schéma général sera une copie de l'organigramme de contrôle.

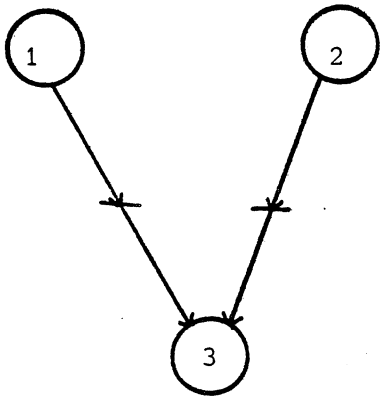
1) Passage en séquence

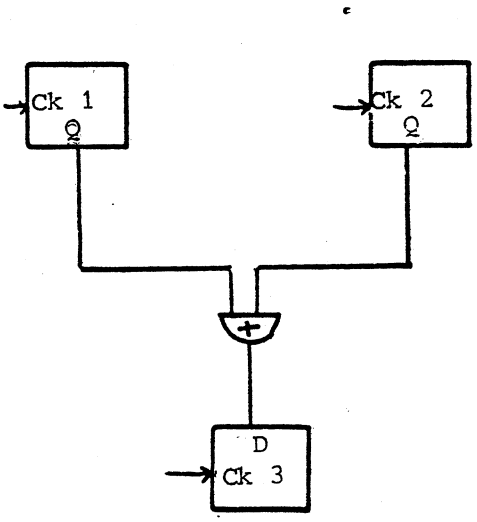
2) Eclatement





3) Jonction





Le graphe de contrôle de la figure 10 en III.3 sera alors réalisé par le schéma de principe de la figure 31.

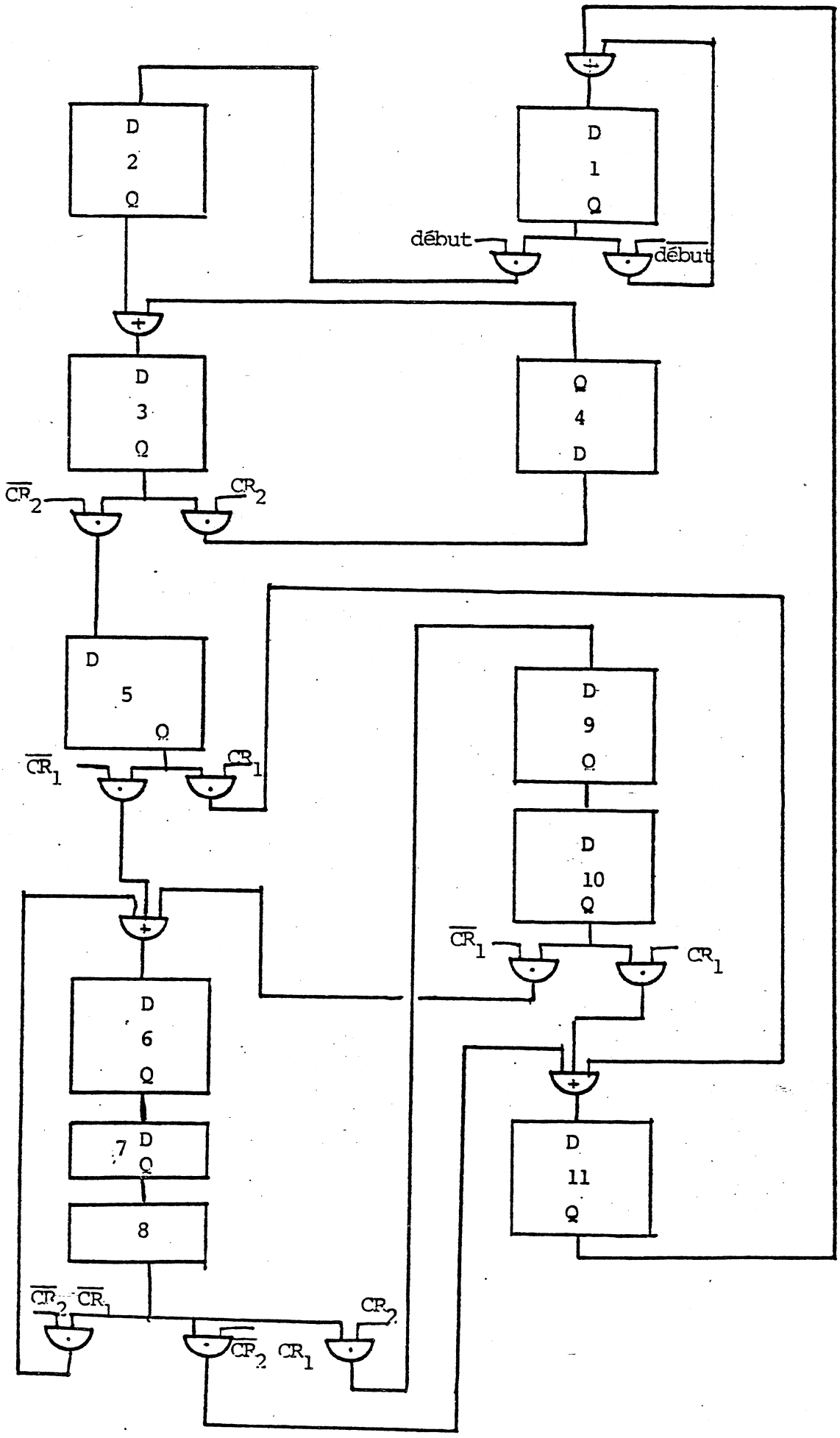
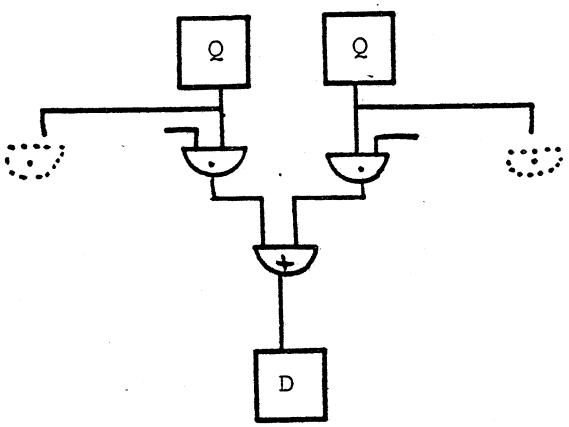
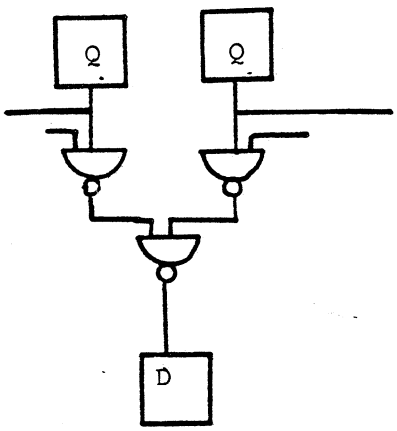
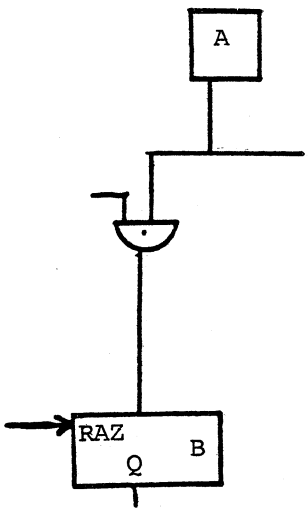


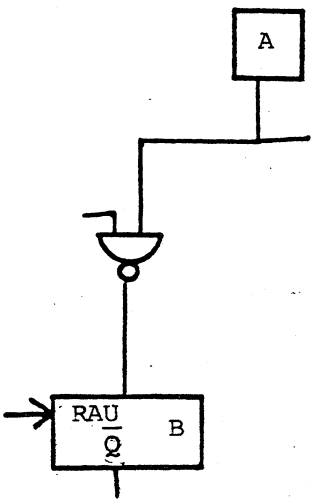
Figure 31 : Schéma de principe de la partie contrôle

Ce schéma utilise des portes ET et OU. Pour une réalisaion en portes NAND on sera amené à utiliser dans cet ordre les 3 règles de modification suivantes :

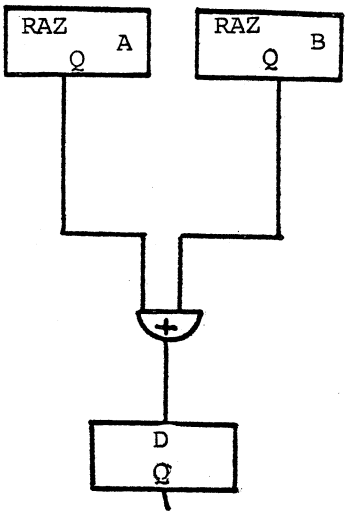


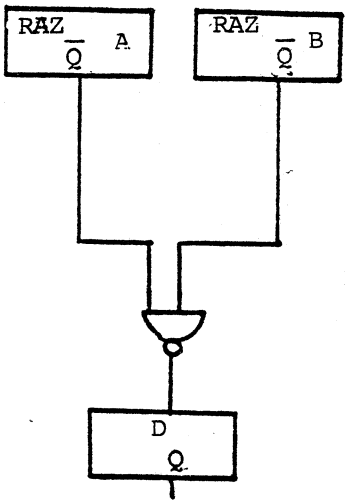






La bascule B devient cette fois active à 0. On l'initialisera donc à "1" et on s'intéressera à sa sortie $\overline{Q}$.

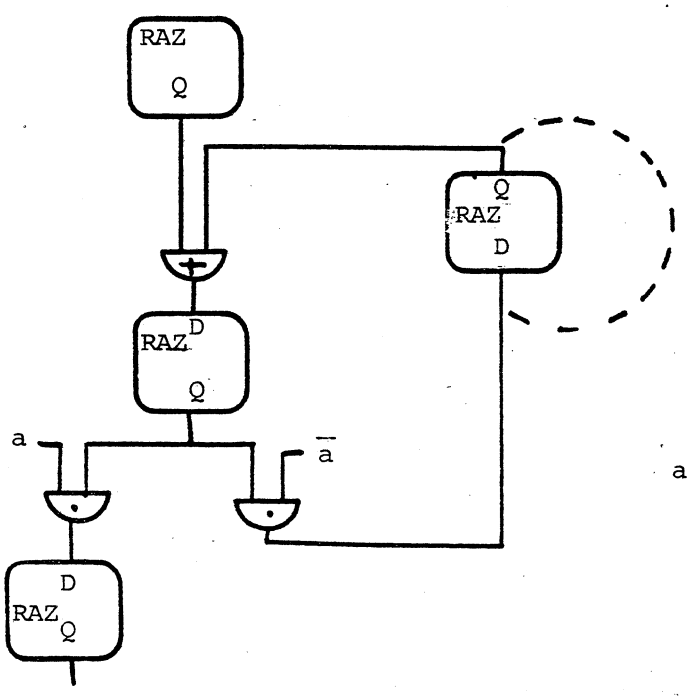


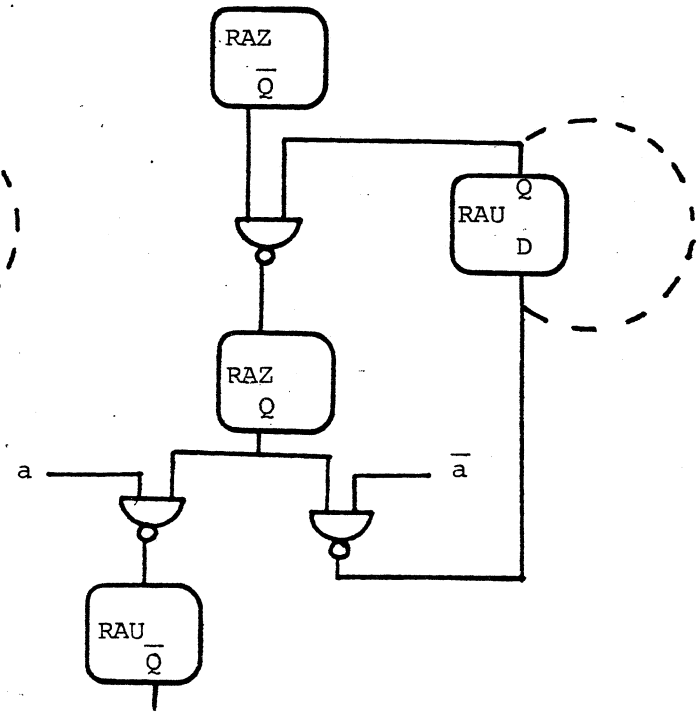


Les bascules A et B sont toujours actives à 1 (donc initialisées à 0) mais on s'intéresse à leur sortie $\overline{Q}$.

Il est à noter que plusieurs de ces règles peuvent coéxister sur la même bascule.

Exemple





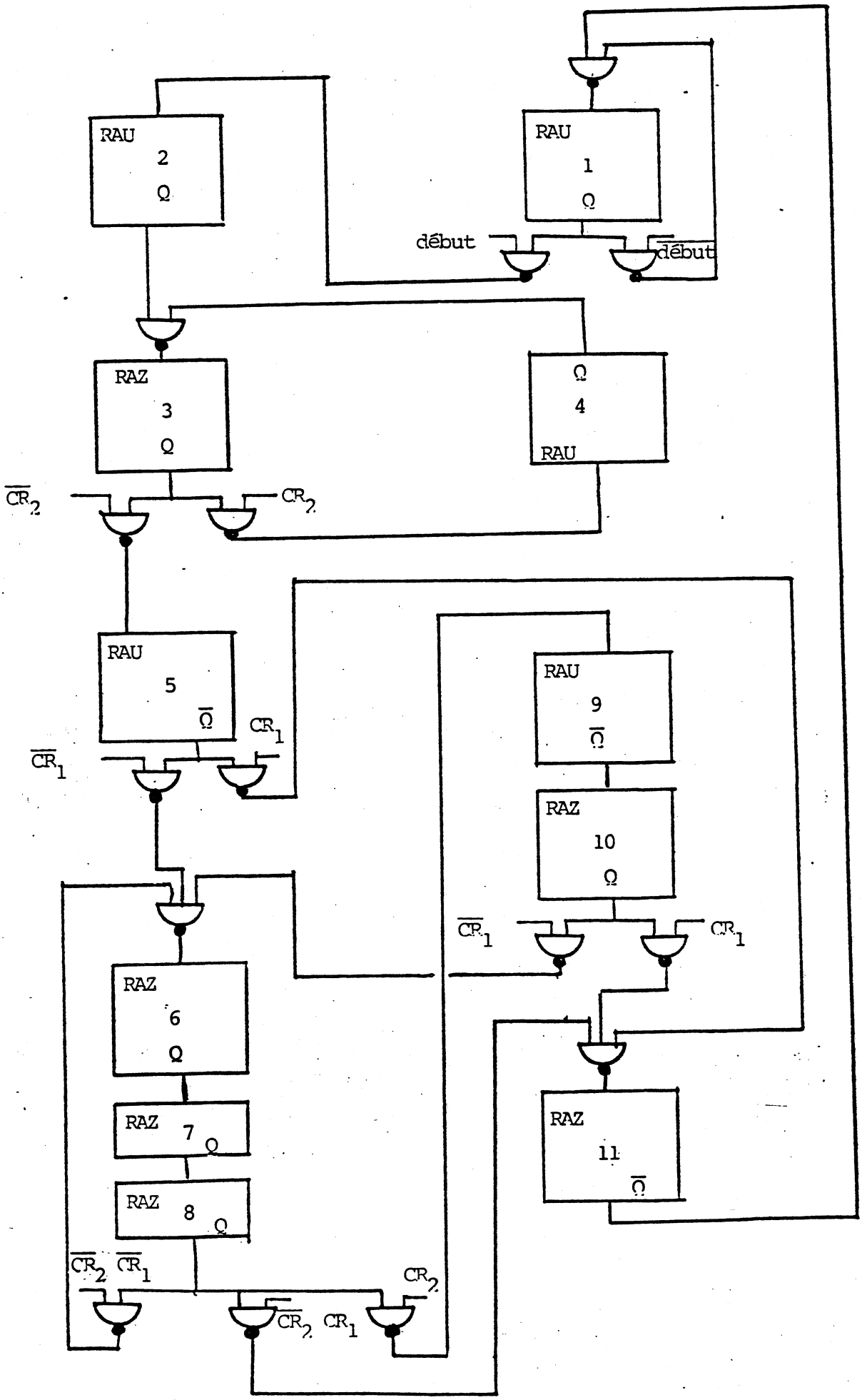


Figure 32 : Schémas en portes NAND de la partie contrôle

Le schéma d'ensemble de la figure 31 devient celui de la figure 32.

Equations des commandes

On notera par

$C4 = 2 + \overline{4} + \overline{6}$ le fait que

C4 = 1 dans l'état 2
0 dans les états 4 et 6
ϕ ailleurs.

On choisira alors de réaliser physiquement 0 soit par 0, soit par 1. En choisissant ϕ=0, on obtient :

C4 = 2
C5 = 4
C2 = 2 = C4
C3 = 6
C6 = 7+10
C7 = 8+9
C8 = 10
OP1 = 10 = C8
OP2 = $\overline{5}$ (ici ϕ = 1)
C1 = 11 (ou $\overline{1}$)

ch0 = $11\overline{h}$ + début $\overline{h}$
ch1 = $(9+2)\overline{h}$
ch2 = $(5+6+7+10)\overline{h}$
ch3 = $(2+4+6)\overline{h}$

Attention ceci ne tient pas compte du fait que certaines bascules sont actives à 0.
Les commandes seront générées ainsi :

C4 = 2 sortie Q2
bascule 2 active à 0

Ampli inverseur

C3 = 6 sortie Q6
bascule 6 active à 1

Le signal Fin = 11 H

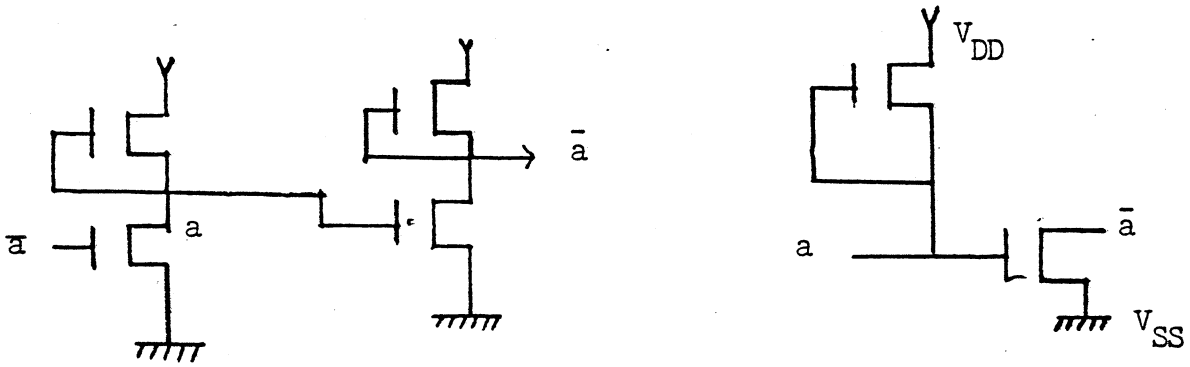
VII - 2. Schémas en portes symboliques MDMOS : spécification de niveau 5

7.2.1. Symbolisme fonctionnel MDMOS

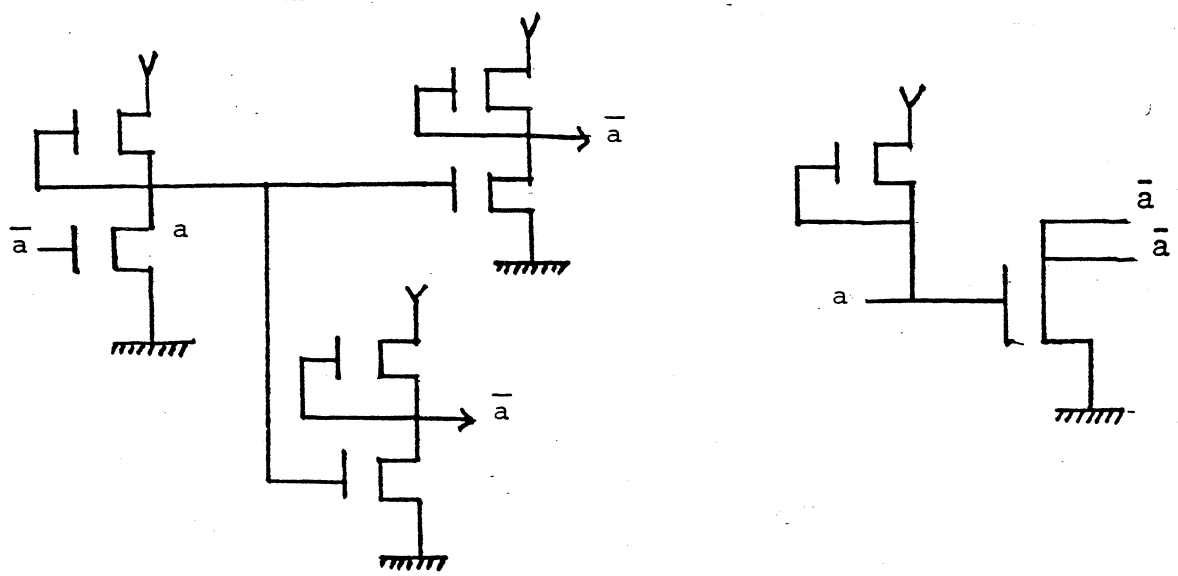
A - Définition

Le symbolisme fonctionnel MDMOS se fait sur grille avec des symboles qui sont les portes MDMOS.

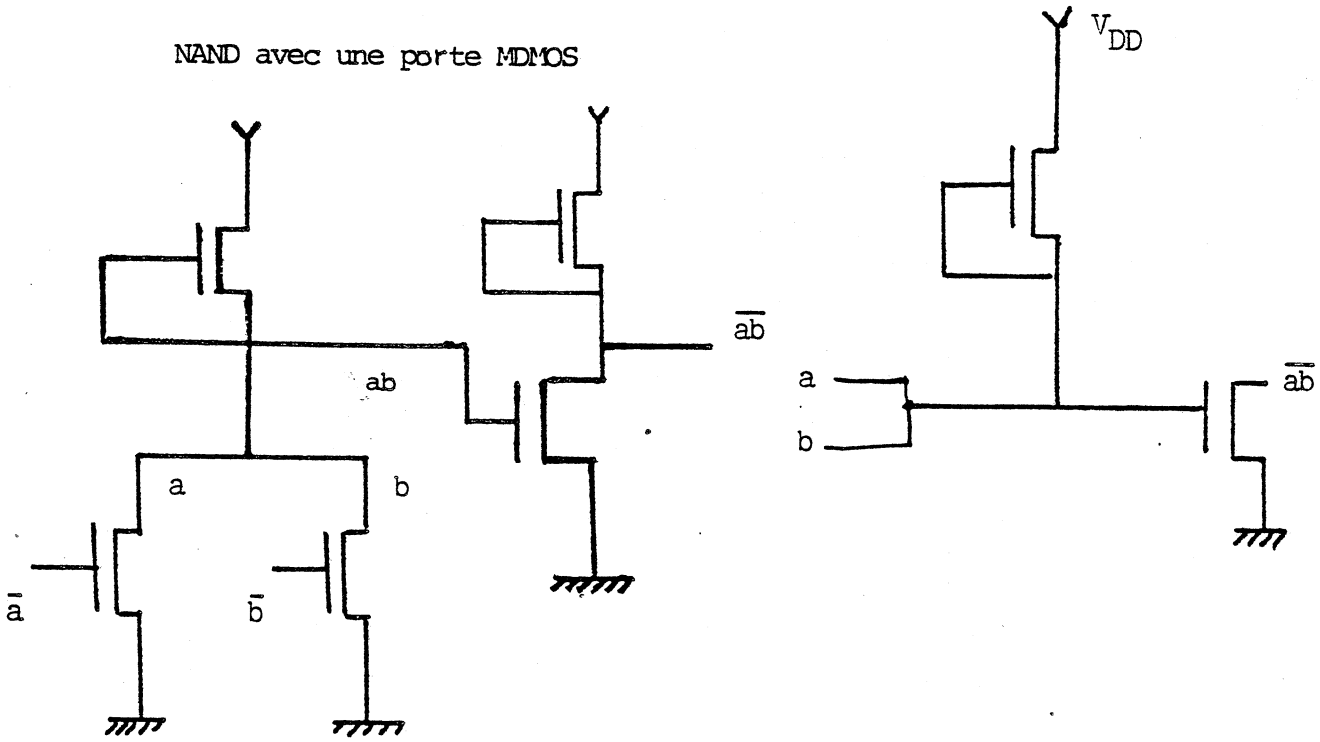
Porte MDMOS (un seul drain)



Porte MDMOS (plusieurs drains)

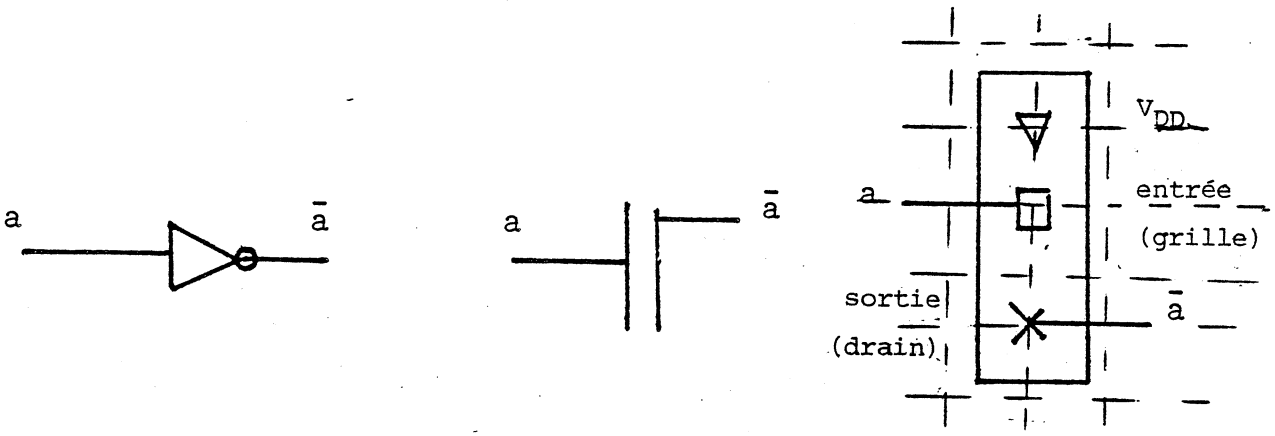


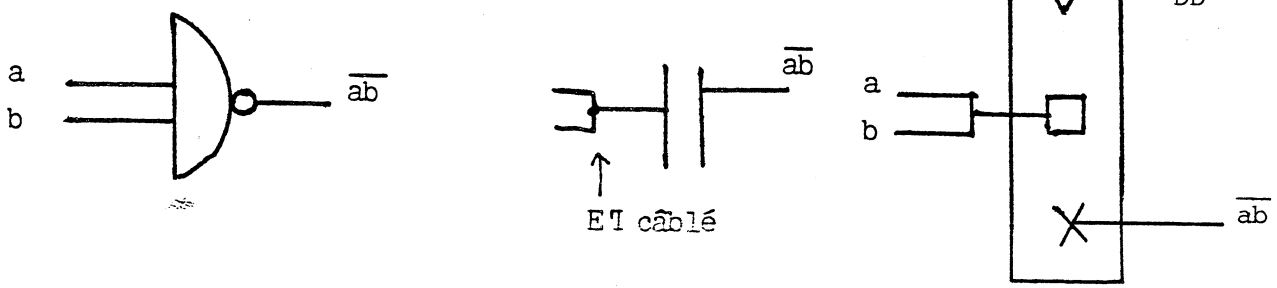
NAND avec une porte MDMOS



différentes représentations :

schéma logique schéma en "peigne" schéma symbolique

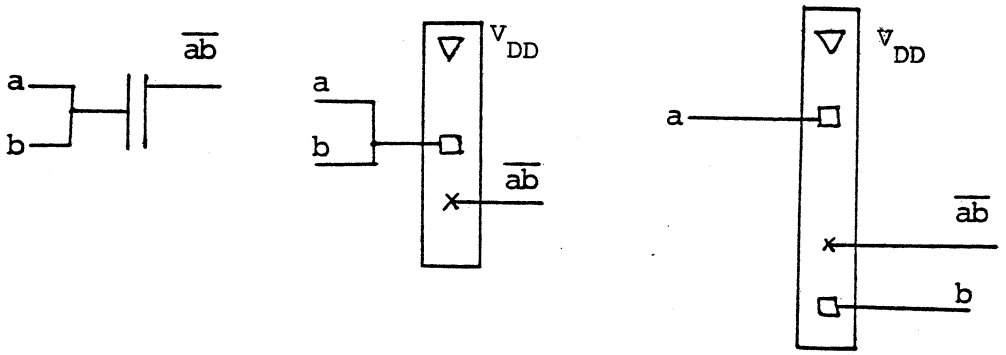




B - Règles d'utilisation

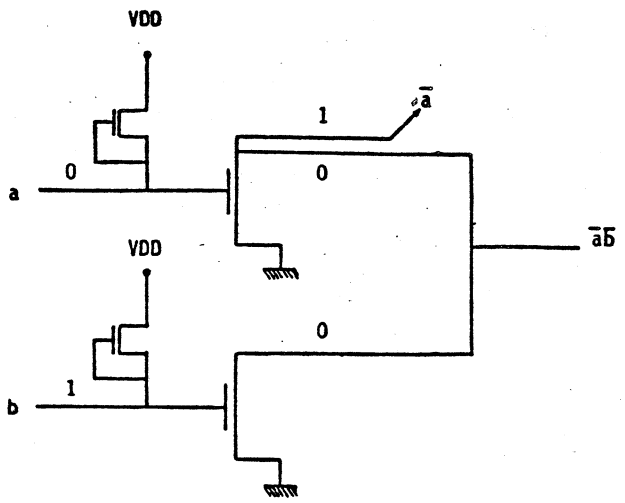
La porte MDMOS a une structure électrique en NOR (transistors signaux en parallèle), mais la synthèse s'effectue en NAND.

Le symbolisme MDMOS permet de réaliser des "ET câblés" :



Sur les figures précédentes, les valeurs des sorties sont indiquées pour les cas où les portes considérées sont prises isolément.

En effet, ces valeurs dépendent également des sorties d'autres portes auxquelles elles sont éventuellement connectées (à cause du ET câblé). Ainsi, une porte MDMOS peut voir ses drains portés à des potentiels différents, le potentiel n'est pas le même sur toutes les grilles attaquées :



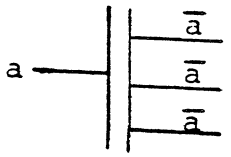
Lorsqu'une sortie d'une porte P attaque les grilles de plusieurs portes Pi, on utilise autant de drains en sortie de P que de grilles des portes Pi à attaquer:

- Les différents signaux de sortie d'une porte peuvent être utilisés avec d'autres sorties et réaliser au moyen de ET câblés des fonctions logiques ne devant attaquer qu'une seule porte.

- La taille du transistor signal étant fixée, le niveau zéro en sortie de P, ainsi que le temps de descente du signal sont dégradés.

Il n'existe donc pas de bifurcation, un drain n'attaque qu'une seule grille, d'où :

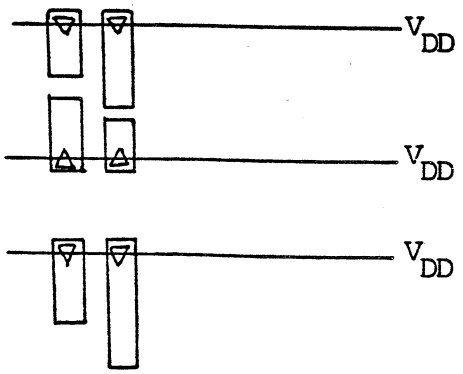
- les transistors signaux sont multi-drains :



- un inverseur existe sur l'entrée des éléments courants (bascules, multiplexeurs) : une seule grille est attaquée.

Pour la même raison (ET câblé), un plot attaque uniquement des inverseurs.

C - L'implantation peut se faire avec des cellules à une ou deux lignes :

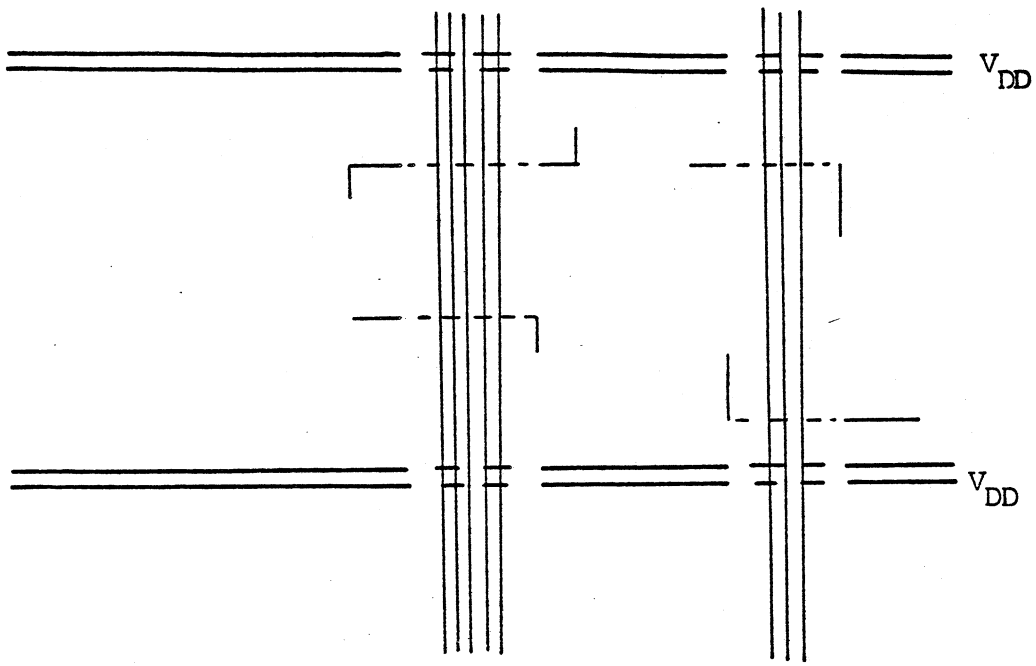


cellule à 2 lignes
(le plus souvent utilisée)

cellule à 1 ligne

La source des transistors signaux est commune et reliée à la masse.

La même ligne d'alimentation est utilisée pour deux cellules adjacentes, les contacts injecteurs peuvent se superposer. L'organisation générale d'un circuit se fait de la façon suivante :

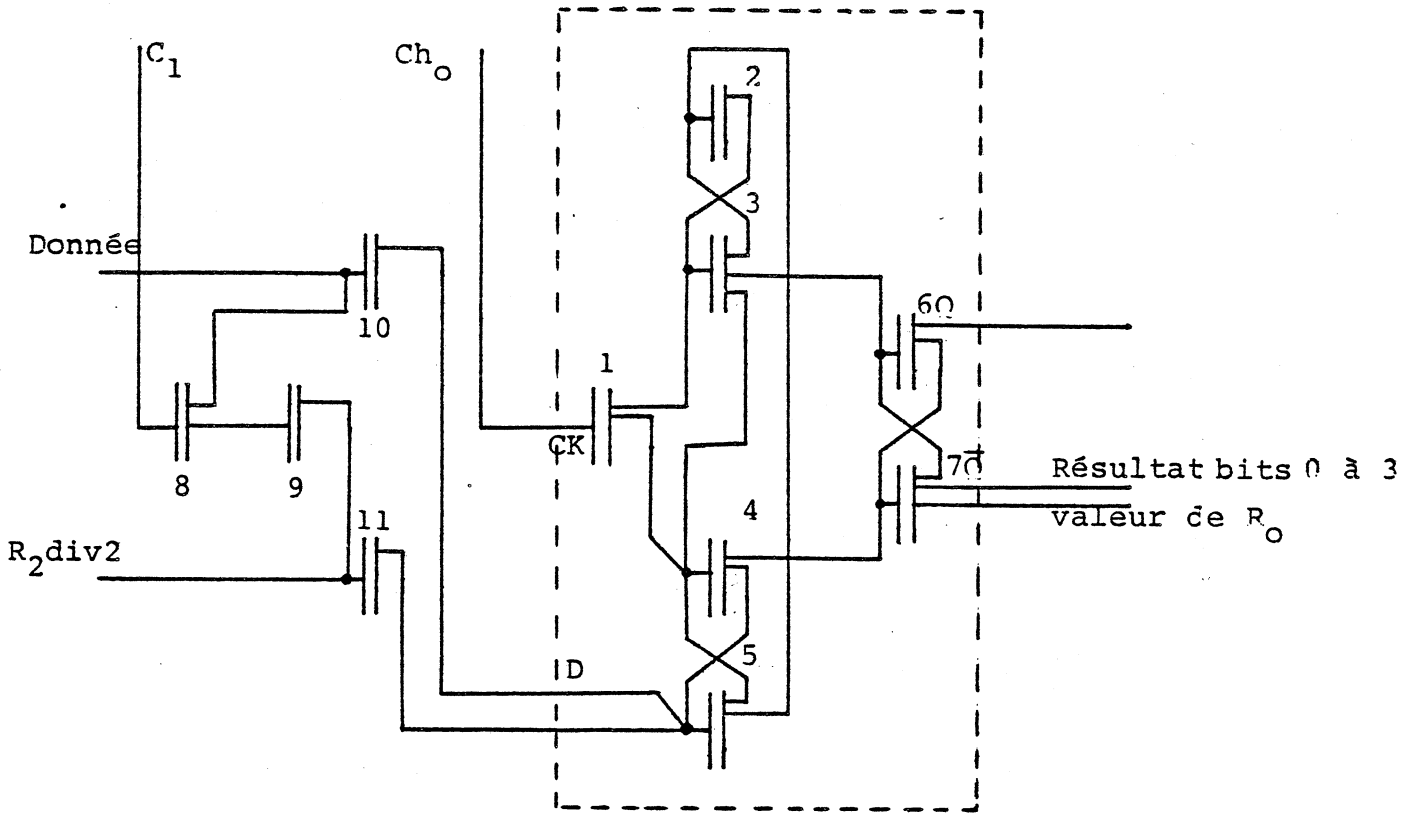


Les commandes sont en aluminium, ce sont les connexions et les alimentations qui sont en passages diffusés. On regroupe les commandes pour diminuer le nombre de passages diffusés. Si la cellule est large, il faut régénérer le VSS: en aluminium comme une commande.

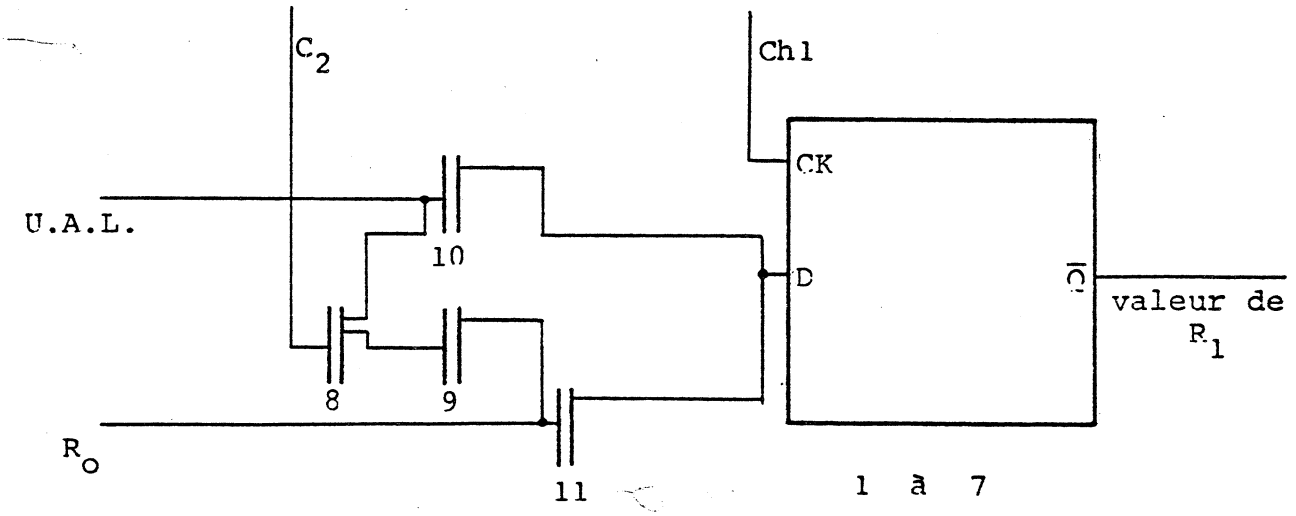
7.2.2. Partie opérative

A - Schéma des différents modules en portes MDMOS

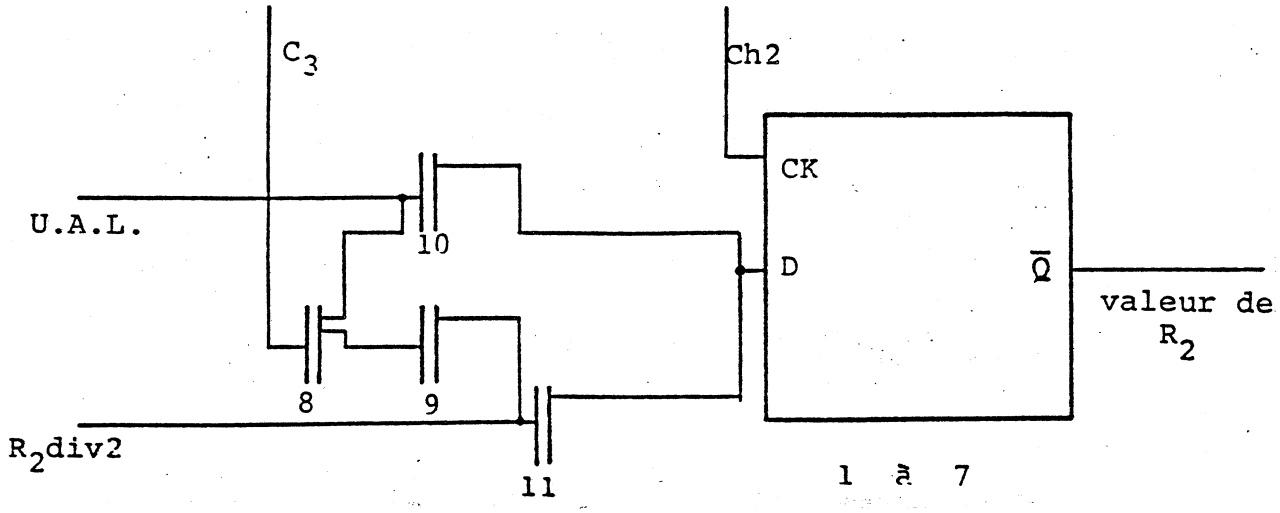
1) Module M1 : bits 0 à 7 : 11 portes



2) Module M2 : bits 0 à 7 : 11 portes



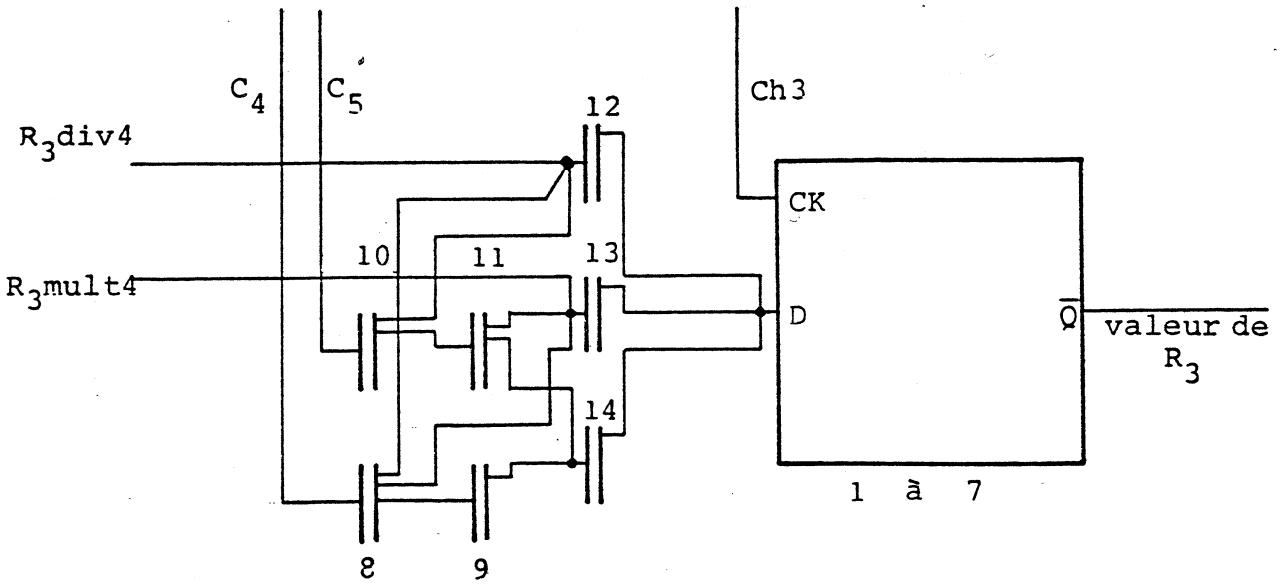
3) Module M3 : bits 0 à 7 : 11 portes
bit 8 : 9 portes



Particularité : pour le bit 8, l'entrée "R2 div 2" est à 0 ; on supprime les portes 9 et 11, et la connexion "R2 div 2".

4) Module M4 : bit 0 : 13 portes
bits 1, 7 et 8 : 11 portes
bits 2 à 6 : 12 portes

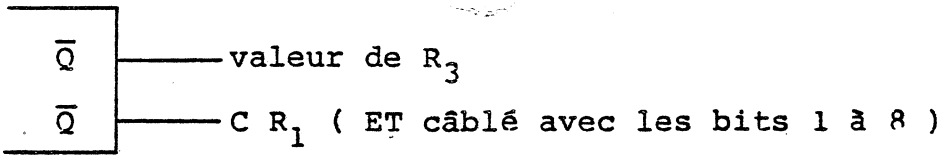
bit 0 :



bits 1 à 8 :

On a le même schéma avec :

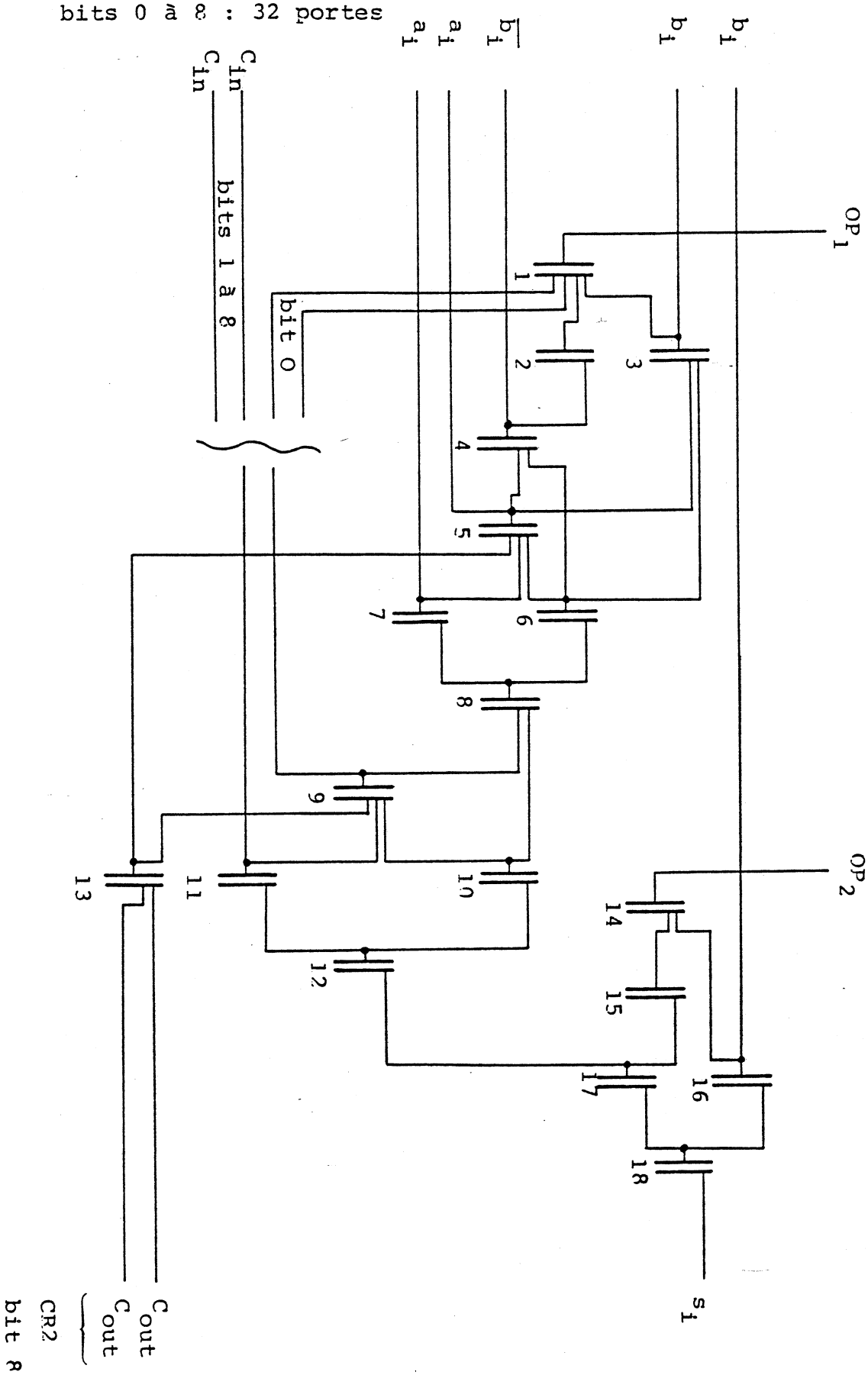
- suppression des portes 9 et 14 ("1" arithmétique)
- une sortie supplémentaire



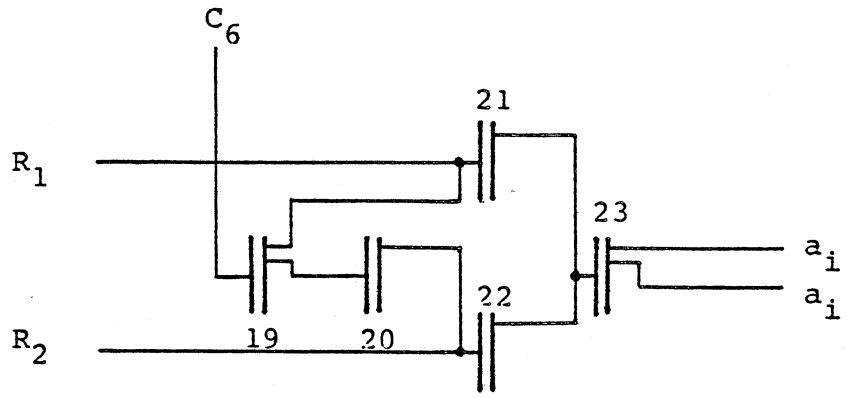
Particularités : Pour les bits 0 et 1, l'entrée "R3 mult 4" est à 0, on supprime la porte 13 et la connexion "R3 mult 4".
Pour les bits 7 et 8, l'entrée "R3 div 4" est à 0, on supprime la porte 12 et la connexion "R3 div 4".

5) Module M5 : U.A.L.

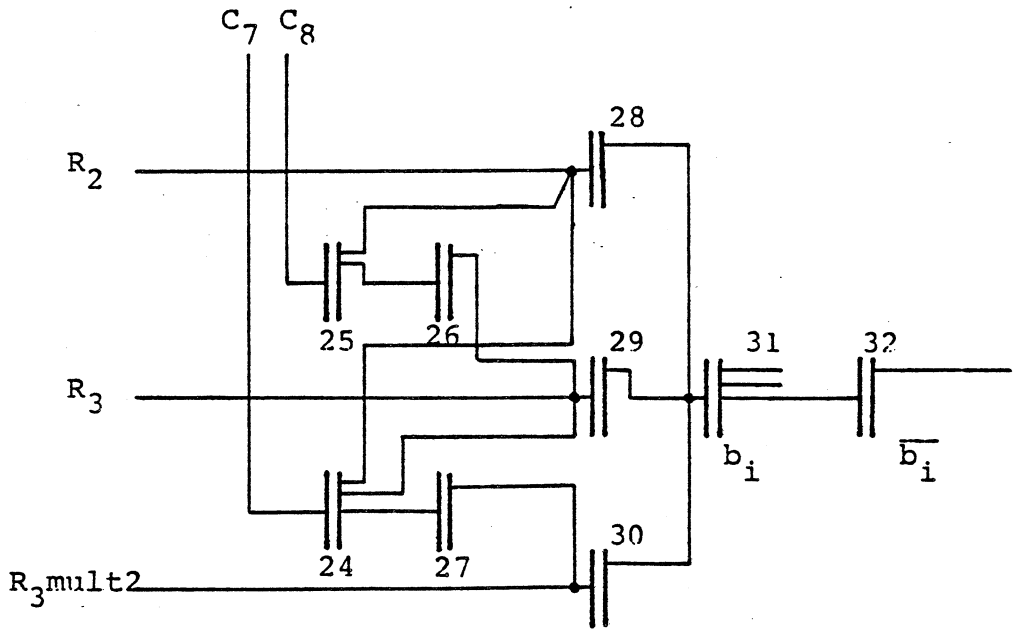
bits 0 à 8 : 32 portes



Accès entrée A



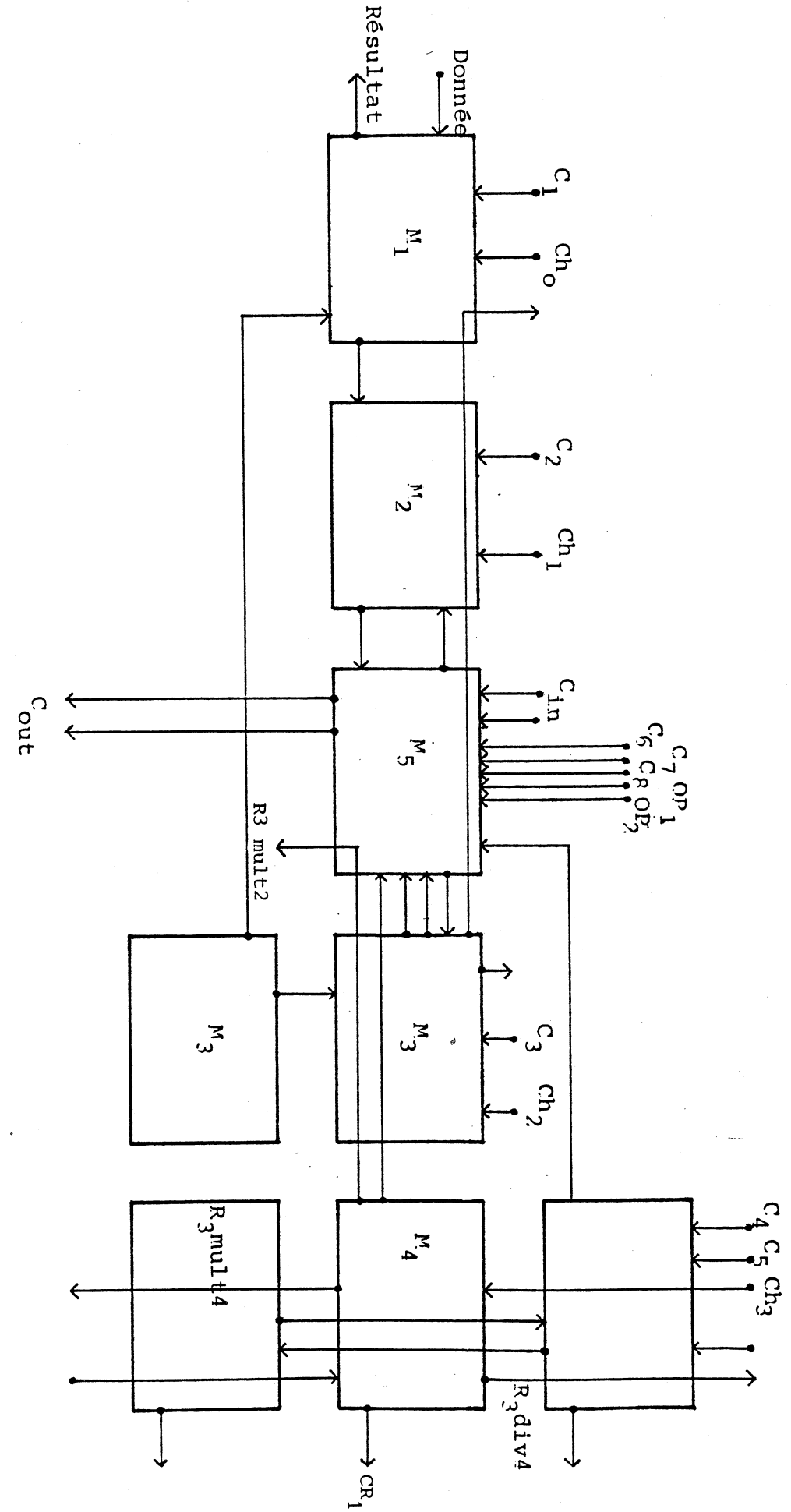
Accès entrée B



Particularité :

Pour le bit 8, le module M_2 n'existe pas, on supprime la connexion "R_1".

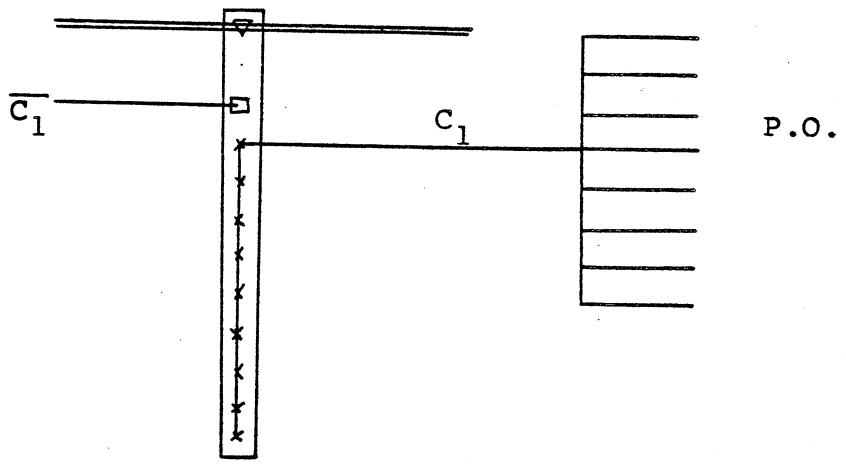
ii) Connexions entre les modules pour un bit de la P.O.



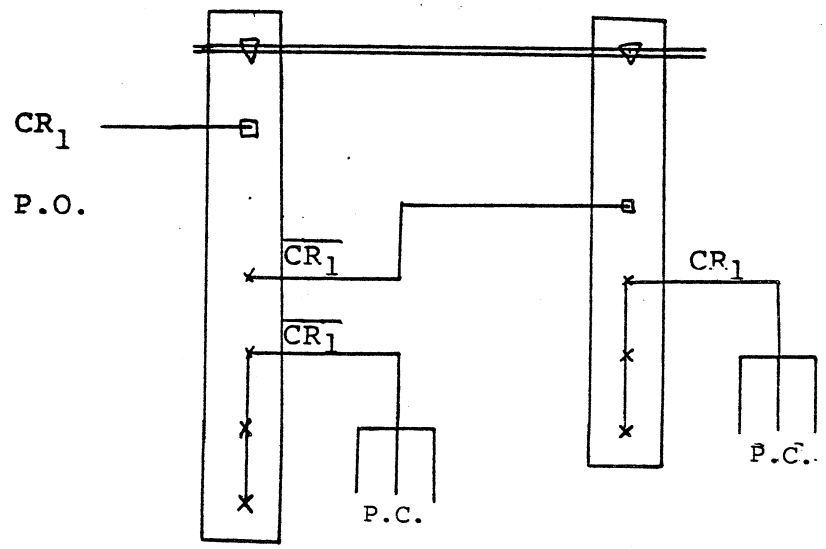
7.2.3. Partie contrôle

A - Interface Partie contrôle - Partie opérative : Partie amplificatrice

i) En symbolique MDMOS, un drain n'attaque qu'une seule grille. Les commandes émises par la partie contrôle attaquent les modules répétitifs de la partie opérative, donc un certain nombre de portes. Ces commandes seront amplifiées. Les portes amplificatrices sont réalisées par la mise en série de drains, elles sont inverseuses. Les commandes C1, C2, ch0, ch1 attaquent 8 portes. Les autres commandes attaquent 9 portes. On aura pour C1, par exemple :

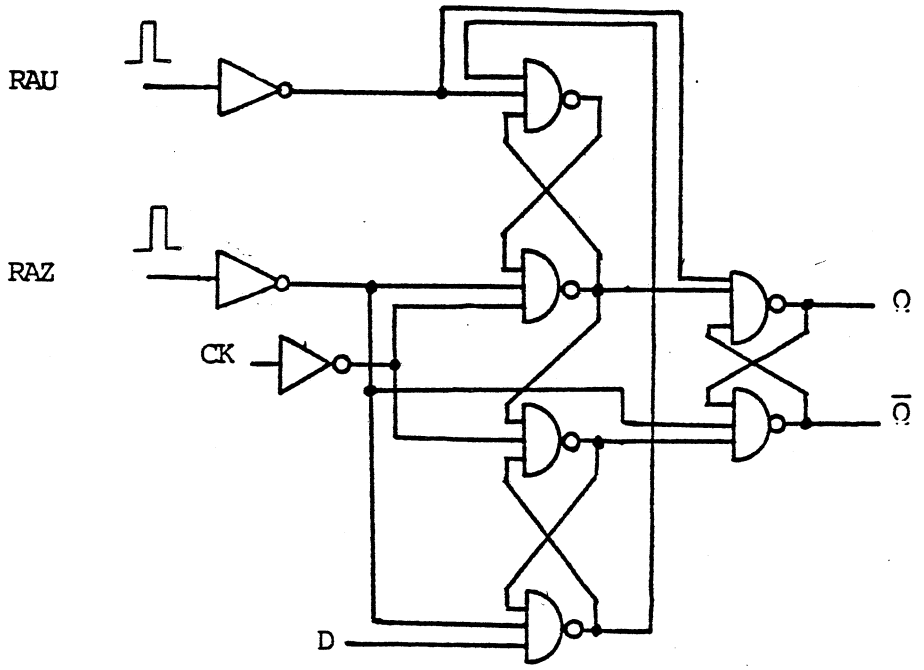


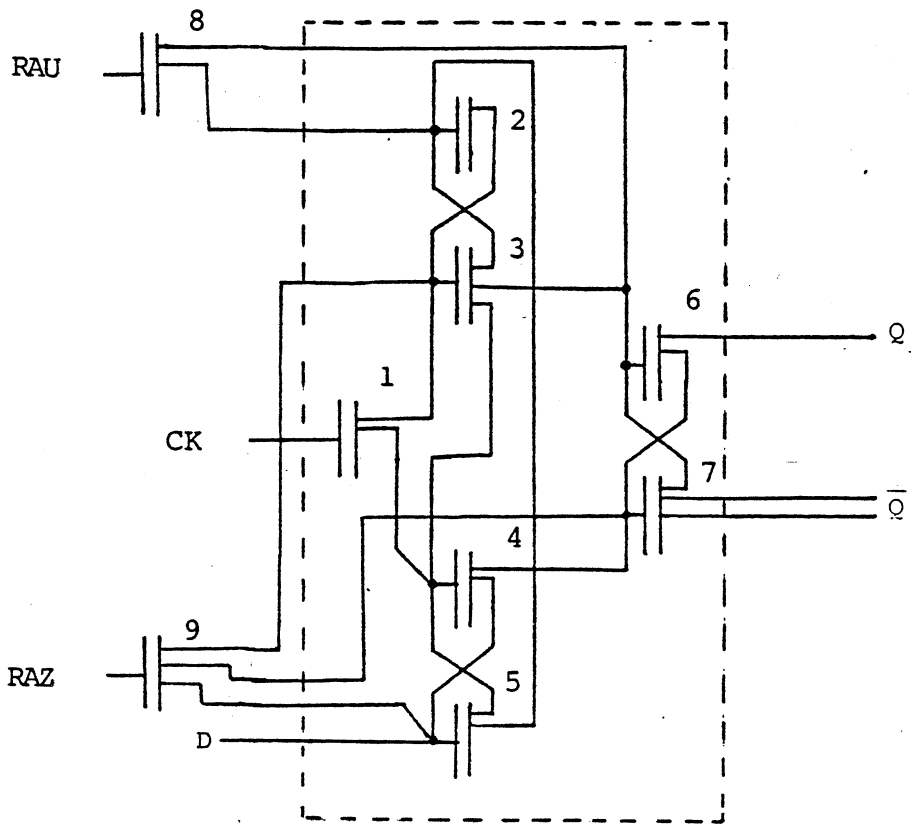
ii) De la même façon, les compte-rendus émis par la partie opérative attaquent plusieurs portes de la partie contrôle : CR1, $\overline{CR1}$ et $\overline{CR2}$ attaquent 3 portes ; CR2 2 portes (voir la figure 32). On aura pour CR1, par exemple :



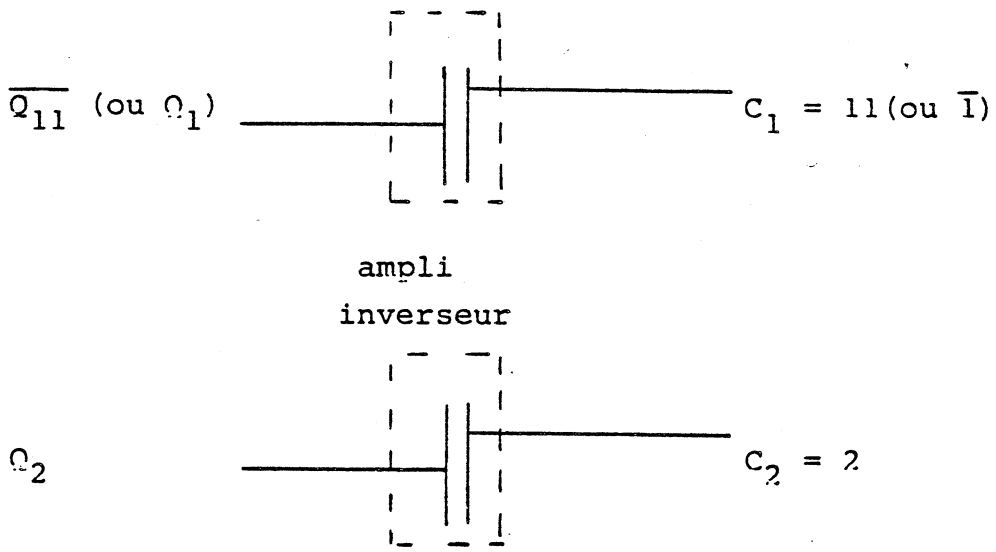
B - Partie contrôle

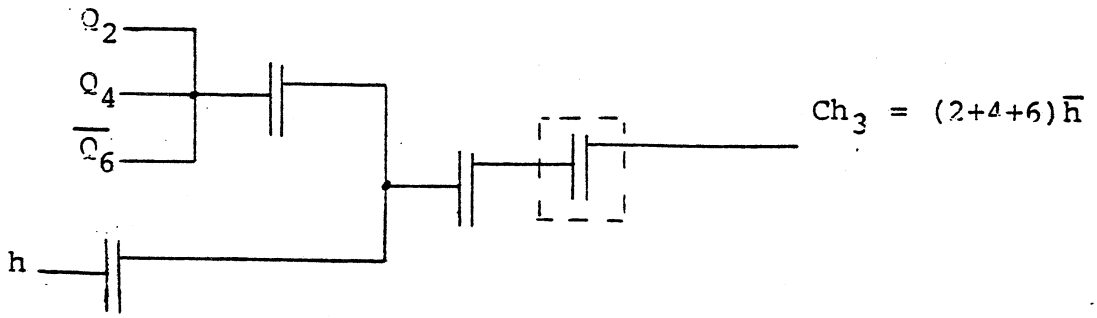
i) Le schéma de la partie contrôle avec une synthèse en NAND (figure 32) donne directement le schéma en MDMOS. on utilise le schéma logique de la bascule à front descendant donné en 7.1.1.b, avec en plus la remise à zéro et la remise à un. On obtient les schémas suivants :





ii) La commande sera inversée avec les portes amplificatrices et suivant l'activation de la bascule à 1 ou à 0, on sortira respectivement sur $\overline{Q}$ ou Q. Les commandes seront générées de la façon suivante :

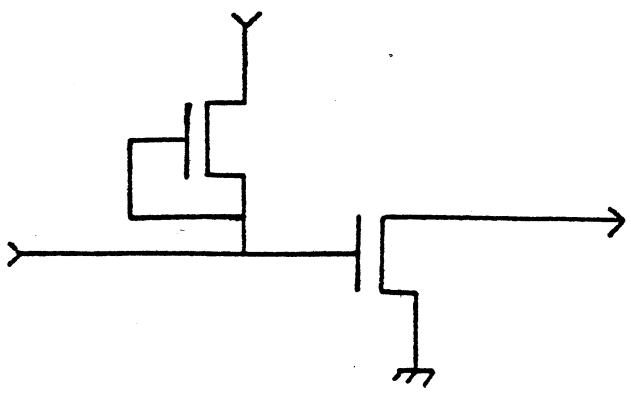




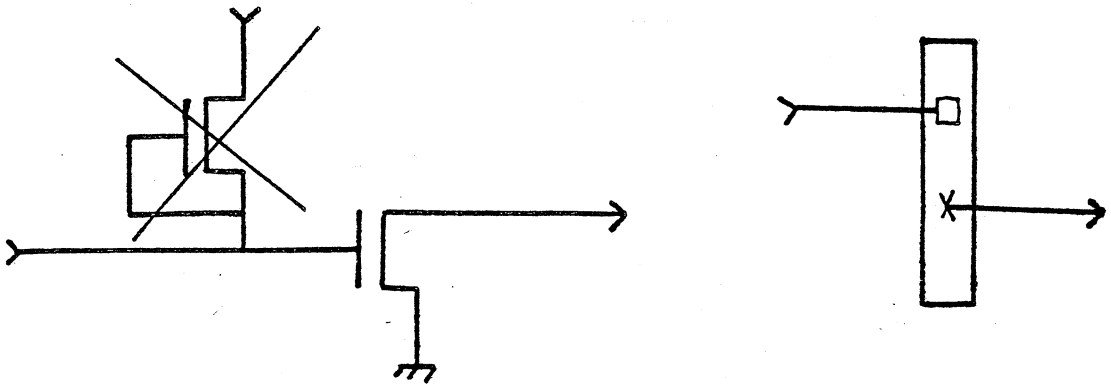
7.2.4. Compatibilité MDMOS-NMOS

Ce circuit est réalisé en technologie NMOS, avec le symbolisme fnctionnel MDMOS. Il faut tenir compte de la compatibilité MDMOS-NMOS pour relier les plots d'entrée-sortie au reste du circuit.

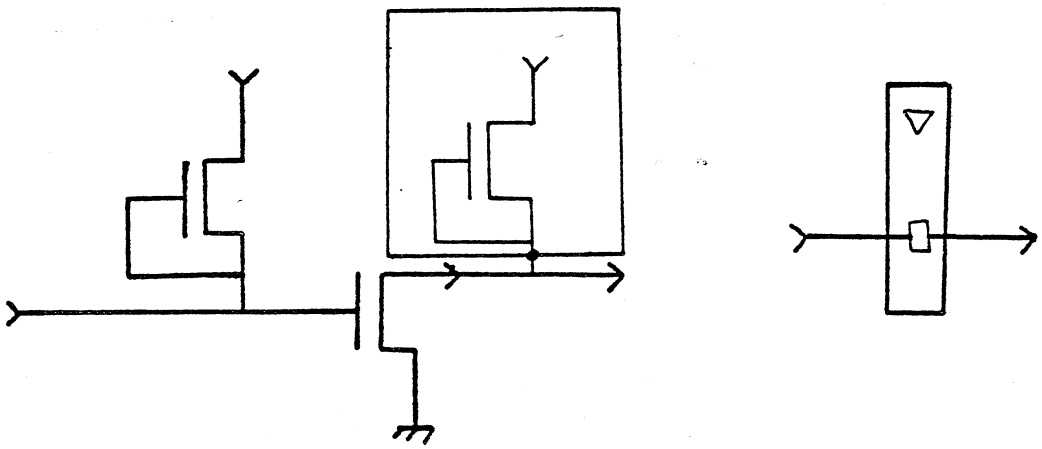
La porte MDMOS possède le transistor de charge de la porte précédente :



On supprime le transistor de charge sur les entrées. Tous les plots d'entrée sont suivis d'une porte sans injecteur :



On ajoute un transistor de charge seul sur les sorties. Tous les plots de sortie sont précédés d'une porte possédant uniquement un transistor de charge.



i) Le circuit a 11 plots d'entrée (cf. la figure 10 du § IV) : les signaux Initialisation, Début, Horloge arrivant sur la partie contrôle ; les 8 bits de données et le signal Début arrivant sur la partie opérative.

Un plot d'entrée ne doit pas attaquer un "ET câblé" (NAND), il attaque un ou plusieurs inverseurs sans transistor de charge (cf. § 7.2.1).

ii) Le circuit a 5 plots de sortie : le signal Fin venant de la partie contrôle et les 4 bits de résultats venant de la partie opérative.

Chaque plot de sortie aura une sortie basse impédance non inverseuse. On ajoute un transistor de charge seul avant chacun de ces plots.

VII - 3. Schémas en portes symboliques MDMOS dimensionnées : specification de niveau 6

L'intérêt du symbolisme fonctionnel MDMOS réside dans la largeur constante de la porte pour une gamme de courants. Dans une porte MDMOS, les transistors signaux sont de taille fixe, seules les dimensions du transistor de charge varient suivant la rapidité désirée (d'où le courant).

A - Dimension des portes MDMOS utilisées (pour CMP)

Les dimensions possibles pour la porte MDMOS (cf. le dessin des masques (figure 33)), sont :

transistor signal : longueur du canal $Ls = 2\lambda$; largeur du canal $Ws = 8\lambda$

transistor de charge : largeur du canal $Wc = 2\lambda$; longueur du canal $2\lambda \leqslant Lc \leqslant 8\lambda$

La longueur du canal du transistor de charge peut prendre 7 valeurs :

$Lc = 2\lambda, 3\lambda, 4\lambda, 5\lambda, 6\lambda, 7\lambda$ et 8λ

Remarques : 2λ est la dimension minimale permise par la technologie.

Si le dernier site de la porte est un drain, ce transistor signal a la largeur de canal : $Ws = 16\lambda$ (il est donc plus rapide).

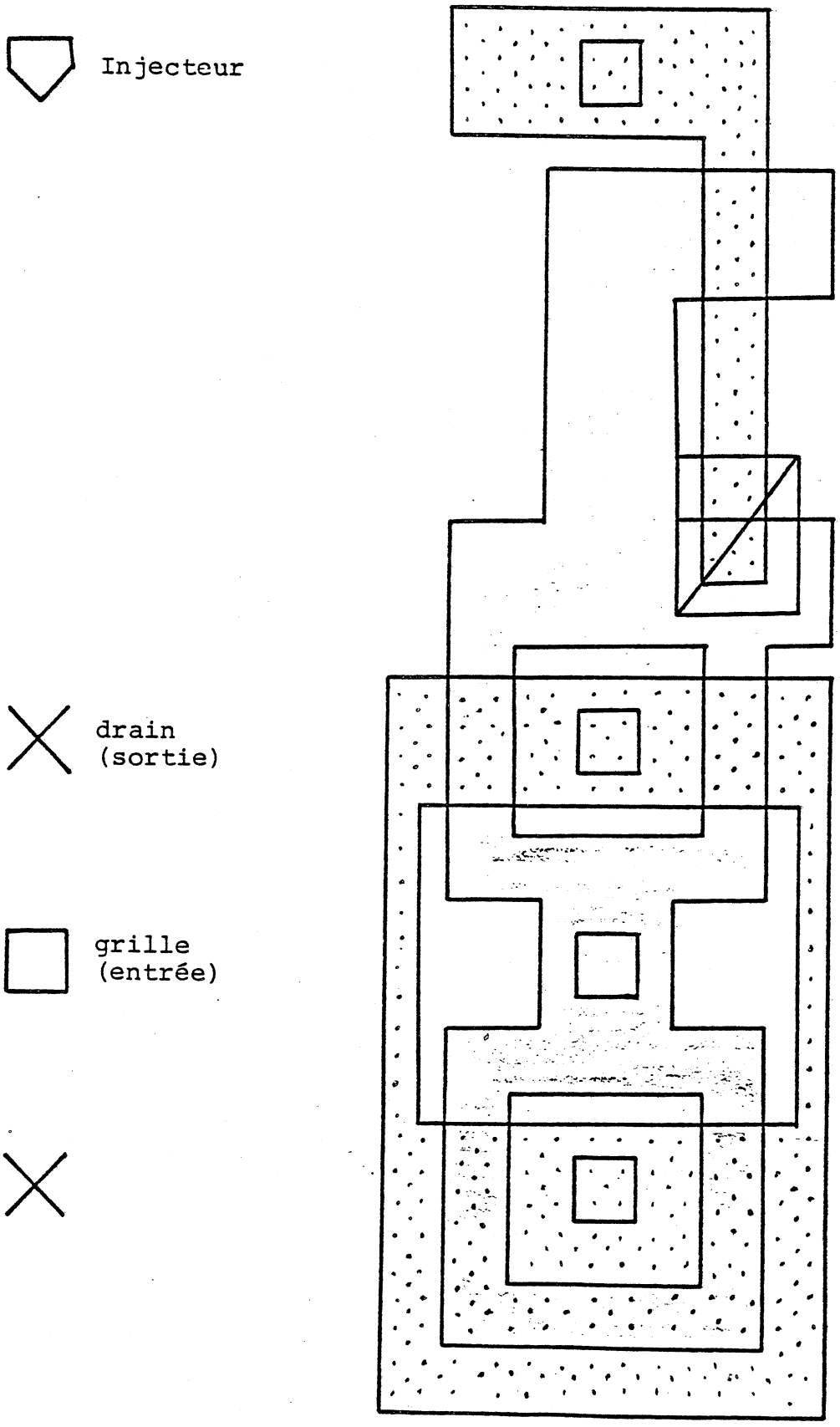


Figure 33 : dessin des masques de la porte MDMOS

B - Portes amplificatrices : mise en série de drains

Lorsqu'une sortie d'une porte P attaque les grilles de plusieurs portes Pi, la taille du transistor signal étant fixé, le niveau "0" en sortie de P et la rapidité sont dégradés.

C'est pourquoi on utilise autant de drains en sortie de P que de grilles des portes Pi (cf. 7.2.3).

C - Dimensionnement du transistor de charge

On fait les calculs avec les paramètres électriques de CMP, $\lambda = 3\mu$ et le pas de grille = 7 (CMP 82)

Le calcul du temps de propagation donne pour une porte :

$t(ns) \approx 100\ C/\gamma c\ (pF)$

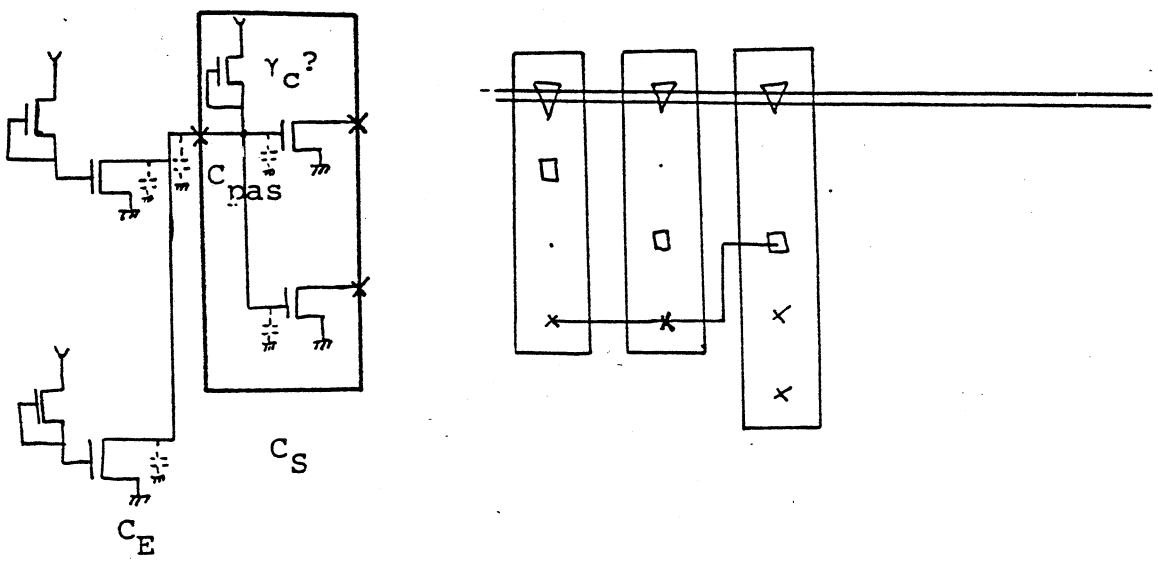
avec C : capacité à calculer

$$C = \frac{wc}{Lc}$$

wc : largeur du canal du transistor de charge.

Lc : longueur du canal du tranistor de charge.

On calcule la capacité C :



La capacité C est égale à :

$$C = Cpas + \text{Œ} + CS$$

avec :

Cpas est la capacité de la connexion attaquant la porte considérée ;

Cpas = Calu + Cdiff où Calu est la capacité de la portion aluminium et Cdiff la capacité de la portion diffusion éventuelle.

Œ : capacité des drains attaquant la porte considérée
CS : capacité de grille des transistors signaux de la porte considérée.

Le calcul de Œ et CS est facile puisque les transistors signaux sont de taille fixe.

Les paramètres de CMP nous permettent de calculer facilement :

$$Calu/Npas = 10^{-2} \text{ pF}$$

$$\text{Œ}/NE = 3.10^{-2} \text{ pF}$$

$$\text{Œ}/NS = 9.10^{-2} \text{ pF}$$

Il suffit donc de calculer :

Npas : le nombre de pas de grille de la connexion attaquant la porte considérée
NE : le nombre de drains attaquant la porte considérée

NS : le nombre de drains de la porte considérée

Pour l'extracteur, on calcule une valeur moyenne pour la capacité C avec :

Npas = 10 NE = NS = 2

d'où Cpas = 10.10^{-2} pF

CE = 6.10^{-2} pF

CS = 18.10^{-2} pF

et C = 34.10^{-2} pF

On a t(ns) # 100 C/γc (pF). Ayant calculé la capacité C, on choisit le temps de propagation t = 70ns, on trouve les dimension du transistor de charge :

On prendra γc = 1/2 pour toutes les portes, on a Wc = 2λ d'où Lc = 4λ. On sait que le temps de propagation t croît si la capacité C augmente, mais on ne cherche pas à avoir un circuit rapide.

VII - 4. Implantation du circuit

Le pré-implantation de la partie opérative par blocs répétitifs a été faite en 7.1.1 au moment de sa conception logique. Ceci permet de réduire la conception de ces modules à un seul bit (cf. Figure 30).

Pour minimiser les connexions entre la partie opérative et la partie contrôle (commandes et compte-rendus), on implantera la partie contrôle "sous" la partie opérative. Cette partie contrôle sera constitué de cellules MDMOS a une ou deux lignes, de même largeur que celles de la partie opérative et pourra utiliser la place vide des blocs manquants M1 et M2 pour le dernier bit (bit 8).

Une tranche de la partie opérative (un bit) est constituée de 80 portes. La partie contrôle (partie amplificatrice comprise) comprend 157 portes (cf. l'évaluation de surface ci-après). On aura donc deux cellules pour la partie contrôle : une cellule à deux lignes pour le séquenceur, une deuxième cellule à une ou deux lignes à laquelle on peut ajouter la place vide en bas de la partie opérative pour la partie amplificatrice comprenant le câblage des commandes à envoyer et l'amplification de celles-ci et des compte-rendus.

Le circuit (partie opérative, partie contrôle, connexions entre celles-ci) est totalement défini en portes symboliques MDMOS dimensionnées.

On le décrit en SYMBOLE : logiciel de conception de circuits intégrés en symbolique MDMOS avec génération automatique de la description des masques en LUCIE (à l'aide de TRADUC), on obtient alors le dessin au micron du circuit, il reste à placer les plots d'entrée-sortie et leurs connexions au reste du circuit.

Le ciruit a 18 plots d'entrée-sortie :

- le plot de masse VSS
- le plot d'alimentation VDD
- 11 plot d'entrée :
  - . Initialisation, Début, Horloge
  - . 8 bits de données
- 5 plots de sortie :
  - . Fin
  - . 4 bits de résultats.

On dispose ces plots suivant la surface de ceux-ci, la longueur et la largeur du ciruit et en minimisant les connexions entre les plots et le circuit.

Un exemple d'implantation de ce circuit est donné dans les pages suivantes : pré-implantation (figure 34), schéma en portes symboliques MDMOS de la partie opérative et de la partie contrôle (figure 35), dessin au micron du circuit complet (figure 36). Cet exemple a été réalisé par des étudiants avec les règles simplifiées du service CMP, l'évaluation de surface effectuée pour une technologie 6µ n'est donc pas représentative de cet exemple. De plus, ce circuit n'a pas été optimisé (beaucoup de place perdue).

Evaluation de surface

L'évaluation de surface est faite avec une technologie NMOS 6µ à déplétion en charge.

Dans cet exemple, tous les éléments du circuit sont réalisés en logique aléatoire avec une implantaion dans un symbolisme fonctionnel : le MDMOS. La

surface de ceux-ci est alors évaluée en prenant la densité d'intégration calculée comme le nombre de transistors par mm2 : D = 800 tr/mm2.

L'estimation de la surface totale de la puce, avec une précision de ± 20 %, se fait comme suit :

S1 surface active (transistors) = surface de la logique aléatoire
S2 surface occupée par les interconexions = 0,8 S1
S3 surface occupée par les plots d'entrées/sorties = N/10
avec N : nombre de plots d'entrée/sortie
S surface de travail = S1 + S2
SPO = S1PO + S2PO
SPC = S1PC + S2PC
S surface totale de la puce = S1 + S2 + S3 = Stravail + S3

On calcule le nombre de portes MDMOS du circuit (cf. VII-2.). Les portes amplificatrices compteront pour deux portes. On prend une moyenne de trois transistors par porte.

Pour la partie opérative, on compte le nombre de portes maximal pour un bit, on se sert du découpage en modules fait en VII-1.1. sur la figure 30.

Modules	M1	M2	M3	M4	M5
Nombre de portes MDMOS pour un bit	13	11	11	13	32
Nombre de bits	8	8	9	9	9

Une tranche de partie opérative (un bit) est constituée de 80 portes, la partie opérative : 696, donc 2088 transistors d'où :

S1 P.O = 2088/800 = 2,61 mm2

S2 P.O = 0,8 x 2,61 = 2,09 mm2

S P.O = 4,7 mm2

La partie contrôle comprend 157 portes (la partie amplificatrice comprise : 32 portes), donc 471 transistors, d'où :

S1 P.C = 471/800 = 0,59 mm2

S2 P.C = 0,8 x 0,59 = 0,47 mm2

S P.C = 1,06 mm2

On obtient :

Stravail = S P.O + S P.C = 5,76 mm2

Le circuit a 18 plots d'entrée/sortie :

S3 = N/10 = 18/10 = 1,8 mm2

La surface totale estimée de la puce sera :

Stotale = Stravail + S3 = 7,56 mm2

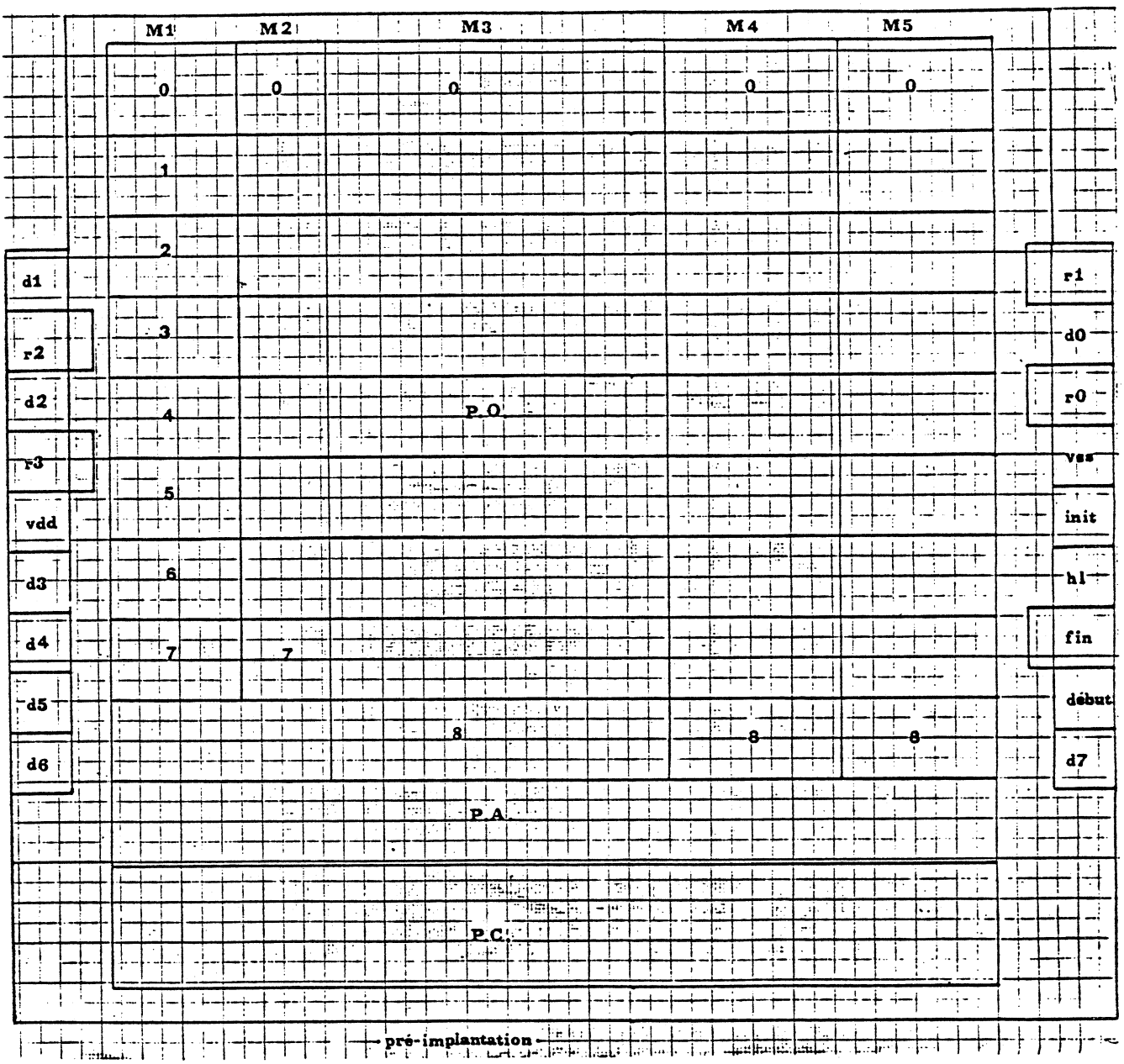

M1
M2
M3
M4
M5
0
1
2
3
4
5
6
7
8
d1
r2
d2
r3
vdd
d3
d4
d5
d6
P.O
P.A
P.C
pré-implantation
r1
d0
r0
vss
init
h1
fin
début
d7

Figure 34

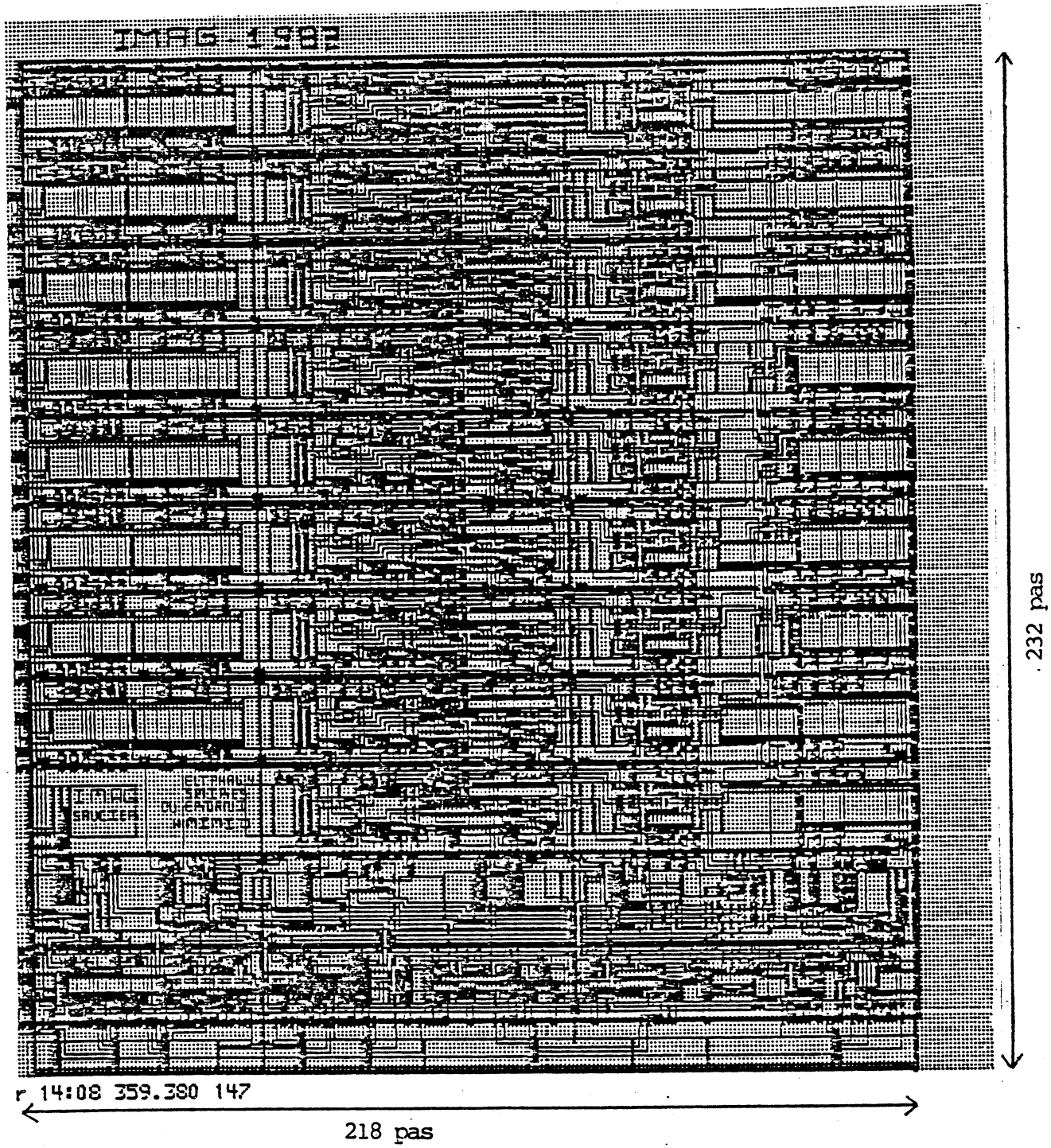


Figure 35 : Schéma en portes symboliques MDMOS du circuit (sans les plots d'entrée/sortie) 1 pas = 7λ ; λ = 3μ

$$S_{travail} = 218 \times 232 \times 7^2 \times 3^2 \, \mu^2 = 22{,}3 \, mm^2$$

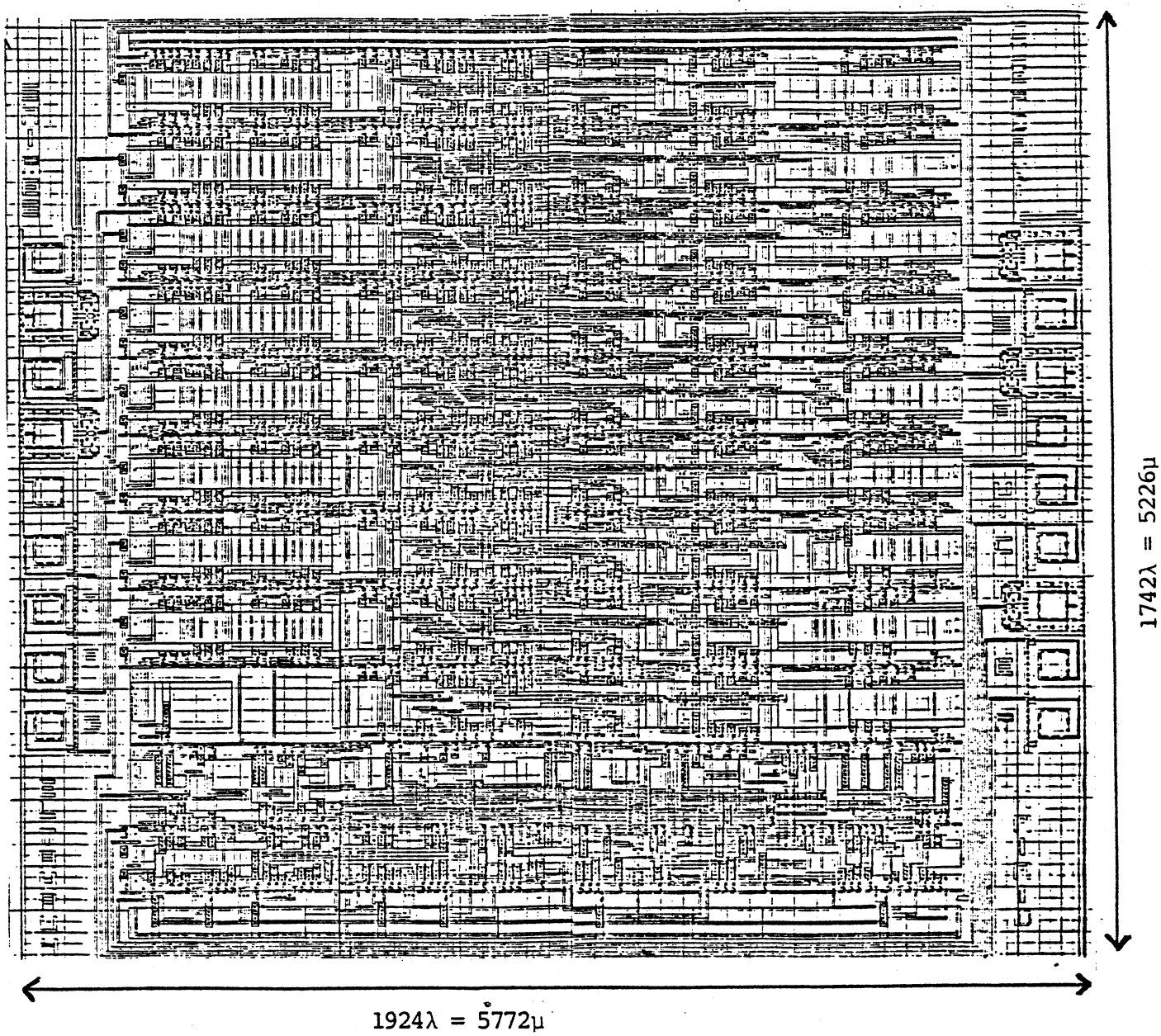


Figure 36 : Dessin au micron du circuit complet

$S_{totale} = 30,17\ mm^2$

$\lambda = 3\mu$

CONCLUSION

Nous avons souhaité mettre en évidence dans cette thèse l'intérêt des méthodes de conception très systématiques pour les circuits intégrés de haute complexité en recherchant une synthèse entre les aspects architecturaux, logiques, topologiques et électriques. Dans un premier chapitre, nous avons montré que la préparation de cellules élémentaires (porte complexe MOS) doit faire un compromis entre le compactage de la cellule et les considérations de connectique. Des réalisations systématiques autorisant des connexions sur les deux faces d'une cellule ont été proposées. Le deuxième chapitre a mis l'accent sur une implantation en Silicium d'arbres de décision binaire. Il s'agit d'une transcription d'un modèle de haut niveau directement en circuit intégré. Enfin le troisième chapitre illustre l'ensemble d'une démarche de conception d'un circuit complexe (circuit à contrôleur). Dans ce dernier chapitre, les modèles à différents niveaux, les passages sûrs d'un niveau à un autre, l'implantation systématique sont exposés sur l'exemple concret d'un circuit. L'ensemble des méthodes et outils proposés vise la conception sûre et rapide de circuit complexe.

BIBLIOGRAPHIE

(AKE 78) AKERS S.B.,
"Binary decision diagrams"
IEEE Transactions on Computers, Vol.C-27, n°6, pp.509-516, juin 1978.

(AKE 79) AKERS S.B.,
"Communication interne"
General Electric Company, Electronics Laboratory Syracuse, New York (USA), 1979.

(AMB 81) AMBLARD P., SAUCIER G.,
"Conception de séquenceurs"
L'onde Electrique, Mars 1981.

(BEL 79) BELLON C., SAUCIER G.,
"Structure des calculateurs"
Polycopié, Université de Grenoble, 1979.

(BEL 81) BELLON C., SAUCIER G.,
"Conception d'un séquenceur de microprocesseurs"
RAIRO Automatique, vol.15, n°2, 1981.

(BLA 79) BLAKESLEE T.R.,
"Digital design with standard MSI & LSI Design techniques for the microcomputer age"
John Wiley & Sons, 2nd Edition, 1979.

(CAL 62) CALDWELL S.H.,
"Switching circuits and logical design"
John Wiley & Sons, Inc. 1962.

(CER 79) CERNY E., MANGE D., SANCHEZ E.,
"Synthesi of minimal binary decision trees"
IEEE Transactions on Computers, vol.C-28, n°7, juillet 1979.

(CMP 82) "Notice du CMP GCIS"
Octobre 1982.

(CMP 83) "Notice du CMP CMOS"
Octobre 1983.

(CRO 82) CROUCH K.W., SISKIND J.M., SOUTHARD J.R.,
"Generating Custom High Performance VLSI Designs from succinct algorithmic descriptions"
Conference on advanced research in VLSI, 25-27 janvier 1982, pp.28-39.

(DAV 83) DAVID R., SILVA M.,
"Binary Decision graphs for implementation of boolean functions"
Rapport interne, Laboratoire d'Automatique, Ecole Nationale Supérieure d'Ingénieurs Electriciens de Grenoble, à paraître.

(DUN 81) DUNLOP A.E.,
"Slim : the translation of Symbolic Layouts into Mask Data"
Journal of Digital System, vol.V, n°4, 1981.

(EIC 82) EICHENBERGER P., MATHEWS R., NEWKIRK J.,
"A target language for silicon compilers"
COMPCON SPRING, Février 1982.

(EL-Z 77) EL-ZIQ Y.M.,
"Logic design automation of MOS combinational networks with fan-in, fan-out constraints"
14th Design Automation Conference, Juin 1977, pp 240-249.

(GON 83) GONCALVES N.F., DE MAN H.J.,
"NORA : a racefree dynamic CMOS technique for pipelined logic structures"
IEEE Journal of Solid State Circuits, vol.SC-18, n°3, juin 1983.

(HAN 72) HANAN M., KURTZBERG J.M.,
"Placement techniques"
Design Automation of Digital Systems, Edited by M.A. Brever, vol.1, Prentice Hall, 1972, pp.211-282.

(KRA 82) KRAMSECK R.H., LEE C.M., LAW H.S.,
"High speed compact circuits with CMOS"
IEEE J. Solid State Circuits, vol.SC-17, pp.614-619, juin 1982.

(KUB 83) KUBITZ W.J., LUHUKAY J.F.P., WONG F.L.,
"An automatic NMOS cell synthesis system"
ICCAD, Santa Clara, septembre 1983.

(KUN 68) KUNTZMANN J.,
"Algèbre de Boole"
Editions Dunod, Paris 1968.

(LEB 80) LEBLOND A., SERRERO G., VERDILLON A.,
"Automatic layout of symbolic MDMOS circuits"
1st ICC Conference, 1-3 octobre 1980, Port Chester, New York pp.772-776.

(LIU 75) LIU T.K.,
"Synthesis algorithms for 2 level MOS networks"
IEEE Transactions on computers,vol.C 24, n°1, janvier 1975.

(LOT 79) LOFTI Z., TOSSERA A.,
"Utilisation de multiplexeurs pour la réalisation de fonctions logiques"
L'onde Electrique 1979, vol.59, n°11.

(MAJ 78) MAJOS J., LARDY J.C.,
"The multidrain MOS transistor"
4th ESSIRC European Solid State Circuit Conference, 18-21 sept. 1978., Amsterdam, pp.133-135.

(MAL 81) MALLADI R., SERRERO G.,
"OASIS : An automatic design tool for symbolic layout of circuits"
7th ESSIRC, Septembre 1981, Freiburg (RFA), pp.31-33.

(MAL 82) MALLADI R.,
"Conception électrique et implantation de circuits intégrés"
Thèse de Docteur Ingénieur, INP Grenoble, Janvier 1982.

(MAN 78) MANGE D.,
"Arbres de décision pour systèmes logiques câblés ou programmés"
Bulletin ASE/UCS, t.69, n°22, pp.1238-1243, 1978.

(MAN 80) MANN J.H., SZANTO L.,
"Simulation of MOS IC's specified by topography of their masks"
4th European Conference on Electrotechnics, 24-28 mars 1980, pp.119-121.

(MAT 83) MATOS J.S., OLDFIELD J.V.,
"Binary decision diagrams from abstract representations to physical implementations"
20th Design Automation Conference, juin 1983, pp.567-570.

(MAT) MATOS J.S.,
"Binary decision diagram as a tool for logic implementation"
PhD, University of Syracuse,en préparation.

(MEA 80) MEAD C., CONWAY L.,
"Introduction to VLSI System"
Addison-Wesley, 1980.

(MOR 82) MORET B.M.E.,
"Decision trees and diagrams"
Computing surveys, vol.14, n°4, décembre 1982.

(MUR 82) MUROGA S.,
"VLSI system design"
John Wiley & Sons, 1982.

(PER 67) PERRIN J.P., DENOUETTE M., DACLIN E.,
"Systèmes combinatoires, Introduction aux systèmes séquentiels"
Editions Dunod, Paris 1967.

(RED 72) REDDY R.M., CHAN S.P.,
"Transient Analysis of Complementary MOS IC inverter"
Electronics 1973, vol. 34, n°5, pp.701-708.

(SAN 78) SANCHEZ E., MANGE D.,
"Synthèse des fonctions logiques avec des multiplexeurs"
Digital Processes, n°1, vol.4, 1978.

(SAU 82) SAUCIER G.,
"Conception descendante et choix de l'architecture des circuits à très haute intégration"
RAIRO Automatique/Systems Analysis and Control, vol.16, n°3, 1982, pp.233-257.

(S.E. 81) S.E. Director et all,
"A design methodology and computer aids for digital VLSI systems"
IEEE Trans. on Circuits and Systems, July 1981.

(SER 82) SERRERO G.,
"Implantation symbolique automatisée de circuits intégrés"
Thèse de Docteur Ingénieurs, INP de Grenoble, mars 1982.

(THU 82) THUAU-MAUPETIT G.,
"Conception descendante d'un circuit : étude de cas (extracteur de racine carrée)"
Rapport interne n°302, laboratoire IMAG, mai 1982.

(THU 83) THUAU G., SAUCIER G.,
"Logical and topological design in MOS technology"
European Conference on "Electronic Design Automation", Londres, à paraître mars 1984.

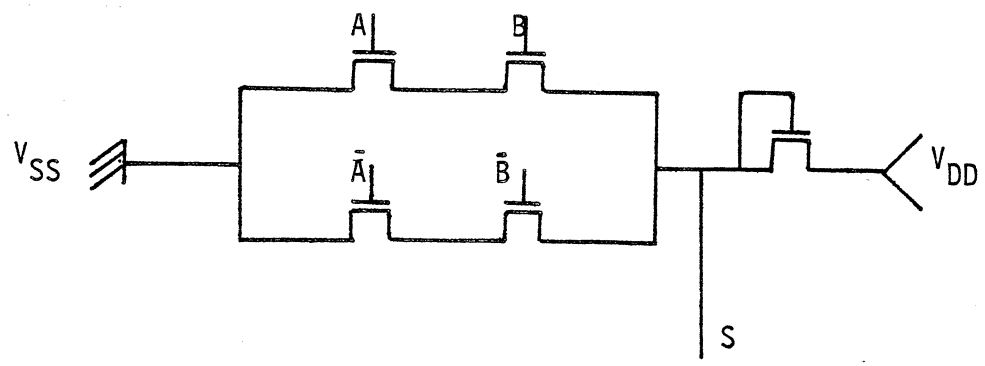
(UEH 81) UEHARA T., VAN CLEEMPUT W.M.,
"Optimal layout of CMOS functional arrays"
IEEE Transactions on computers, vol.C 30, n°5, mai 1981.

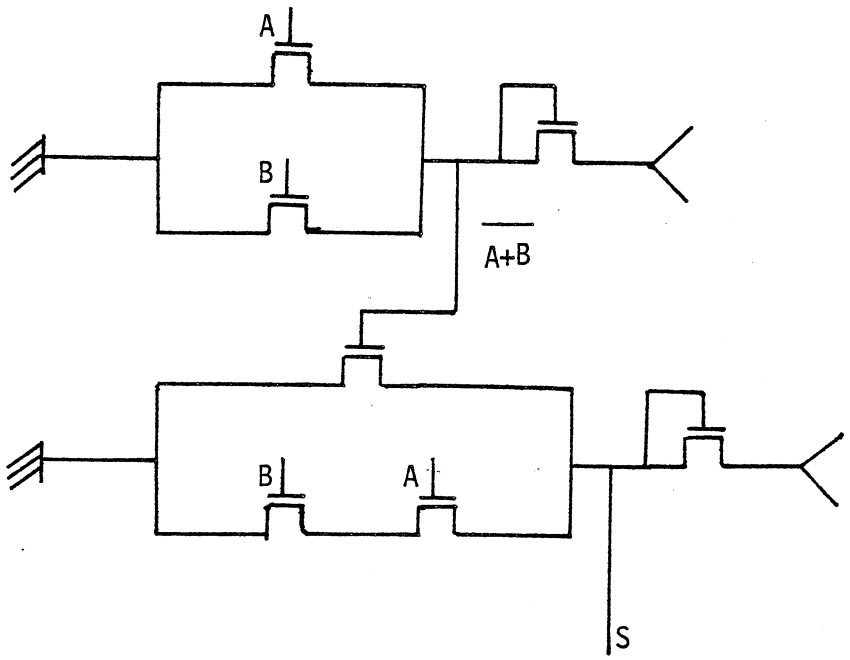
(WIL 74) WILSON D.C., SMITH II R.J.,
"An experimental Comparison of Force Directed Placement techniques"
Design Automation Workshop, 1974, pp.194-199.

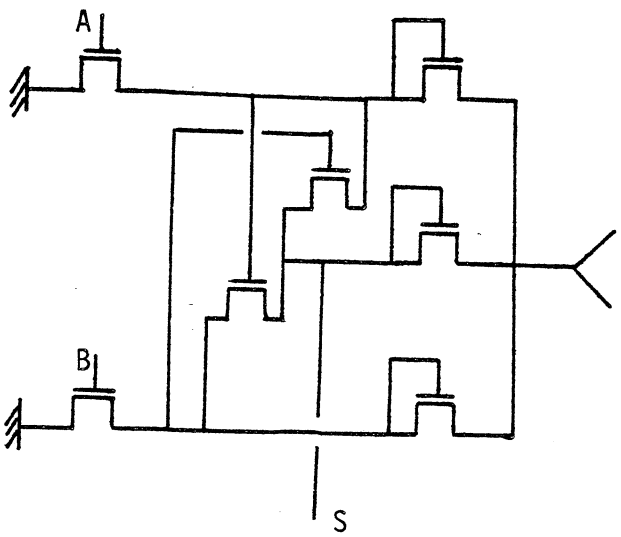
Annexe 1 : Portes complexes NMOS couramment utilisées

Portes complexes NMOS à 2 variables

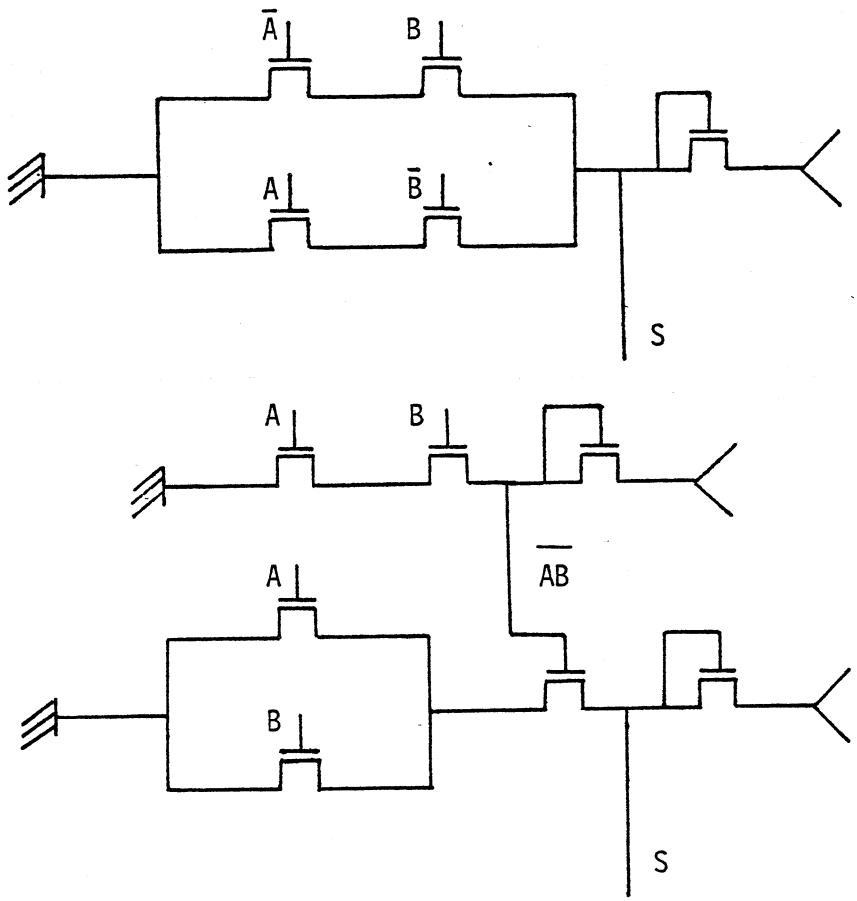
- OU exclusif, disjonction $F = \bar{A}B + A\bar{B} = \overline{AB + \bar{A}\bar{B}} = \overline{\overline{A + B} + AB}$





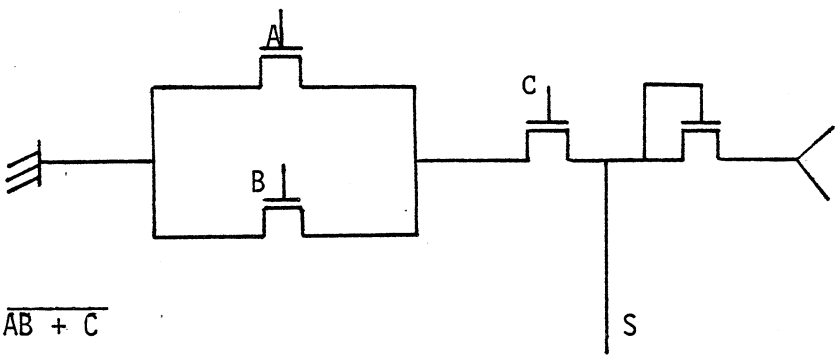


- Conjonction $F = AB + \bar{A}\bar{B} = \overline{A\bar{B} + \bar{A}B} = \overline{\overline{AB}(A + B)}$

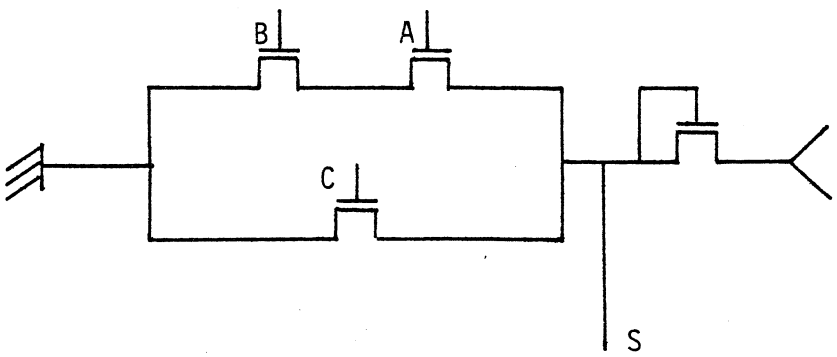


Portes complexes NMOS à 3 variables

- $F = \overline{(A + B)C}$

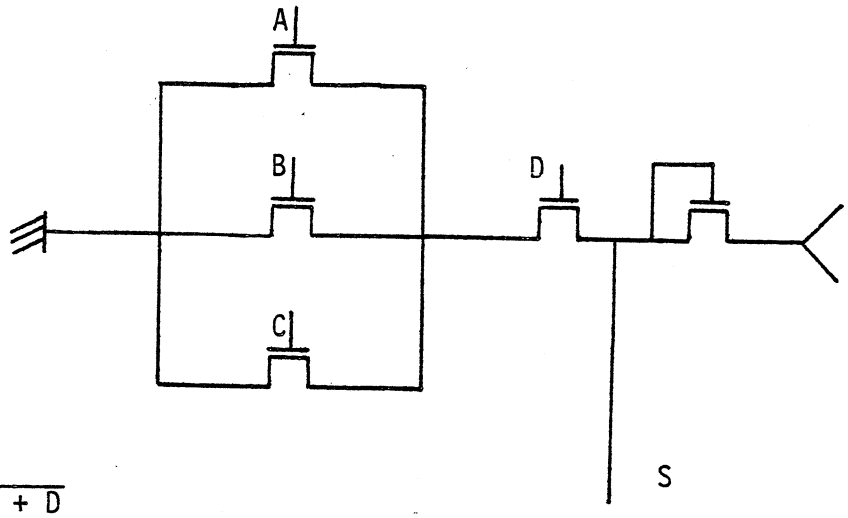


- $F = \overline{AB + C}$

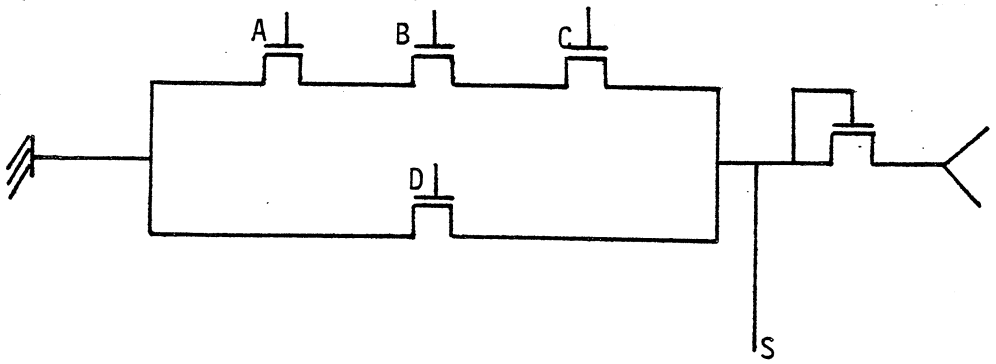


Portes complexes NMOS à 4 variables

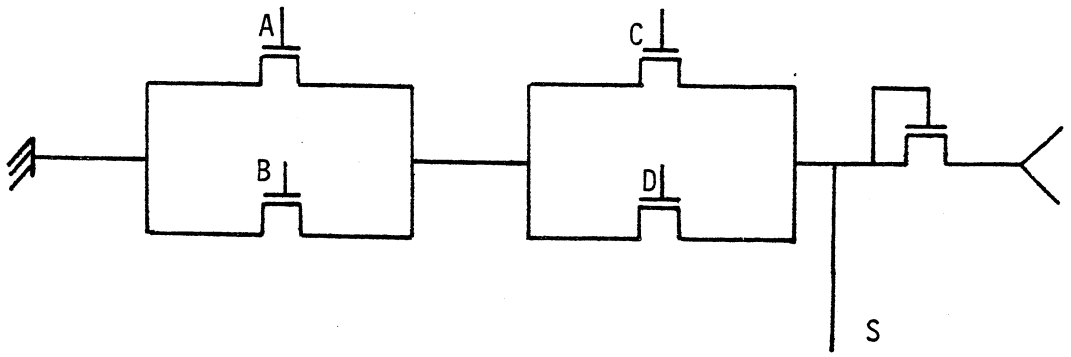
- $F = \overline{(A + B + C)D}$



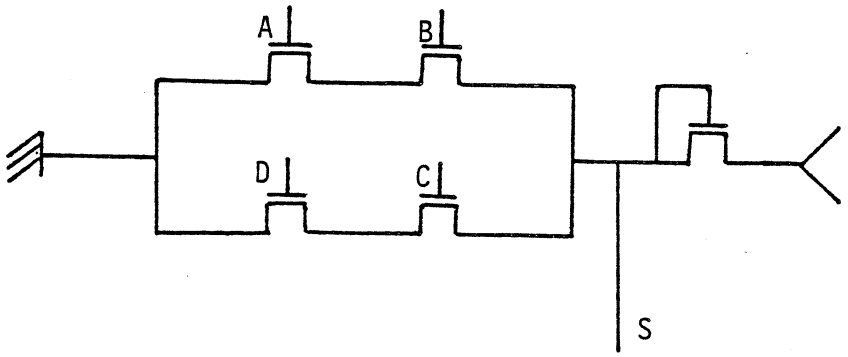
- $F = \overline{ABC + D}$



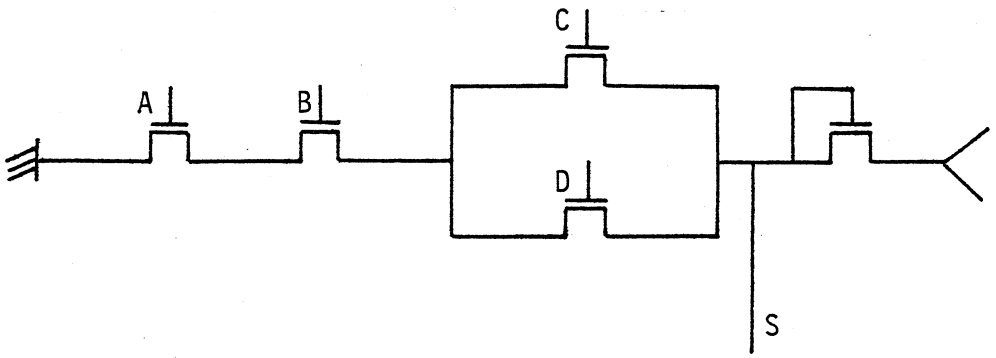
- $F = \overline{(A + B)(C + D)}$



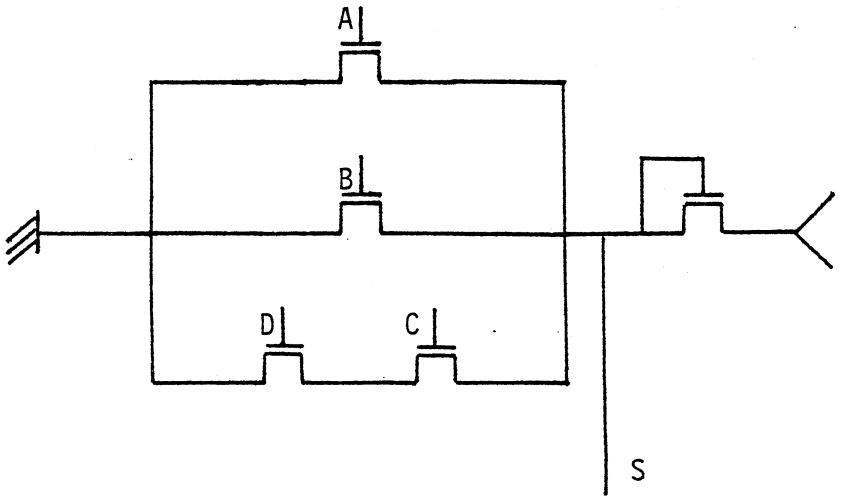
- $F = \overline{AB + CD}$



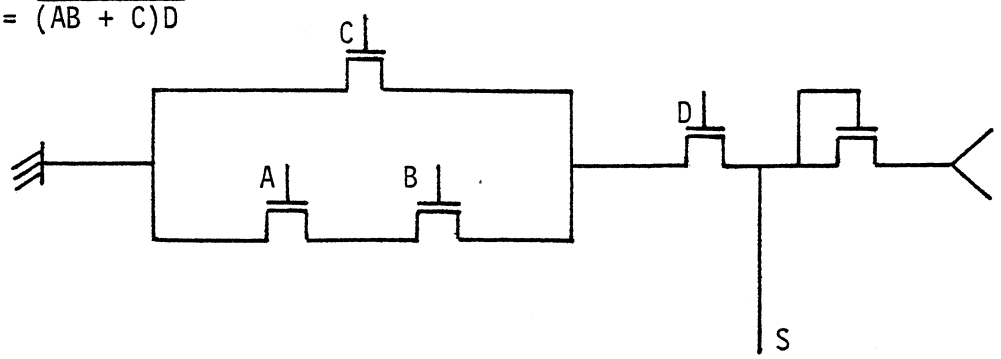
- $F = \overline{AB\ (C + D)}$



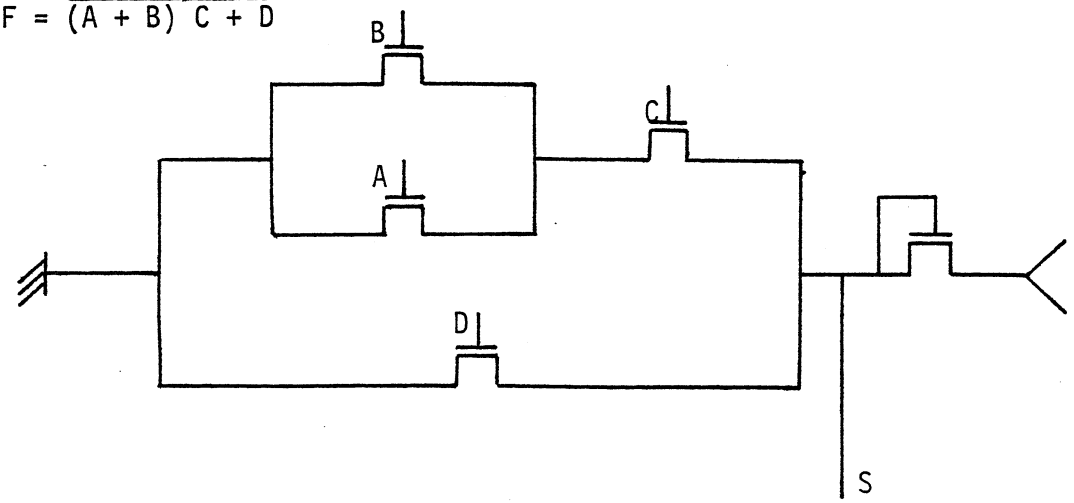
- $F = \overline{A + B + CD}$



- $F = \overline{(AB + C)D}$

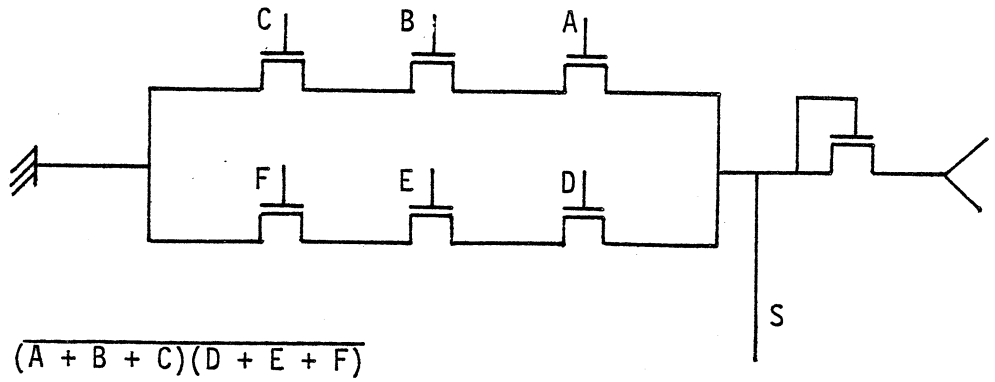


- $F = \overline{(A + B)\ C + D}$

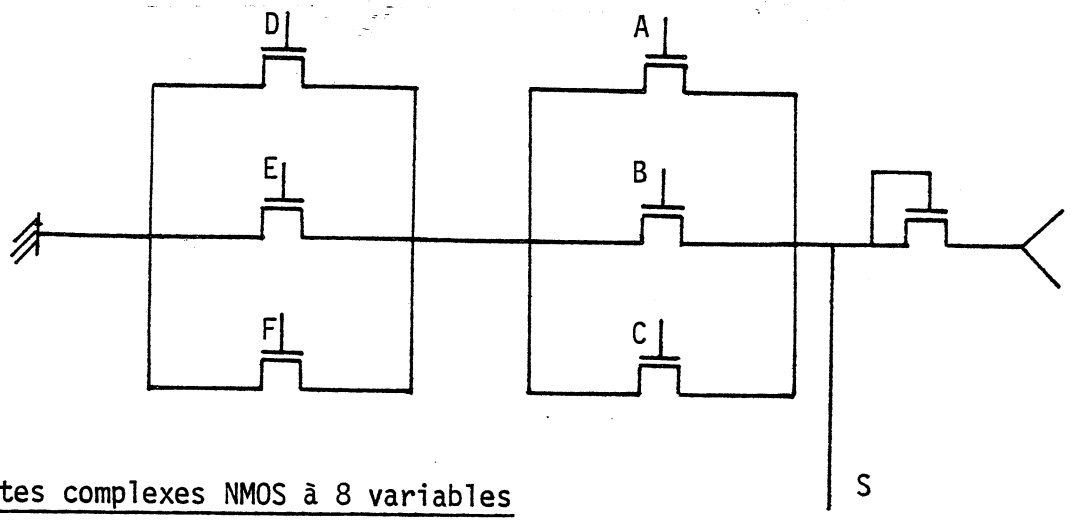


Portes complexes NMOS à 6 variables

- $F = \overline{ABC + DEF}$

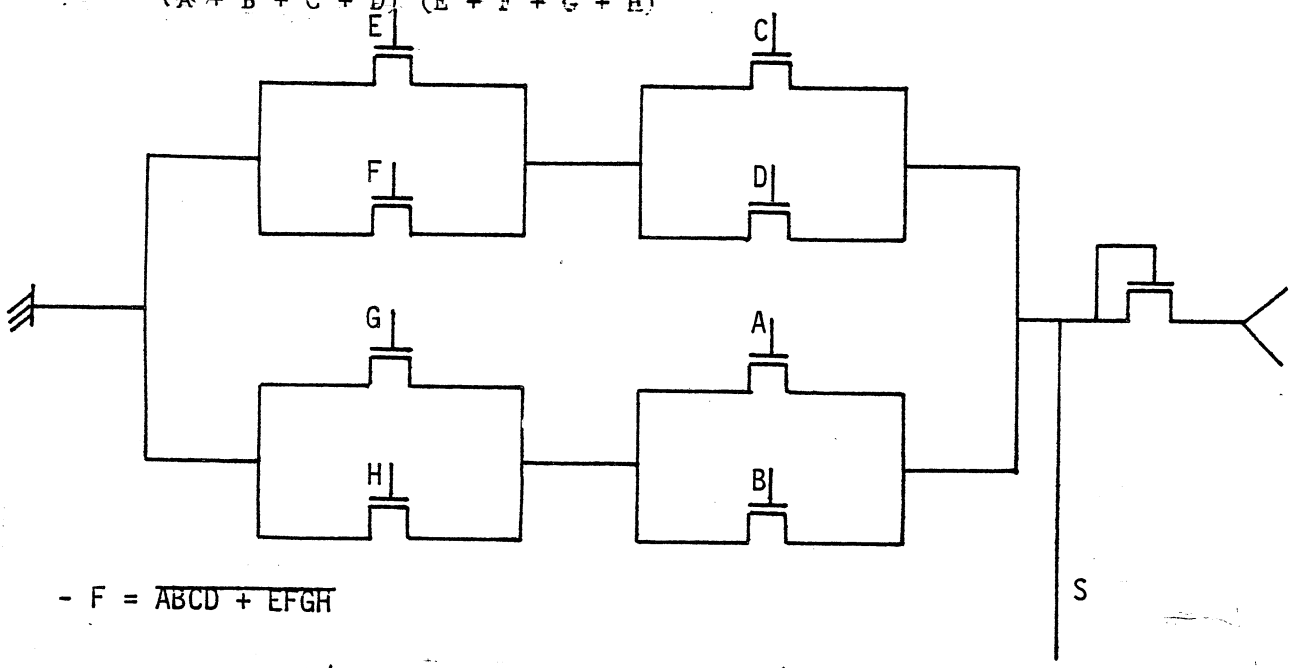


- $F = \overline{(A + B + C)(D + E + F)}$

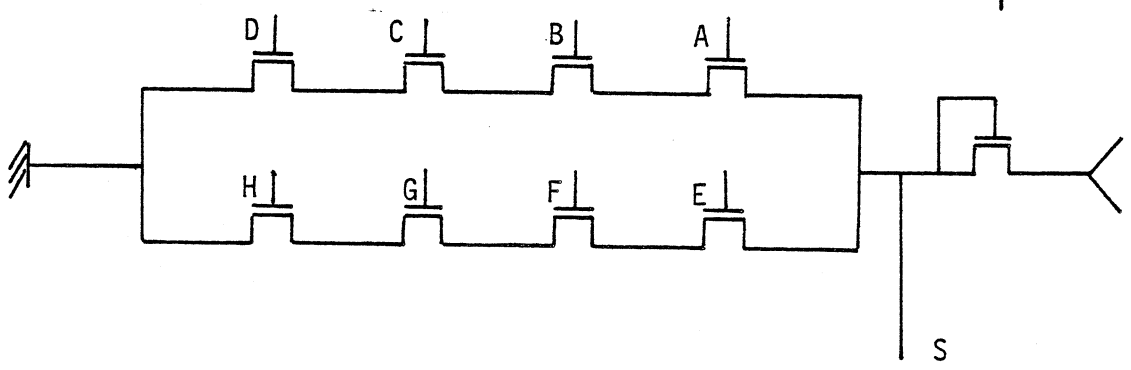


Portes complexes NMOS à 8 variables

- $F = \overline{(A + B + C + D)(E + F + G + H)}$



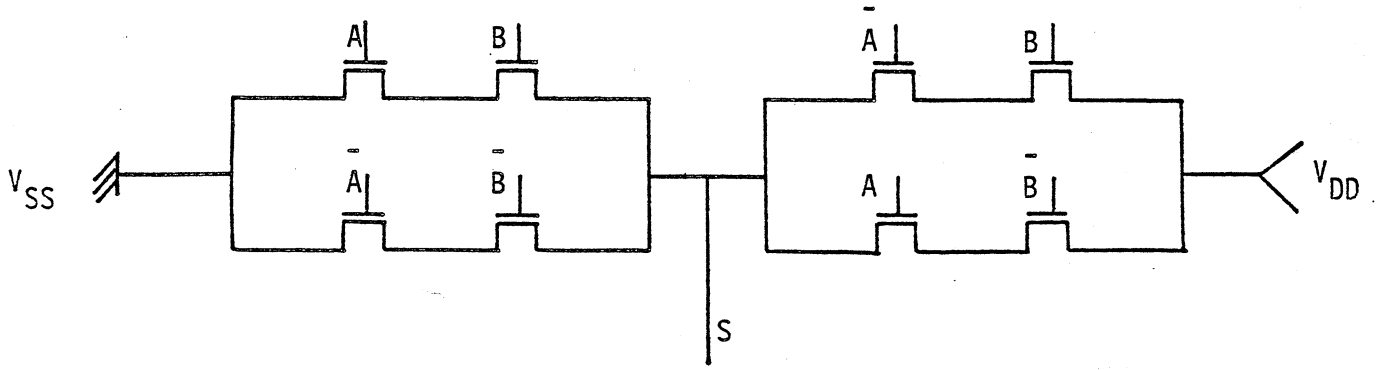
- $F = \overline{ABCD + EFGH}$

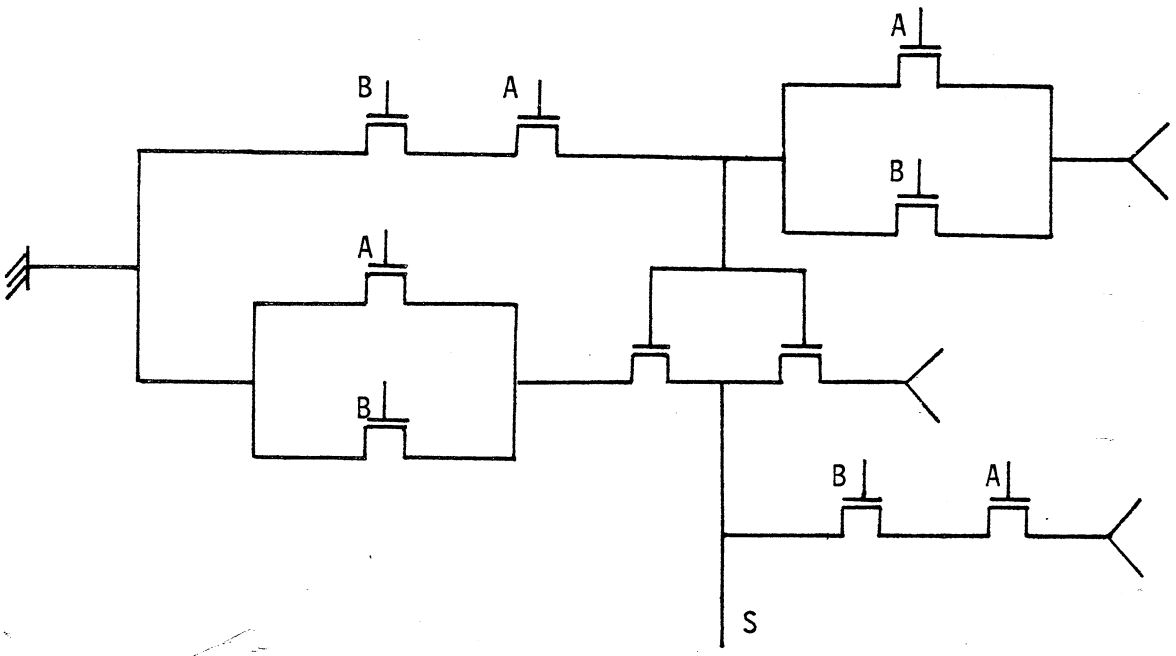


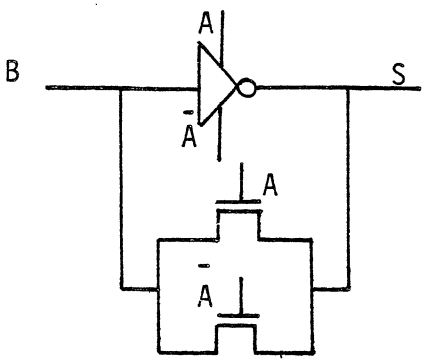
Annexe 2 : Portes complexes CMOS couramment utilisées

Portes complexes CMOS à 2 variables

- OU exclusif, disjonction $F = \bar{A}B + A\bar{B} = \overline{AB + \bar{A}\bar{B}} = \overline{\overline{A+B} + AB}$

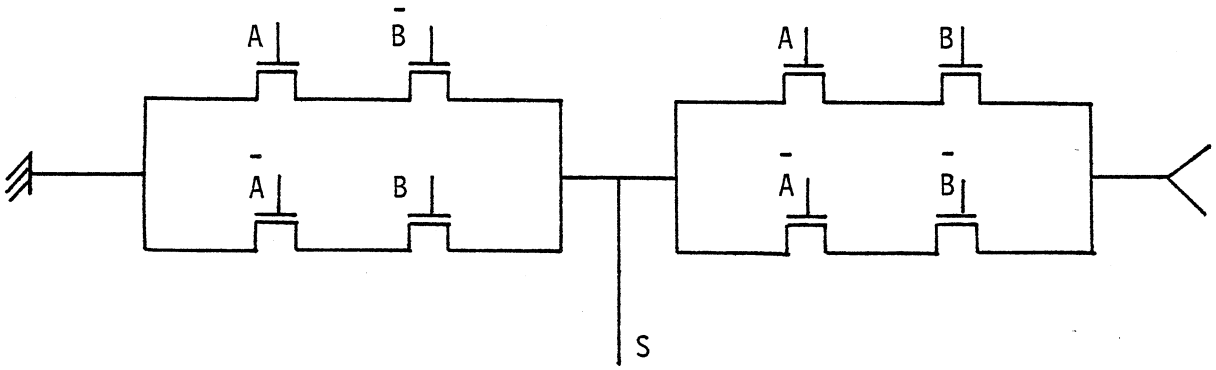


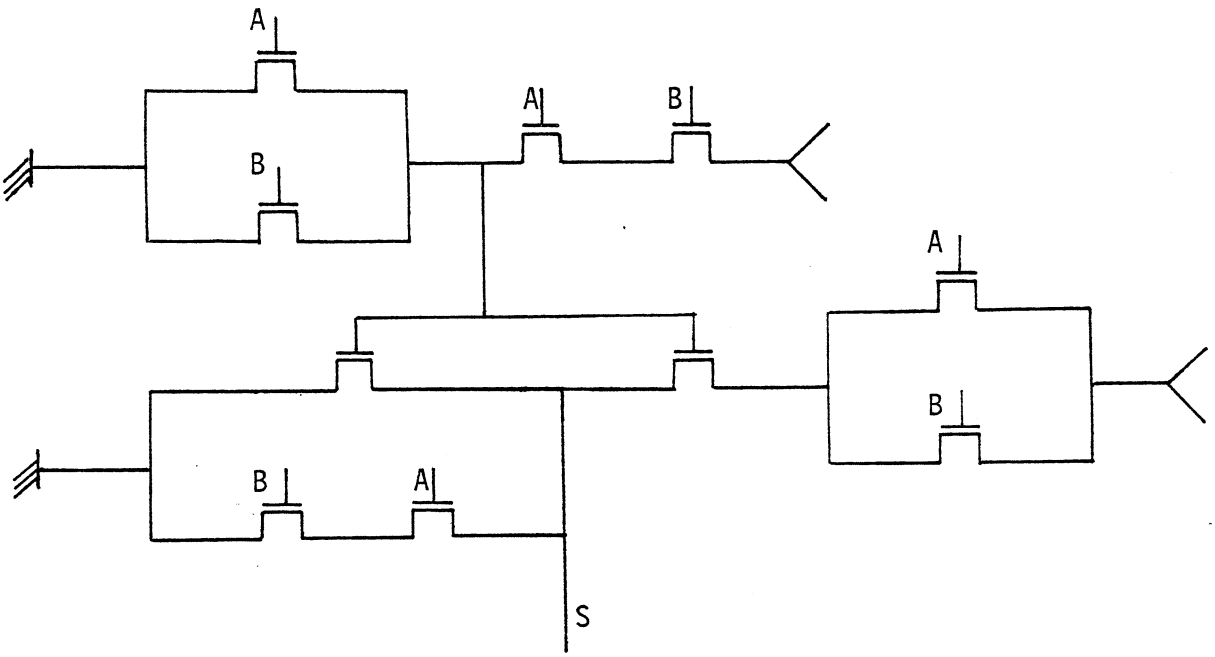




- conjonction

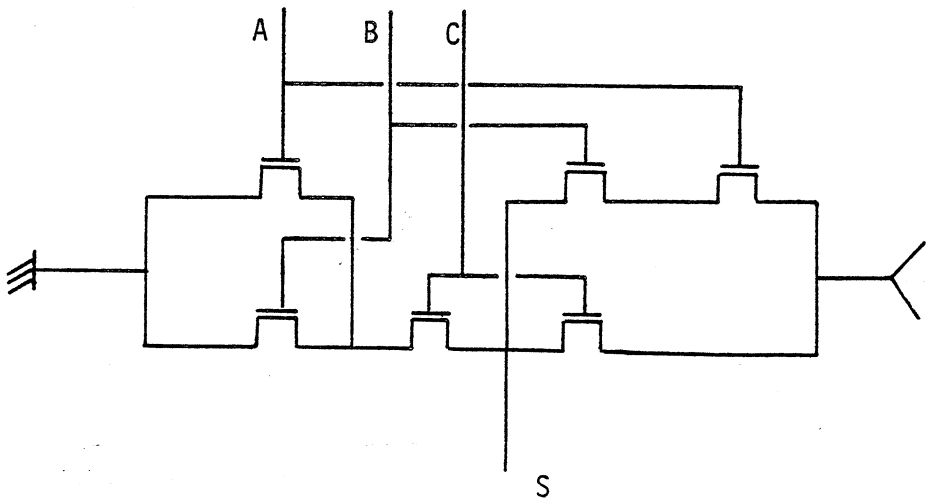
$$F = AB + \bar{A}\bar{B} = \overline{A\bar{B} + \bar{A}B} = \overline{\overline{AB}(A + B)}$$



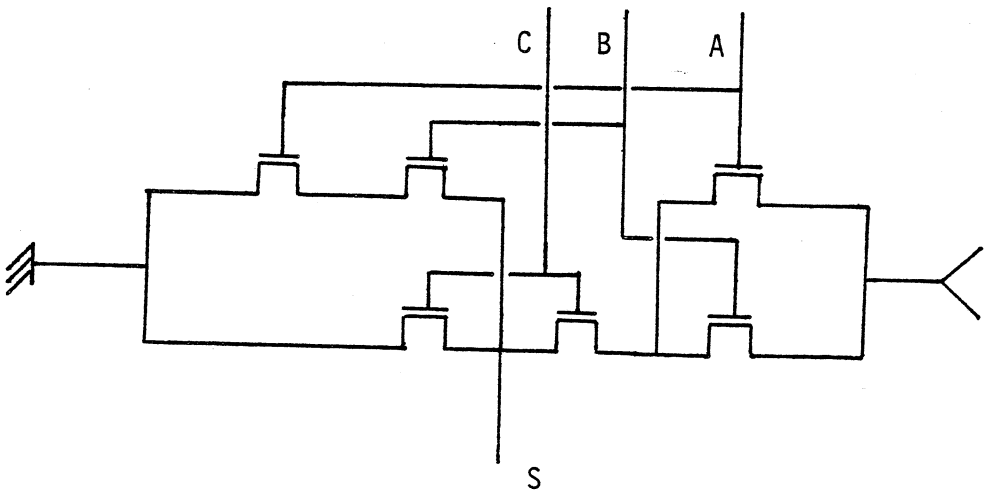


Portes complexes CMOS à 3 variables

- $F = \overline{(A + B)C}$

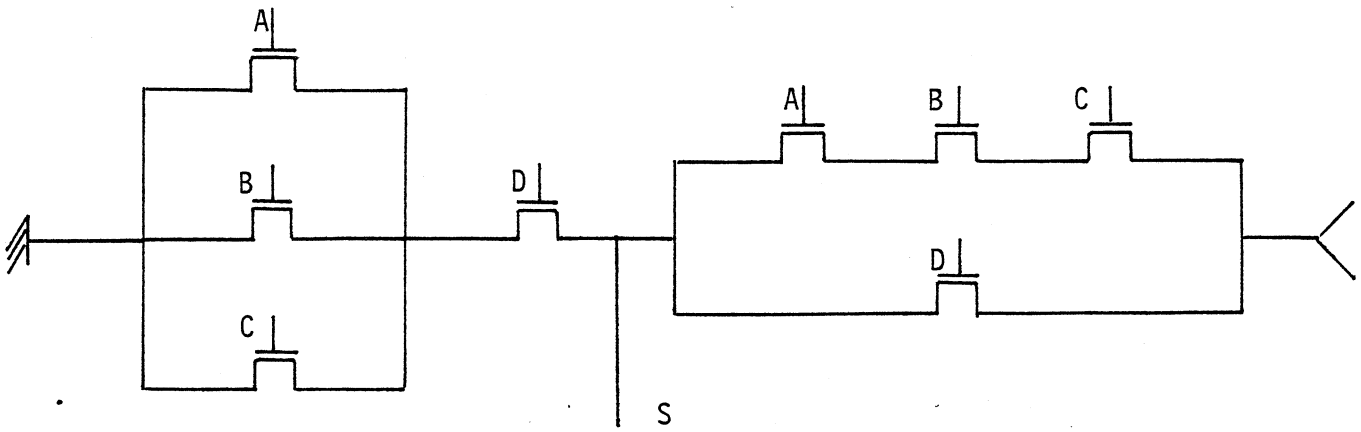


- $F = \overline{AB + C}$

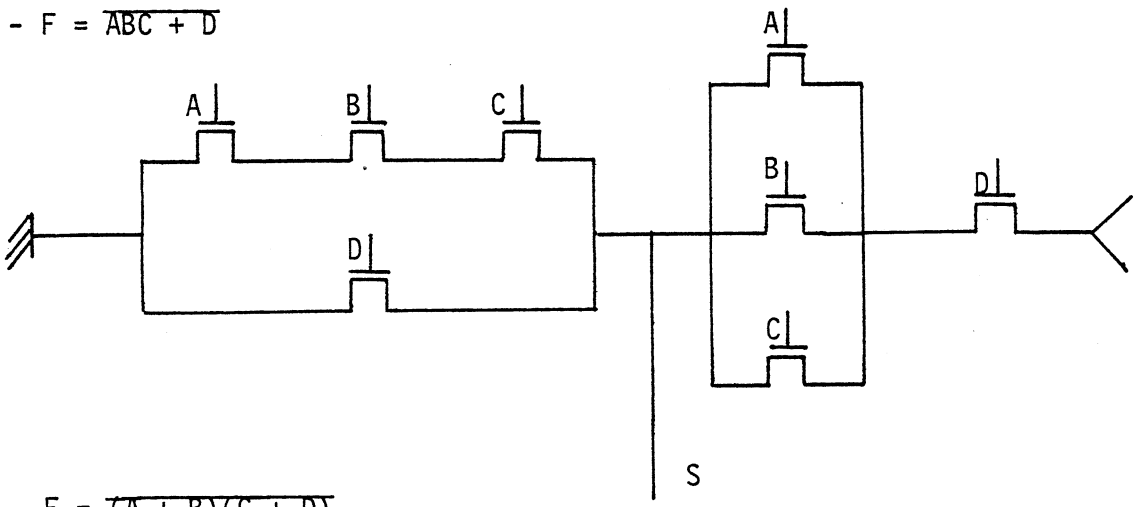


Portes complexes CMOS à 4 variables

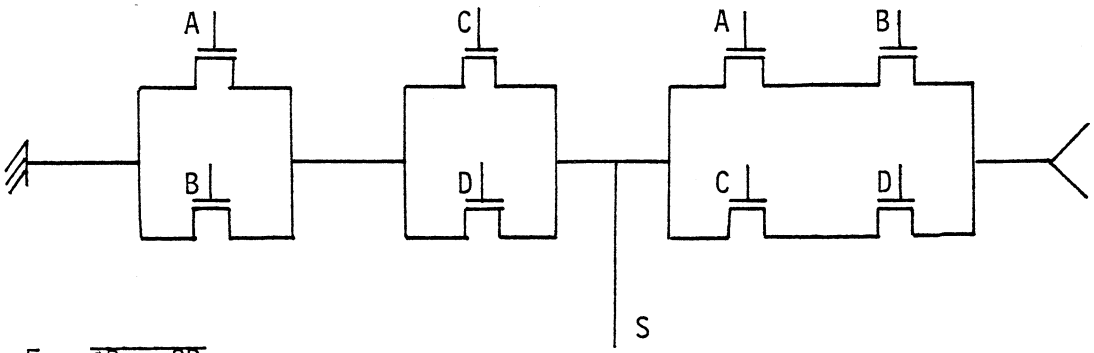
- $F = \overline{(A + B + C)D}$



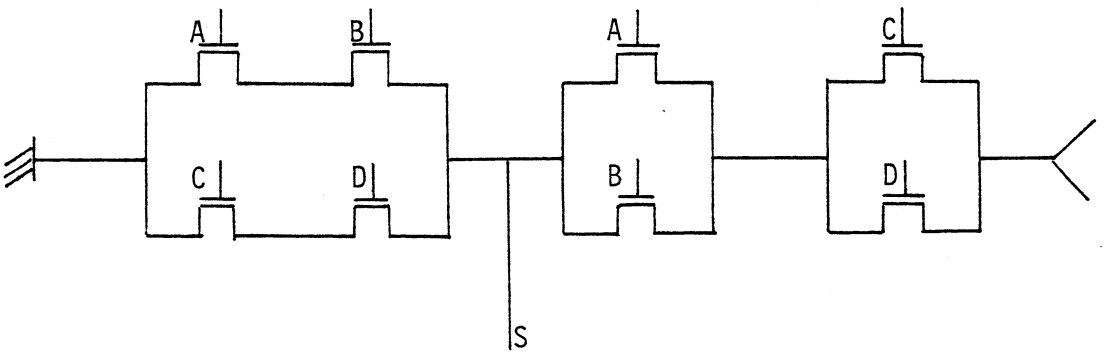
- $F = \overline{ABC + D}$



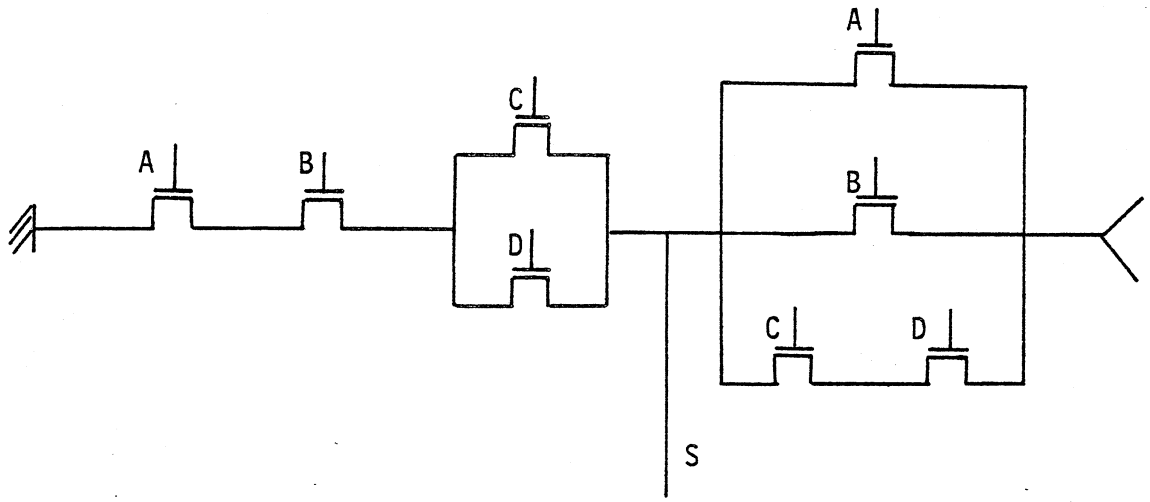
- $F = \overline{(A + B)(C + D)}$



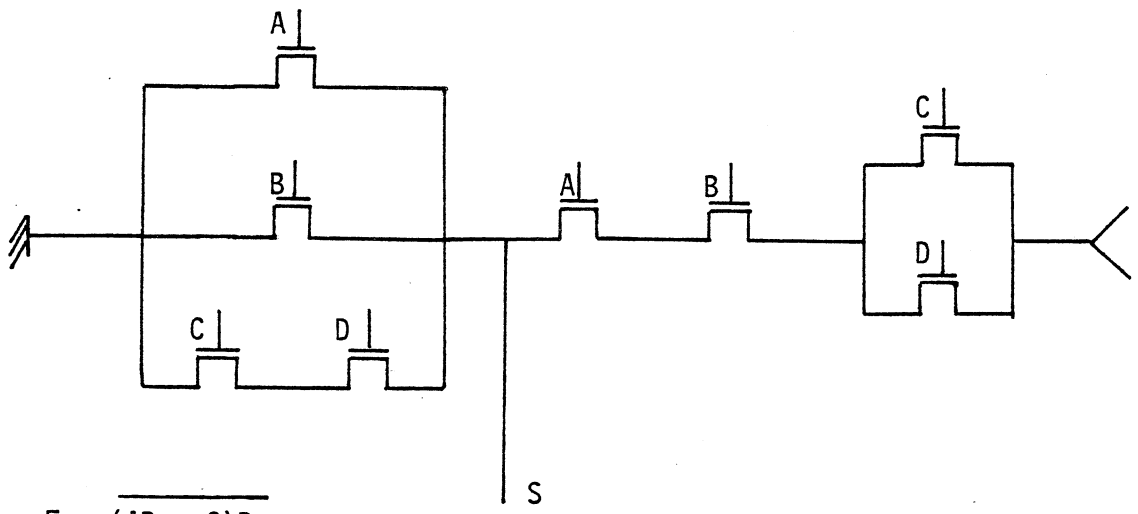
- $F = \overline{AB + CD}$



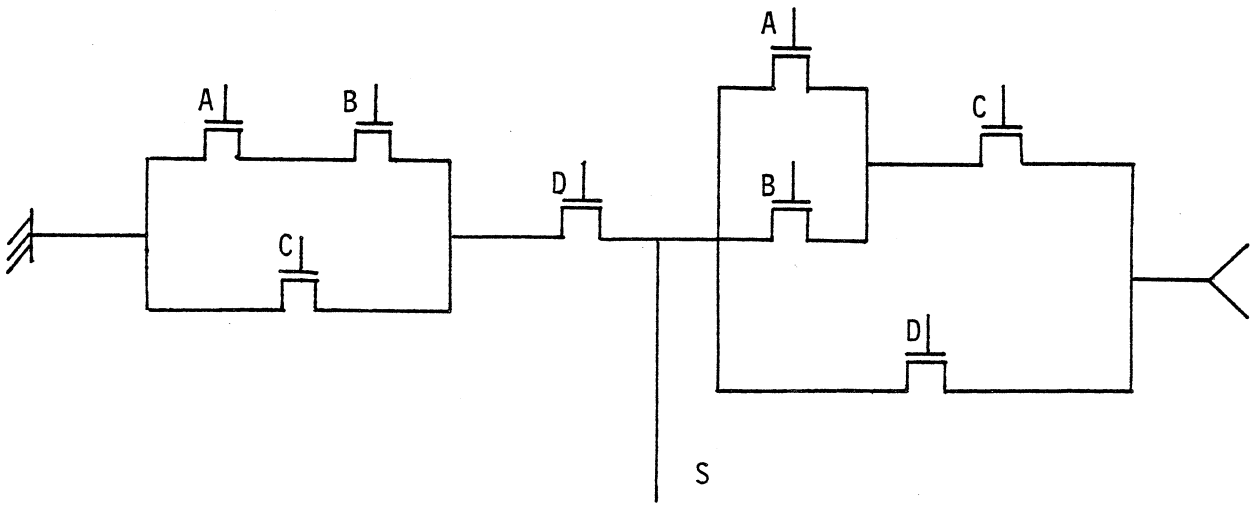
- $F = \overline{AB + (C + D)}$



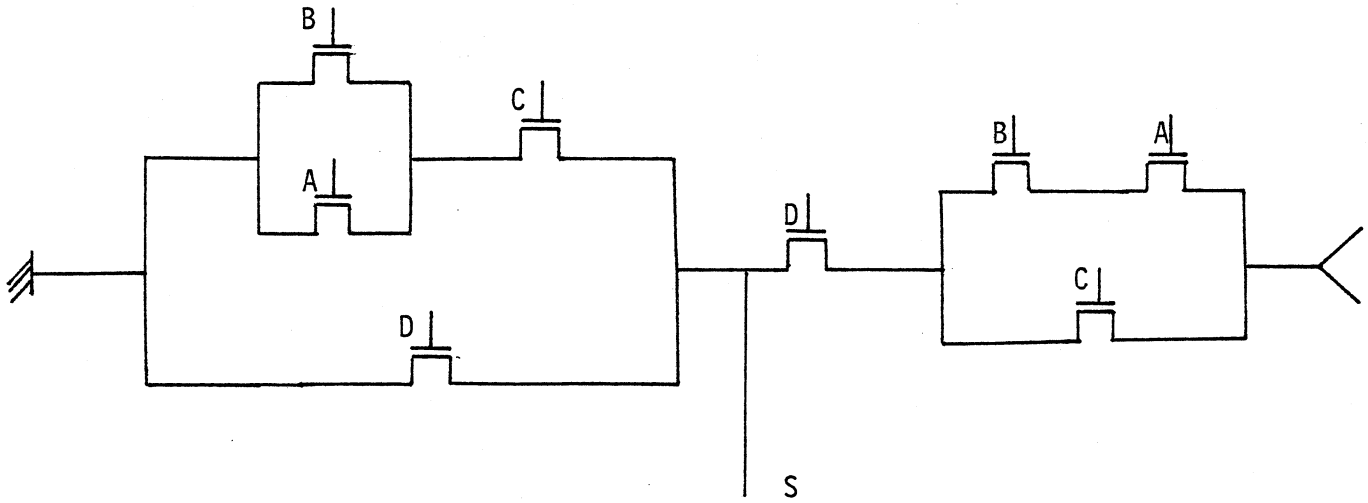
- $F = \overline{A + B + CD}$



- $F = \overline{(AB + C)D}$

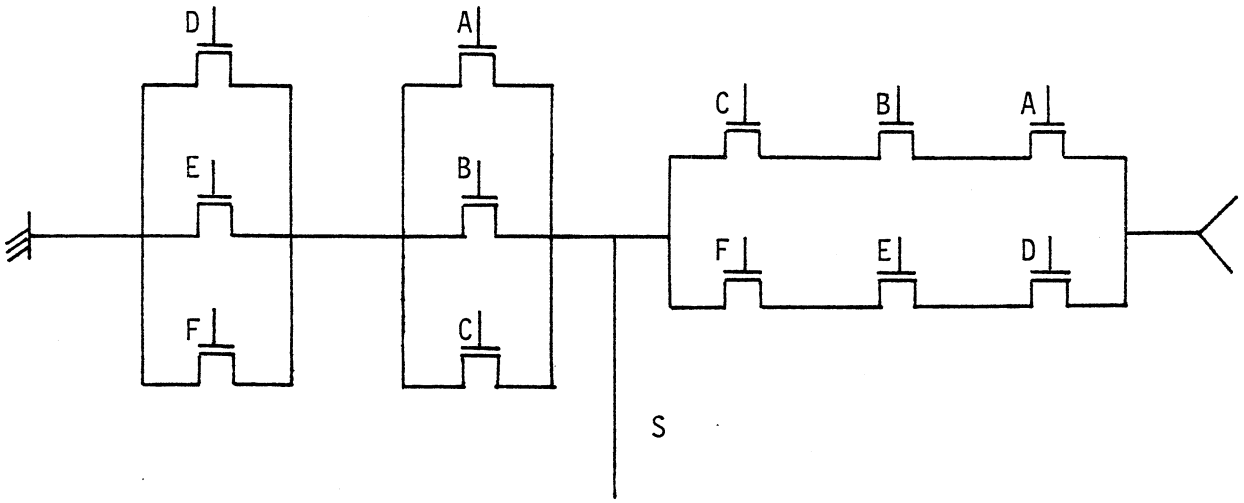


- $F = \overline{(A + B)C + D}$

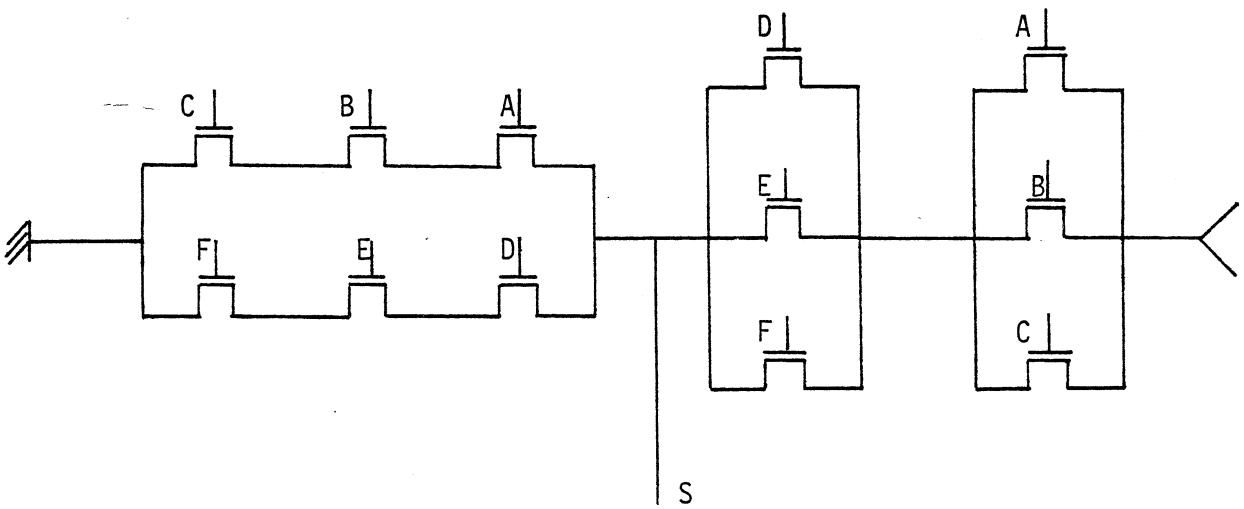


Portes complexes CMOS à 6 variables

- $F = \overline{(A + B + C)(D + E + F)}$

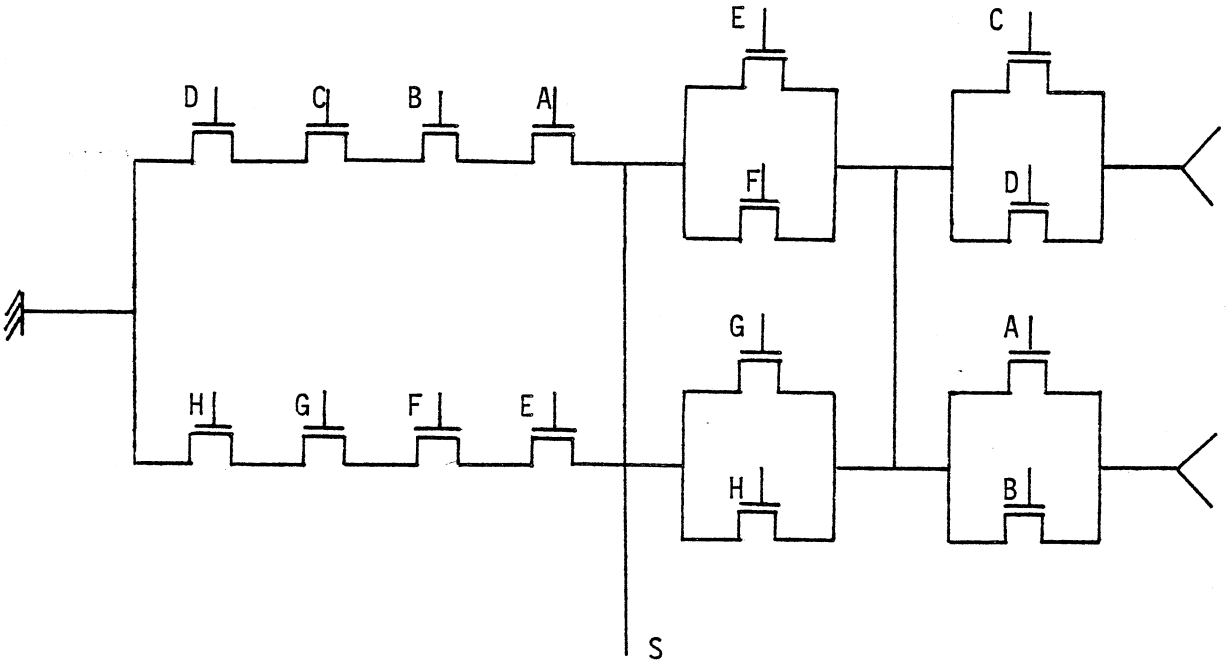


- $F = \overline{ABC + DEF}$



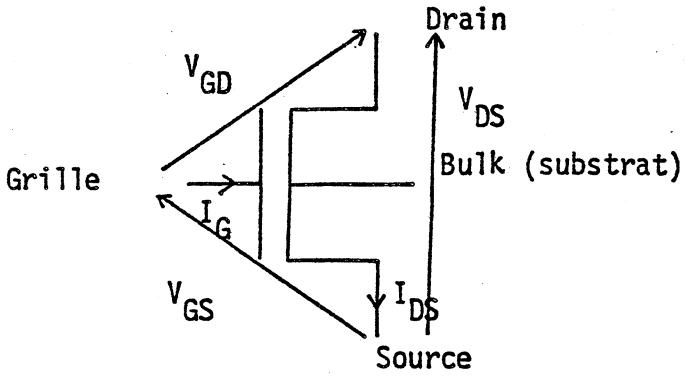
Portes complexes CMOS à 8 variables

- $F = \overline{ABCD + EFGH}$

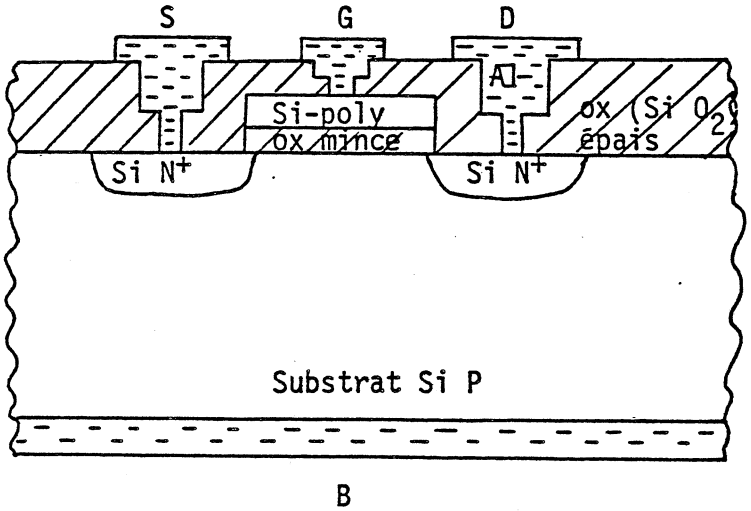


Annexe 3 : Le transistor MOS canal N

I - Symbole électrique



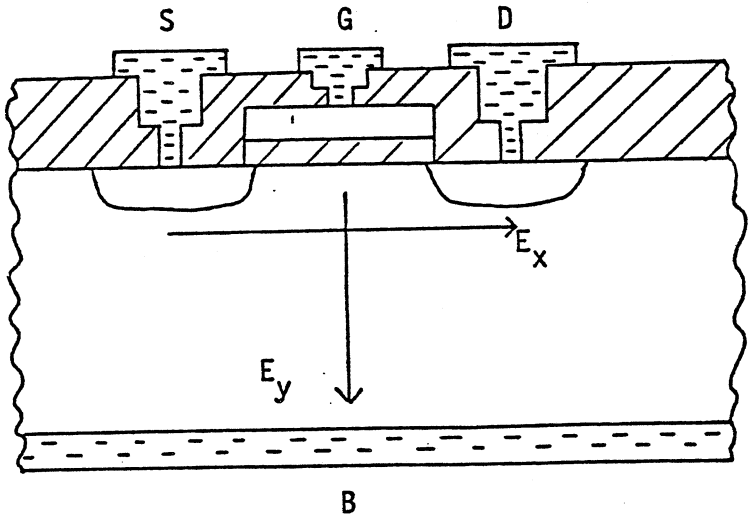
II - Principe de fonctionnement



Soit V_{DS} une tension appliquée entre le drain et la source :

* si il n'y a aucune tension sur la grille, alors une des deux diodes constituées par les jonctions Si P - Si N^+ situées entre le drain et la source est bloquée, d'où $I_{DS} = 0$

* si l'on applique une tension V_{GS} suffisante, cette tension V_{GS} additionnée de V_{BS} crée un champ électrique transversal E_y qui repousse les trous majoritaires et attire les électrons minoritaires du Si P vers la couche d'oxyde entre les deux zones Si N^+, il y a conduction entre D et S par l'intermédiaire du canal ou couche d'inversion N apparu dans le Si P par effet de champ.

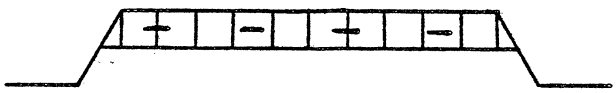


La tension V_{GS} doit d'abord neutralisée les charges fixes ⊕ situées dans le canal et attirer un nombre suffisant d'électrons avant que celui-ci soit du type N, V_{GS} doit donc être supérieur à un certain seuil : V seuil.

A V_{GS} donné ($V_{GS} > V_{seuil}$), on a la répartition de porteurs suivante dans le canal en fonction de V_{DS}.

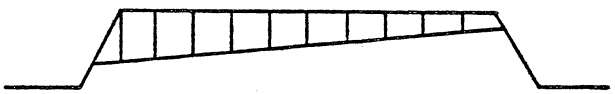
1) $V_{DS} \ll V_{GS} - V_{seuil}$

Le champ longitudinal E_x dû à V_{DS} est très faible par rapport au champ transversal E_y. La répartition des porteurs dans le canal ne dépend que de $(V_{GS} - V_{seuil})$, on aura $I_{DS} = \alpha (V_{GS} - V_{seuil}) V_{DS}$



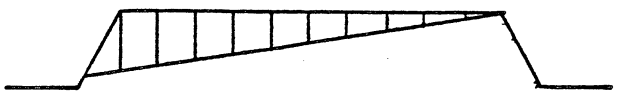
2) $V_{DS} < V_{GS} - V_{seuil}$

Il y a déformation progressive de la répartition des porteurs.



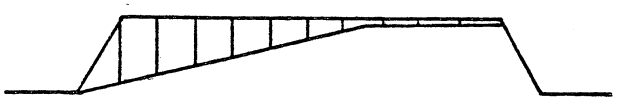
3) $V_{DS} = V_{GS} - V_{seuil}$

On atteint alors la tension de pincement : $V_{GS} - V_{seuil}$

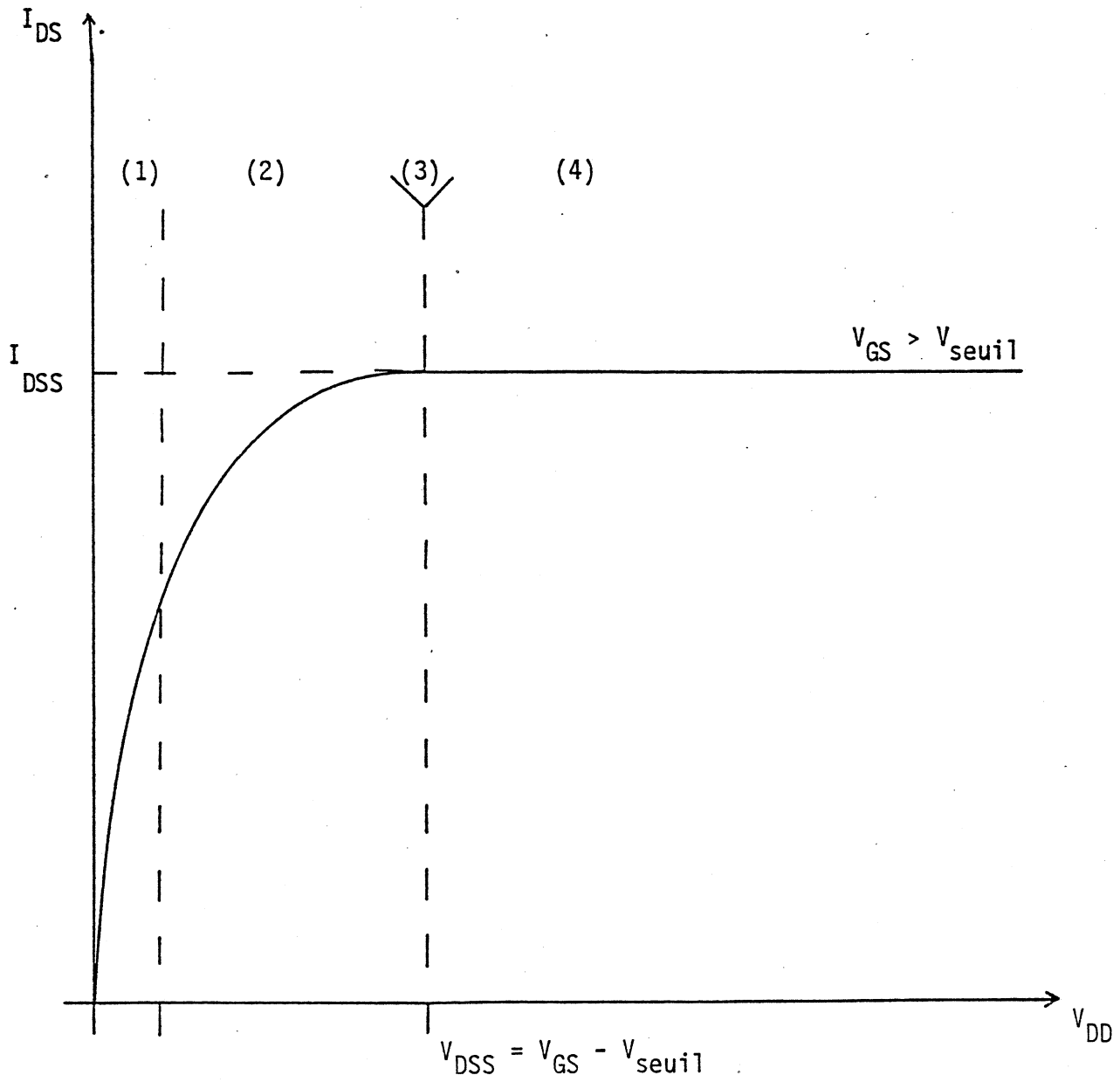


4) $V_{DS} > V_{GS} - V_{seuil}$

A partir de la tension de pincement $(V_{GS} - V_{seuil})$: si V_{DS} augmente, on a alors pincement et perçage, ce qui correspond à la saturation

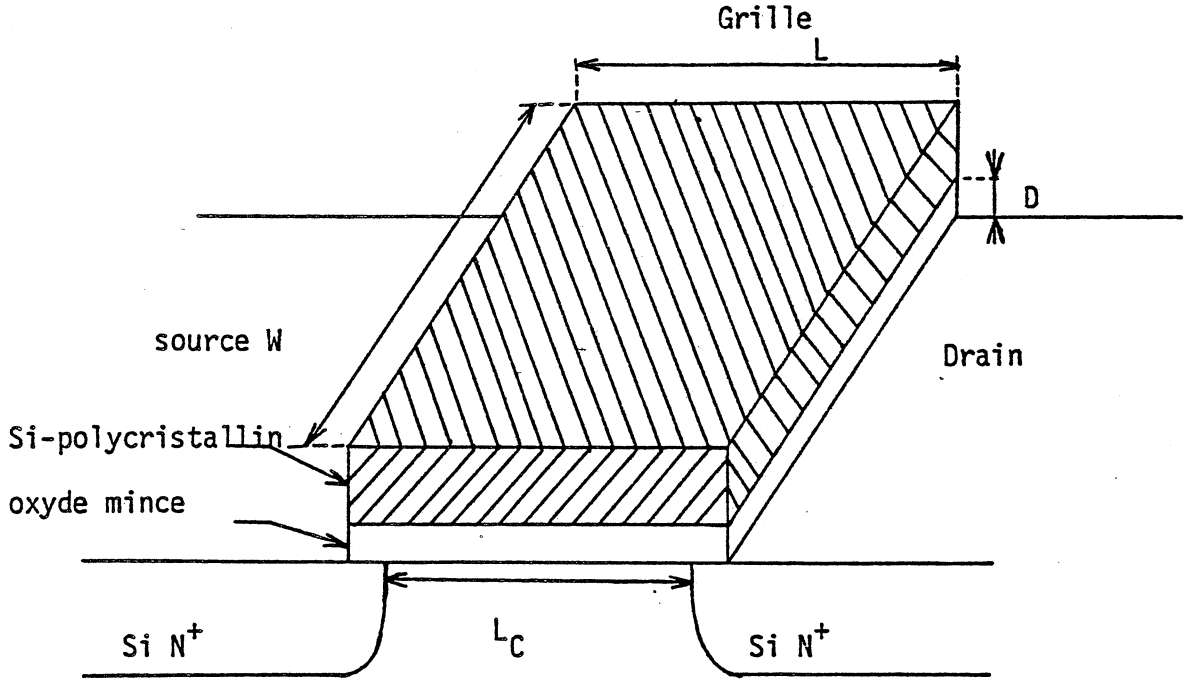


Avec cette répartition de porteurs dans le canal, on obtient la courbe $I_{DS}(V_{DS})$ à V_{GS} donné ($V_{GS} > V_{seuil}$) :



III-Caractéristiques et équations du courant

On définit les dimensions de la grille et du canal du transistor MOS :

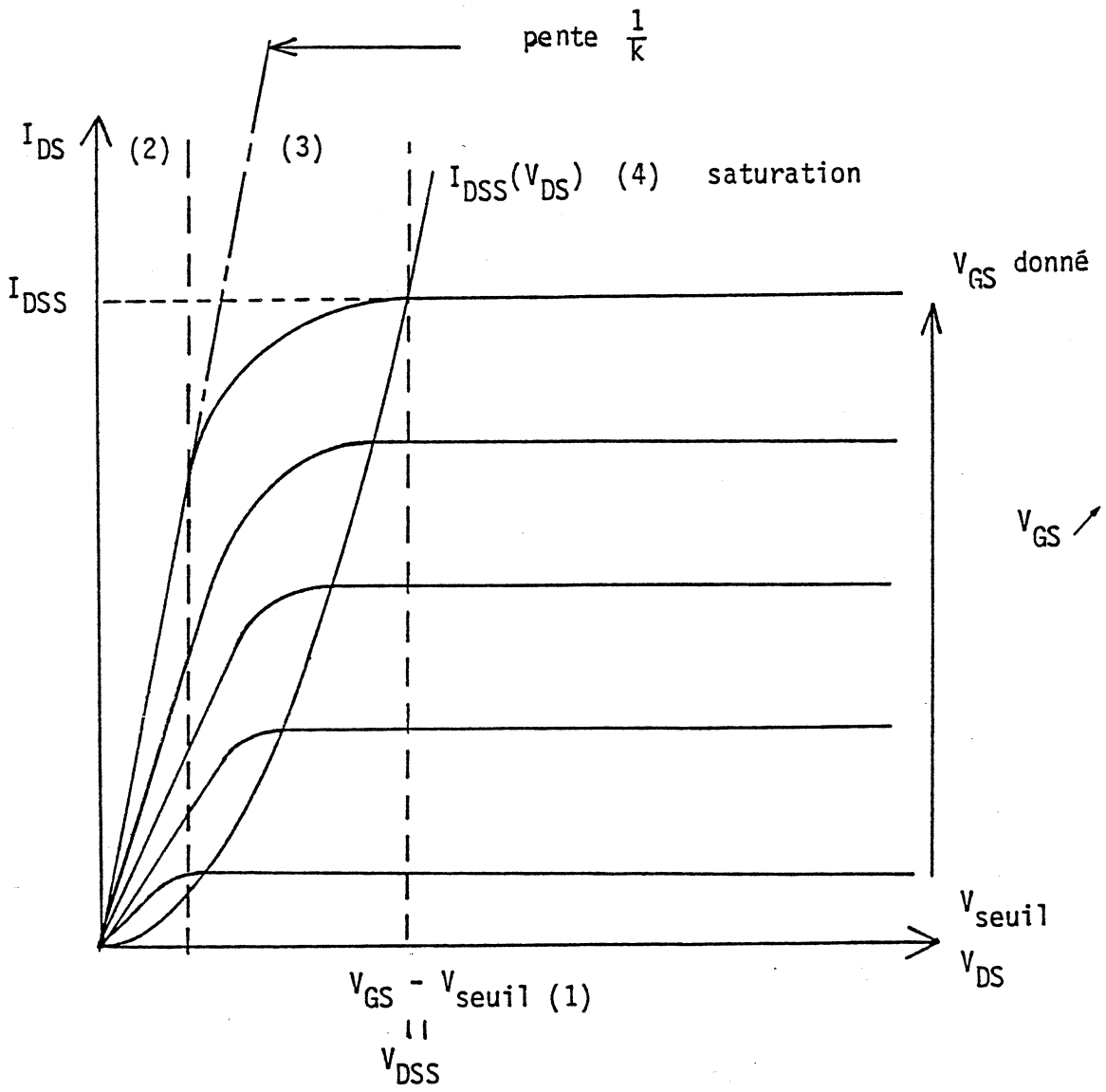


L, W : longueur, largeur de la grille

L_C, W_C : longueur, largeur du canal

D : épaisseur de l'oxyde mince

On distinguera différentes zones sur les caractéristiques $I_{DS}(V_{DS})$:



(1) $V_{GS} < V_{seuil}$

Le canal n'étant pas encore formé, on a $I_{DS} = 0$

Le transistor est bloqué

(2) - (3) $V_{GS} > V_{seuil}$ avec $V_{DS} < V_{GS} - V_{seuil}$

— On a :

$$I_{DS} = \bar{\mu}_n C_{ox} \frac{W_c}{L_c} \left[(V_{GS} - V_{seuil}) V_{DS} - \frac{V_{DS}^2}{2} \right]$$

$\bar{\mu}_n$: mobilité moyenne des porteurs (électrons)

avec C_{ox} : capacité de la couche d'oxyde mince par unité de surface

$C_{ox} = \frac{\varepsilon}{D}$, où ε est la permittivité de l'oxyde

W_c, L_c : largeur et longueur du canal

Le transistor est conducteur.

— Dans la zone 2, $V_{DS} << V_{GS} - V_{seuil}$:

On a alors :

$$I_{DS} = \bar{\mu}_n C_{ox} \frac{W_c}{L_c} (V_{GS} - V_{seuil}) V_{DS}$$

On est dans la région ohmique de la caractéristique :

$$R = \frac{1}{\bar{\mu}_n C_{ox} \frac{W_c}{L_c} (V_{GS} - V_{seuil})}$$

R est ajustable avec V_{GS}.

(4) $V_{GS} > V_{seuil}$ avec $V_{DS} \geq V_{GS} - V_{seuil}$

Le transistor est saturé. Il fonctionne en générateur de courant.

On cherche I_{DS} maximum : I_{DSS} à partir de l'expression de I_{DS} précédente :

$$\frac{dI_{DS}}{dV_{DS}} = \overline{\mu}_n \, C_{ox} \, \frac{W_c}{L_c} \, (V_{GS} - V_{seuil} - V_{DS})$$

$$\frac{dI_{DS}}{dV_{DS}} = 0 \text{ pour } V_{DS} = V_{GS} - V_{seuil}$$

$$\text{alors : } I_{DSS} = \overline{\mu}_n \, C_{ox} \, \frac{W_c}{L_c} \, \frac{(V_{GS} - V_{seuil})^2}{2}$$

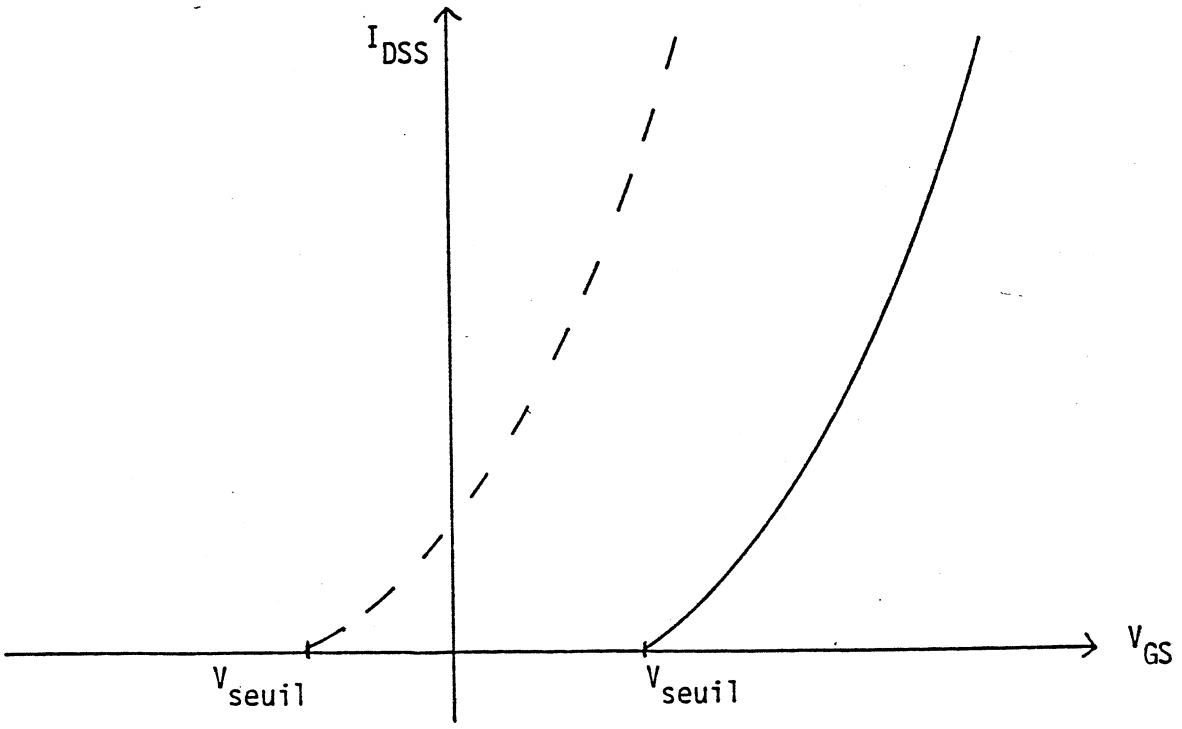
On a aussitôt $I_{DSS}(V_{DS})$:

$$I_{DSS} = \overline{\mu}_n \, C_{ox} \, \frac{W_c}{L_c} \, \frac{V_{DS}^2}{2} \quad : \text{parabole}$$

On a la courbe $I_{DSS}(V_{GS})$:

$$I_{DSS} = \overline{\mu}_n \, C_{ox} \, \frac{W_c}{L_c} \, \frac{(V_{GS} - V_{seuil})^2}{2}$$

calculé précédemment : parabole centrée sur V_{seuil}.



On peut ajuster la valeur de V_{seuil} lors de la fabrication du transistor.
Il existe deux familles de transistors MOS :

* MOS à enrichissement : $V_{seuil} > 0$, à $V_{GS} = 0$ le transistor est bloqué.

* MOS à déplétion : $V_{seuil} < 0$, à $V_{GS} = 0$ le transistor est conducteur. Il existe un pré-canal de type N.

Remarques sur le calcul de l'expression du courant :

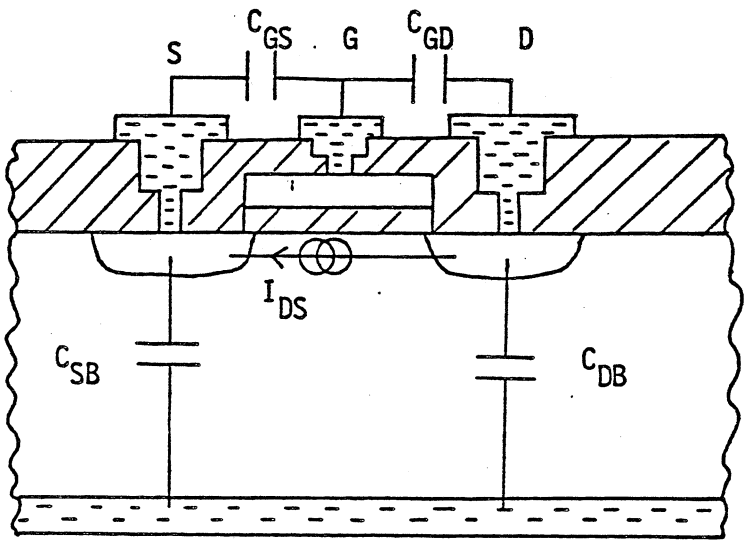
$$I_{DS} = \bar{\mu}_n C_{ox} \frac{W_c}{L_c} \left[(V_{GS} - V_{seuil}) V_{DS} - \frac{V_{DS}^2}{2} \right]$$

* La grille du transistor MOS est isolée (oxyde : Si 02). On a : I_G = quelques pA (10^{-12} A), donc $I_G \ll I_{DS}$, $I_G \neq 0$

* Les diffusions de la source et du drain sont fortement dopées (N^+) pour minimiser les résistances séries qui limitent le courant I_{DS}. Pour le calcul, on prend ces résistances égales à zéro, on ne considère que la résistance de grille.

* Dans l'expression de V_{seuil}, la tension Bulk-Source V_{BS} intervient.

IV - Modèle électrique

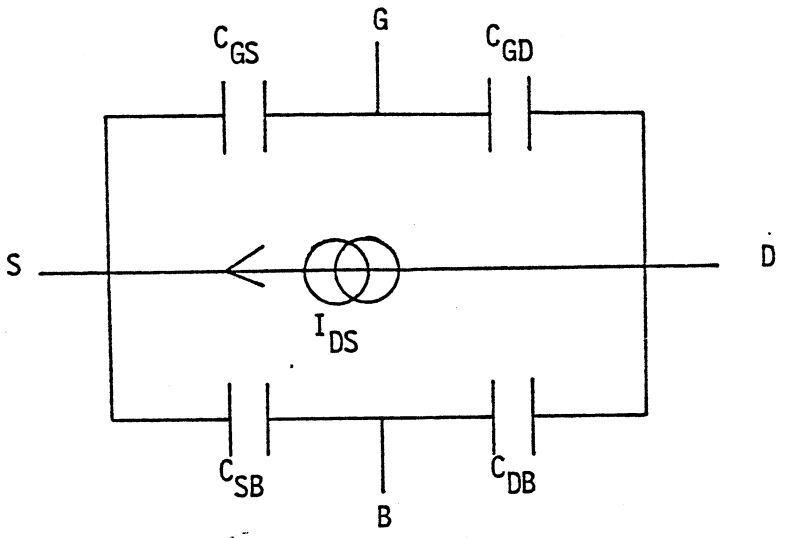


I_{DS} : courant calculé précédemment fonction de V_{GS}, V_{GD}, V_{BS}

C_{SB} et C_{DB} : capacité de diodes en inverse

C_{GS} et C_{GD} : capacité de débordement
et
capacité active de grille C_{ox}

L'ordre de grandeur de ces capacités est de quelques 10^{-3} à 10^{-4} pF/μ^2



AUTORISATION DE SOUTENANCE

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

VU le rapport de présentation de Madame G. SAUCIER, Professeur

Madame THUAU Ghislaine

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

Fait à Grenoble, le 17 novembre 1983

Le Président de l'INP-G

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

P.O. le Vice-Président,