Contribution à la vérification des circuits intégrés dans un environnement multivalué

J.-P. Caisso

THESE

présentée par

Jean-Paul CAISSO

pour obtenir le titre de **DOCTEUR**

de l'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

(arrêté ministériel du 5 Juillet 1984)

Spécialité : ***Microélectronique***

CONTRIBUTION A LA VERIFICATION DES CIRCUITS INTEGRES DANS UN ENVIRONNEMENT MULTIVALUE

Date de soutenance : 16 Novembre 1987

Composition du Jury :

Messieurs	Pierre	GENTIL	*Président*
	Bernard	COURTOIS	
	Christian	MASSON	
	Daniel	ETIEMBLE	*Rapporteurs*
	Christian	LANDRAULT	

Thèse préparée au sein du Laboratoire TIM3.

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Année 1987

Président : Georges LESPINARD

Vice-Présidents : Jean-Marie PIERRARD
Jean-Pierre VERJUS

Professeurs des Universités

BARIBAUD	Michel	ENSERG
BARRAUD	Alain	ENSIEG
BAUDELET	Bernard	ENSPG
BEAUFILS	Jean-Pierre	ENSEEG
BESSON	Jean	ENSEEG
BLIMAN	Samuel	ENSERG
BLOCH	Daniel	ENSPG
BOIS	Philippe	ENSHMG
BONNETAIN	Lucien	ENSEEG
BOUVARD	Maurice	ENSHMG
BRISSONNEAU	Pierre	ENSIEG
BRUNET	Yves	IUFA
BUYLE-BODIN	Maurice	ENSERG
CAILLERIE	Denis	ENSHMG
CAVAIGNAC	Jean-François	ENSPG
CHARTIER	Germain	ENSPG
CHENEVIER	Pierre	ENSERG
CHERADAME	Hervé	UFR PGP
CHERUY	Arlette	ENSIEG
CHIAVERINA	Jean	UFR PGP
CHOVET	Alain	ENSERG
COHEN	Joseph	ENSERG
COUMES	André	ENSERG
DARVE	Félix	ENSHMG
DELLA-DORA	Jean-François	ENSIMAG
DEPORTES	Jacques	ENSPG
DOLMAZON	Jean-Marc	ENSERG
DURAND	Francis	ENSEEG
DURAND	Jean-Louis	ENSIEG
FONLUPT	Jean	ENSIMAG
FOULARD	Claude	ENSIEG
GANDINI	Alessandro	UFR PGP
GAUBERT	Claude	ENSPG
GENTIL	Pierre	ENSERG
GREVEN	Hélène	IUFA
GUERIN	Bernard	ENSERG
GUYOT	Pierre	ENSEEG
IVANES	Marcel	ENSIEG
JAUSSAUD	Pierre	ENSIEG
JOUBERT	Jean-Claude	ENSPG
JOURDAIN	Geneviève	ENSIEG
LACOUME	Jean-Louis	ENSIEG
LESIEUR	Marcel	ENSHMG
LESPINARD	Georges	ENSHMG
LONGEQUEUE	Jean-Pierre	ENSPG
LOUCHET	François	ENSEEG
MASSE	Philippe	ENSIEG
MASSELOT	Christian	ENSIEG
MAZARE	Guy	ENSIMAG
MOREAU	René	ENSHMG
MORET	Roger	ENSIEG
MOSSIERE	Jacques	ENSIMAG
OBLED	Charles	ENSHMG
OZIL	Patrick	ENSEEG
PARIAUD	Jean-Charles	ENSEEG
PAUTHENET	René	ENSIEG
PERRET	René	ENSIEG
PERRET	Robert	ENSIEG
PIAU	Jean-Michel	ENSHMG
POUPOT	Christian	ENSERG
RAMEAU	Jean-Jacques	ENSEEG
RENAUD	Maurice	UFR PGP
ROBERT	André	UFR PGP
ROBERT	François	ENSIMAG
SABONNADIERE	Jean-Claude	ENSIEG
SAUCIER	Gabrielle	ENSIMAG
SCHLENKER	Claire	ENSPG
SCHLENKER	Michel	ENSPG
SERMET	Pierre	ENSERG
SILVY	Jacques	UFR PGP
SIRIEYS	Pierre	ENSHMG
SOHM	Jean-Claude	ENSEEG
SOLER	Jean-Louis	ENSIMAG
SOUQUET	Jean-Louis	ENSEEG
TROMPETTE	Philippe	ENSHMG
VEILLON	Gérard	ENSIMAG
ZADWORNY	François	ENSERG

Professeur Université des Sciences Sociales (Grenoble II)

BOLLIET	Louis

Personnes ayant obtenu le diplôme d'HABILITATION A DIRIGER DES RECHERCHES

BECKER	Monique
BINDER	Zdenek
CHASSERY	Jean-Marc
COEY	John
COLINET	Catherine
COMMAULT	Christian
CORNUEJOLS	Gérard
DALARD	Francis
DANES	Florin
DEROO	Daniel
DIARD	Jean-Paul
DION	Jean-Michel
DUGARD	Luc
DURAND	Robert
GALERIE	Alain
GAUTHIER	Jean-Paul
GENTIL	Sylviane
PLA	Fernand
GHIBAUDO	Gérard
LADET	Pierre
LATOMBE	Claudine
LE GORREC	Bernard
MADAR	Roland
MULLER	Jean
NGUYEN TRONG	Bernadette
TCHUENTE	Maurice
VINCENT	Henri

Chercheurs du C.N.R.S

Directeurs de recherche 1ère Classe

CAILLET	Marcel	JORRAND	Philippe
CARRE	René	LANDAU	Ioan
FROCHART	Robert	MARTIN	

Directeurs de recherche 2ème Classe

ALEMANY	Antoine	EUSTATHOPOULOS	Nicolas
ALLIBERT	Colette	JOUD	Jean-Charles
ALLIBERT	Michel	KAMARINOS	Georges
ANSARA	Ibrahim	KLEITZ	Michel
ARNAUD	Michel	KOFMAN	Walter
BINDER	Gilbert	LEJEUNE	Gérard
BOREL	Roland	MERMET	Jean
BORNARD	Guy	MUNIER	Jacques
CALMET	Jacques	SENATEUR	Jean-Pierre
DAVID	René	SUERY	Michel
DRIOLE	Jean	TEDOSIU	
ESCUDIER	Pierre	WACK	Bernard

Personnalités agrées à titre permanent à diriger des travaux de recherche (décision du conseil scientifique)

E.N.S.E.E.G

BERNARD	Claude	MALMEJAC	Yves
CHATILLON	Catherine	MARTIN GARIN	Régina
CHATILLON	Christian	SAINFORT	Paul
COULON	Michel	SARRAZIN	Pierre
		SIMON	Jean-Paul
FOSTER	Panayotis	TOUZAIN	Philippe
HAMMOU	Abdelkader	URBAIN	Georges

E.N.S.E.R.G

BOREL	Joseph

E.N.S.I.E.G

DESCHIZEAUX	Pierre	PERARD	Jacques
GLANGEAUD	François	REINISCH	Raymond

E.N.S.H.G

BOIS	Daniel	ROWE	Alain
		VAUCLIN	Michel
MICHEL	Jean-Marie		

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

BERT	Didier		
COURTIN	Jacques		
COURTOIS	Bernard	SIFAKIS	Joseph

E.F.P.-G

CHARUEL	Robert

C.E.N.G

CADET	Jean	NIFENECKER	Hervé
COEURF	Philippe	PERROUD	Paul
DELHAYE	Jean-Marc	PEUZIN	Jean-Claude
DUPUY	Michel	TAIEB	Maurice
JOUVE	Hubert	VINCENDON	Marc
NICOLAU	Yvan		

Laboratoires extérieurs

C.N.E.T

DEMOULIN	Eric	MERCKEL	Gérard
DEVINE		PAULEAU	Yves
GERBER	Roland		

****

Je tiens à remercier

M. Pierre GENTIL, Directeur du Centre Interuniversitaire de Micro-Electronique et Professeur à l' E.N.S.E.R.G., qui me fait l'honneur de présider le jury de cette thèse ;

M. Bernard COURTOIS, Directeur de Recherche au C.N.R.S. et Directeur-Adjoint du Laboratoire TIM3, qui m'a accueilli dans l'Equipe d'Architecture des Ordinateurs, et a toujours suivi mon travail avec beaucoup d'efficacité et de bienveillance ;

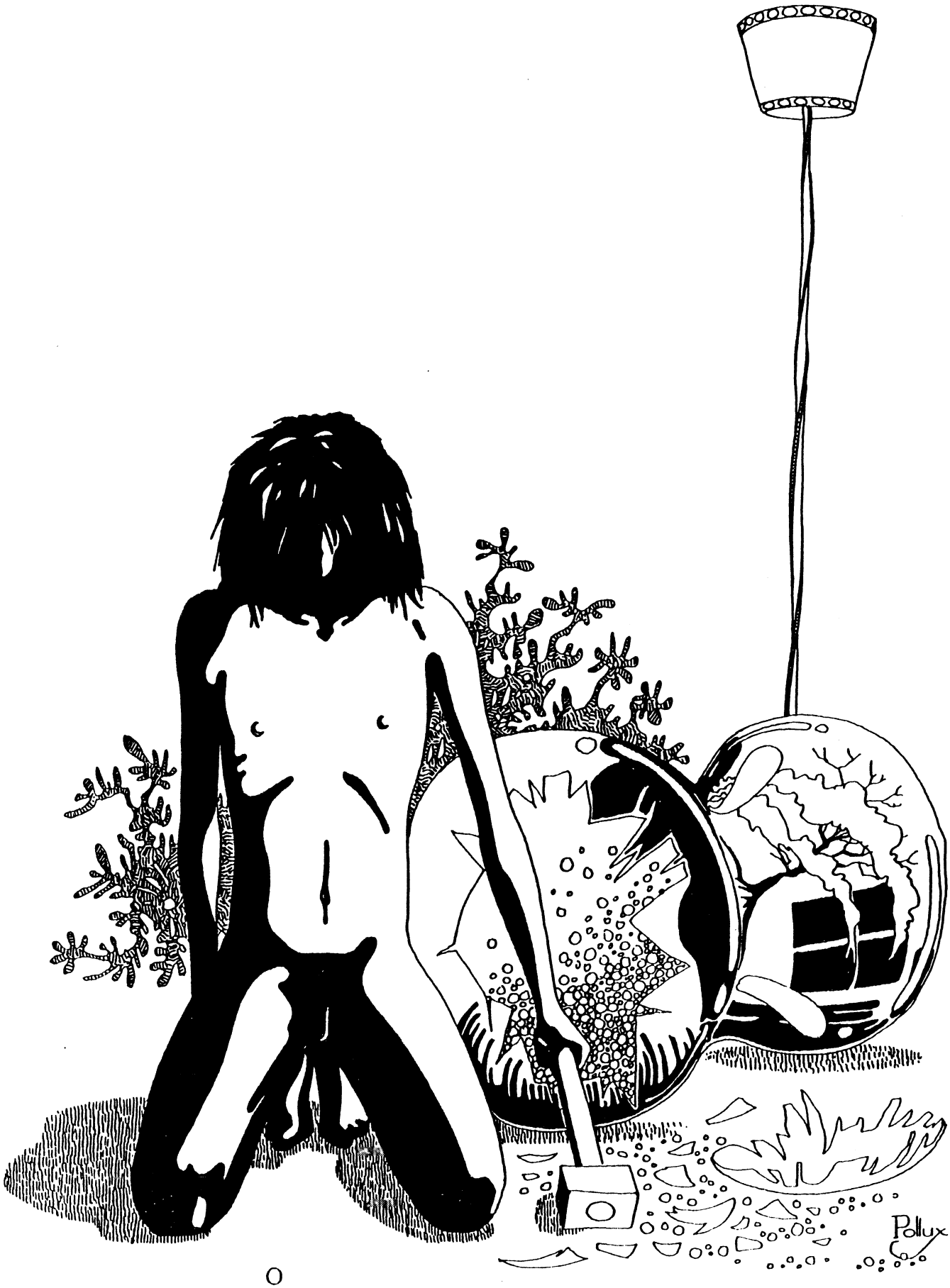
M. Daniel ETIEMBLE, Professeur à l'Université Pierre et Marie Curie à Paris, et M. Christian LANDRAULT, Chargé de Recherche au C.N.R.S., qui ont accepté d'être rapporteurs et membres du jury de cette thèse, et m'ont aidé par leurs conseils ;

M. Christian MASSON, Chef du Service Développements Avancés CAO VLSI Groupe à BULL, qui a accepté d'être membre du jury de cette thèse ;

Tout le personnel du Laboratoire TIM3, pour son amitié.

Le signe &, outre la juxtaposition qu'il suggère, possède une harmonie de courbes, une retenue dans son louvoiement, qui lui confèrent un certain pouvoir désarmant, et préparent le lecteur à la magie de la surprise.

Roland Cloutier
"Dissertations sémiologiques".
Ed. Clavirisus.


Pollux

à Catherine

INTRODUCTION

INTRODUCTION

La vérification de la conception d'un circuit VLSI ("Very Large Scale Integration") est capitale et englobe plusieurs disciplines. Dans ce document, le terme de vérification est utilisé pour désigner la vérification de la conception et la vérification de la fabrication. Seule la simulation est toutefois concernée par le premier type de vérification, tandis que la vérification de la fabrication est considérée comme accomplie par un test de fin de fabrication "hors-ligne"[1].

Un circuit est simulé avant d'être fabriqué, pour que son fonctionnement soit vérifié (vérification de la conception). Pour cela, un programme utilisant comme base de données une représentation virtuelle du circuit, où sont présentes toutes les caractéristiques nécessaires - cette représentation est appelée un modèle -, simule son fonctionnement et vérifie que les divers résultats sont conformes au cahier des charges initial. Le test hors-ligne du circuit intervient après la fabrication, et est utilisé pour déterminer les circuits défectueux. Le circuit dans ce cas n'est plus modélisé : le test est appliqué sur le circuit réel. Le test est limité par les possibilités d'accès au circuit[2] : un circuit VLSI devenant de plus en plus petit, on ne peut observer les résultats n'importe où, comme cela est possible lors de la simulation de circuits dont le niveau de description est assez fin. Ainsi, à moins de prévoir des zones spéciales où une pointe de test trouvera sa place, seuls les plots de sortie du circuit (pattes) peuvent être examinées sans dommage. Il est par conséquent indispensable que l'état de ces sorties reflète le fonctionnement interne du circuit. Notamment, l'effet sur les sorties d'une panne éventuelle à l'intérieur du circuit doit être connu, et l'état des sorties doit être différent selon qu'il y a une panne ou non. La génération de vecteurs de test englobe généralement un type spécial de simulation, appelée simulation de pannes. Celle-ci permet de connaître l'effet de l'injection d'une panne dans un circuit, en particulier sur ses sorties. Associée à un générateur de vecteurs de test, elle est utilisée pour déterminer toutes les pannes détectables par un vecteur de test particulier.

(1) les autres types de test sont le test "en-ligne" qui concerne la détection des défaillances d'un circuit au cours de son utilisation, et le test de mise au point ("debugging").

(2) il existe d'autres techniques, pour la mise au point de prototypes, qui permettent un bien meilleur accès au circuit : par exemple, l'utilisation d'un faisceau d'électrons permet d'accéder à un grand nombre des nœuds internes du circuit.

*
* *

La Conception Assistée par Ordinateur est destinée à devenir de plus en plus automatisée par l'utilisation de systèmes complets d'outils, qui permettront de dessiner et de vérifier un circuit de manière transparente. De manière idéale, le concepteur spécifiera un circuit au niveau fonctionnel, et le système génèrera automatiquement le dessin des masques et vérifiera de façon descendante la description à chacun des niveaux (niveau fonctionnel, niveau logique, niveau des transistors, niveau des masques). Toutefois, ces outils n'étant pas opérationnels aujourd'hui, la vérification se fait généralement par des extractions successives d'un niveau à l'autre, depuis le niveau des masques jusqu'au niveau fonctionnel. Les outils de vérification, dans un tel environnement, doivent être parfaitement adaptés au type du circuit. En outre, les niveaux de description étant très différents entre eux, il est nécessaire de prévoir des outils spécialisés par niveau : bien qu'il existe des outils de vérification "multi-niveaux" qui utilisent le même modèle pour tous les nivcaux de description, certains niveaux, à cause de leur caractère spécifique, demandent un modèle spécialisé pour que la vérification soit concluante.

En particulier, parmi tous les niveaux de description, le niveau des transistors, appelé dans la suite niveau "interrupteur" ("switch level"), joue un rôle prépondérant. En effet, la description d'un circuit à ce niveau est suffisamment précise pour autoriser une simulation très efficace, si le modèle est bien construit. Un circuit décrit au niveau "switch" se présente sous la forme d'un réseau de transistors (interrupteurs) qui sont reliés entre eux par des lignes équipotentielles (nœuds). La spécification plus ou moins fine des interrupteurs et des lignes peut varier d'une description à l'autre (par exemple, le programme de simulation SPICE demande des caractéristiques technologiques et structurales très complètes), mais la structure essentielle reste inchangée. La structure particulière de ce type de réseau induit un traitement approprié, qui doit s'adapter au caractère bi-directionnel des interrupteurs : une porte logique peut être unidirectionnelle, un interrupteur en tant que tel ne l'est jamais. La principale conséquence de ce caractère est que le sens des courants électriques qui traversent un réseau d'interrupteurs est a priori inconnu ; or la modélisation d'un tel réseau, pour être correcte, doit pouvoir retracer la propagation d'un signal. L'ignorance des courants, inhérente à la description, doit conduire dans la modélisation à une méthode de calcul spéciale. En effet, le sens des courants peut être déterminé, mais cela oblige toujours à pré-traiter le réseau de transistors et parfois à le transformer en un réseau de niveau plus élevé, donc "dénaturé" par rapport au réseau initial. Mais la solution la plus "pure" consiste à accepter cette ignorance et à créer des outils permettant de la contourner. Ces derniers, en exprimant l'isolation d'une ligne équipotentielle ou au contraire ses connexions avec les alimentations, permettent de définir une "hiérarchie" entre une ligne et ses voisines (via des interrupteurs), de façon à déterminer implicitement le sens de propagation des signaux.

La précision du niveau interrupteur autorise l'utilisation d'une algèbre d'états plus riche que l'algèbre booléenne du niveau logique. Bien plus, les résultats de la simulation seront intéressants à condition que l'algèbre tire parti des possibilités et aussi des obligations de ce niveau. Aussi l'algèbre doit-elle comporter un assez grand nombre d'états, modélisant les tensions et les intensités des signaux : c'est la raison pour laquelle les qualificatifs "multivalué" et "switch" sont quasi inséparables. On utilise ainsi des couples (valeur, force) où la valeur représente la tension et la force l'intensité du signal. Les valeurs et les forces employées doivent s'inscrire dans une définition logique : il est important de comprendre pourquoi telle valeur ou telle force est présente dans un modèle, ce qui n'est pas toujours le cas dans les simulateurs existants.

*
* *

L'objectif du travail exposé dans ces pages est de spécifier un système de simulation, de simulation de pannes et de génération de vecteurs de test pour des circuits anarchiques de petites tailles (jusqu'à dix mille transistors), qui soit mieux adapté aux contingences d'un réseau d'interrupteurs que les systèmes déjà décrits. De fait, il semble utile de décrire un outil de vérification utilisant un environnement "switch" multivalué, qui soit réaliste et surtout entièrement justifié par le type du réseau. Le premier but de ce travail est précisément d'exprimer ces contingences afin de définir logiquement les états et l'algorithme de calcul qui détermine la propagation des signaux. En ce qui concerne la simulation, les spécifications se restreignent à l'évaluation des signaux, et laissent de côté ses aspects informatiques, tels que la gestion des événements. De même, la gestion des listes de pannes et leur couverture ne sont pas abordées dans les spécifications de la simulation de pannes. Enfin, les résultats concernant la génération des vecteurs de test ne peuvent être considérés que comme une étude préliminaire.

*
* *

Le premier chapitre décrit la construction de certaines algèbres multivaluées en termes d'extensions mathématiques d'une algèbre booléenne, puis expose un certain nombre d'algèbres utilisées actuellement. La première partie explique la théorie de l'extension des algèbres utilisée pour créer des algèbres multivaluées. Les deux extensions proposées sont applicables directement à la simulation multivaluée pour créer des variables indéterminées et des variables temporelles. La première extension crée des variables $x = \{x_1, \ldots, x_n\}$, qui peuvent prendre une valeur quelconque dans l'ensemble des x_i (modélisation des indéterminées) ; la deuxième extension crée des n-uplets $x = (x_1, \ldots, x_n)$, qui prennent chronologiquement toutes les valeurs x_i (modélisation des aléas). La deuxième partie décrit l'utilisation des treillis dans la simulation multivaluée, qui s'appuie sur une de leurs

propriétés (chaque paire d'éléments d'un treillis possède une borne supérieure), et décrit l'opérateur # associé. La troisième partie expose les algèbres d'états utilisées par plusieurs simulateurs existants. Celles-ci sont présentées sous forme de tableaux, afin que leurs points communs soient visibles directement. Après une description plus précise des simulateurs, la légitimité de leurs algèbres est discutée.

*
* *

Le deuxième chapitre présente la construction d'un modèle adapté à la simulation "switch" multivaluée et utilisant une algèbre d'états et un algorithme de calcul justifiés. Dans les deux premières parties, le domaine d'utilisation d'un simulateur multivalué et le principe de la simulation des réseaux d'interrupteurs sont rapidement étudiés de manière à définir quelques critères de choix et à déterminer les outils de base nécessaires. Dans la troisième partie, l'algèbre d'états utilisée par le simulateur est construite en fonction des besoins spécifiques de la modélisation des circuits CMOS et NMOS. Cette algèbre est associée à un opérateur de calcul, noté &, destiné à déterminer les états de part et d'autre d'un transistor. Cet opérateur est défini dans la quatrième partie sous forme de tableaux. Dans la cinquième partie, l'adjonction d'un modèle temporel au système est étudiée : quatre solutions sont exposées, correspondant à différents degrés de précision dans les calculs. La sixième partie décrit sommairement l'algorithme de calcul. La septième partie expose la modélisation de certains dispositifs spéciaux.

*
* *

Le troisième chapitre est consacré à l'extension du modèle à la simulation des pannes. La première partie illustre l'intérêt de simuler les pannes au niveau interrupteur. En effet, la précision de ce niveau permet de traiter des pannes inaccessibles au niveau logique. La deuxième partie présente trois techniques de modélisation pouvant être utilisées pour la simulation de pannes. L'une d'elles, qui consiste en l'enrichissement du premier modèle multivalué, est décrite de manière approfondie. La troisième partie expose les pannes qui peuvent être simulées et leur modélisation.

*
* *

Le quatrième chapitre décrit l'utilisation du modèle de simulation de pannes pour la génération automatique de vecteurs de test. Dans la première partie sont spécifiés les outils qui sont ajoutés au modèle, ainsi que la sensibilisation des chemins du siège de la panne vers les sorties et vers les entrées. La seconde partie expose la mise en évidence de l'anomalie pour chacune des pannes envisagées.

*
* *

Le système de vérification décrit dans les pages qui suivent est destiné à faire partie d'un ensemble complet d'outils de conception, dans lequel la description des circuits est hiérarchique. Cette hiérarchie n'est pas abordée ici, car l'accent est mis sur la vérification de petits circuits. Mais elle peut être directement inclue dans les spécifications car ces circuits peuvent être considérés comme étant des cellules au sein d'une hiérarchie. Le seul point à ajouter concerne les appels aux cellules appelées et la remontée vers la cellule appelante, ce qui n'est pas du tout spécifique du traitement d'un réseau d'interrupteurs.

La précision obtenue dans la vérification telle qu'elle est spécifiée est intermédiaire entre celle d'une vérification logique et celle d'une vérification électrique. Ainsi, une vérification multivaluée au niveau interrupteur pourra être accomplie pour affiner des résultats du niveau logique, tandis qu'une vérification électrique permettra par exemple de résoudre des indéterminations mises en lumière par la vérification multivaluée.

CHAPITRE 1

LES ALGEBRES MULTIVALUEES

1 Les algèbres multivaluées

Le travail présenté dans ce document utilise des algèbres multivaluées, c'est-à-dire des algèbres comprenant plusieurs éléments, et notamment d'autres que 0, 1, et X. Avant que celles-ci ne soient définies, certaines règles d'extensions permettant de construire de telles algèbres à partir d'une algèbre simple doivent être décrites afin que le concept d'algèbre multivaluée soit bien cerné. Ensuite, les algèbres utilisées par quelques simulateurs existants seront sommairement décrites.

1, 1 Extensions d'algèbres logiques

Les modèles utilisés en simulation emploient des ensembles de variables qui, ainsi qu'il sera vu au chapitre [1, 3], semblent n'avoir aucun rapport entre eux, du moins quant au choix des différentes valeurs de signal. En fait, ces ensembles sont créés pour la plupart d'après des règles mathématiques d'extension d'algèbres qui sont exposées ici [HAY84].

On définit une algèbre logique comme un couple (A,Φ), où $A = \{a_1, \ldots, a_n\}$ est un ensemble de valeurs logiques (ensemble-Base), et $\Phi = \{\phi_1, \ldots, \phi_m\}$ est un ensemble d'opérateurs ϕ_i de A^{m_i} vers A.

Exemple :

l'algèbre booléenne est définie par :

$$(B_2 = \{0, 1\}, \Phi = \{ET, OU, NON\}).$$

L'utilisation en simulation d'une logique multivaluée consiste dans certains cas en l'emploi de l'extension d'une algèbre de base. Cette extension concerne l'ensemble A d'une part, et l'ensemble Φ d'autre part.

1, 11 Extensions de l'ensemble A

On utilise en simulation deux extensions possibles :

1, 111 Extension "indéterminée"

A' est inclus dans 2^A (ensemble des sous-ensembles de A) :

$A' = \{a'_i\} / a'_i = \{a_{i1}, \ldots, a_{ik}\}$ où a_{ij} appartient à A pour $j = 1, \ldots, k$.

A' est appelé un ensemble d'indéterminées (ensemble-I) et a'_i une valeur indéterminée (valeur-I). L'interprétation naturelle de la valeur logique a'_i est de la considérer comme la valeur d'un signal x incertain ou inconnu pouvant prendre n'importe quelle valeur dans l'ensemble $\{a_{i1}, \ldots, a_{ik}\}$. On ne connaît pas la valeur de x, mais on sait qu'elle appartient à a'_i.

Exemple :

La valeur indéterminée X employée dans certains simulateurs est la valeur-I {0,1} de B_2. Dans ce cas, X prend soit la valeur 0, soit la valeur 1 ; ce n'est donc pas un état intermédiaire.

1, 112 Extension "temporelle"

A' est inclus dans A^n :

$$A' = \{a'_i\} / a'_i = (a_{i1}, \ldots, a_{in}),\ a_{ij} \text{ élément de A.}$$

A' est appelé un ensemble produit cartésien (ensemble-P) et a'_i une valeur produit cartésien (valeur-P). Si la valeur-P a'_i est affectée à la variable x, alors chaque a_{ij} représente l'état de x sous l'une des n conditions que l'on spécifie.

Exemple :

a'_i représente l'état de x pour n instants successifs $t_1, \ldots, t_n$. Un signal montant de 0 à 1 est symbolisé par (0, 1), un signal descendant par (1, 0).

Cette extension est surtout utilisée dans la modélisation des transitions. Dans ce cas, contrairement à l'extension précédente, le signal prend successivement toutes les n valeurs a_{ij}. On connaît donc toujours sa valeur, il n'y a pas d'indétermination.

1, 12 Extensions de l'ensemble Φ

Après avoir défini les extensions possibles de l'ensemble A, on définit celles de l'ensemble Φ. Ces extensions ne changent pas les éléments de l'ensemble Φ mais sont le fait d'une application possible sur les nouveaux ensembles A'.

1, 121 Extension sur les ensembles-I

pour A' inclus dans 2^A, on a :

$$\begin{aligned}\phi(a'_1, \ldots, a'_m) = \{&\phi(a_{11}, a_{21}, \ldots, a_{m1}),\\ &\phi(a_{12}, a_{21}, \ldots, a_{m1}),\\ &\phi(a_{13}, a_{21}, \ldots, a_{m1}),\\ &\ldots\\ &\phi(a_{1n_1}, a_{21}, \ldots, a_{m1}),\\ &\phi(a_{11}, a_{22}, \ldots, a_{m1}),\\ &\phi(a_{12}, a_{22}, \ldots, a_{m1}),\\ &\ldots\\ &\phi(a_{1n_1}, a_{2n_2}, \ldots, a_{mn_m})\},\end{aligned}$$

où $a'_i = \{a_{i1}, \ldots, a_{in_i}\}$ est élément de A' et a_{ij} est élément de A pour $1 \leq i \leq m$ et $1 \leq j \leq n_i$. Le membre de droite est un ensemble de $\prod n_i$ valeurs ϕ ($1 \leq i \leq m$) dont les opérandes sont obtenus en combinant les éléments de chacun des a'_i, de toutes les manières possibles.

Exemple :

$\phi((0, 1), (0, 1, I)) = (\phi(0, 0), \phi(0, 1), \phi(0, I), \phi(1, 0), \phi(1, 1), \phi(1, I))$,
où I est une valeur indéterminée.

En utilisant l'indéterminée X définie plus haut (X = (0, 1)), on a :
ET(0, X) = ET((0), (0, 1)) = (ET(0, 0), ET(0, 1)) = (0, 0) = 0.
NON(X) = (NON(0), NON(1)) = (1, 0) = X.

1, 122 Extension sur les ensembles-P

pour A' inclus dans A^n, on a :

$$\phi(a'_1, \ldots, a'_m) = (\phi(a_{11}, \ldots, a_{m1}), \phi(a_{12}, \ldots, a_{m2}), \ldots, \phi(a_{1n}, \ldots, a_{mn})),$$

où $a'_i = (a_{i1}, \ldots, a_{in})$ est élément de A'.

Exemple :

NON(0, 1) = (NON(0), NON(1)) = (1, 0) : un signal montant se transforme en signal descendant.

1, 13 Utilisations des extensions -I et -P

Les programmes de simulation utilisent souvent des ensembles combinés -P et -I, qui sont généralement construits au fur et à mesure des besoins pour analyser une classe limitée de types de signal. Les modèles traitant plus spécialement les phénomènes temporels utilisent surtout la deuxième extension (A^n).

1, 131 Définition des états indéterminés

On trouve un bon exemple d'utilisation des deux extensions dans la définition des états d'indétermination. En effet, il est facile de trouver la description formelle des variables employées dans les simulateurs. Deux sortes d'indéterminées sont employées, correspondant respectivement à la première et à la deuxième extension (on ajoute généralement dans les simulateurs une troisième valeur indéterminée "totale", résultant d'un conflit grave et pouvant couvrir tout l'ensemble analogique [0 volt, 5 volts].

1, 1311 Indéterminée restreinte à un ensemble

$I = a'_i$: I prend l'une des valeurs de l'ensemble a'_i. Par exemple, la valeur indéterminée X = {0, 1} fait partie de cette classe : X prend soit la valeur 0, soit la valeur 1, mais on ne sait pas laquelle des deux. Cette valeur X est en général une valeur d'initialisation ou une valeur indifférente, généralement introduite en entrée de circuit.

1, 1312 Indéterminée de transition

I = valeur de transition. Elle est utilisée dans le cas où une variable change d'état. L'emploi d'une telle variable est justifié lorsqu'on doit considérer le temps de transition du signal. Ce signal change en fait de façon discrète, en général de 0 à 1 ou de 1 à 0. Dans ce cas, cette valeur de transition peut être modélisée ainsi : I = {(0, 1), (1, 0)}.

1, 132 Analyse des phénomènes transitoires

Un aléa est présent dans un circuit si certains changements en entrée provoquent un signal erroné mais transitoire (pic) en sortie. Ces aléas résultent de combinaisons défavorables entre les temps de transition des signaux d'entrée et les délais de propagation dans le dispositif. Ils obligent le concepteur à effectuer une simulation temporelle détaillée. Les aléas peuvent être détectés par l'emploi d'une logique multivaluée qui inclut des valeurs logiques explicites pour les signaux de transition. Le nombre et la représentation de ces valeurs sont variables d'un simulateur à l'autre[1].

(1) voir la table des états du chapitre [1, 31].

1, 1321 Aléas statiques

Un aléa statique se produit dans un signal sans transition. On utilise ici la deuxième extension (A' inclus dans A^3), où un signal est représenté par (a_1, a_2, a_3), chacun des a_i étant l'état du signal à trois instants différents. On représente un signal pouvant contenir un aléa statique par la variable $x = (a, X, a) = \{(a, a, a), (a, \neg a, a)\}$. Le premier triplet est le signal correct, le second est le signal erroné.

On obtient sur l'ensemble {0, 1} l'ensemble A_6 :

$$A_6 = \{(0, 0, 0), (1, 1, 1), (1, X, 1), (0, X, 0), (0, X, 1), (1, X, 0)\}.$$

(0, 0, 0) et (1, 1, 1) représentent les signaux normaux,
(0, X, 0) et (1, X, 1) représentent les signaux dans lesquels un aléa statique est présent,
(0, X, 1) et (1, X, 0) représentent les signaux de transitions.

Exemple :

une transition de 0 à 1 est représentée par

$$x = (0, X, 1) = \{(0, 0, 1), (0, 1, 1)\},$$

où (0, 0, 1) représente une transition lente et (0, 1, 1) une transition rapide.

L'ensemble A_6 est inclus dans $(B_2 x 2^B 2 x B_2)$ et est fermé pour toutes les opérations définies dans B_2.

1, 1322 Aléas dynamiques

Un aléa dynamique se produit dans une transition. Un pic à 0 dans une transition 0-1 est représenté par le quadruplet (0, 1, 0, 1). Un pic à 1 dans une transition 1-0 est représenté par (1, 0, 1, 0).

Une transition 0-1 normale est représentée par l'ensemble :

$$\{(0, 0, 0, 1), (0, 0, 1, 1), (0, 1, 1, 1)\}.$$

Une transition 1-0 normale est représentée par l'ensemble :

$$\{(1, 1, 1, 0), (1, 1, 0, 0), (1, 0, 0, 0)\}.$$

Une transition 0-1 pouvant comporter un aléa est représentée par la variable :

$$(0, T, 1),$$

où T est l'ensemble des couples (0, 0), (0, 1), (1, 0), (1, 1), c'est-à-dire B_2^2.

De même, (1, T, 0) représente une transition 1-0 pouvant comporter un aléa.

On obtient ainsi l'ensemble Λ_8 pour représenter les aléas dynamiques :

$$\Lambda_8 = \{(0,P,0),(1,Q,1),(0,R,1),(1,S,0),(0,T,0),(0,T,1),(1,T,0),(1,T,1)\},$$

où P, Q, R, S, T sont des ensembles de transitions : P = {(0, 0)}, Q = {(1, 1)}, R = {(0, 0), (0, 1), (1, 1)} (transition 0-1), S = {(1, 1), (1, 0), (0, 0)} (transition 1-0), T = B_2^2.

1, 2 Treillis hiérarchiques - Forces logiques

1, 21 Définition des variables-couples

Les ensembles de valeurs ci-dessus sont appliqués aux systèmes dans lesquels un seul paramètre a une signification logique (par exemple la tension). Il est apparu depuis peu qu'il était fort utile de prendre en compte la "force logique" d'un nœud, en plus de sa valeur logique. Cette force modélise le courant que peut délivrer le nœud. Une variable logique est donc représentée par un couple (v, f) (valeur, force). L'ensemble V_m des valeurs v peut être défini à l'aide des extensions du chapitre [1, 1], et l'ensemble des forces est de la forme F_n = {1, 2, ... , n}.

1, 22 Représentation par treillis hiérarchique

On représente ces variables (v, f) à l'aide d'un treillis. Soit par exemple l'ensemble de valeurs V_4 = {0, 1, I, Z}, où Z est la valeur haute impédance et I la valeur indéterminée totale[1]. La hiérarchie entre ces différentes valeurs, à un même niveau de force i (i élément de F_n), est donnée par le treillis T_i de la figure 1-1.

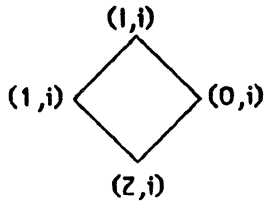


figure 1-1 : treillis de V_4 au niveau de force i.

Cela signifie que la superposition sur un nœud des valeurs 0 et 1 à un même niveau de force crée un "problème" (en l'occurence, le résultat est (I, i)). Ce phénomène modélise ainsi un court-circuit entre l'alimentation Vdd et la masse Gnd. Ce court-circuit, suivant le niveau de force des signaux 0 et 1 concernés, peut être destructif ou être anihilé par un signal de force plus grande. Dans tous les cas, la valeur est l'indéterminée I.

(1) voir la remarque du chapitre [1, 131].

Pour obtenir le treilllis T_n associé à V_4xF_n, qui représente tous les états logiques disponibles et leur niveau d'influence, on concatène les treillis T_i [fig. 1-2].

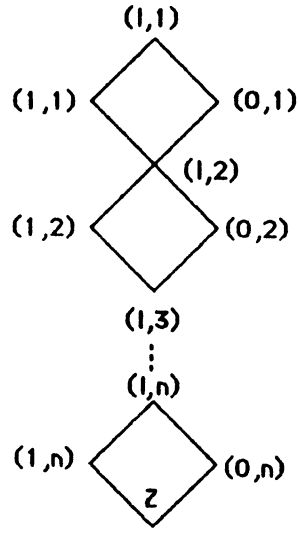


figure 1-2 : treillis complet de V_4xF_n.

Pour cette concaténation, on pose :

(Z, i) = (I, i+1) et Z = (Z, n) (l'état le plus fort est (I, 1).

1, 23 Adjonction de l'opérateur

On utilise ces treillis pour calculer l'état final d'un nœud où arrivent plusieurs signaux d'états différents. Pour cela, on définit un opérateur permettant de résoudre les conflits de superposition. Il s'agit de l'opérateur borne supérieure # (least-upperbound).

Soient $(v_1, f_1), \ldots, (v_k, f_k)$ k signaux arrivant sur un nœud. L'état final de celui-ci est :

$$(v, f) = \#((v_1, f_1), \ldots, (v_k, f_k)),$$

c'est-à-dire le plus faible état de T_n tel que $(v_i, f_i) \leq (v, f)$, $i = 1, \ldots, k$.

Exemple :

soit le treillis T_2 donné par la figure 1-3. On a alors :

#((0, 1), (0, 2)) = (0, 1), #((0, 1), (1, 1)) = (I, 1), #((I, 2), (1, 2)) = (I, 2).

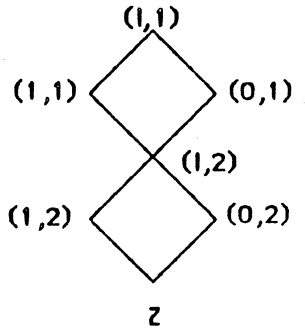


figure 1-3 : treillis T_2.

1, 3 Présentation de modèles de simulateurs

Dans cette partie, plusieurs simulateurs sont présentés sous trois aspects différents : d'abord une suite de tableaux rend compte des états utilisés, puis une comparaison effectuée entre ces états fait ressortir leurs analogies et leur rapport avec les extensions définies dans le chapitre [1, 1]. Enfin, les simulateurs exposés sont décrits de manière plus approfondie.

1, 31 Tables des états de différents simulateurs

Certains simulateurs emploient des couples (valeur, force), qui sont donnés ici en deux tableaux juxtaposés. Le nombre d'états disponibles dans ce cas est le produit du nombre de valeurs par le nombre de forces de signal.

Simulateur LOGDET : 8 états [ALM84]

ETAT	DESCRIPTION
0	
1	
E	erreur
R	0 à 1
F	1 à 0
Z	haute impédance
H	transition vers Z
L	transition depuis Z

Système CADOC : 7 états [AMB83]

ETAT	DESCRIPTION
0	
1	
X	indifférent
Z	haute impédance
U	inconnue
R	front montant
F	front descendant

Simulateur MURPHY : 25 états [BAN83]

VALEUR	FORCE
hard-1	entrée
hard-0	fort
soft-1	faible
soft-0	charge-stockage
X	erroné

Simulateur de M. A. Breuer & R. L. Harrison : 8 états [BRE74]

ETAT	DESCRIPTION
00hp	aléa statique sur 0
11hp	aléa statique sur 1
01hp	aléa dynamique en montée
10hp	aléa dynamique en descente
00hf	signal statique 0 sans aléa
11hf	signal statique 1 sans aléa
01hf	0 à 1 sans aléa
10hf	1 à 0 sans aléa

Simulateur MOSSIM II : 3 états [BRY84]

ETAT	DESCRIPTION
0	
1	
X	0 à 1 ou 1 à 0

Simulateur hiérarchique de M. C. Chen & C. A. Mead : 4 états [CHE83]

VALEUR	DESCRIPTION
0	
1	
X	intermédiaire
⊥	inconnu

Simulateur THEMIS : 10 états [DOS84]

VALEUR	FORCE
0	conducteur
1	résistif
X	capacitif
Z	

Simulateur de E. B. Eichelberger : 3 états [EIC65]

VALEUR
0
1
½ (transition)

Simulateur de G. Fantauzzi : 9 états [FAN74]

ETAT	DESCRIPTION
↑	0 à 1
↓	1 à 0
T	1
F	0
/	0 à 1 avec aléa
\	1 à 0 avec aléa
∩	0 à 0 avec aléa
U	1 à 1 avec aléa
*	ambiguë

Système SMILE : 16 états [GON84]

VALEUR	FORCE
0	entrée
	actif
1	résistif
	flottant
U	indéterminé
Z	

Système DAX : 2, 4, ou 6 états [HOD84]

ETAT	DESCRIPTION	
0		(niveau comportemental)
1		
H	1	(niveau fonctionnel)
L	0	
X	indéterminée	
Z	haute impédance	
H		(niveau transistor)
L		
X		
Z		
h	H ou Z	
l	L ou Z	
+ 7 forces		

Simulateur CSASIM : 7 états [KAW84]

ETAT	DESCRIPTION
0	0 fort
1	1 fort
U	indéterminée fort
$\widetilde{0}$	0 faible
$\widetilde{1}$	1 faible
$\widetilde{U}$	indéterminée faible
Z	haute impédance

Système DECSIM : 2 ou 4 états suivant le niveau [KEA84]

ETAT	
0 1	(haut niveau)
0 1 U Z	(niveau transistor)

Simulateur de D. W. Lewis : 5 états [LEW72]

ETAT	DESCRIPTION
0	
1	
0/1	0 à 1
1/0	1 à 0
½	transition avec aléa

Simulateur de E. J. McCluskey : 3 états [McC81]

VALEUR
0
1
U

Simulateur de V. Ramachandran : 9 états [RAM83]

VALEUR	FORCE
0	i (entrée)
1	p (charge)
U	w (faible)

. **Simulateur de P. Stevens et G. Arnout : 15 états [STE83]**

VALEUR	FORCE
0	E externe
	D conducteur
1	R résistif
	L grande impédance
X	Z haute impédance

Système FIDEL : 5 types [TAH84]

TYPE DE SIGNAL
entrée
sortie
entrée/sortie (bidir.)
état
entier

Simulateur TEGAS : 5 états [THO75]

ETAT	DESCRIPTION
0	0 à 0 (statique)
1	1 à 1 (statique)
U	0 à 1
D	1 à 0
E	transition avec aléa

Simulateur de J. Watanabe & al. : 7 états [WAT80]

ETAT	DESCRIPTION
1	1 statique
0	0 statique
x	inconnue statique
$\tilde{1}$	1 dynamique
$\tilde{0}$	0 dynamique
z	inconnue dynamique
E	erroné

1, 32 Commentaires sur les modèles existants

La grande diversité de dénominations dans les algèbres d'états, que l'on peut constater dans cette présentation en tableaux, ne s'accompagne pas forcément d'une incohérence entre les solutions. En particulier, les analogies entre les états décrivant les transitions avec ou sans aléas sont évidentes [fig. 1-4]. On peut noter que les valeurs des simulateurs [LEW72] et [THO75] sont en fait les ensembles P, Q, R, S, T. Par exemple, l'ensemble T décrit effectivement une transition (de 0 à 1 ou de 1 à 0) contenant un pic, et symbolisée dans ces deux simulateurs par 1/2 et E. La définition d'un quadruplet est nécessaire pour préciser de quelle transition il s'agit, et donc pour déterminer la qualité de l'aléa - statique ou dynamique.

description	codage	BRE74	BRY84	EIC65	FAN74	LEW72	THO75
0 statique	(0,P,0)	00hf	0	0	F	0	0
1 statique	(1,Q,1)	11hf	1	1	T	1	1
0 à 1	(0,R,1)	01hf	X	1/2	↑	0/1	U
1 à 0	(1,S,0)	10hf	X	1/2	↓	1/0	D
aléa stat. 0	(0,T,0)	00hp			∩		
aléa stat. 1	(1,T,1)	11hp			U		
aléa dyn. 0	(0,T,1)	01hp			/	1/2	E
aléa dyn. 1	(1,T,0)	10hp			\	1/2	E

figure 1-4 : comparaison entre les états utilisés.

1, 33 Description détaillée des simulateurs présentés

Système LOGDET [ALM84]

Ce système est une amélioration de la version précédente qui ne traitait pas les signaux bi-directionnels. Les blocs bi-directionnels sont ici remplacés par des blocs unidirectionnels connectés de façon adéquate.

Le calcul des états se fait en deux fois : d'abord le recensement des événements en entrée donne lieu au calcul des sorties, puis un délai est affecté à chaque transition.

Une distinction est faite entre les transitions "contiguës" et "non contiguës" (par exemple, la transition 0->Z est décomposée en 0->H->Z, et un délai est affecté à chacune). Les transitions illégales (F->1) sont transformées en message d'erreur (->E) avec un délai minimum t_m. Le délai affecté ne dépend que de la transition observée en sortie. Il s'agit donc plutôt d'un délai de commutation plus que de propagation.

Système CADOC [AMB83]

Le système CADOC est un outil de conception possédant un ensemble d'outils de simulation et de spécification fonctionnelle, à différents niveaux. Afin que les circuits complexes puissent être traités, le système permet le partitionnement hiérarchique (les différentes cellules peuvent être imbriquées) en plusieurs sous-circuits, chacun étant considéré comme une ressource fonctionnelle et pouvant être modélisé à tout niveau (du niveau algorithmique temporisé au niveau structural).

Les variables utilisées dans CADOC sont des échéanciers représentant une suite de couples (valeur, date) permettant un contrôle temporel fin. Les types primitifs sont "entiers" (0, 1, X, Z, U) et "fronts" (R, F). Les opérations et fonctions disponibles sont les opérations logiques et arithmétiques, des primitives de contrôle temporel qui fournissent les dates absolues des événements, et d'autres qui vérifient la stabilité d'une variable dans le temps. On dispose également de la fonction "change", qui détecte le changement d'état d'une variable, et permet la spécification des ressources combinatoires.

Exemple de modélisation d'une porte OU [fig 5]. Les traits horizontaux représentent les transitions : C n'est activé que si A ou B change.

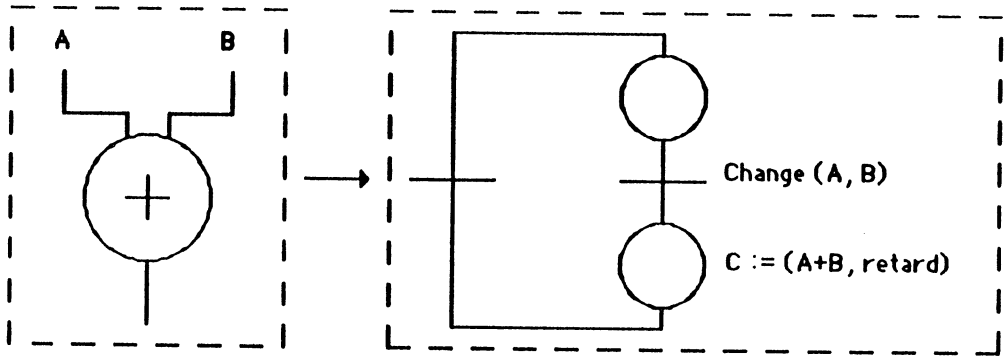


figure 1-5 : modélisation d'une porte OU.

La simulation d'une ressource utilise un générateur d'environnement effectuant la saisie des stimuli de simulation et rendant compte de la connexité des ressources.

Simulateur logique MURPHY [BAN83]

Ce simulateur a été conçu pour modéliser les effets des défauts physiques. Il peut simuler les courts-circuits entre deux terminaux d'un transistor, les grilles flottantes, les faux-contacts entre source et drain, les faux-contacts de toutes sortes. L'état d'un nœud est décrit par un couple (a, b), où a est la condition du nœud et b sa valeur logique. Les niveaux logiques utilisés "soft_1" et "soft_0" représentent les signaux dégradés en tension.

L'algorithme est fondé sur la recherche d'un état stationnaire du circuit après chaque changement des entrées ; il utilise la fonction T, calculant le nouvel état des transistors, et la fonction O, qui ordonne le réseau en vue de la boucle suivante. L'état stable est atteint lorsqu'il n'y a plus de changement d'une répétition sur l'autre. Pour alléger la compilation, le circuit est partitionné en groupes ; ceux-ci sont traités dans un ordre donné par un algorithme de classement "orienté-événement" ("event-driven"), en fonction de leurs interdépendances.

Simulateur de M. A. Breuer & al. [BRE74]

Dans cette méthode, l'état d'une ligne est représenté par la paire $a_n b_{n+1}$, dans laquelle a_n élément de l'ensemble {0, 1, x} est l'état de la ligne au temps t_n, et où b_{n+1} élément de l'ensemble {0, 1, x} est celui de la ligne au temps t_{n+1}.

On définit sur une ligne un état d'aléa du temps t_n au temps t_{n+1} qui peut être : hf (hazard free), hp (hazard present) ou hsu (hazard status unknown). Par exemple, une ligne notée 01hp signale qu'un aléa dynamique peut être présent sur cette ligne lors de la transition de 0 à 1.

On définit aussi la condition de nécessité hf, notée *, indiquant qu'il est essentiel pour la fiabilité du dispositif qu'une ligne soit exempte d'aléa. Cette condition * n'est utilisée que pour les entrées des bascules, dans le cas où un aléa pourrait forcer la bascule dans un état stable erroné.

La propagation de ces conditions hf, hp, hsu, et * est faite en fonction des portes traversées, et permet de prévoir les aléas statiques et dynamiques. Cela autorise donc une sélection des séquences de test, celles qui manifestement donneraient lieu à des aléas remettant en cause la fiabilité des résultats de simulation eux-mêmes étant éliminées.

Simulateur MOSSIM II [BRY84]

Le calcul des états est analogue à celui de E. B. Eichelberger : mise à X des entrées changeantes, répétition jusqu'à l'état stable, mise à leur valeur finale de ces mêmes entrées, répétition jusqu'à l'état stable. Tout nœud dont l'état logique final dépend des délais du circuit reste à X, indiquant une erreur possible de délai. Un aléa (statique) sur un nœud est indiquée par la séquence 0X0 ou 1X1.

Le simulateur teste le respect d'une "discipline temporelle" dans la conception, pour laquelle le comportement séquentiel du circuit ne repose pas sur des délais relatifs à travers les éléments logiques ou les connexions, à la condition que les signaux d'horloge soient assez "lents" pour permettre au circuit de se stabiliser après chaque transition. Le circuit simulé est décrit en trois parties (niveau interrupteur) : un modèle réseau représentant le circuit, un modèle formel décrivant le comportement logique du réseau, et un algorithme pour simuler ce comportement. Le modèle réseau représente les nœuds et les interrupteurs, leur état et leurs liaisons. Les nœuds sont soit des nœuds "entrée" (signal fort), soit de "stockage", leur état étant dans ce cas déterminé par l'opération du réseau.

L'état stationnaire du circuit est calculé par un algorithme de relaxation (propagation de vagues) de complexité temporelle linéaire fonction du nombre de transistors. Le modèle utilise une algèbre ternaire T = {0, 1, X}. Il y est défini un opérateur borne supérieure : la borne supérieure d'un ensemble de valeurs ternaires est égale à 1 (ou 0) si et seulement si tous les éléments ont la valeur 1 (ou 0), et est égale à X dans le cas contraire. Cette opération agit donc comme une opération de cohérence, X représentant l'état incohérent.

On définit également l'extension ternaire f^t d'une fonction booléenne $f : B^n \rightarrow T$:

$f^t : T^n \rightarrow T$,

$f^t(a_1, \ldots, a_n) = b.sup \{f(b_1, \ldots, b_n), b_i$ élément de $B=\{0, 1\}, b_i \leq a_i$, pour tout $i\}$.

Cela signifie que lorsque des arguments a_i de f^t sont égaux à X, la fonction génère une valeur booléenne (0 ou 1) si et seulement si elle aurait cette même valeur booléenne avec les a_i mis à 0 ou 1, de toutes les manières possibles (la valeur X représente une valeur incertaine ou ambiguë qui, placée en argument d'une fonction, lui fait générer un X en résultat si et seulement si la sortie est "sensible" à cet argument d'entrée). L'algorithme de simulation ternaire est appliqué aux réseaux de portes logiques, ceux-ci étant mosélisés à l'aide d'extensions ternaires des fonctions représentant chaque porte (détection de rémanence de X à la fin des deux cycles de stabilisation).

Simulateur hiérarchique [CHE83]

Ce simulateur possède deux niveaux de hiérarchie : hiérarchie syntaxique et hiérarchie sémantique. Cette dernière contient des abstractions comportementales des cellules syntaxiques.

Les différents niveaux sémantiques sont les niveaux transistors et nœuds, portes, cellules avec horloge, registres de transfert, ... Une cellule syntaxique, composée d'interconnexions de transistors et de nœuds, peut donner lieu à une cellule sémantique, si une abstraction est désirée. Cette cellule sémantique est utilisée comme un atome dans la hiérarchie syntaxique pour clarifier la spécification et exprimer la localisation de la cellule syntaxique. Cette utilisation peut être faite dans chaque niveau sémantique. Le simulateur n'utilise pas un réseau plat, mais se sert des spécifications hiérarchiques de l'utilisateur. Les cellules sont représentées, dans le langage utilisé par le simulateur (MAINSAIL), par des modules, contenant les spécifications de ses composants, c'est-à-dire son implantation en termes de cellules de bas niveau, et la procédure pour la simuler. On simule une cellule sémantique en simulant tous ses composants jusqu'à obtention d'un état stable. D'une façon générale, les cellules complexes utilisent des simulations récursives de leurs composants, alors que le comportement des cellules simples est simulé directement. Les connexions sont également représentées par des modules contenant des informations sur leur structure et leur type de conduction (unidirectionnelle, bi-directionnelle, ...).

L'algorithme d'obtention de l'état stable appelle toutes les cellules, puis toutes les connexions pour qu'elles transfèrent les données et fournissent des informations sur les interactions entre les cellules connectées ; cette procédure est répétée jusqu'à ce que l'état stable soit atteint.

Simulateur multiniveau THEMIS [DOS84]

Ce simulateur travaille aux niveaux comportemental, registre de transfert, logique et interrupteur. Les primitives du modèle booléen, au lieu d'être construites à partir d'un

ensemble prédéfini de portes (NAND, NOR, ...), sont définies à partir d'équations booléennes données par l'utilisateur, afin d'augmenter la souplesse du système.

THEMIS possède 10 états : 9 couples (v, f) et la valeur haute impédance Z, qui assurent une bonne simulation à tous les niveaux.

Le caractère interactif étant un des buts majeurs de ce simulateur, il était nécessaire de laisser l'accès par l'utilisateur à n'importe quel objet, à tout moment. Ce but est atteint par l'implémentation d'une table permanente de symboles, organisée comme un arbre, qui reflète la hiérarchie du modèle décrivant le circuit. En fait, la simulation interactive est loin d'être indispensable. Les auteurs de THEMIS citent comme inconvénient majeur du différé ("batch"), la trop grande quantité de données et de résultats qui en découlent. Cela a du moins l'avantage de permettre l'étude à loisir des pannes et de leurs effets, alors qu'on doit, avec THEMIS, rester en session et corriger au fur et à mesure des résultats.

Simulateur de E. B. Eichelberger [EIC65]

Ce modèle a été conçu pour détecter les aléas statiques des signaux. Chaque porte est supposée sans délai, mais on associe un délai à chaque entrée, qui est compris dans un intervalle $[0, t_{Max}]$.

L'auteur distingue deux sortes d'aléas statiques :

- **aléa fonctionnel**

Exemple : les fonctions définies par la figure 1-6 peuvent donner lieu à de tels aléas.

z \ xy	00	01	11	10
0	1	0	1	-
1	-	-	-	-

figure 1-6 : fonctions pouvant donner lieu à des aléas fonctionnels.

Pour passer de xyz = 000 (f = 1) à xyz = 110 (f = 1), il se peut, dans le cas où toutes les variables de la fonction ne changent pas en même temps, que la sortie passe par xyz = 010 (f = 0), ce qui provoque un pic à 0 en sortie.

Ce type d'aléa est impossible à éliminer sans modifier la fonction.

- **aléa logique**

L'auteur démontre qu'un réseau (synthèse somme de produits ou produit de sommes) ne contient aucun aléa logique en sortie si et seulement si tous les monômes premiers de la fonction sont inclus dans sa réalisation.

Exemple :

la fonction décrite dans les figures 7 et 8 peut comporter un aléa logique car tous les monômes ne sont pas présents dans sa réalisation.

yz \ wx	00	01	11	10
00	1	1	1	1
01	0	1	1	1
11	0	1	1	0
10	0	1	0	0

figure 1-7 : fonction pouvant comporter un aléa logique.

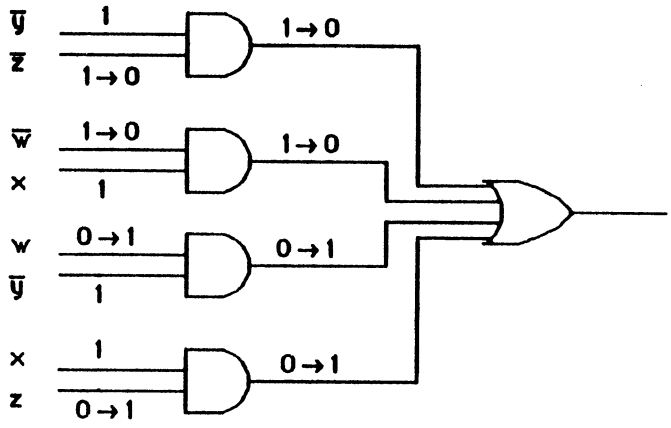


figure 1-8 : réalisation logique de la fonction.

La sortie peut marquer un pic à 0 lors de sa transition 1->1. L'aléa logique présent dans cette réalisation disparaît lorsqu'on ajoute le monôme manquant (x, ¬y).

L'analyse des transitions dans un circuit se fait en deux passages:

- détermination de tous les retours de boucle [fig. 1-9] qui peuvent changer lorsque l'entrée change (utilisation de la valeur 1/2).

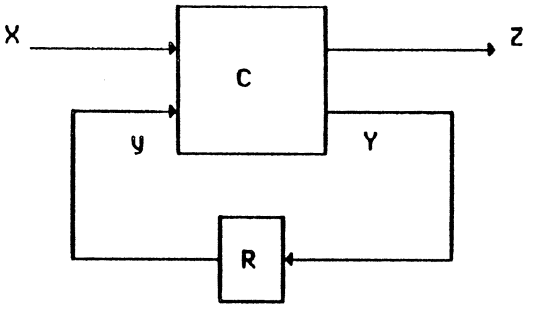


figure 1-9 : traitement des boucles.

Les variables X_i susceptibles de changer sont mises à 1/2. On évalue ensuite les variables Y_i, pour déterminer si certaines ont changé de 1 ou 0 vers 1/2. On change alors les y_i correspondants, puis on répète.

- vérification de la stabilisation de ces signaux à un état déterminé. Les variables qui étaient à 1/2 sont mises à leur valeur finale 0 ou 1, la répétition étant la même que ci-dessus.

Modèle à neuf valeurs de G. Fantauzzi [FAN74]

Ce modèle est conçu pour traiter les aléas statiques et dynamiques pouvant avoir lieu dans les circuits. Il se restreint aux circuits composés de portes simples (NAND, NOR, INVERSEUR, ...) et de bascules RS, et ne comprenant pas de boucles autres que celles contenues dans les bascules.

Système de simulation multiniveau SMILE [GON84]

Ce système de simulation opère à plusieurs niveaux, le plus bas étant le niveau logique. Les éléments logiques utilisés sont des primitives MOS (portes de transmission, transistors de charge, transistors de signal), et des éléments plus complexes (registres, compteurs, mémoires, PLAs, additionneurs) qui sont décrits fonctionnellement à un niveau plus élevé. La simulation logique distingue le traitement des états et l'étude du comportement temporel des éléments. Le traitement des étatss est fait à l'aide de 16 états : 15 couples (valeur, force) et l'état "tristate" (Z). L'emploi d'une force logique "indéterminée" ne semble pas utile ; en effet, les éléments à simuler sont tous décrits ; il se peut que l'on ignore la valeur logique d'un signal (initialisation, conflit), mais on connaît toujours le ou les éléments qui délivrent ce signal, et donc la force de celui-ci (la force logique d'un signal dépend de l'élément qui le délivre : capacité, résistance, interrupteur...). La simulation du comportement temporel distingue les délais de lignes, de commutation et de propagation dans un élément. Ces délais sont affectés en fonction de l'entrée et de la sortie de l'élément, et de la fonction qu'il active. La distinction entre les signaux montants et descendants est faite

automatiquement par le simulateur. Les aléas sont analysés et filtrés, et les conflits sur les bus (écritures simultanées, ...) ainsi que les oscillations sont identifiés et signalés à l'utilisateur.

Simulateur multiniveau MSIM sur système DAX [HOD84]

DAX permet une conception hiérarchique montante et descendante. Le simulateur multiniveau MSIM implanté fait partie intégrante du système. L'utilisateur peut effectuer une simulation à tous les étages de description, en choisissant le niveau de détail le plus adéquat.

Les sept forces de conduction disponibles en plus des six états de signal sont affectés en fonction de la taille des transistors. Les réseaux de transistors sont automatiquement extraits à partir des données d'implantation physique. La définition d'un état contient, en plus de la valeur, une condition de conduction : B (bi-directionnel), P (conduction possible), R (mode réception). Les valeurs h et l servent justement à signaler qu'une ligne est en conduction possible.

Dans ce simulateur, l'accent a été mis sur la possibilité de correction et de localisation interactives des pannes. Pour cela, plusieurs facilités sont fournies :

- accessibilité de tous les états des objets pendant la simulation,
- mécanisme de trace pour aider à la localisation,
- langage de test de haut niveau pour développer les séquences de données de test interactivement,
- possibilité d'isoler ou de sélectionner des parties du circuit pour une simulation plus fine,
- interruption possible de la simulation à tout moment pour faire le point et changer les données si nécessaire,
- indications sur les lignes dont l'état reste indéterminé,
- liste des lignes concernées en cas d'oscillations.

Simulateur multivalué CSASIM [KAW84]

Ce simulateur applique la modélisation CSA (Connector-Switch-Attenuator), se servant de primitives plus complexes qu'au niveau interrupteur. Le modèle CSA contient plusieurs éléments de base : les connecteurs, les interrupteurs et les atténuateurs. Les connecteurs relient les interrupteurs (transistors de signal) entre eux. Les atténuateurs sont des éléments résistifs (transistors de charge).

Le modèle utilise l'ensemble de valeurs $V_4 = \{0, 1, I, Z\}$ auquel on adjoint deux niveaux de force, donnant le treillis hiérarchisé :

$$T_2 = \{I, 1, 0, \tilde{I}\text{-faible}, 1\text{-faible}, 0\text{-faible}, Z\}.$$

- logique de connexion

Lorsque k signaux dont les états sont $x_1, ..., x_k$ arrivent sur un même nœud, on calcule l'état z de celui-ci à l'aide de l'opérateur # :

$z = \#(x_1, ..., x_k)$ = la plus faible valeur de T_2 telle que $x_i \leq z$, pour tout i = 1, ..., k (opérateur borne supérieure).

Le treillis T_2 est défini par la figure 1-10.

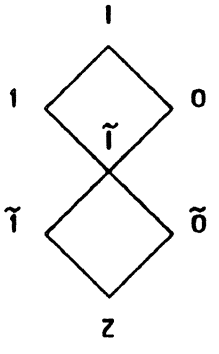


figure 1-10 : treillis T_2.

Les primitives sont représentées par les symboles suivants [fig. 1-11] :

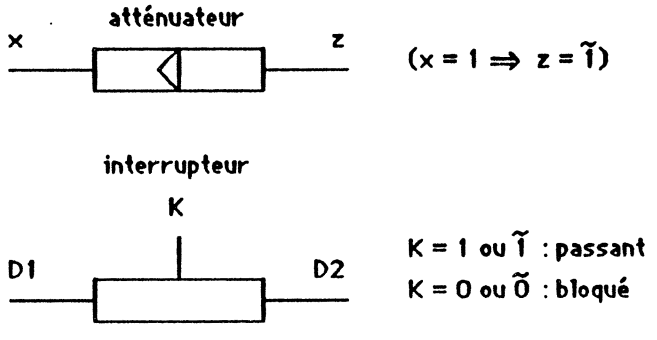


figure 1-11 : symboles des primitives statiques.

- comportement dynamique

Les retards dans la propagation du signal sont principalement dûs aux capacités

parasites qui limitent les vitesses de commutation des signaux. Le système CSA utilise des puits W [fig. 1-12] pour modéliser ces retards.

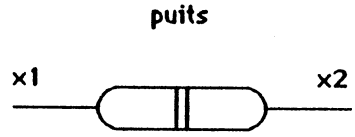


figure 1-12 : symbole des capacités.

Les états x1 et x2 de ces puits sont donnés dans le tableau suivant [fig. 1-13], dans le cas où le puits est d'abord déchargé et isolé, avant d'être connecté :

x2 \ x1	0	1	I	Z
0	s_Z	s_1	s_1	s_Z
1	s_1	s_Z	s_1	s_Z
I	s_1	s_1	s_1	s_Z
Z	s_Z	s_Z	s_Z	s_Z

figure 1-13 : états des pôles des puits.

L'état de décharge est s_Z, les états de charge sont s_1, s_0, et s_I.

Les temps de charge et de décharge sont aussi représentés dans le modèle CSA. Un signal fort crée une charge ou une décharge rapide, et celles associées aux signaux faibles sont lentes [fig. 1-14].

x1	(x2 = 0)	temps
0	décharge rapide à s_Z	τ_F
1	charge rapide à s_1	τ_R
0	décharge lente à s_Z	$\tilde{\tau_F}$
1	charge lente à s_1	$\tilde{\tau_R}$

figure 1-14 : détermination des temps de charge et de décharge des puits.

Il n'y a pas de temps associé à la charge "incomplète" s_I.

- représentation d'un transistor

La représentation symbolique d'un transistor telle qu'elle pourrait être utilisée par un modèle plus précis est donnée par la figure 1-15. Sur ce schéma, A_1 est la résistance d'entrée, A_2 est la résistance source-drain, et W est la capacité grille-substrat. Les atténuateurs et les puits ne servent à modéliser que les sources dominantes de résistances et de capacités parasites.

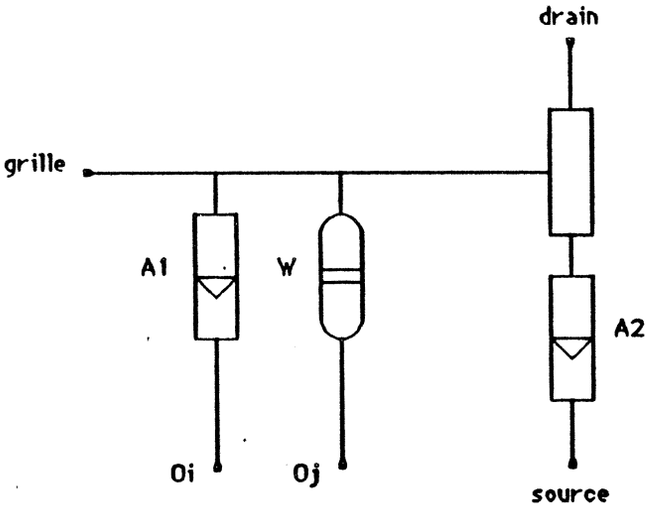


figure 1-15 : représentation symbolique d'un transistor.

- modélisation des pannes

Alors que les pannes de courts-circuits et de coupures sont en général modélisées par des données ou des circuits artificiels introduits dans la simulation, le modèle CSA représente ces pannes de façon directe.

Exemple :

On veut modéliser une panne de coupure sur l'une des sources de la porte NOR CMOS de la figure 1-16. La ligne d, normalement connectée à la masse, est ici coupée, et chacune des deux extrémités prend la valeur Z (déconnectée). Lorsque x_1x_2 = 10, le transistor S_3 délivre la valeur Z à la sortie z au lieu de la valeur 0. Cela a pour effet d'empêcher la décharge normale des deux puits W_p et W_n. Par exemple, si à l'instant t-1, z avait la valeur 1 (x_1x_2 = 00), W_n avait la valeur 1 à son extrémité variable. A l'instant t, z garde la valeur 1 maintenue par W_n, ce qui permet de détecter la panne.

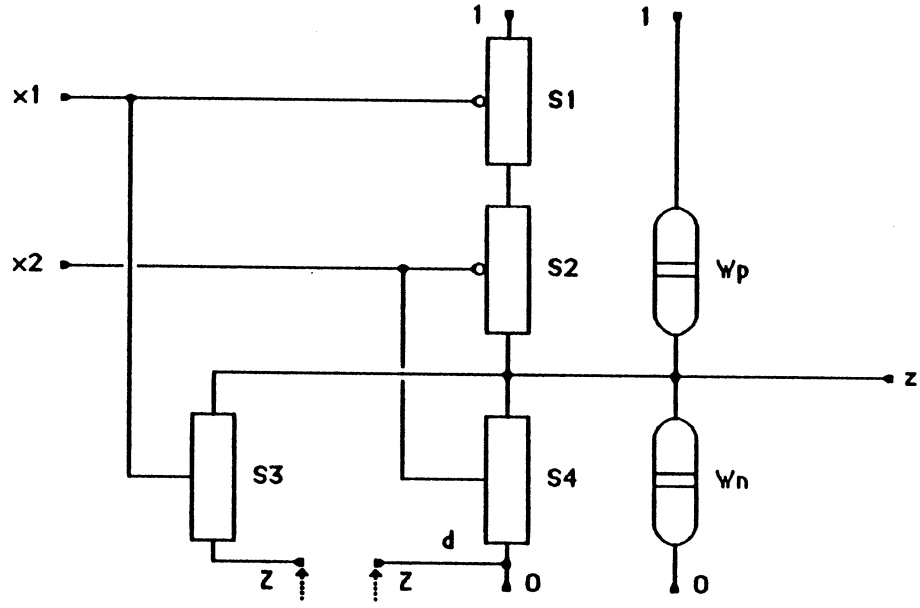


figure 1-16 : porte NOR avec une coupure sur une ligne.

Le système de modélisation de pannes collées utilisé dans CSA s'appelle GSSL (general single stuck-line). Il traite aussi bien les lignes bi-directionnelles que les lignes unidirectionnelles. Les types de collages modélisés sont regroupés dans la figure 1-17.

v1	v2	type de collage
0	0	court-circuit avec la masse
1	1	court-circuit avec l'alimentation
Z	Z	circuit ouvert
Z	0	collage à 0 } le signal se propage de gauche à droite
Z	1	collage à 1 }
0	Z	collage à 0 } le signal se propage de droite à gauche
1	Z	collage à 1 }
0	1	} collages divers à v1v2
1	0	
1	-	
-	1	

avec :

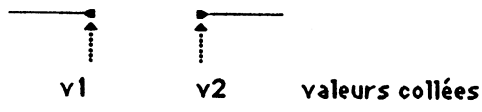


figure 1-17 : liste des collages GSSL.

Il est de plus possible de représenter des tendances aux collages en injectant comme valeurs de collage $v_1 v_2$ des valeurs faibles. Ce genre de collage est surtout utilisé associé à une capacité, pour modéliser un changement de temps de charge ou de décharge, et donc pour simuler des pannes de délais.

Exemple [fig. 1-18 et 1-19] :

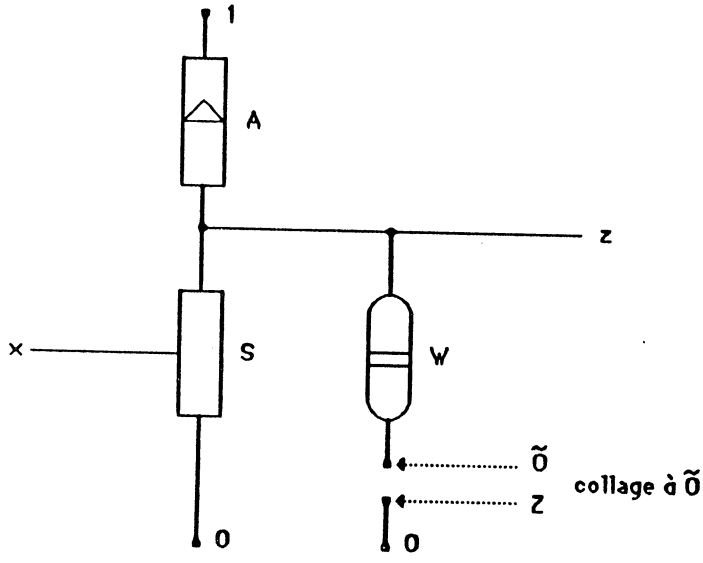


figure 1-18 : modélisation d'une tendance à un collage.

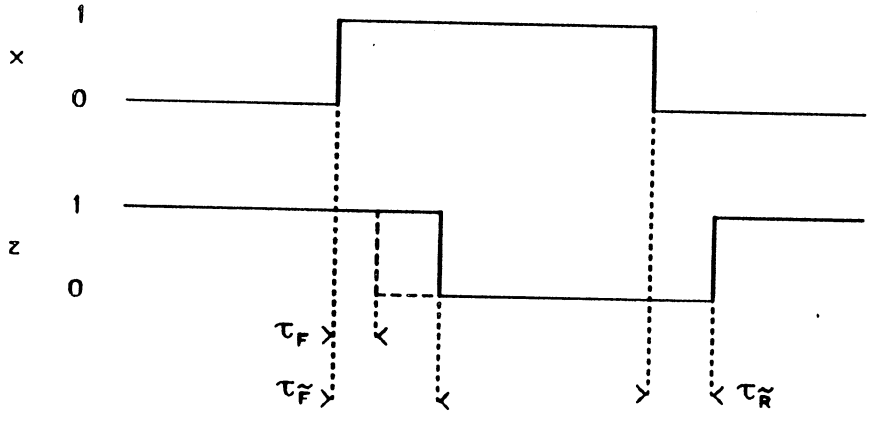


figure 1-19 : augmentation de délai dû au collage à 0-faible.

- évaluation du signal

On associe à chaque ligne du circuit deux signaux x_1 et x_2 allant dans des sens opposés. x_1 et x_2 sont calculés séparément, puis l'état final réel est donné par $x = \#(x_1, x_2)$.

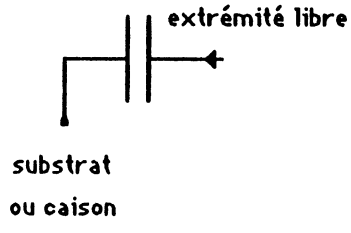
Un puits possède en entrée et en sortie une charge électrique fixe correspondant à une variation de courant dans le temps. Cette charge diminue vers 0 au fur et à mesure que le

puits se charge ou se décharge. Il est supposé dans le modèle CSA que l'une des extrémités d'un puits est connecté à une source statique (substrat ou alimentation). Le signal dynamique apparaît seulement à l'autre extrémité. Les signaux dynamiques n'ont aucun effet sur le comportement statique du modèle CSA : les puits se contentent d'introduire des délais dans tout changement de signal à son extrémité variable.

- remarques sur le modèle CSA

- un condensateur ne peut se charger que s'il existe une différence de potentiel entre ses deux pôles. Si, après ce chargement, l'une des extrémités se trouve isolée (= Z), la charge du condensateur maintient cette différence de potentiel. Un état de charge se caractérise donc en partie par les deux potentiels appliqués à ses pôles. Si la différence de potentiel entre les deux pôles est nulle, le condensateur se décharge (ou reste déchargé). Si l'une des deux extrémités est ensuite isolée, l'absence de charge électrique maintient cette égalité de potentiel. Il existe donc plusieurs cas de décharge, suivant que les pôles sont à (1, 1) ou (0, 0). Les états de charge et de décharge d'un condensateur, dans le cas simplifié où l'une des extrémités est fixe, sont résumés dans la figure 1-20, dans laquelle :

C0 : état de charge mémorisant un 0 ;
C1 : état de charge mémorisant un 1 ;
D0 : état de décharge mémorisant un 0 ;
D1 : état de décharge mémorisant un 1.



ETAT DE LA CAPACITE		
substrat ou caisson / extrémité libre	0 V	5 V
0 V	D0	C0
5 V	C1	D1

figure 1-20 : états d'un condensateur.

La modélisation CSA est donc incomplète de ce point de vue : les états de charge ne suffisent pas à retrouver les valeurs des pôles, soit (1, 0), soit (0, 1). De même, l'état de décharge s_z est insuffisant: les valeurs des pôles dans ce cas peuvent être (1, 1) ou (0, 0).

- une capacité délivre toujours un signal de niveau faible, plus faible que celui d'un signal délivré par un transistor de charge. Pour assurer une modélisation précise, il faudrait donc prévoir trois niveaux de forces. Il n'y en a que deux dans le modèle CSA.

- un autre inconvénient du modèle CSA, en ce qui concerne son utilisation pour les circuits CMOS, est qu'il travaille sur l'ensemble de valeurs $V_4 = \{0, 1, I, Z\}$. Cela ne suffit pas pour traiter des signaux dégradés : à cause de la perte de seuil sur un signal haut traversant un transistor N et sur un signal bas traversant un transistor P, il faudrait utiliser en CMOS, comme cela est exposé par la suite, l'ensemble $V_6 = \{0, 1, 0^+, 1^-, I, Z\}$ (dans lequel $0^+ \approx 0{,}8$ volt et $1^- \approx 4{,}2$ volts).

Système DECSIM [KEA84]

Le système à deux valeurs {0, 1} pour la simulation comportementale est une option du système normal à 4 valeurs. L'algorithme de simulation au niveau transistor calcule les tensions et les intensités réelles ; ces valeurs sont ensuite converties dans l'ensemble ternaire {0, 1, U}, les intensités servant à donner des indications sur la force des signaux. De ce fait, DECSIM se trouve assez proche des simulateurs électriques : l'abstraction des signaux est faite après les calculs sur les signaux réels, et non dans la modélisation du circuit, où l'on affecte des états virtuels aux éléments du modèle.

Simulateur à 5 valeurs de D. W. Lewis [LEW72]

Il est supposé dans ce modèle que les délais dûs à la longueur des connexions sont négligeables devant ceux des portes logiques. Le calcul des états et celui des délais de propagation sont séparés, ces derniers étant associés aux sorties des portes. La fonction délai est représentée par un vecteur de stockage où les différentes valeurs d'une variable y au cours de temps sont empilées, les deux extrêmes étant la valeur courante y(t) et celle que prend y après délai, c'est-à-dire y(t-délai). Ce vecteur est mis à jour à chaque transition : la valeur y(t) "vieillit" dans la pile, la valeur y(t-délai) est remplacée.

Simulateur logique de E. J. McCluskey [McC81]

On définit le vecteur d'entrée courante $X(t) = [a_i]$ et le vecteur d'entrée suivante $X(t+1) = [c_k]$, où a_i et c_k sont égaux à 0 ou 1. On définit également le vecteur de pseudo-entrée $X(t^+) = [b_j]$, où $b_j = a_i$ si et seulement si $a_i = c_k$(pas de changement), et $b_j = U$ sinon. La réponse de la ligne z à la séquence d'entrée $E = X(t)\ X(t^+)\ X(t+1)$ est $S = Z(t)\ Z(t^+)\ Z(t+1)$.

Pour détecter les aléas statiques, on simule le circuit avec E en entrée ; il existe un aléa sur la ligne z si et seulement si la séquence S obtenue est 0U0 ou 1U1. Un raisonnement analogue est appliqué pour les circuits séquentiels, la rémanence d'un U indiquant la présence d'un aléa.

Simulateur de V. Ramachandran [RAM83]

Ce simulateur se restreint aux circuits ne contenant pas de boucles pendant une phase d'horloge (les horloges sont commandées de telle manière que les circuits de retour des boucles soient coupés lors de leurs phases "off"). Le réseau est d'abord mis sous la forme d'un arbre [fig. 1-21] :

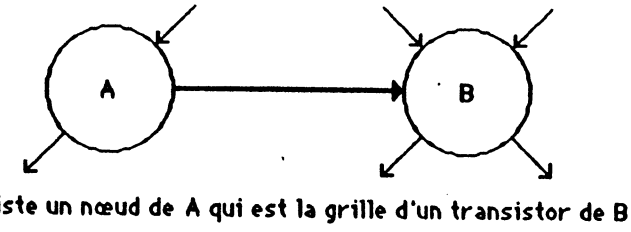


figure 1-21 : arbre modélisant le circuit.

Avec la restriction ci-dessus (découpage des boucles), ce graphe est acyclique. On peut donc ordonner les groupes d'un tel graphe :

si la grille d'un transistor conduit à un groupe i,
si sa source et son drain conduisent à un groupe j,
alors i < j.

Avec cet ordre, la simulation de chaque phase d'horloge peut être faite en un passage à travers le réseau.

Simulateur "switch" orienté-objet [ROY85]

Ce simulateur fait partie d'un environnement hiérarchique de conception. Cet environnement englobe différentes applications compatibles entre elles, l'ensemble réduisant la variété de bases de données. Ces applications fondent leur compatibilité sur l'utilisation commune d'un système de données, où l'objet de base est la cellule, représentée différemment suivant l'application, mais gardant extérieurement la même apparence pour conserver la cohérence de description au sein de l'environnement. Le langage d'implantation du système est le langage MAINSAIL.

Les objets (cellules) utilisés dans le simulateur sont de deux types :

- les cellules-feuilles, atomiques, qui sont au niveau le plus bas de la hiérarchie, sont construites à partir d'éléments comme les interrupteurs, les connecteurs...

- les cellules-composées sont formées de cellules-feuilles et d'interconnexions.

Ces objets de base étant de deux types, le système comporte en fait deux simulateurs "switch" : un simulateur de cellules-feuilles et un simulateur de cellules-composées. Ces simulateurs sont "orientés-événement" ("event-driven"), et permettent ainsi la modélisation de circuits asynchrones à délais multiples. Chaque cellule contrôle sa propre liste d'événements, ce qui assure l'autonomie entre elles.

Le simulateur de cellules-feuilles utilise un modèle linéaire de transistor, fondé sur la méthode CSA de Hayes [HAY83] et légèrement modifié en fonction de certains résultats de Bryant. L'évaluation du signal passe par un pré-traitement des groupes de transistors (réseaux de connexions et d'influence), délivrant des diagrammes de décision. Ceux-ci permettent une évaluation rapide des signaux parcourant ces groupes de transistors.

Le simulateur de cellules-composées détermine le bon séquencement des événements entre les cellules-feuilles (contrôle asynchrone distribué entre elles), et propage les signaux d'une cellule à l'autre.

Simulateur logique BIMOS [STE83]

Ce simulateur accepte comme primitives les portes unidirectionnelles, les blocs fonctionnels de haut niveau, aussi bien que les transistors bi-directionnels. Chaque nœud est défini par un couple (v, f) ; sont présentées ici les différentes forces et leur signification :

E (external) : force des entrées externes (V_{SS} et V_{DD}),

D (driven) : force d'une sortie de porte chargée ou déchargée par une source de courant,

R (resistive) : force d'une sortie de porte chargée ou déchargée à travers une résistance,

L (large high impedance) : information stockée par une grande capacité,

Z (high impedance) : information stockée par une petite capacité.

Ces forces sont présentées en ordre décroissant. Dans ce système de variables, un inverseur NMOS délivre un D0 et un R1, un inverseur CMOS délivre un D0 et un D1. Lorsqu'un événement se produit à l'une des entrées du circuit, on détermine les nouvelles valeurs internes et en sortie par un algorithme de relaxation. Pour alléger ce calcul, on partitionne le circuit en sous-circuits unilatéraux, en coupant toutes les lignes unidirectionnelles. Les effets bilatéraux causés par les lignes bi-directionnelles sont alors locaux aux sous-circuits.

Les délais de propagation sont calculés après le calcul des nouveaux états, qui est fondé sur un modèle à délai nul. Les délais de montée et de descente (temps entre l'événement en entrée et le changement en sortie) sont spécifiés pour les transistors et les portes. Lorsque plusieurs portes chargent un même nœud, on prend le délai maximal de celles-ci pour caractériser le nœud (pire cas).

Langage de description de circuits intégrés fonctionnel (FIDEL) [TAH84]

Ce langage permet de spécifier fonctionnellement des systèmes de processeurs, des puces ou des parties de puces. FIDEL étant destiné à la conception descendante, il n'y a aucune relation ascendante entre un niveau de description et ceux qui lui sont inférieurs.

Une simulation fonctionnelle ne requiert pas les mêmes variables qu'une simulation plus détaillée ; cela explique le choix des états : "entrée/sortie" (bi-directionnelle), "entrée", "sortie", qui représentent des signaux de communication entre les différents modules fonctionnels, "état" et "entier" qui sont des valeurs internes.

La gestion du temps englobe les délais de propagation et les contraintes temporelles. Celles-ci sont régulièrement vérifiées, et un signal d'alarme est envoyé si l'une d'elles n'est pas respectée. Toutes ces manipulations sont gérées par un algorithme de haut niveau.

Simulateur logique TEGAS [THO75]

TEGAS est un système de simulation logique et de génération de test prenant en compte le comportement temporel des circuits. Il possède trois niveaux de précision et de modélisation.

Le plus fin utilise 5 valeurs, permettant de représenter les transitions (0-1 et 1-0), et 2 valeurs de délais (maximum et minimum). Quatre tables de mémoire décrivent un signal, contenant sa valeur future, les erreurs potentielles de celle-ci, sa valeur actuelle, et les erreurs potentielles correspondantes. Les valeurs courante et future sont toujours égales, sauf dans le cas où le signal se trouve dans un état de transition.

Deux méthodes de détection de pannes sont disponibles dans TEGAS. La première consiste à interroger chaque signal qui change d'état et que l'on désire observer, pour déterminer si une panne y est visible. La détection se fait par comparaison entre certains bits des mots mémoire représentant les signaux. Dans la seconde méthode, les signaux observables ne sont vérifiés que pendant certaines périodes fixes. Elle est utilisable pour les circuits totalement synchrones, ou lorsqu'on attend l'obtention de l'état stable du circuit pour observer les pannes.

Simulateur de J. Watanabe & al. à 7 valeurs [WAT80]

La valeur dynamique 1 caractérise une charge de capacité, et la valeur dynamique 0 une absence de charge. Cela ne reflète pas exactement le comportement d'une capacité, ainsi qu'il est exposé précédemment, et ne peut concerner que les capacités mises à la masse. Ces deux valeurs convergent vers z (inconnue dynamique) après un temps T_h de maintien.

La modélisation du signal (ensemble d'états) est proche de celle de la méthode CSA, mais comporte trois forces : signal fort, signal atténué (transistor de charge) et signal faible délivré par les capacités.

1, 4 Récapitulation de la modélisation multivaluée

Il faut bien distinguer les deux parties principales de la modélisation multivaluée. D'une part, la définition des états (les valeurs et les forces qui définissent les valeurs statiques, dynamiques, ...) ; d'autre part, la définition des phénomènes temporels et la spécification des outils et des variables nécessaires à leur traitement (délais de propagation fixes ou calculés, ...).

1, 41 Valeurs et forces des états

Les valeurs sont définies en vue de représenter le plus fidèlement possible dans le cadre d'un modélisation discrète, les tensions que l'on peut trouver dans un circuit VLSI.

L'important dans cette démarche est de choisir une valeur indéterminée, qui permet de couper court à toutes les spéculations floues (par exemple, un inverseur reconnaît-il la tension 2 V en entrée comme un 1 ou un 0 ? ...), et d'avoir suffisamment de valeurs pour pouvoir traiter ce que l'on veut (par exemple, en CMOS, on peut choisir de modéliser les tensions altérées 0,8 V 4,2 V, car en théorie, elles font conduire respectivement un transistor N et un transistor P).

Le rôle important joué par les capacités parasites et par la taille des transistors peut justifier la détermination de plusieurs forces et d'un opérateur de superposition de signaux (noté #), calculant le nouvel état d'une ligne où il y a conflit. Cet opérateur, qui peut être utilisé différemment selon l'algorithme choisi, est inhérent à la structure de treillis des algèbres multivaluées (tout couple d'états possède une borne supérieure et une borne inférieure). Ces treillis définissent sans équivoque la hiérarchie entre tous les états. Les calculs qui en découlent sont intemporels.

1, 42 Délais de propagation et phénomènes temporels

La détermination des délais de propagation permet de mieux cerner les erreurs de temporisation. Cela est utile entre autres pour définir un temps minimal d'attente sur une ligne subissant un changement, pour être sûr de la stabilité de l'état final (par exemple, dans les automates, il est primordial de ne pas envoyer des ordres erronés ; on temporise donc pour assurer le bon déroulement des étapes).

On traite aussi les temps de charge et de décharge des capacités parasites, qui entrent dans le calcul du temps de réponse des portes. On obtient ainsi le temps de propagation des portes s'ajoutant au temps de propagation des lignes. Les variables utilisées sont "simples", par exemple t_m et t_d (temps de montée et de descente), ou encore un délai fixe unitaire ("unit-delay") qui s'ajoute à toute transition.

L'utilisation de valeurs "composées" est justifiée lorsqu'on veut connaître ou prévoir le comportement d'un nœud au cours du temps. L'étude des aléas est une excellente application de cette méthode : par exemple, un passage de 0 à 1 est noté (0, 1), un pic à 0 dans une transition de 0 à 1 est noté (0, 1, 0, 1) ... Ces valeurs ne doivent pas être confondues avec les ensembles de valeurs possibles (par exemple, {0, 1} pour définir X), ni avec les couples (v, f) utilisés dans les treillis (la valeur v du couple (v, f) peut être une valeur composée, quoiqu'il n'y en ait pas d'exemple existant).

CHAPITRE 2

SPECIFICATIONS D'UN SIMULATEUR "SWITCH" MULTIVALUE

2 Spécifications d'un simulateur "switch" multivalué

Ces spécifications ne concerne que l'évaluation des signaux ; les autres aspects de la simulation tels que la gestion informatique des états et des événements ne sont pas abordés dans ce document. La définition de ces spécifications comprend deux parties principales : la détermination de l'algèbre utilisée et celle du calculs des signaux. Mais ces deux points ne peuvent être traités sans une étude préalable du domaine d'utilisation et du principe général de fonctionnement d'un simulateur "switch" multivalué.

2, 1 Utilisation d'un simulateur multivalué

On peut considérer que la simulation multivaluée se situe entre la simulation logique et la simulation électrique. En effet, l'utilisation d'un nombre de valeurs supérieur à trois (algèbre ternaire {1, 0, X}) la rapproche de la simulation électrique, tandis que le maniement de valeurs discrètes appartient davantage au domaine logique. Un simulateur électrique opère à partir de formules électriques du fonctionnement des transistors, et requiert un assez grand nombre de spécifications technologiques. On ne peut donc qu'être moins précis si l'on n'utilise pas ces formules : la résolution des états en simulation multivaluée ne demande qu'un calcul simple qui prend en compte, de manière virtuelle dans un modèle, la résistance des transistors et les tensions qui leur sont appliquées.

Le nombre de forces logiques modélisant ces résistances fait l'objet d'un compromis entre une précision convenable et un maniement aisé. On pourrait définir un niveau de force par taille de transistor (W/L) et obtenir ainsi une grande précision ; mais le nombre de transistors différents existant dans un circuit ne permet pas toujours d'exploiter cette possibilité : les calculs seraient trop lourds et trop lents pour être acceptables. De même, un nombre trop petit de forces ne donne pas une précision suffisante.

Tous ces choix participent de l'utilisation souhaitée d'un simulateur multivalué. A priori, l'intérêt d'un tel outil est de permettre une vérification sur des parties réduites d'un circuit, qui peut éventuellement se suffire à elle-même, et qui indique avec une précision satisfaisante si ces parties doivent être simulées plus finement.

2, 2 Principe de la simulation "switch"

Le simulateur décrit dans les pages qui suivent utilise un modèle de circuit décrit au niveau interrupteur. La base de données de ce modèle consiste en un réseau de transistors connectés par des lignes équipotentielles, appelées nœuds : à un instant donné, un nœud a une valeur unique, et cela partout dans le circuit. L'élément principal donnant lieu au calcul des états est donné par la figure 2-1.

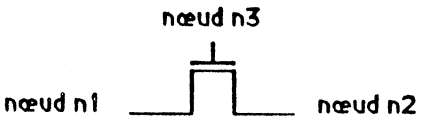


figure 2-1 : élément de base du modèle.

La propagation d'un signal se réduit alors à deux opérations principales :

- rendre un transistor bloqué
- rendre un transistor passant,

avec les opérations de mise à jour qui en découlent.

Choisir de travailler au niveau interrupteur conduit implicitement à assumer le fait que les transistors MOS sont symétriques et donc complètement bi-directionnels. Cela n'est pas souvent le cas dans les simulateurs "switch" existants : on suppose connu le sens des courants (cela signifie qu'après une étude du circuit, on décide que le courant peut n'avoir que tel sens sur telle ligne), ce qui réduit le circuit à une suite de lignes influant sur des grilles de transistors (la représentation en arbres de [RAM83] est proche d'un tel réseau). Cela est vrai si on supprime les portes de transmission, ou du moins si on en tient compte dans un contexte unidirectionnel : par exemple, on crée des sous-réseaux unidirectionnels reliés entre eux par des dispositifs bi-directionnels qui sont traités à part. Transformer les circuits de cette manière simplifie les calculs (c'est pourquoi elle est si répandue, malgré la difficulté posée par la décision du sens des courants lors de l'étude du circuit), mais elle laisse de côté tous les problèmes de conflit en sortance (où se posent justement les problèmes de bi-directionnalité) pour les retrouver toujours à une étape ou à une autre [cf. fig. 2-2], outre le fait qu'elle ne reflète pas exactement le principe même du niveau de travail, illustré par son nom : niveau interrupteur.

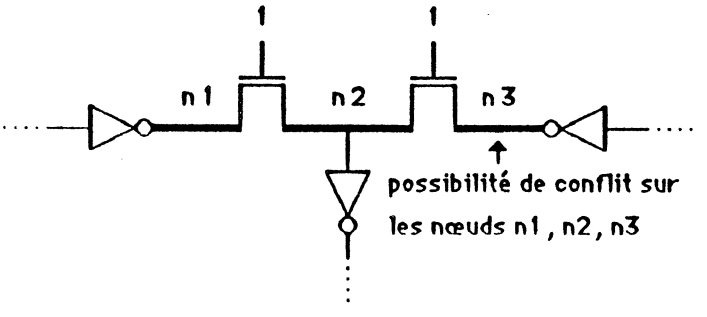


figure 2-2 : conflit possible en sortie des deux inverseurs.

Ici, aucune hypothèse n'est faite, et les problèmes de bi-directionnalité sont traités de manière naturelle.

Les opérations de base définies précédemment sont l'ouverture (le transistor devient bloqué) et la fermeture (le transistor devient passant) d'un transistor.

L'opération essentielle est la fermeture d'un transistor, car c'est elle qui donne lieu au calcul d'états le plus complet. Elle peut se résumer par les faits suivants [fig. 2-1] :

- au temps t0, n3 := 1 (resp. 0) si le transistor est de type N (resp. P).

Si on applique un délai unitaire à toute commutation[1] :

- au temps t0+1, le transistor est passant. Donc, au temps t0+1, n1 et n2 changent d'état. Le problème est de connaître ces deux nouveaux états, qui ne sont pas forcément identiques. En effet, dans le cas où le transistor est très résistif, il doit y avoir une différence de force entre les deux sources[2] pour modéliser cette résistance, et dans certains cas, le signal traversant le transistor peut être dégradé par suite d'une perte de seuil entre les deux sources.

Comme on ne connaît pas le sens du courant, on ne peut savoir à priori lequel des deux nœuds impose sa valeur à l'autre. Il faut donc, avant d'appliquer une quelconque opération calculant les deux nouveaux états, déterminer si l'un des nœuds prévaut sur l'autre, ou si chacun est capable de modifier l'autre. Ce point sera traité dans la description du calcul des signaux. Auparavant, une analyse complète des besoins de la simulation "switch"multivaluée de circuits construits en technologie CMOS et NMOS doit être réalisée pour permettre de définir une algèbre d'états pertinente.

2, 3 Définition de l'algèbre multivaluée

L'algèbre utilisée doit s'adapter au principe de calcul exposé plus haut. Le calcul des états consiste ainsi, à partir de deux anciens états de part et d'autre d'un transistor, à découvrir les deux nouveaus états, créés par la superposition des deux anciens. Cela diffère de l'opération # décrite par J. P. Hayes [HAY83], qui à partir de deux états, trouve le nouvel état résultant de leur superposition [fig. 2-3].

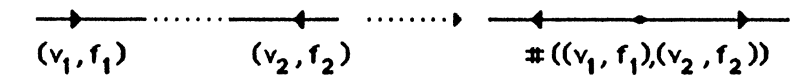


figure 2-3 : principe de la superposition décrite dans [HAY83].

(1) voir le chapitre 2, 54.

(2) le terme "drain" n'est pas utilisé à cause de l'ambiguïté qu'il pourrait créer, les transistors étant considérés comme parfaitement symétriques.

L'opération qui convient ici est une nouvelle opération de calcul de signaux, notée &, dont le principe est décrit par la figure 2-4, dans laquelle :

$((v'_1, f'_1), (v'_2, f'_2)) = \&((v_1, f_1), (v_2, f_2),$ type_transistor, taille_trans., grille_trans.).

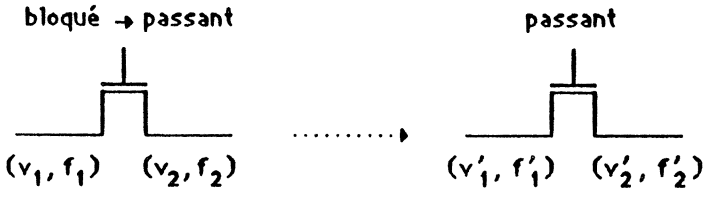


figure 2-4 : principe de l'opération &.

Pour simplifier l'écriture des états, ceux-ci seront notés de la façon suivante :

- (v_i, f_i) = état(ni) = v_{fi} ;

- (v'_i, f'_i) = nouvel_état(ni) = $v_{f'i}$.

L'opération & peut utiliser implicitement l'opération # pour déterminer les nouveaux états de part et d'autre du transistor, mais en tenant compte des ajustements de seuils et de forces éventuels.

2, 31 Ensemble des valeurs

2, 311 Technologie CMOS

On choisit de faire apparaître, en plus des valeurs 0 et 1 d'un simulateur logique, les valeurs 0^+ et 1^- représentant des tensions altérées pouvant être délivrées lors du passage d'un signal à travers un transistor. Cette altération se traduit par une perte de potentiel due à la différence d'un seuil (V_{thn} et V_{thp}) existant naturellement entre la source et la grille d'un transistor, comme il est montré dans la figure 2-5 :

- $V_{grille} - V_{source} \leq V_{thp}$, donc $V_{source} \geq V_{grille} - (-1) \approx 1\ (0^+)$;
- $V_{grille} - V_{source} \geq V_{thn}$, donc $V_{source} \leq V_{grille} - (1) \approx 4\ (1^-)$.

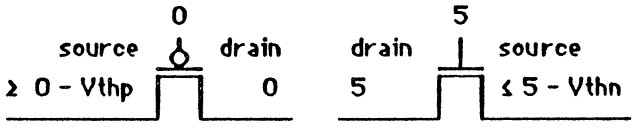


figure 2-5 : perte de seuil dans un transistor.

Le seuil des transistors P et N étant approximativement -1 V et +1 V, 0^+ et 1^- valent respectivement 1 V et 4 V. Dans la réalité, ces valeurs peuvent varier sensiblement d'une technologie à l'autre, ou d'un circuit à l'autre, et se rapprocher de la tension 2, 5 V. Pour les exemples de simulation SPICE donnés dans les annexes, une interprétation parfois large de ces deux valeurs a été nécessaire pour garder la cohérence au sein de l'ensemble d'états. L'intérêt des valeurs 1^- et 0^+ est de faire conduire à la fois un transistor N et un transistor P ; en cela, elles paraissent semblables. Pourtant, prises dans le contexte des réseaux d'interrupteurs, ces valeurs sont différentes, pour la force et la valeur des signaux concernés.

2, 3111 Influence de 1^- et de 0^+ sur la force d'un signal

Ces deux valeurs sont différentes du point de vue de la force des signaux, car la résistance des transistors commandés par de telles valeurs peut changer :

- un transistor N conduit mieux commandé par un 1^- que par un 0^+, car 0^+ est plus proche que 1^- du seuil du transistor (théoriquement il lui est égal);

- un transistor P conduit mieux commandé par un 0^+ que par un 1^-, pour une raison analogue.

Exemple [fig. 2-6] :

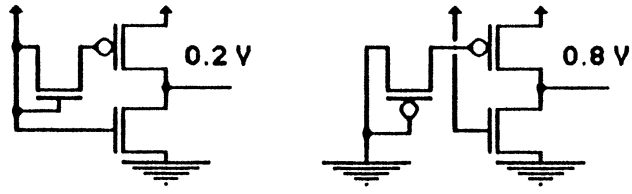


figure 2-6 : différence de conduction pour 0^+ et 1^-.

Le transistor P conduit dans les deux cas, et empêche le signal de sortie de tomber complètement à 0. La différence entre les deux tensions vient de ce que le transistor P commandé par 0^+ conduit mieux, et altère davantage le niveau de sortie. On interprétera ainsi la tension 0,2 V comme la valeur 0, et la tension 0,8 V comme une valeur indéterminée. Cette interprétation, qui peut sembler abusive, s'appuie sur la différence entre les intensités du courant qui traverse les transistors (il y a plus de risque de claquage dans le second cas). Les résultats de la simulation de ce circuit sont donnés en Annexe 1.

Comme la résistance d'un transistor N (resp. P) augmente lorsqu'il est commandé par un 0^+ (resp. 1^-), un signal traversant ce transistor perd en force. Cela permet de

modéliser le comportement d'un inverseur CMOS dont l'entrée est 1^- ou 0^+ [fig. 2-7]. Les deux tensions de sortie, valant 0, 2 V et 4, 8 V, ont été vérifiées par une simulation SPICE. La détermination des états des nœuds, et en particulier des forces qui sont indiquées sur la figure, font référence de manière anticipée au chapitre [2, 32].

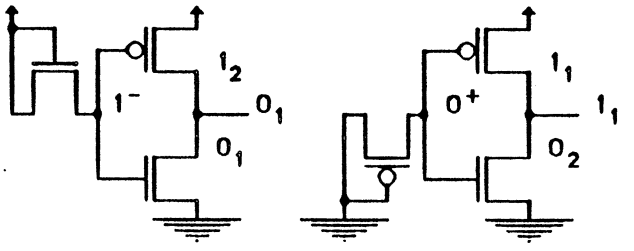


figure 2-7 : comportement d'un inverseur commandé par des valeurs altérées.

2, 3112 Influence de 1^- et de 0^+ sur la valeur d'un signal

Les deux valeurs altérées sont également différentes quant à leur influence sur la valeur des signaux. En effet, la perte de seuil que subit un signal traversant un transistor peut créer dans certains cas des dégradations en cascades [fig. 2-8]. Il est impossible de faire état dans le modèle de ces pertes de seuil en cascade, à cause du nombre trop élevé de valeurs que cela occasionnerait, d'autant qu'une semblable disposition de transistors est extrêmement rare, même dans une configuration de simulation de pannes. On pourrait tout de même approcher ce phénomène, comme il est montré dans la figure 2-9.

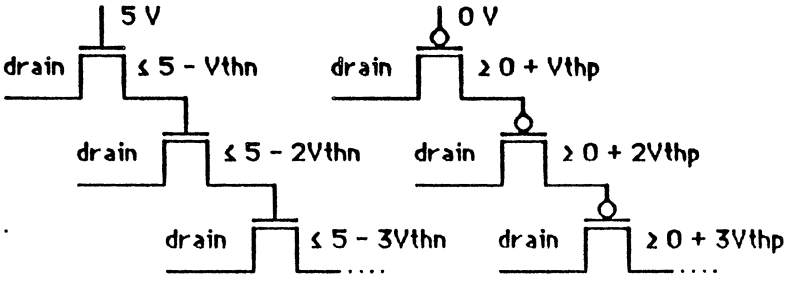


figure 2-8 : dégradations en série.

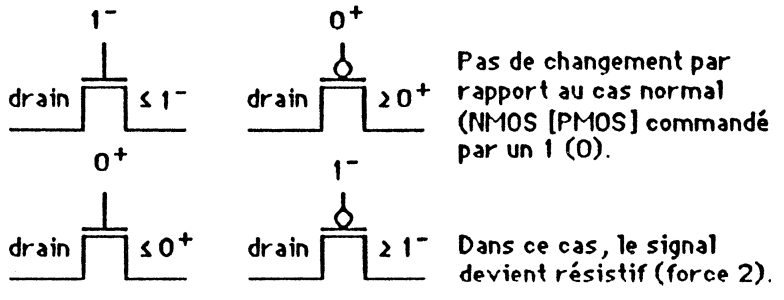


figure 2-9 : modélisation des dégradations en cascade.

2, 3113 Détermination de l'algèbre finale

En plus des quatre valeurs 0°, 1, 0^{+}, 1^{-}, on utilise une valeur indéterminée I et une valeur d'initialisation X (la valeur I n'est pas concernée par les considérations des chapitres [2, 3111] et [2, 3112] ; cette convention est choisie par souci de généralité : la valeur indéterminée I représente [0 V, 5 V]). On obtient donc l'ensemble V_6 :

$$V_6 = \{0, 1, 0^{+}, 1^{-}, I, X\}.$$

La hiérarchie entre ces valeurs au sein de l'ensemble V_6, prise dans le sens de l'opération #, est définie par la figure 2-10.

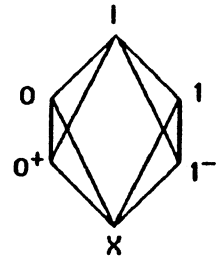


figure 2-10 : treillis représentant l'ensemble V_6.

Cette hiérarchie est justifiée par la résolution de la porte de transfert CMOS de la figure 2-11. Dans cette porte, il y a superposition permanente d'un niveau altéré avec le niveau correct correspondant (0^{+}, 0 ; 1^{-}, 1). Or, la sortie se trouve toujours aux niveaux corrects (c'est en cela que réside l'intérêt du doublage N-P dans les portes de transfert CMOS), ce qui indique la prépondérance de ces derniers. Les deux niveaux altérés se trouve donc "inférieurs" aux deux niveaux corrects 0 et 1.

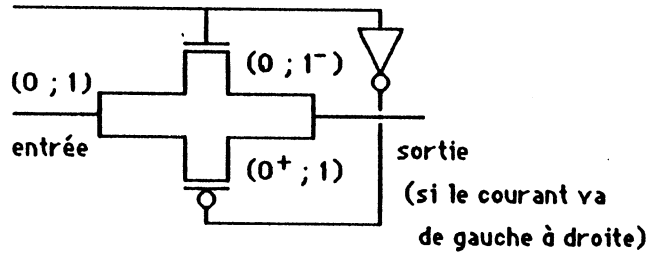


figure 2-11 : porte de transfert CMOS.

Ces deux niveaux altérés sont réellement intéressants à cause de l'incidence grave qu'ils peuvent avoir sur le fonctionnement de certaines portes [fig. 2-12] :

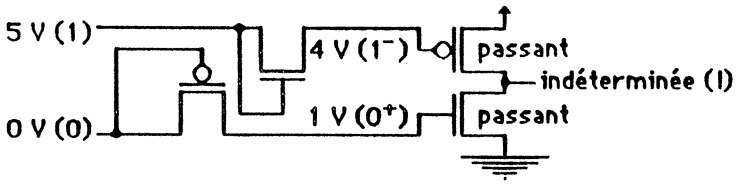


figure 2-12 : incidence des niveaux altérés.

Les signaux transmis par les deux transistors de transfert sont 1^- et 0^+. Ces niveaux suffisent pour faire conduire les deux transistors P et N correspondants. Leur source prend respectivement la valeur 1 et 0, créant un conflit à leur jonction dont le résultat après superposition est modélisé par la valeur I. D'un point de vue électrique (SPICE), la sortie se trouve à un niveau de potentiel instable et intermédiaire.

2, 312 Technologie NMOS

Il est important de noter que l'utilisation des niveaux altérés ne concerne que la technologie CMOS. En effet, en NMOS, le 0^+ n'existe pas car il n'y a pas de transistor P, et il n'y a jamais de conflit entre un 1 et un 1^-. Le cas est étudié ici [fig. 2-13] pour la superposition de la sortie d'un inverseur et celle d'une porte de transfert.

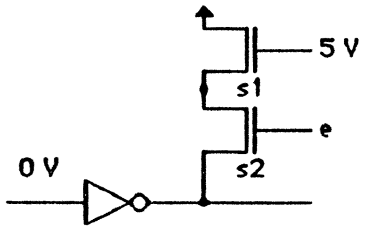


figure 2-13 : superposition en NMOS de 1 et de 1^-.

En NMOS, la sortie à l'état haut d'un inverseur prend, par référence anticipée au chapitre [2, 32], l'état 1_2, car le transistor déplété (de charge) est plus résistif que le transistor enrichi (de décharge). C'est d'ailleurs la condition nécessaire pour que le fonctionnement des circuits soit correct, car dans le cas où l'entrée de l'inverseur est à l'état haut, les deux transistors de l'inverseur conduisent ; la résistance du transistor enrichi doit être plus petite que celle du transistor déplété pour que le niveau de sortie soit 0.

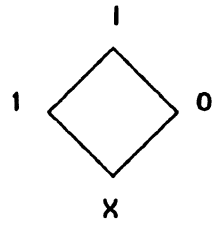


figure 2-14 : hiérarchie de l'ensemble V_4.

2, 32 Ensemble des forces

Le choix du nombre de forces utilisées dans une algèbre multivaluée est capital, car de lui dépend l'efficacité de la simulation ; il fait l'objet d'un compromis entre une grande précision, permise par l'emploi d'un grand nombre de forces, et une simulation rapide, favorisée par un petit nombre de forces. Il est donc nécessaire de rechercher d'abord les comportements que l'on désire modéliser, et ceux dont le traitement serait inutile ou trop complexe, pour en déduire un nombre optimal de forces.

2, 321 Modélisation des signaux statiques

Les signaux statiques dans un circuit sont créés par une connexion avec l'une des deux alimentations. Les forces qui permettent de représenter leurs effets introduisent donc la notion de résistance. Le problème est ainsi de déterminer le nombre de résistances du modèle. Ces résistances ne concernent, au niveau de description choisi, que les transistors, et plus précisément, le rapport W/L (largeur du canal/longueur du canal). Il apparaît que pour conserver l'atout de rapidité de la simulation multivaluée, il est préférable de sacrifier à la précision ; en effet, celle-ci n'est généralement souhaitée que pour des problèmes critiques sur de petites parties, et peut ainsi être obtenue par le recours à un simulateur électrique. Malgré tout, pour que ce type de simulation soit avantageuse par rapport à la simulation logique, une certaine hiérarchie entre les signaux doit être définie, en fonction du chemin de résistances qu'ils traversent. C'est pourquoi on choisit d'introduire deux forces statiques, la première (force 1) caractérisant un signal traversant un transistor "Standard", la seconde (force 2) étant attribuée à un signal traversant un transistor "Long" (W/L petit). Reste à l'utilisateur le classement des transistors du circuit en deux groupes, suivant leur taille.

2, 322 Modélisation des signaux dynamiques

Les signaux dynamiques sont délivrés par des capacités. Celles-ci sont chargées ou déchargées par des signaux statiques, et influent momentanément sur le comportement du circuit. La hiérarchie entre les signaux dynamiques est donc fonction de la taille des capacités. On choisit deux tailles différentes, induisant deux forces dynamiques (forces 3 et

4) ; l'établissement d'une hiérarchie possible au sein des signaux capacitifs permet de modéliser les problèmes de partage de charge (deux capacités isolées influent l'une sur l'autre), qui peuvent être critiques lorsque les capacités commandent la grille d'un transistor. Là aussi, l'utilisateur affecte aux nœuds la capacité qui convient.

En conclusion, on se contente de quatre forces, qui constituent l'ensemble F_4 :

- force 1 : celle des entrées et des signaux transmis par les transistors de signal et les transistors "Standard" ;

- force 2 : celle des signaux transmis par les transistors de charge ou les transistors ayant un faible rapport W/L (transistors "Longs") ;

- force 3 : celle des signaux délivrés par les grosses capacités ;

- force 4 : celle des signaux délivrés par les petites capacités ;

La hiérarchie complète (au sens de l'opération #) sur l'ensemble V_6xF_4 et sur l'ensemble V_4xF_4 est donnée par la figure 2-15.

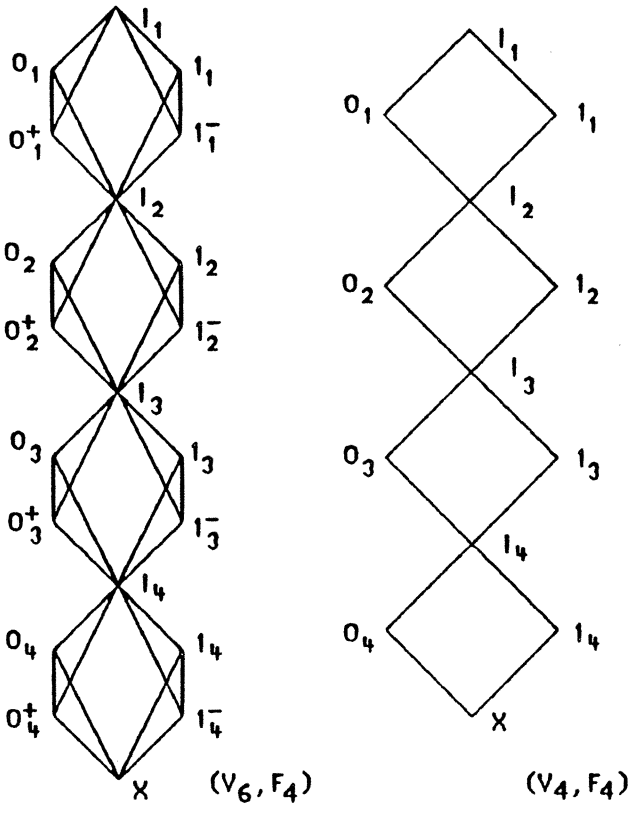


figure 2-15 : treillis de $V_6 \times F_4$ et de $V_4 \times F_4$.

Ces forces correspondent à des forces qui existent dans d'autres simulateurs. Ce sont les forces :

- entrée & fort, faible, charge, stockage de [BAN83],
- conducteur, résistif, capacitif de [DOS84],
- actif & entrée, résistif, flottant de [GON84],
- entrée, faible, charge de [RAM83],
- conducteur, résistif, capacitif de [ROY85],
- externe & conducteur, résistif, grande impédance, haute impédance de [STE83].

2, 33 Algèbres définitives

2, 331 Technologie CMOS

L'algèbre définitive comporte 21 états :

$$\mathcal{A} = \{1_1, 0_1, 1^-_1, 0^+_1, 1_2, 0_2, 1^-_2, 0^+_2, 1_3, 0_3, 1^-_3, 0^+_3, 1_4, 0_4, 1^-_4, 0^+_4, I_1, I_2, I_3, I_4, X\}.$$

2, 332 Technologie NMOS

Dans ce cas, il est inutile de retenir les valeurs 0^+ et 1^-[(1)]. L'algèbre définitive pour la technologie NMOS comprend 13 états :

$$\mathcal{A}' = \{1_1, 0_1, 1_2, 0_2, 1_3, 0_3, 1_4, 0_4, I_1, I_2, I_3, I_4, X\}.$$

2, 4 Calcul des signaux

Le calcul des signaux consiste essentiellement à définir l'opérateur &, défini comme suit [cf. chapitre 2, 3] :

$$(v_f'_1, v_f'_2) = \&(v_{f1}, v_{f2}, \text{type\_transistor, taille\_trans., grille\_trans.}).$$

Le type du transistor est indispensable pour connaître les pertes de seuil qui peuvent survenir, sa taille et sa valeur de grille servent à déterminer les changements éventuels de force et de valeur du signal (la force du signal présent sur la grille n'est pas considérée dans le calcul, car elle n'a aucun effet sur la conduction ; ainsi, un signal capacitif (force 3 ou 4) pourra influer sur l'état d'un transistor (modélisation possible des grilles flottantes)).

(1) voir le chapitre 2, 312.

Avant de décrire le calcul des signaux, et afin de justifier l'utilité de l'opérateur & dans le système retenu, il semble utile de montrer par des exemples en quoi l'opérateur # est insuffisant employé seul, et de décrire les outils de calcul nécessaires compte-tenu du caractère bi-directionnel des transistors.

2, 41 Insuffisance de l'opérateur

Le premier exemple [fig. 2-16] montre qu'il est indispensable avant toute action de calcul, de savoir si les nœuds en jeu sont isolés des alimentations :

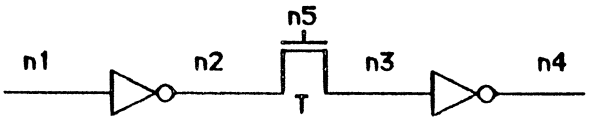


figure 2-16 : vérification de l'isolation des nœuds.

temps t0 : état(n5) = état(n4) = état(n1) = 1_2, état(n2) = état(n3) = 0_1 (NMOS).
t1 : état(n5) := 0_1.
t1+1 : état(T):= bloqué (délai unitaire).
t2 : état(n1) := 0_1.
t2+1 : état(n2) := 1_2.
t3 : état(n5) := 1_2.
t3+1 : état(T):= passant, état(n2) = état(n3) := #(0_1, 1_2) = 0_1, au lieu de 1_2.

Dans la réalité, n2 s'impose à n3, car sa valeur est réellement créée par une source d'alimentation (Vdd), alors que la valeur de n3 n'existe qu'à titre de rémanance capacitive.

Il faut donc dans ce cas mettre n3 à 0_3 ou à 0_4 (capacitif) au temps t1+1. Mais cela n'est possible que si l'on est sûr que n3 est effectivement isolé, c'est-à-dire qu'il n'a aucun contact de près ou de loin avec une alimentation.

Le fait qu'un nœud soit connecté à une alimentation n'entraîne pas forcément que le signal correspondant ait une force 1 (maximale). En effet, si celui-ci traverse un élément plus résistif, il peut acquérir une force inférieure (force 2 ; on réserve les forces 3 et 4 aux signaux capacitifs des nœuds isolés). Mais un signal de force 1 ou 2 traduit l'existence potentielle d'un courant non capacitif, et c'est cela qui importe.

Le second exemple [fig. 2-17] montre à l'inverse qu'un nœud isolé (son signal est de force 3 ou 4) peut s'imposer à un autre lorsque celui-ci lui est uniquement connecté :

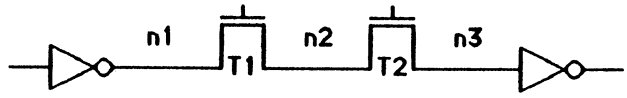


figure 2-17 : une valeur capacitive peut s'imposer.

La seule connexion de n3 est celle qui le relie à n2. Lorsque T1 devient bloqué, n2 garde sa valeur, mais sa force devient égale à 3 ou à 4. Au moment de propager cette nouvelle force vers n3, si on superpose de but en blanc l'état de n2 avec l'ancienne valeur de n3, qui a une force 1 ou 2, il sera anthilé par celui-ci, alors qu'il doit s'imposer, et propager son isolation à n3.

2, 42 Outils nécessaires au calcul des signaux bi-directionnels

Les deux problèmes exposés ci-dessus obligent à savoir déterminer une isolation, au sens large du terme : un nœud peut être isolé, même en ayant des connexions passantes vers d'autres nœuds, si l'ensemble est isolé.

Pour vérifier qu'un nœud est isolé, deux choix sont possibles : soit passer en revue, à chaque fois que c'est nécessaire, toutes les connexions du nœud et de ses voisins pour voir s'il n'est pas connecté à une alimentation ou à une entrée, soit disposer de deux variables indiquant ses connexions à Vdd, à Gnd, et aux entrées.

La deuxième solution est la plus réalisable, car elle évite la recherche systématique et obligatoire de la première : les variables sont mises à jour au fur et à mesure des opérations d'ouverture et de fermeture des transistors. Les variables créées sont au nombre de deux : on regroupe dans une première liste les informations sur la connexion avec Vdd et avec les entrées à l'état haut, et dans une seconde celles sur la connexion avec Gnd et les entrées à l'état bas.

Ces deux variables se présentent sous forme de listes :

- $VDD(n) = \{\dots , (n, n_1, \dots , n_m, N, T_1, \dots , T_{m+1}), \dots \}$,
- $GND(n) = \{\dots , (n, n'_1, \dots , n'_n, N', T'_1, \dots , T'_{n+1}), \dots \}$,

où N (resp. N') est soit une entrée, soit Vdd (resp. Gnd), où les nœuds $n_1, \dots , n_m$ (resp. $n'_1, \dots , n'_n$) forment le chemin de n à N (resp. N'), à travers les transistors $T_1, \dots , T_{m+1}$ (resp. $T'_1, \dots , T'_{n+1}$). Il y a un transistor de moins que de nœuds formant le chemin, n et N (resp. N') étant compris.

2, 421 Fermeture d'un transistor

Un nœud est donc caractérisé, à un instant donné, par trois variables dynamiques : état, VDD et GND. Les étapes de la fermeture d'un transistor sont les suivantes :

- fermer le transistor (état(T) := passant) ;

- appliquer l'opérateur & sur n1 et n2, c'est-à-dire trouver leur nouvel état, en fonction de leurs listes VDD et GND.

- mettre à jour VDD et GND des deux nœuds : si VDD(n1) n'est pas vide, on ajoute à VDD(n2) le contenu de VDD(n1) (sauf les listes contenant le nœud n2 et le transistor T), en ajoutant aux listes le nœud n1 et le transitor T. La démarche est similaire pour la mise à jour de la liste VDD(n1) et des listes GND.

2, 422 Ouverture d'un transistor

Les actions d'ouverture sont les suivantes :

- ouvrir le transistor (état(T) := bloqué).

- mettre à jour les variables VDD et GND de n1 et n2 : supprimer dans VDD(n2) et dans GND(n2) toutes les listes contenant le nœud n1 et le transistor T ; si VDD(n2) ou GND(n2) est alors modifiée, affecter à n2 la force 3 ou 4, suivant la capacité qui lui est associée ; propager cette suppression aux voisins de n2 jusqu'à ce que toutes les listes contenant n1 et T soient supprimées ; affecter la force 3 ou 4 aux nœuds dont la liste VDD ou GND est modifiée. Cette affectation est nécessaire, bien qu'elle puisse paraître prématurée : en effet, il se peut que les deux listes VDD et GND des nœuds dont la force devient 3 ou 4 ne soient pas vides ; dans ce cas, la propagation de la suppression aboutit nécessairement à une alimentation, une entrée ou un nœud dont les listes ne changent pas ; l'application de & permet alors de déterminer le nouvel état réel des nœuds. La démarche est analogue pour la mise à jour de VDD(n1) et de GND(n1).

Exemple :

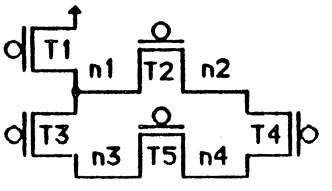


figure 2-18 : utilisation des listes VDD et GND.

Les variables VDD des nœuds du circuit de la figure 2-18 sont les suivantes :

- *VDD(n1) = {(n1, Vdd, T1)}*
- *VDD(n2) = {(n2, n1, Vdd, T2, T1), (n2, n4, n3, n1, Vdd, T4, T5, T3, T1)}*
- *VDD(n3) = {(n3, n1, Vdd, T3, T1), (n3, n4, n2, n1, Vdd, T5, T4, T2, T1)}*
- *VDD(n4) = {(n4, n3, n1, Vdd, T5, T3, T1), (n4, n2, n1, Vdd, T4, T2, T1)}.*

L'état des nœuds est : état(n1) = état(n2) = état(n3) = état(n4) = 1_1.

Si le transistor T5 devient bloqué, on supprime dans VDD(n4) les listes contenant n3 et T5, et on supprime dans VDD(n3) les listes contenant n4 et T5 :

- *VDD(n4) := {(n4, n2, n1, Vdd, T4, T2, T1)}, état(n4) :=* 1_3 ;
  on propage cette suppression aux voisins "connectés" de n4 (sauf n3) :
  VDD(n2) := {(n2, n1, Vdd, T2, T1)}, état(n2) := 1_3 ;
  on propage cette suppression aux voisins "connectés" de n2 (sauf n4) :
  VDD(n1) reste inchangé, donc on applique & entre n1 et n2, ce qui donne :
  état(n2) := 1_2 *; on propage cet état à n4 : état(n4) :=* 1_2 ;

- *VDD(n3) := {(n3, n1, Vdd, T3, T1)}, état(n3) :=* 1_3 ;
  on propage cette suppression aux voisins "connectés" de n3 (sauf n4) :
  VDD(n1) reste inchangé, donc on applique & entre n1 et n3, ce qui donne :
  état(n3) := 1_1.

Le simulateur de [AMA85] utilise une méthode voisine de celle exposée ici, en recherchant avant la simulation les chemins de conduction possibles, au sein de sous-réseaux unidirectionnels (tous les chemins possibles entre les nœuds importants et les nœuds sources (entrées, alimentations)) ; ainsi, chaque nœud possède sa propre liste de chemins. Les variables VDD et GND dynamiques de la méthode présente servent à déterminer un chemin de conduction vers les alimentations, mais qui soit effectif, et non simplement possible (ces connexions varient au cours de la simulation). Le simulateur spécifié ici utilise aussi, comme il est vu dans le chapitre [2, 6], deux variables statiques qui décrivent l'environnement de chaque nœud.

2, 43 Définition de l'opération &

L'opérateur & est décrit dans ce chapitre pour chacune des deux technologies CMOS et NMOS. Il est d'abord défini pour les cas où les deux nœuds peuvent se modifier l'un l'autre, c'est-à-dire lorsqu'ils sont tous les deux connectés, de près ou de loin, à une alimentation. Ensuite, & est défini pour les cas où l'un des nœuds est isolé.

2, 431 Technologie CMOS

2, 4311 Les deux nœuds sont reliés à une alimentation

L'opération dépend du type du transistor, de sa taille, et de la valeur de sa grille. On choisit pour faciliter la programmation de décrire & par des tableaux. Il y a ainsi un tableau par type, par taille, et par grille de transistor. Le nombre de tableaux est réduit par la réunion de plusieurs cas identiques : par exemple, les cas où pour un transistor N de taille Standard, la grille est à 1, à 1^- et à I sont décrits par le même tableau.

Le premier tableau a été rempli d'après les résultats de simulations SPICE (jointes en annexe 2) sur des circuits décrits par la figure 2-19, où les signaux en n1 et en n2 sont délivrés par des alimentations via des transistors adéquats. La grille du transistor est commandée par un échelon de tension, afin d'obtenir en résultat les deux états initiaux v_{f1} et v_{f2}, et les deux états finals $v_{f}'_1$ et $v_{f}'_2$.

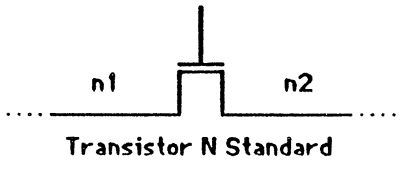


figure 2-19 : type de circuit simulé pour la détermination de &(, , N, S, 1 ou 1^- ou I).

Exemple : détermination de &(1_1, $0^+{}_1$, type N, taille S, grille 1 ou 1^- ou I) :

état(n1) = $v_{f1} = 1_1$ [V(11) dans l'annexe 2] [fig. 2 20] est délivré par un transistor P de taille standard et état(n2) = $v_{f2} = 0^+{}_2$ [V(12)] est délivré par un long transistor P (10 fois plus résistif). Les deux nouveaux états sont :

- $v_{f}'_1 = \#(1_1, 0^+{}_2) = 1_1$;
- $v_{f}'_2 = \#(1_1 - V_{thn}, 0^+{}_2) = \#(1^-{}_1, 0^+{}_2) = 1^-{}_1$.

La simulation donne les résultats suivants, qui correspondent aux calculs théoriques sur les états :

- $v_{f1} = v_{f}'_1 = 5$ V, *sauf à la fermeture du transistor central, où la ligne subit une légère aspiration vers la masse.*
- $v_{f2} = 1{,}8$ V, *interprétée dans ce cas comme un* 0^+. *Cette valeur monte ensuite après la fermeture du transistor central ;*
- $v_{f}'_2 = 2{,}9$ V, *interprétée comme un* 1^-.

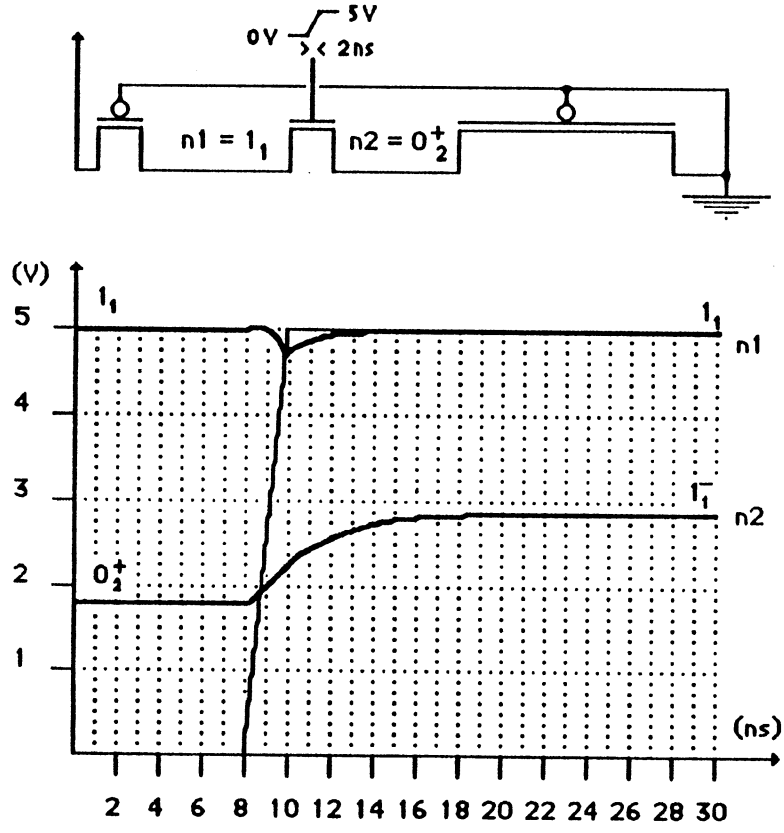


figure 2-20 : circuit simulé pour le premier tableau et résultat de la simulation SPICE.

L'opération & est ici définie par six tableaux, dont les entrées ne représentent pas la totalité de $\mathcal{A}x\mathcal{A}$. En effet, ces tableaux ne traitent que les cas où les deux lignes sont reliées à une alimentation ; les états de force 3 et 4, qui caractérisent les signaux capacitifs, ne sont donc pas concernés.

Les six tableaux traitent les cas suivants :

(i) Transistor N, de taille S (standard), commandé par 1, 1^-, I.
(ii) Transistor N, de taille L (long), commandé par 1, 1^-, I.
(iii) Transistor P, de taille S, commandé par 0, 0^+, I.
(iv) Transistor P, de taille L, commandé par 0, 0^+, I.
(v) Transistor N, de taille S ou L, commandé par 0^+.
(vi) Transistor P, de taille S ou L, commandé par 1^-.

Les tableaux étant symétriques, seule une moitié a été remplie. D'autre part, la "longueur" d'un transistor concerne non pas seulement la longueur du canal, mais le rapport L/W (longueur/largeur), qui est proportionnel à sa résistance.

(i) Transistor de type N, de taille S (standard), commandé par 1, 1^-, I

Certains choix ont été nécessaires dans plusieurs cas, à cause de l'ambiguïté des résultats électriques et, bien sûr, de l'imprécision - indispensable - du modèle.

Exemple :

pour définir &(1_1, 1_2, N, S, 1), une simulation plus précise a été nécessaire pour obtenir le résultat. Celui-ci ($v_f'_1 = 1_1$, $v_f'_2 = 1^-_1$) est en parfaite harmonie avec le modèle choisi, mais semble en désaccord avec les résultats électriques. D'un point de vue théorique, le résultat est calculé ainsi :

$$v_f'_1 = \#(1_1, 1_2) = 1_1 \; ; \; v_f'_2 = \#(1^-_1, 1_2) = 1^-_1.$$

Le circuit simulé dans ce cas est le suivant [fig. 2-21] :

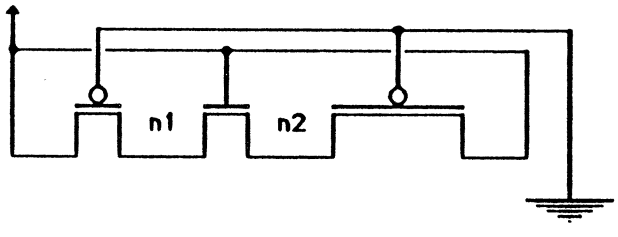


figure 2-21 : circuit simulé pour déterminer &(1_1, 1_2, N, S, 1).

Le nœud n1 est effectivement à une tension correspondant à 1, mais le nœud n2 se trouve à 5 V. Pour éclaircir ce point, on a superposé l'état de n2 avec l'état 0_2, fourni par un transistor résistif commandé par n3 [fig. 2-22] :

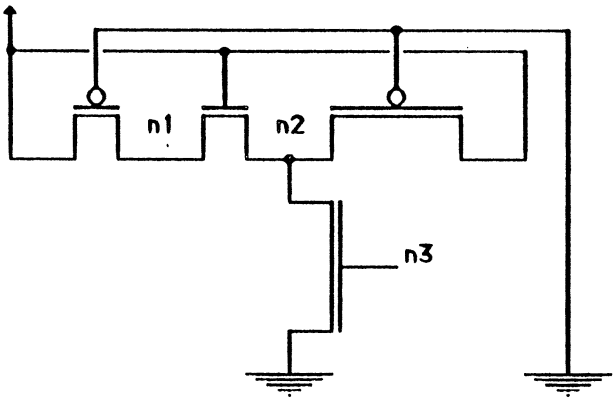


figure 2-22 : résolution d'une contradiction apparente.

Lorsque n3 est égal à 1, n2 descend brusquement, mais s'arrête à la même valeur de tension que le nœud n4 du petit circuit annexe [fig. 2-23], dont l'état est 1^-_1. L'état de n2 est donc également 1^-_1, malgré la tension de 5 V qu'il affiche. On pourrait ainsi dire que l'état de n2 est 1_2 "à concurrence de 1^-_1" : une superposition de 1_2 avec 0_2 crée momentanément une indétermination (la tension du nœud n2 baisse), mais cette indétermination cesse dès que n2 atteint 1^-. Dans le tableau, on n'a bien sûr rendu compte que de l'état 1^-_1.

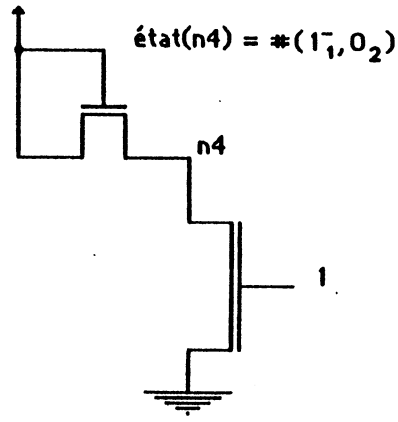
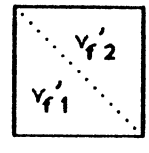


figure 2-23 : superposition de 1^-_1 et de 0_2.

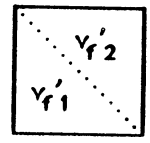
Il se peut aussi que des tensions modélisées par la valeur indéterminée soient en réalité tout-à-fait déterminées (0 V ou 5 V). Le choix dans ces conditions s'est appuyé sur l'importance des consommations de courant .

V_{f2} / V_{f1}	1_1	0_1	1^-_1	0^*_1	1_2	0_2	1^-_2	0^*_2	I_1	I_2
1_1	1_1	I_1	1^-_1	I_1	1^-_1	1^-_1	1^-_1	1^-_1	I_1	1^-_1
0_1		0_1	I_1	0_1	0_1	0_1	0_1	0_1	I_1	0_1
1^-_1			1^-_1	I_1	1^-_1	1^-_1	1^-_1	1^-_1	I_1	1^-_1
0^*_1				0^*_1	0^*_1	0^*_1	0^*_1	0^*_1	I_1	0^*_1
1_2					1_2	I_2	1^-_2	I_2	I_1	I_2
0_2						0_2	I_2	0_2	I_1	I_2
1^-_2							1^-_2	I_2	I_1	I_2
0^*_2								0^*_2	I_1	I_2
I_1									I_1	I_1
I_2										I_2



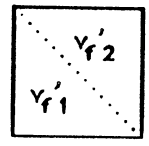
(ii) Transistor de type N, de taille L (long), commandé par 1, 1^-, I

V_{f1} \ V_{f2}	1_1	0_1	1_1^-	0_1^*	1_2	0_2	1_2^-	0_2^*	I_1	I_2
1_1	1_1	0_1 1_1	1_1^- 1_1	0_1^* 1_1	1_2 1_1	I_2 1_1	1_2^- 1_1	I_2 1_1	I_1 1_1	I_2 1_1
0_1		0_1	1_1^- 0_1	0_1^* 0_1	I_2 0_1	0_2 0_1	I_2 0_1	0_2 0_1	I_1 0_1	I_2 0_1
1_1^-			1_1^-	0_1^* 1_1^-	1_2 1_1^-	I_2 1_1^-	1_2^- 1_1^-	I_2 1_1^-	I_1 1_1^-	I_2 1_1^-
0_1^*				0_1^*	I_2 0_1^*	0_2 0_1^*	I_2 0_1^*	0_2^* 0_1^*	I_1 0_1^*	I_2 0_1^*
1_2					1_2	I_2 1_2	1_2^- 1_2	I_2 1_2	I_1 1_2	I_2 1_2
0_2						0_2	I_2 1_2^-	0_2 0_2^*	I_1 I_2	I_2 I_2
1_2^-							1_2^-	I_2 1_2^-	I_1 I_2	I_2 I_2
0_2^*								0_2^*	I_1 I_2	I_2 I_2
I_1									I_1	I_2 I_1
I_2										I_2



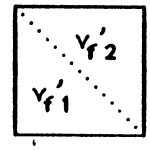
(iii) Transistor de type P, de taille S, commandé par 0, 0^+, I

V_{f1} \ V_{f2}	1_1	0_1	1_1^-	0_1^*	1_2	0_2	1_2^-	0_2^*	I_1	I_2
1_1	1_1	I_1 1_1	1_1 1_1	I_1 1_1	1_1 1_1	1_1 1_1	1_1 1_1	1_1 1_1	I_1 1_1	1_1 1_1
0_1		0_1	I_1 0_1	0_1^* 0_1	0_1^* 0_1	0_1^* 0_1	0_1^* 0_1	0_1^* 0_1	I_1 0_1	0_1^* 0_1
1_1^-			1_1^-	I_1 1_1^-	1_1^- 1_1^-	1_1^- 1_1^-	1_1^- 1_1^-	1_1^- 1_1^-	I_1 1_1^-	1_1^- 1_1^-
0_1^*				0_1^*	0_1^* 0_1^*	0_1^* 0_1^*	0_1^* 0_1^*	0_1^* 0_1^*	I_1 0_1^*	0_1^* 0_1^*
1_2					1_2	I_2 1_2	1_2 1_2	I_2 1_2	I_1 1_2	I_2 1_2
0_2	#					0_2	I_2 1_2^-	0_2^* 0_2^*	I_1 I_2	I_2 I_2
1_2^-							1_2^-	I_2 1_2^-	I_1 I_2	I_2 I_2
0_2^*								0_2^*	I_1 I_2	I_2 I_2
I_1									I_1	I_1 I_1
I_2										I_2



(iv) Transistor de type P, de taille L, commandé par 0, 0^+, I

V_{f1} \ V_{f2}	1_1	0_1	1_1^-	0_1^*	1_2	0_2	1_2^-	0_2^*	I_1	I_2
1_1	1_1	0_1	1_1^-	0_1^*	1_2	I_2	1_2	I_2	I_1	I_2
0_1		0_1	1_1^-	0_1^*	I_2	0_2	I_2	0_2^*	I_1	I_2
1_1^-			1_1^-	0_1^*	1_2	I_2	1_2^-	I_2	I_1	I_2
0_1^*				0_1^*	I_2	0_2	I_2	0_2^*	I_1	I_2
1_2					1_2	I_2	1_2	I_2	I_1	I_2
0_2						0_2	I_2	0_2^*	I_1	I_2
1_2^-							1_2^-	I_2	I_1	I_2
0_2^*								0_2^*	I_1	I_2
I_1									I_1	I_2
I_2										I_2



(v) Transistor de type N, de taille S ou L, commandé par 0^+

Le tableau est le même que pour le cas (ii), puisqu'un 0^+ en grille d'un transistor N augmente sa résistance. Il n'a pas été tenu compte, pour les cas (v) et (vi), des remarques du chapitre [2, 3112] ni de la modélisation possible des pertes de seuil dans ce cas particulier. Toutefois, cette modélisation est utilisée dans le cas où l'un des nœuds est isolé[1].

Exemple :

la résolution de &(1_1, 1^-_2, type N, taille S ou L, grille 0^+) devrait donner selon cette possibilité 1_1 pour n1 et #(0^+_2, 1^-_2) = I_2 pour n2. Des simulations SPICE ont montré la prépondérance de 1^-_2 dans ce cas.

(vi) Transistor de type P, de taille S ou L, commandé par 1^-

Le tableau est le même que pour le cas (iv).

Les tableaux décrits précédemment traitent les calculs de signaux sur les nœuds internes aux circuits. Les cas où l'un des deux nœuds est une alimentation [fig. 2-24] (l'autre nœud étant relié à une alimentation), sont décrits par les deux premières lignes des tableaux ci-dessus ($v_{f1} = 1_1$ et $v_{f1} = 0_1$), en ne considérant que la variation de l'état de n2. Vdd et Gnd sont des nœuds immuables, dont l'état est implicite. Si l'on décide que les entrées ont également une force infinie, les calculs d'états les faisant intervenir suivent le même principe.

(1) voir le chapitre 2, 4312, cas (vii).

2, 4312 L'un des nœuds est isolé

On décide qu'un nœud n1 dont on est en train de propager l'état, doit s'imposer (à condition que sa force soit 1 ou 2) sur un de ses voisins n2 lorsque celui-ci n'est pas relié de façon indépendante de n1 à une alimentation. Cela englobe les deux cas décrits dans la figure 2-24.

Sur ces deux schémas, les pointillés représentent un chemin formé de nœuds reliés entre eux par des transistors. Dans le second cas, n2 n'est pas relié à une source d'alimentation autrement que par n1 (le chemin de rebouclage n'a pas de connexion avec une alimentation). On vérifie que le nœud n2, si sa force lors de l'application de & est 1 ou 2, est isolé au sens large (tel qu'il est défini ci-dessus), par une recherche de n1 dans les variables VDD(n2) et GND(n2). S'il y existe une liste ne contenant pas n1, cela signifie que n2 n'est pas isolé ; on utilise pour appliquer l'opérateur & entre n1 et n2 le cas du chapitre [2, 4311]. Si toutes les listes contiennet n1, n2 est isolé ; si la force de n1 est 1 ou 2, son état s'impose ; si sa force est 3 ou 4, son état est confronté à l'état de n2.

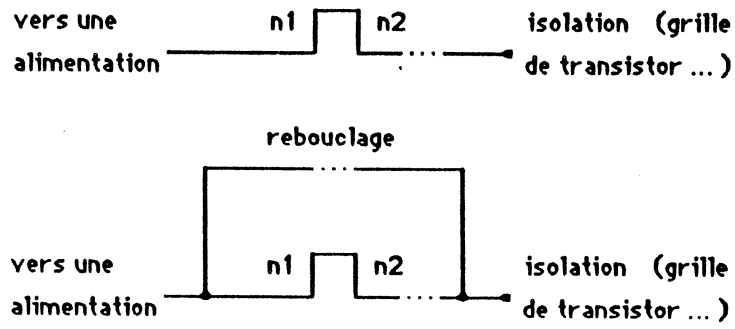


figure 2-24 : différents cas d'isolation.

Si l'un des nœuds est isolé (ici n2), l'autre (n1) peut imposer son état au cours de la propagation du signal. Le calcul, lorsque la force de n1 est 1 ou 2, suit le tableau suivant :

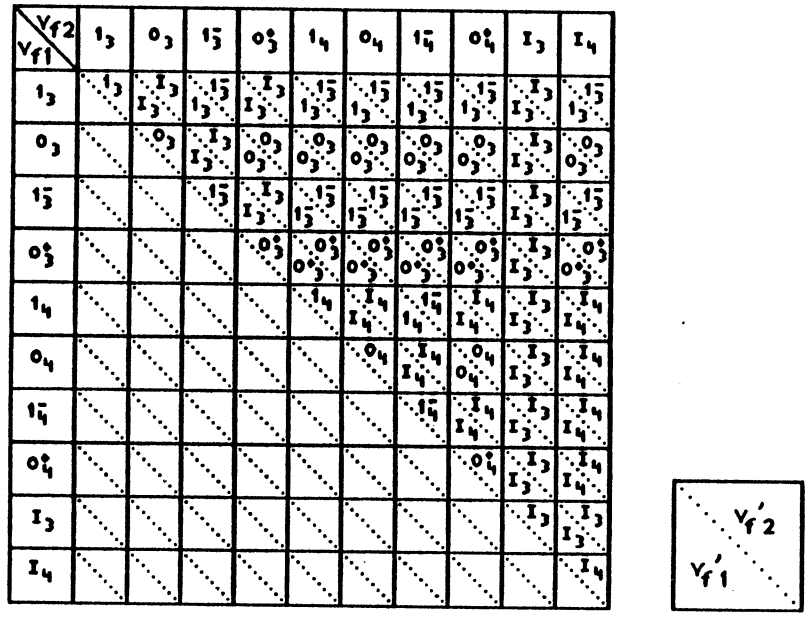
(vii)

trans	V_{f1}	1_1	0_1	1_1^-	0_1^*	1_2	0_2	1_2^-	0_2^*	I_1	I_2
NS $1,1^-,I$	$V_{f'2}$	1_1^-	0_1	1_1^-	0_1^*	1_2^-	0_2	1_2^-	0_2^*	I_1	I_2
NL $1,1^-,I$		1_2^-	0_2	1_2^-	0_2^*	1_2^-	0_2	1_2^-	0_2^*	I_2	I_2
NSL 0^*		0_2^*	0_2	0_2^*	0_2^*	0_2^*	0_2	0_2^*	0_2^*	I_2	I_2
PS $0,0^*,I$		1_1	0_1^*	1_1^-	0_1^*	1_2	0_2^*	1_2^-	0_2^*	I_1	I_2
PL $0,0^*,I$		1_2	0_2^*	1_2^-	0_2^*	1_2	0_2^*	1_2^-	0_2^*	I_2	I_2
PSL 1^-		1_2	1_2^-	1_2^-	1_2^-	1_2	1_2^-	1_2^-	1_2^-	I_2	I_2

L'ancien état de n2 est indifférent, puisque n1 s'impose. L'état X (initialisation) n'est pas concerné, car il n'est pas propageable. On ne demande de propager l'état d'un nœud à ses voisins que lorsqu'il a changé ; si c'est le cas, le nouvel état ne peut être X, puisqu'il est le plus faible de tous. Les cas où le nœud qui s'impose est une alimentation sont décrits dans le tableau par les deux premières colonnes ($v_{f1} = 1_1$ et $v_{f1} = 0_1$).

Lorsque le nœud n1 a la force 3 ou 4, le calcul est défini d'une manière analogue aux cas (i), (ii), (iii), (iv), (v) et (vi), la force 3 remplaçant la force 1 et la force 4 remplaçant la force 2. On donne seulement à titre indicatif le tableau correspondant au cas (i) :

(viii) Transistor N Standard commandé par 1, 1⁻, I



2, 432 Technologie NMOS

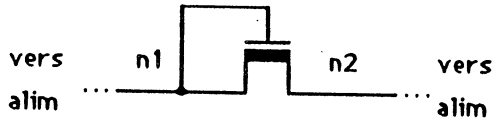
En NMOS, les transistors sont enrichis ou déplétés. La principale différence entre ces deux sortes réside dans leur résistance (plus grande pour les déplétés), et dans leur tension de seuil (négative pour les déplétés). Le calcul utilise l'ensemble $\mathcal{A}$.

On distingue le cas où les deux nœuds sont reliés à une alimentation et celui où l'un des nœuds est isolé. Pour les transistors déplétés, dans les deux cas, la grille du transistor est reliée indifféremment à l'une de ses sources, car un transistor déplété conduit toujours (tension de seuil négative). La résistance d'un transistor déplété est plus grande que celle d'un transistor enrichi[1], et occasionne un changement de force dans le signal qui le traverse.

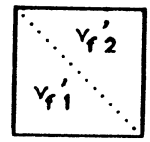
(1) voir la remarque sur le fonctionnement des circuits NMOS dans le chapitre 2, 312.

2, 4321 Les deux nœuds sont reliés à une alimentation

ix) Transistor déplété

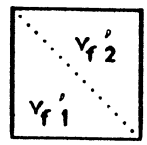


V_{f1} \ V_{f2}	1_1	0_1	1_2	0_2	I_1	I_2
1_1	1_1	0_1 1_1	1_2 1_1	I_2 1_1	I_1 1_1	I_2 1_1
0_1		0_1	I_2 0_1	0_2 0_1	I_1 0_1	I_2 0_1
1_2			1_2	I_2 I_2	I_1 I_2	I_2 I_2
0_2				0_2	I_1 I_2	I_2 I_2
I_1					I_1	I_2 I_1
I_2						I_2



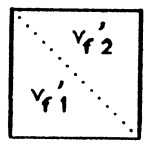
(x) Transistor enrichi, de taille S, commandé par 1, I

V_{f1} \ V_{f2}	1_1	0_1	1_2	0_2	I_1	I_2
1_1	1_1	I_1 1_1	1_1 1_1	1_1 1_1	I_1 1_1	1_1 1_1
0_1		0_1	0_1 0_1	0_1 0_1	I_1 0_1	0_1 0_1
1_2			1_2	I_2 I_2	I_1 I_2	I_2 I_2
0_2				0_2	I_1 I_2	I_2 I_2
I_1					I_1	I_1 I_1
I_2						I_2



(xi) Transistor enrichi, de taille L, commandé par 1, I

V_{f1} \ V_{f2}	1_1	0_1	1_2	0_2	I_1	I_2
1_1	1_1	0_1 1_1	1_2 1_1	I_2 1_1	I_1 1_1	I_2 1_1
0_1		0_1	I_2 0_1	0_2 0_1	I_1 0_1	I_2 0_1
1_2			1_2	I_2 I_2	I_1 I_2	I_2 I_2
0_2				0_2	I_1 I_2	I_2 I_2
I_1					I_1	I_2 I_1
I_2						I_2



2, 4322 L'un des nœuds est isolé

(xii) Transistor déplété

v_{f1}	1_1	0_1	1_2	0_2	I_1	I_2
$v_{f'2}$	1_2	0_2	1_2	0_2	I_2	I_2

(xiii) Transistor enrichi

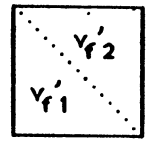
v_{f1}	1_1	0_1	1_2	0_2	I_1	I_2
$v_{f'2}$	1_1	0_1	1_2	0_2	I_1	I_2

Comme précédemment, les cas où l'un des deux nœuds est une alimentation ou une entrée sont décrits par les deux premières colonnes ($v_{f1} = 1_1$ et $v_{f1} = 0_1$).

Si la force de n1 est 3 ou 4, le calcul est décrit par le tableau suivant, pour les deux types de transistor :

(xiv)

v_{f1} \ v_{f2}	1_3	0_3	1_4	0_4	I_3	I_4
1_3	1_3	I_3	1_3	1_3	I_3	1_3
0_3		0_3	0_3	0_3	I_3	0_3
1_4			1_4	I_4	I_3	I_4
0_4				0_4	I_3	I_4
I_3					I_3	I_3
I_4						I_4



L'utilisation de ces tableaux décrivant l'opération & dans différents cas de figures, les recoupements à effectuer, les simplifications d'emploi à y apporter ne concernent que la programmation et l'implémentation du simulateur, ce qui n'est pas le but de ce travail. Celui-ci se restreint à une définition "propre" et surtout logique d'une algèbre multivaluée et d'un algorithme d'évaluation des états, qui s'adaptent à la simulation "switch", et dont chaque élément a sa raison d'être qui correspond à une réalité physique dans le fonctionnement des circuits VLSI.

2, 5 Modèle temporel

Si l'on désire inclure dans le simulateur un modèle temporel, on peut le faire de quatre manières principales différentes :

- on effectue un calcul précis en utilisant des formules caractéristiques. Les calculs que l'on peut trouver dans un simulateur électrique sont trop compliqués pour être applicables ici : il serait absurde de dater les évènements précisément sur un ensemble de tensions discrètes, dont la détermination reste par essence approximative. Les programmes de calculs temporels existants (Crystal [OUS83], ...) qui sont moins compliqués que ceux des simulateurs électriques, le sont encore trop pour ce que l'on désire.

- on effectue des approximations importantes sur l'utilisation des formules électriques des délais, afin de diminuer le temps de compilation. On s'aperçoit vite que, si les transistors sont traités un par un et non au sein d'un groupe de connexion, l'imprécision doit être trop importante pour que la complexité du calcul soit justifiée, par rapport au nombre d'états de l'algèbre utilisée.

- on utilise une algèbre spéciale employée pour le calcul des délais uniquement, qui en plus des niveaux 0 et 1, comporte des niveaux intermédiaires représentant les niveaux de marge de bruit de la technologie, dont l'influence sur les caractéristiques dynamiques des circuits est importante [BAY84].

- on suppose que les délais qui concernent les transistors sont unitaires, comme dans le simulateur MOSSIM II [BRY84]. Cela signifie qu'un transistor change d'état une unité de temps δ après le changement de sa grille : par exemple, si la valeur de la grille d'un transistor N passe de 0 à 1, les nouveaux états de part et d'autre du transistor sont calculés après le temps δ, qui est la plus petite valeur temporelle possible, et correspond du point de vue algorithmique à une étape de calcul. Si la grille passe de 1 à 0, le transistor devient bloqué après le temps δ. Cette méthode a l'avantage d'être rapide, bien que ne permettant de détecter que des aléas grossiers. De plus, les caractéristiques individuelles des composants n'ont pas besoin d'être connues, et la gestion du temps, parce qu'elle ne s'appuie sur aucune valeur réelle, n'est pas assujettie à une contingence technologique particulière.

Les quatre chapitres qui suivent donnent des précisions sur ces quatre solutions, et exposent les points forts et les points faibles de chacune, relativement à l'emploi qui en serait fait en simulation multivaluée. La quatrième méthode est retenue pour la simulation normale qui ne demande pas de calculs temporels précis. Dans le cas où certains problèmes temporels se posent ou peuvent se poser, l'emploi d'une méthode plus précise est prévue.

2, 51 Calculs précis des délais

Il existe plusieurs programmes de calculs purement temporels ayant chacun pour objectif d'être plus rapide qu'un simulateur électrique comme SPICE, tout en gardant une précision convenable ([AUV87], [LIN84], [MO 84], [OKA83], [OUS83], [OUS84]). Ils s'appuient en majorité sur le calcul des temps de propagation des réseaux RC. Les types de calcul utilisés sont en tout cas trop compliqués pour servir en simulation multivaluée dans le cadre d'utilisation défini précédemment(1), les transistors étant traités isolément.

2, 52 Calculs avec approximations

Deux approximations dans le cas présent sont possibles : soit on suppose que la charge des capacités des circuits se fait à tension constante, et on obtient des résultats utilisant $e^{-t/RC}$, soit on suppose qu'elle se fait à intensité constante. Cette dernière solution est plus proche de la réalité, un transistor en charge ou décharge fonctionnant comme une source de courant et non comme une source de tension. Les calculs en découlant sont exposés dans la suite, pour la technologie NMOS et pour la technologie CMOS. Le but est de calculer dans chaque cas un temps de montée (charge), et un temps de descente (décharge). Dans le modèle, on ne prend en compte que les capacités de grille, les autres étant supposées négligeables. Il faut donc calculer un temps de montée et un temps de descente typiques qu'il suffira de rajouter à la date d'un évènement pour en déduire la date du résultat.

2, 521 Technologie NMOS

2, 5211 Calcul de t_m

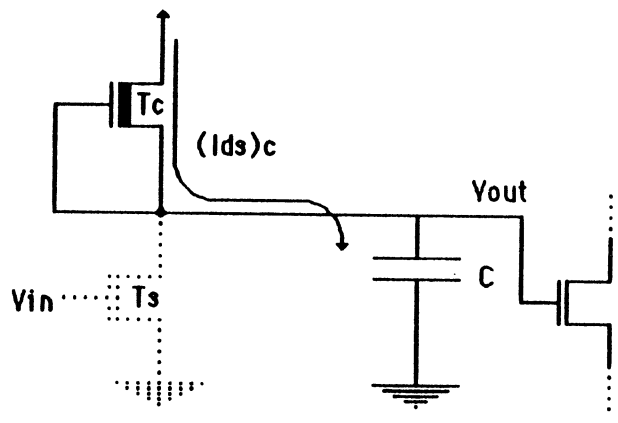


figure 2-25 : détermination du temps de montée.

De façon simplifiée, la charge de la capacité de sortie d'un inverseur NMOS se fait en deux temps [fig. 2-25] :

(1) voir le chapitre 2, 1.

- le transistor de charge T_c est saturé, jusqu'à ce que la tension V_{out} en sortie atteigne V_{dd}- $|V_{thc}|$ (V_{dd} est la tension d'alimentation, V_{thc} la tension de seuil du transistor de charge). En effet, il y a saturation si $V_{out} < V_{dd}$- $|V_{thc}|$. La valeur de l'intensité du courant traversant T_c et chargeant la capacité C est donnée par :

$$(I_{ds})_c = 1/2\, \gamma_c K_c (V_{thc})^2$$

avec $K_c = \mu_c.Cox$ (μ_c est la mobilité des porteurs du transistor de charge et Cox la capacité poly/oxyde mince/substrat).

- T_c passe en régime linéaire ($V_{out} > V_{dd}$- $|V_{thc}|$). L'intensité du courant dans ce cas est donné par :

$$(I_{ds})_c = 1/2\, \gamma_c K_c [2\,|V_{thc}|\,(V_{ds})_c - (V_{ds})_c^2]$$

où $(V_{ds})_c$ est la tension aux bornes de T_c : $(V_{ds})_c = V_{dd}$- V_{out}.

Ici, une approximation de t_m suffit. On suppose donc que le transistor de charge est toujours saturé. On utilise la première relation pour le calcul.

La charge de la capacité est donnée par $dQ = C\, dV_{out} = i\, dt$, d'où $dt = C/i\, dV_{out}$, soit :

$$t_m = \int_0^{t_m} dt = \frac{2C}{\gamma_c K_c} \int_{V_l}^{V_h} \frac{dV_{out}}{(V_{thc})^2}$$

$$t_m = \frac{2C\,(V_h - V_l)}{\gamma_c\, K_c\, (V_{thc})^2}$$

V_l est la tension basse où commence le calcul ; V_h est la tension haute correspondant à l'instant où l'on suppose que la charge est finie. On prend habituellement $V_l = 10\%\ V_{dd}$ et $V_h = 90\%\ V_{dd}$.

Les données numériques choisies sont celles de la technologie NMOS/CMP85 dont certaines valeurs sont arrondies :

$V_{thc} \approx -1{,}5$ V
$C = S_s.Cox$ (S_s est la surface du transistor signal attaqué)

$\mu_c = 8{,}75 \cdot 10^{-2}\ m^2/V.s$
$\mu_s = 8 \cdot 10^{-2}\ m^2/V.s$

Dans la technologie NMOS/CMP85, le rapport γ_s/γ_c préconisé est égal à 2, 25, on choisit ainsi $\gamma_c = 2\lambda/3\lambda$ et $\gamma_s = 3\lambda/2\lambda$, avec $\lambda = 2{,}5\ \mu m$, qui sont des valeurs "ordinaires". La surface S_s est alors égale à $37{,}5\ \mu m^2 = 37{,}5 \cdot 10^{-12}\ m^2$.

On obtient avec ces valeurs :

$$t_m = 4{,}44 \cdot \frac{S_s}{\mu_c \cdot \gamma_c} \approx 2{,}9 \text{ ns.}$$

2, 5212 Calcul de t_d

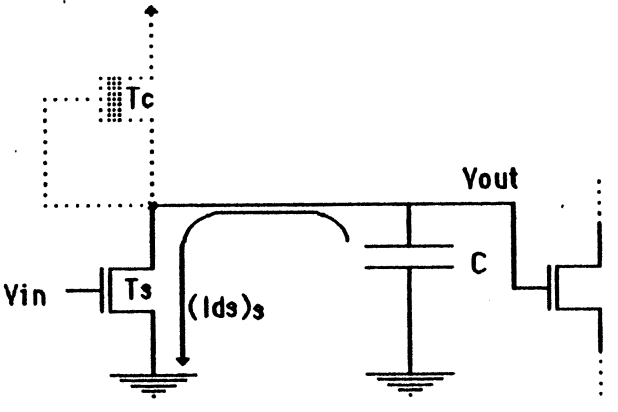


figure 2-26 : détermination du temps de descente.

Pour avoir une approximation du temps de descente, on suppose le transistor signal toujours saturé, et le courant $(I_{ds})_c$ négligeable [fig. 2-26]. La capacité se décharge en créant un courant vers la masse à travers T_s, dont l'intensité est donnée par :

$$(I_{ds})_s = 1/2\ \gamma_s K_s (V_{in} - V_{ths})^2$$

On a donc :

$$t_d = \int_0^{t_d} dt = \int_{V_1}^{V_h} \frac{C\ dV_{out}}{1/2\ \gamma_s K_s (V_{in} - V_{ths})^2}$$

$$t_d = \frac{2C(V_h - V_l)}{\gamma_s K_s (V_{in} - V_{ths})^2}$$

$V_{ths} \approx 0{,}9$ V
$V_{in} = 5$ V

$$t_d = 0{,}6 \frac{S_s}{\gamma_s K_s} \approx 0{,}19 \text{ ns.}$$

2, 522 Technologie CMOS

2, 5221 Calcul de t_m

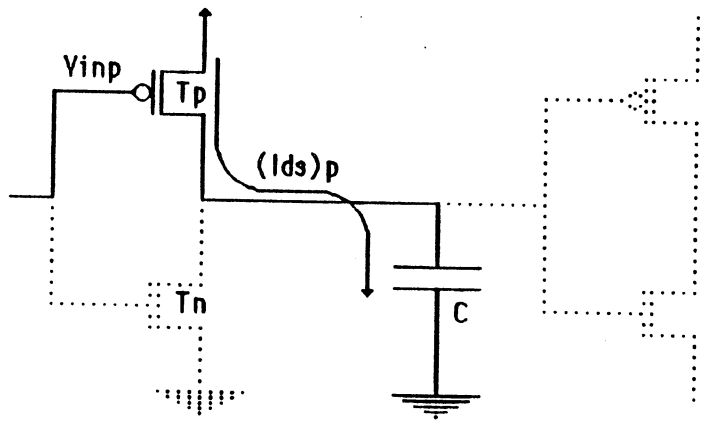


figure 2-27 : détermination du temps de montée.

Pour avoir une approximation de t_m [fig. 2-27], on suppose que T_p est toujours saturé et que T_n est toujours bloqué :

$$(I_{ds})_p = 1/2 \gamma_p K_p (V_{dd} - V_{inp} - |V_{thp}|)^2$$

$$t_m = \frac{C (V_h - V_l)}{(I_{ds})_p} = \frac{S \cdot Cox \cdot (V_h - V_l)}{1/2 K_p \gamma_p (V_{dd} - V_{inp} - |V_{thp}|)^2}$$

$V_h = 4{,}5$ V
$V_l = 0{,}5$ V
$V_{inp} = 0{,}5$ V
$V_{inn} = 4{,}5$ V
$1/2 K_p = 10^{-5}$ A/V^2

$1/2\, K_n = 3 \cdot 10^{-5}\ A/V^2$
$\gamma_p = 12/3\ \gamma_n = 4/3$
$Cox = 10^{-3}\ F/m^2$
$V_{thp} = -0{,}75\ V$
$V_{thn} = 0{,}75\ V$
$S = S_n + S_p$ (surfaces de T_n et de T_p)

$$t_m = 7{,}1 \cdot S \approx 0{,}34\ ns.$$

2, 5222 Calcul de t_d

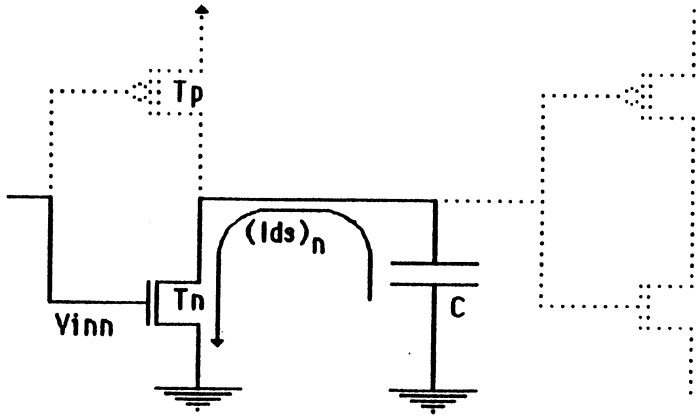


figure 2-28 : détermination du temps de descente.

Pour avoir une approximation de t_d [fig. 2-28], on suppose que T_n est toujours saturé et que T_p est toujours bloqué :

$$(I_{ds})_n = 1/2\, \gamma_n K_n (V_{inn} - V_{thn})^2$$

$$t_d = \frac{C (V_h - V_l)}{1/2\, K_n \gamma_n (V_{inn} - V_{thn})^2}$$

γ_n et γ_p sont choisis pour que $K_n\gamma_n = K_p\gamma_p$; $V_{inn} - V_{thn} = V_{dd} - V_{inp} - |V_{thp}|$. Donc $t_d = t_m$:

$$t_d = 0{,}34\ ns.$$

2, 523 Limites de la méthode

Les quatre temps calculés précédemment ne sont valables qu'à titre indicatif. En effet, plusieurs facteurs les font varier :

- la technologie : les caractéristiques données sont celles de la technologie CMP85 NMOS et CMOS. Les valeurs d'autres technologies sont différentes, et il convient d'en tenir compte dans l'utilisation du simulateur.

- le dimensionnement : celui qui intervient dans les calculs est en quelque sorte un dimensionnement minimal. Mais on peut trouver localement des transistors plus gros dont la capacité de grille est très différente de celles obtenues ici. Pour avoir des résultats utilisables, il faut donc tenir compte en application de ces différences de taille.

- la longueur des connexions : celles-ci peuvent avoir une capacité importante à partir d'une certaine longueur.

- la sortance des portes, qui influe directement sur leur capacité de sortie.

Pour que les temps de montée et de descente soient plus réels, deux solutions sont envisageables :

- on permet à l'utilisateur du simulateur d'entrer en données les caractéristiques technologiques et dimensionnelles des composants en se servant des formules les utilisant. Notamment, on donne la possibilité d'introduire les capacités de ligne par des calculs appropriés, pour les ajouter aux capacités de grille.

- on augmente arbitrairement les temps précédemment calculés pour que leur marge de correction soit plus importante. Par exemple :

- NMOS : $t_m = 5$ ns ; $t_d = 1$ ns,

- CMOS : $t_m = t_d = 1$ ns.

On peut également fondre ces deux solutions en une troisième, où les temps habituels sont constants, mais peuvent être augmentés s'il y a lieu.

2, 53 Utilisation d'une algèbre spéciale

Cette méthode ne concerne pas la technologie NMOS, car les niveaux de marge de bruit MB_H et MB_B ne peuvent y être introduits aisément. En effet, ces deux valeurs dépendent de la géométrie des transistors (W, L) et des caractéristiques de la technologie. Par contre, pour la technologie CMOS, les niveaux de marge de bruit sont beaucoup plus indépendants de la géométrie des transistors et des caractéristiques technologiques. Ainsi, on peut prendre, en gardant une bonne précision $MB_H = MB_B = V_{dd}/2$. L'algèbre utilisée

comporte ainsi trois valeurs $\{V_{dd}, 0, V_{dd}/2\}$. On définit un temps de propagation τ pour un inverseur :

$$\tau = \frac{C \cdot V_{dd}}{4 \beta (V_{dd}/2 - V_{th})^2}$$

où C est la capacité de sortie, $V_{th} = |V_{thn}| = |V_{thp}|$, et $\beta = \mu C_{ox} \cdot W/2L$ [WHI82].

Avec ce modèle, une chaîne d'inverseurs génère une forme d'onde qui ne dépend que du gain des transistors de la chaîne et pas de la forme du signal d'entrée, lorsque le régime stationnaire est atteint [fig. 2-29]. Sur le chronogramme, l'axe des abscisses représente le temps, et chacune des courbes décrit la sortie d'un inverseur. A chaque étage de la chaîne, l'inverseur attaqué commute lorsque la sortie de l'inverseur précédent atteint $V_{dd}/2$.

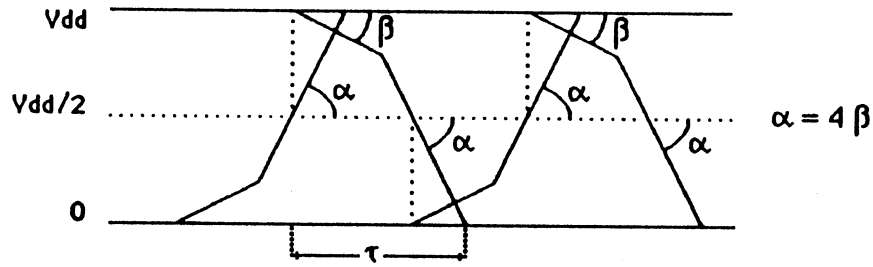


figure 2-29 : forme d'onde propagée par une chaîne d'inverseurs.

2, 54 Calculs avec délais unitaires

Dans ce cas, on utilise toujours une variable de temps, dont la valeur sera associée à un événement donné. L'intervalle de temps δ cité plus haut se réduit simplement à la valeur 1, que l'on ajoute à la date de l'événement courant pour obtenir la date de l'événement immédiatement postérieur. Par exemple si la grille d'un transistor change d'état au temps n, le transistor changera d'état (bloqué ou passant) au temps n+1. Cette méthode ne permet pas de détecter la plupart des pannes de délais, mais sa simplicité justifie son utilisation lors de l'évaluation des états.

Exemple :

soit une porte NAND [fig. 2-30] dont une entrée est retardée par une suite d'inverseurs. Cela crée un aléa en sortie parfaitement détecté (la sortie f passe à 0 pendant deux unités de temps), illustré par le chronogramme de fonctionnement utilisant δ.

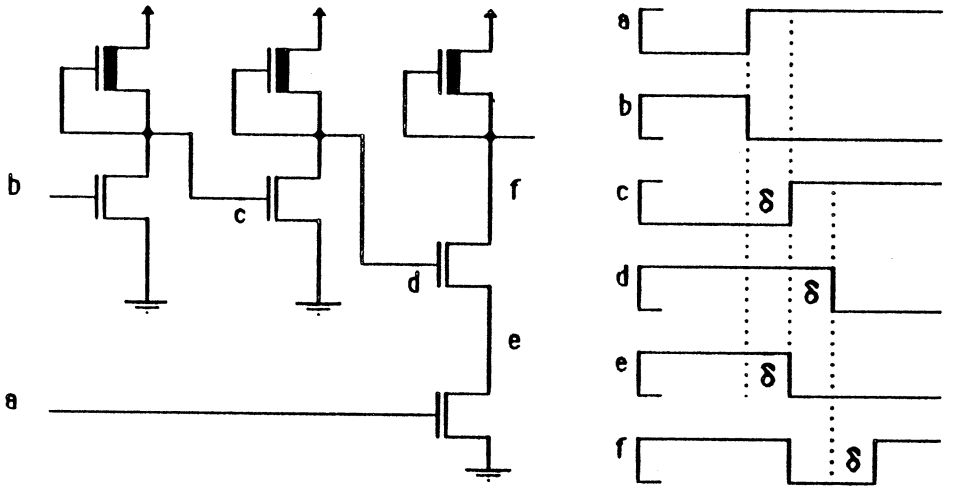


figure 2-30 : aléa détecté à la sortie d'une porte NAND.

Il existe par contre des cas où une panne possible n'est pas détectée, à cause de la grossièreté du modèle à délai unitaire.

Exemple :

soient deux circuits réalisant la même fonction, mais implantés différemment (un inverseur change de place). L'un des deux crée un problème [fig. 2-31] inexistant dans l'autre [fig. 2-32]. Pour chacun des deux circuits, on donne le schéma du circuit, et le chronogramme correspondant. Dans le premier cas, la possibilité de court-circuit en • ne sera pas détectée. Pourtant, dans la réalité, un petit décalage serait inévitable à cause des tailles de transistors et des longueurs de lignes différentes, pouvant causer un court-circuit.

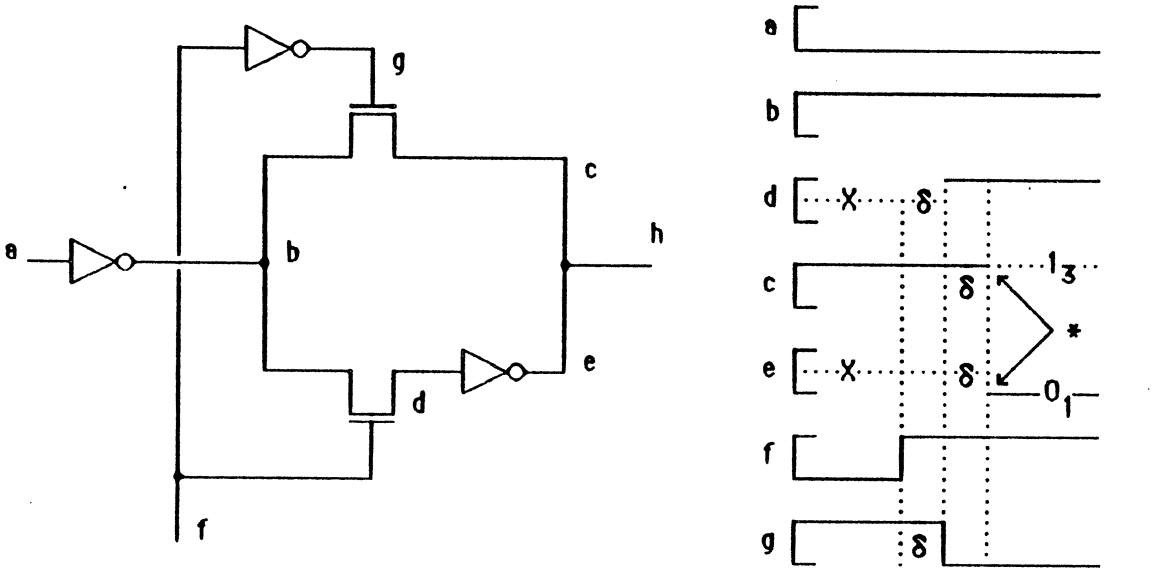


figure 2-31 : la possibilité de court-circuit n'est pas détectée.

Dans le second cas, aucun problème ne se pose, le nœud c étant isolé avant que le nœud e ne passe à 0. La sortie de la porte prend la valeur donnée par la superposition de 1_3 et de 0_1, soit 0_1.

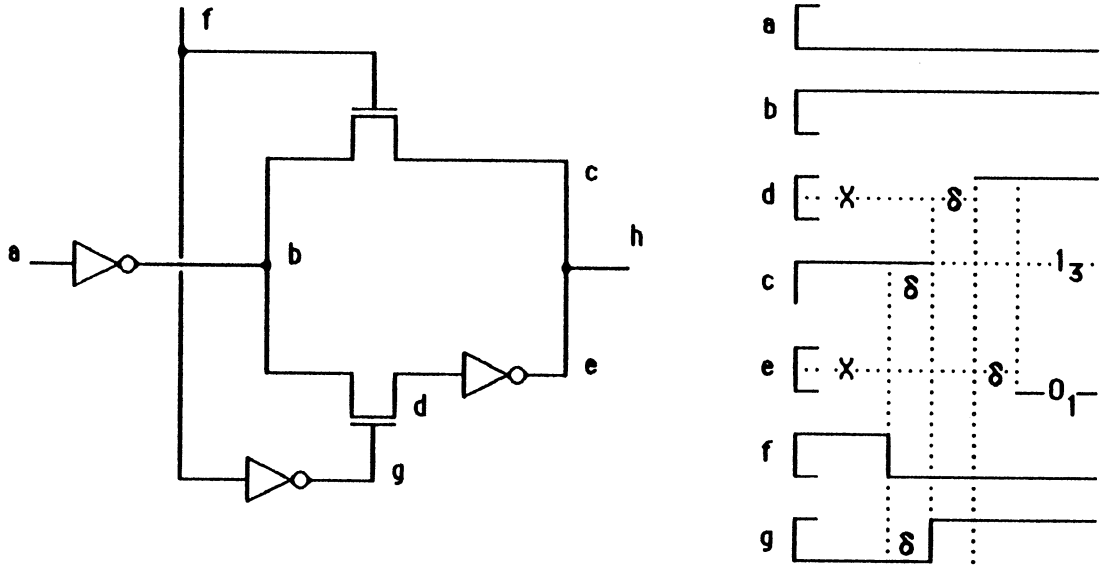


figure 2-32 : exemple sans possibilité de court-circuit.

Quoiqu'il en soit, ce modèle temporel paraît le plus adapté à la simulation "switch" multivaluée comme elle a été définie, du moins pour le calcul des états. L'imprécision qui en résulte est en tout cas équivalente à celle de l'algèbre d'états, et est équilibrée par la vitesse de calcul obtenue.

Pour un calcul plus fin des délais, la troisième méthode est utilisée, car elle allie la simplicité de calcul à une bonne précision. Un tel calcul peut être accompli sur de petites parties lorsque le calcul avec des délais unitaires fait surgir un problème demandant une simulation plus fine.

2, 6 Description succincte de l'algorithme

L'algorithme est orienté-évènement ("event-driven"), et est fondé sur la recherche d'un état stable. On initialise le circuit à simuler en affectant l'état X à tous les nœuds et à tous les transistors. Le type de description d'un circuit s'apparente à celui de SPICE, les caractéristiques technologiques en moins :

Nom_transistor source1 grille source2 type taille.

D'après cette description, on crée pour chaque nœud deux listes :

- voisinage(n) = ((n, g1, n1, T1), (n, g2, n2, T2), ...),
- influence(n) = ((n1, n, n'1, T1), (n2, n, n'2, T2), ...),

décrivant respectivement les nœuds ni qui sont reliées à n via un transistor Ti, et les paires de nœuds ni et n'i reliées entre elles par un transistor Ti dont n est la grille. Ces listes servent par la suite pour la propagation des signaux : voisinage(n) est utilisée pour leur propagation au temps courant (il n'y a pas de délai associé à la "traversée" d'un transistor), influence(n) est utilisée pour leur propagation au temps courant + 1 (un délai unitaire est affecté à la commutation passant-bloqué et bloqué-passant d'un transistor). Lorsqu'un nœud change d'état, celui-ci est propagé, d'abord via les transistors dont il est une source (voisinage), puis par l'intermédiaire des transistors dont il est la grille (influence). Cette propagation consiste à appliquer de façon adéquate l'opérateur & (en fonction de l'isolation ou des connexions aux alimentations des nœuds concernés, par l'étude des listes VDD et GND), et à mettre à jour les variables VDD et GND concernées. Il y a trois sortes de propagations :

- "propager_vois_état" qui propage l'état et les variables VDD et GND d'un nœud vers ses voisins ;
- "propager_infl_état" qui propage l'état et les variables VDD et GND d'un nœud vers ceux qu'il influence ;
- "propager_alim", qui se contente de propager les variables VDD et GND (lorsque l'état du nœud ne change pas, mais qu'une nouvelle connexion à une alimentation est créée).

Un exemple de simulation joint en annexe 3 montre le déroulement de l'algorithme et le traitement naturel des signaux bi-directionnels et des boucles.

2, 7 Modélisation de dispositifs spéciaux

Certains phénomènes ne peuvent être modélisés par le modèle tel qu'il est défini. Ce chapitre est consacré à la description des caractéristiques qu'il faudrait ajouter à celui-ci pour pouvoir simuler quelques dispositifs particuliers qui sont parfois présents dans un circuit. Que ce soit le "bootstrap" (boucle de charge), les effets de partage de charges ou les résistances des lignes, ces particularités entraînent la modélisation des capacités et des résistances.

2, 71 Modélisation des capacités

Une capacité se représente de deux manières [fig. 2-33] : soit elle est introduite dans le dessin par le concepteur sous la forme d'un transistor, soit elle est présente dans le circuit de manière implicite ; dans ce cas, elle n'est représentée par aucun objet réel dans le circuit. Le premier type correspond à une capacité "voulue" du circuit (utilisée dans les "bootstrap"),

le second à une capacité parasite que l'on peut vouloir préciser ou prévoir pour affiner la simulation. Dans le premier cas, la capacité dépend de la taille du transistor utilisé, et peut être calculée automatiquement selon les caractéristiques de la technologie pour le calcul des délais. Dans le second cas, la valeur de la capacité doit être précisée pour être prise en compte dans la simulation.

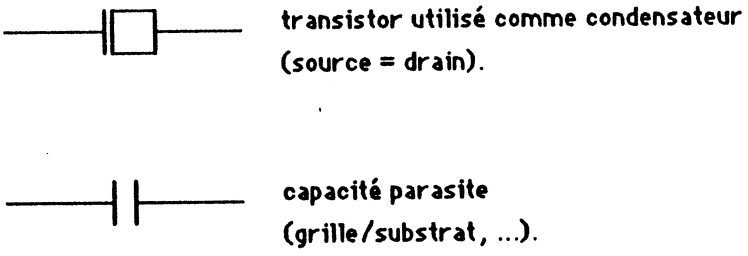


figure 2-33 : deux types de capacités.

En ce qui concerne le calcul des états, la capacité, selon sa valeur, induit une force pour le signal qu'elle est appelée à délivrer temporairement. Par exemple, le problème du partage de charges entre deux capacités, qui n'est pas une panne mais une erreur de conception, oblige à considérer plusieurs forces capacitives (on a définit précédemment les forces 3 et 4) ; de même, la modélisation de l'influence de grosses capacités sur d'autres signaux peut demander d'associer les forces 1 et 2 aux signaux capacitifs.

Exemple :

la figure 2-34 représente un registre connecté à un bus. Suivant la valeur de la capacité, le signal qu'elle délivre lorsqu'on lit le registre peut anihiler le signal correct venant du registre. On retrouve donc les mêmes forces que pour les signaux habituels : 1, 2, 3.

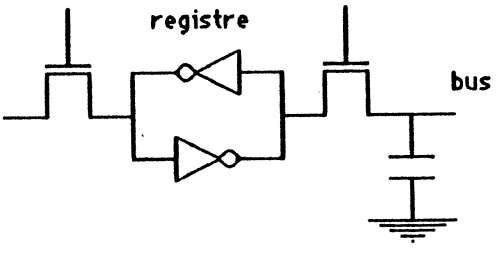


figure 2-34 : problème de partage des charges.

Pour ce phénomène, l'utilisateur peut soit supposer que le signal correct s'imposera finalement (le signal capacitif prend alors la force 3 ou 4), soit décider de mettre en évidence le problème posée par la capacité (le signal capacitif prend alors la force 1 ou 2).

Un effet important des capacités est le couplage capacitif. Cela se produit dans un circuit lorsqu'une capacité parasite est présente entre deux lignes (par exemple, une ligne d'aluminium et une ligne de polysilicium). Dans ce cas, la tension d'une des lignes peut influer sur la tension de l'autre à cause de la capacité qui maintient l'écart entre ces deux tensions : si l'une monte, l'autre monte aussi (et peut dépasser 5 V), si l'une descend, l'autre descend aussi (et peut devenir négative).

Ces effets particuliers (utilisés dans le "bootstrap") demandent pour être modélisés une algèbre dont les valeurs peuvent représenter toutes les tensions entières entre -10 V et +10 V au moins. De plus, le calcul des nouveaux états est particulier, car il doit respecter l'écart de tension qui existe entre les deux pôles de la capacité[1]. Cela entraîne une complexité supplémentaire aggravée par le fait que l'algèbre n'est plus un treillis[2]. Pour ces raisons, la modélisation des couplages capacitifs n'est pas envisagée ici. On se restreint aux cas où l'un des pôles de la capacité est une alimentation.

Le calcul temporel utilisant un délai unitaire ne peut prendre que difficilement en compte une capacité. En effet, une capacité, parasite ou non, induit une variation dans le temps de montée et de descente de la porte située en amont. Il faut donc, à partir des nœuds séparés par la capacité, retrouver les transistors dont il faut modifier le délai de commutation, en fonction de l'état de charge ou de décharge de celle-ci. Le problème est analogue si le calcul temporel utilise des formules plus précises.

2, 72 Modélisation des résistances

Bien qu'on ne puisse calculer la résistance des nœuds (problème posé par les nœuds composés), on peut vouloir spécifier des résistances pour la simulation. Ces résistances ne doivent être spécifiées que si elles peuvent modifier le signal en force, comme le fait un transistor long, et en valeur (V = Ri). Les autres correspondent à des résistances trop faibles pour être simulées avec une algèbre multivaluée.

On peut ainsi considérer les cas donnés dans la figure 2-35. Les petites résistances sont inutilisées, mais elles peuvent être éventuellement modélisées par un transistor long. En effet, les résistances "intéressantes" sont celles qui modifient notablement l'état des nœuds. Le calcul des états de part et d'autre des résistances utilise aussi l'opérateur &, avec les mêmes conditions d'emploi. L'ajout d'une résistance dans un réseau oblige à rajouter aussi un nœud.

(1) voir le fonctionnement des capacités dans la remarque sur le modèle CSA du chapitre 1, 33.

(2) voir les calculs de convergence des calculs sur les algèbres qui sont des treillis du chapitre 3, 21222 et sur les algèbres qui ne sont pas des treillis du chapitre 3, 21224.

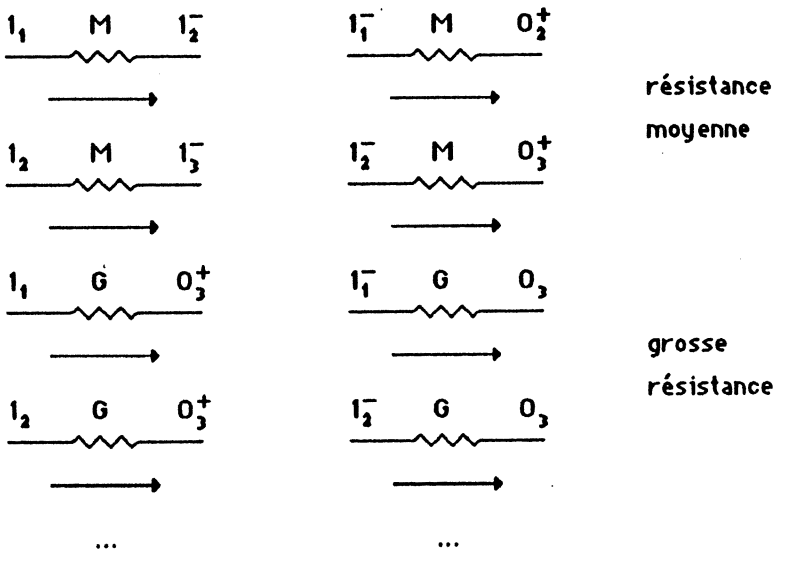


figure 2-35 : une proposition de traitement des résistances.

2, 73 Modélisation du "bootstrap" (NMOS)

Le schéma du "bootstrap" est donné par la figure 2-36.

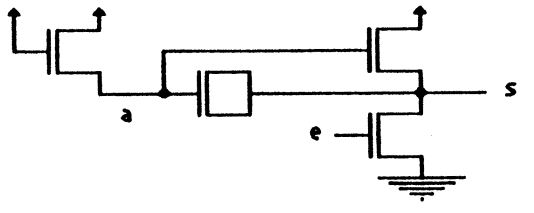


figure 2-36 : schéma du "bootstrap".

C'est un inverseur dont le temps de montée est diminué. Le nœud a, lorsque e = 1, a une tension de 5 - V_{th} . Le transistor placé en capacité entre a et s, au moment où e := 0, garde l'écart de 5 - V_{th} entre a et s. Le transistor de charge conduit tout de suite, créant une augmentation de tension sur s, qui elle-même crée une augmentation de tension sur a, qui peut ainsi monter jusqu'à 7, 5 V environ. Le transistor de charge est alors commandé en grille par une tension de plus en plus élevée, permettant une montée plus rapide de s. Il n'y a pas de perte de seuil sur le transistor de charge, car la tension finale de a est 7, 5 V. Ainsi :

$$V_{GS} \geq V_{th}, \text{ donc } V_S \leq V_G - V_{th} = 7{,}5 - 1 = 6{,}5 \text{ V, donc } V_S = 5 \text{ V.}$$

Une simulation SPICE de ce dispositif montre une différence de 1 ns environ entre le temps de montée obtenu avec le "bootstrap" et celui d'un inverseur standard déplété-enrichi. Sur des inverseurs ayant une grosse capacité de sortie, cet écart devient conséquent.

Pour modéliser le "bootstrap", il faut :

- modéliser le couplage de tension entre les deux pôles ;

- modéliser la valeur de 7,5 V ;

- modéliser la conduction d'un transistor ayant une telle valeur en grille ;

- ajouter une force modélisant celle d'un signal 5 V délivré par un transistor enrichi, qui est inférieure à celle d'un signal 0 V délivré par un transistor enrichi.

Ces différentes contraintes font abandonner la modélisation du dispositif de "bootstrap" au niveau interrupteur. En effet, son fonctionnement est trop sensible pour que le concepteur soit satisfait de résultats trop vagues, et il doit en tout cas le simuler avec un simulateur électrique. On prévoit donc la possibilité, au niveau interrupteur, de remplacer le "bootstrap" par un inverseur normal. Si le calcul temporel n'utilise pas de délai unitaire, le concepteur peut imposer des caractéristiques dynamiques pour cet inverseur, qui peuvent être celles obtenues par la simulation électrique.

CHAPITRE 3

SIMULATION DE PANNES MULTIVALUEE AU NIVEAU "SWITCH"

3 Simulation de pannes multivaluée au niveau "switch"

Depuis que le test des circuits VLSI est devenu crucial, la simulation de pannes a pris beaucoup d'importance, et représente maintenant une étape fondamentale de la conception. En effet, la simulation de pannes doit fournir pour la phase de test un ensemble d'entrées et de résultats qui soit compatible avec les possibilités et la précision du test. Ainsi, le modèle choisi pour représenter les circuits VLSI et leurs pannes doit l'être à bon escient.

Cette partie est consacrée à l'extension du simulateur spécifié jusqu'à présent, à la simulation des pannes, et plus précisément, à l'évaluation des signaux au sein de circuits comportant une panne (les aspects informatiques de la simulation de pannes et la gestion des listes de pannes ne sont pas traités). Le niveau de représentation des circuits VLSI est toujours le même, c'est-à-dire qu'ils sont sous la forme de réseaux d'interrupteurs. La technologie CMOS est seule concernée par cette étude, mais la technique de simulation qui y est décrite peut être aisément appliquée à la technologie NMOS. La première partie expose brièvement la nécessité de modéliser les pannes à ce niveau plutôt que d'utiliser des modèles existant à un niveau supérieur (logique, ...). La seconde partie présente trois techniques de modélisation, étudiées spécialement dans le but d'être intégrées au niveau "switch". Enfin, la troisième partie donne la liste des pannes à considérer dans le modèle.

3, 1 Motivations de la modélisation des pannes au niveau "switch"

Bien que le simulateur doive s'inscrire dans un ensemble hiérarchique d'outils de conception, il serait illusoire d'essayer, au niveau logique, de déterminer toutes les pannes pouvant survenir au niveau interrupteur. En effet, il n'y a pas assez d'informations dans la description d'une porte logique pour en déduire la correspondance entre les pannes possibles au niveau logique et les pannes possibles dans la représentation de cette porte sous forme de réseau d'interrupteurs. La raison principale de cette limitation est l'absence de bijection entre les lignes équipotentielles des deux niveaux : une ligne peut exister au niveau logique et ne pas exister au niveau interrupteur, et inversement.

Exemple :

soit une porte logique et sa représentation au niveau interrupteur [fig. 3-1]. Les lignes existant aux deux niveaux portent le nom ci, celles n'existant qu'au niveau logique portent le nom li, celles n'existant qu'au niveau interrupteur portent le nom ni. L'impossibilité de représenter certaines pannes logiques au niveau inférieur est évidente : ainsi, un collage sur la ligne l1 ne peut être simulé dans le réseau d'interrupteurs. Inversement, le nœud n1, qui peut être le siège d'un collage, n'a pas de correspondant au niveau logique.

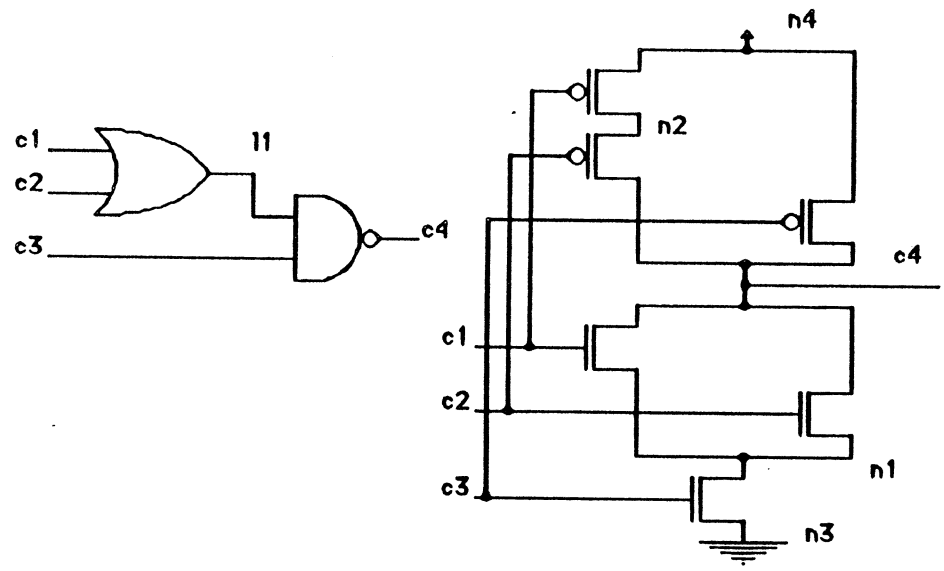


figure 3-1 : une porte logique et sa représentation en technologie CMOS.

Il existe ainsi nécessairement une différence entre la couverture des pannes des deux niveaux : une séquence de vecteurs peut couvrir toutes les pannes dans un circuit logique et ne pas couvrir toutes les pannes dans le réseau de transistors correspondant.

Exemple :

pour résumer cette situation, on donne une fonction logique avec sa représentation en réseau d'interrupteurs [fig. 3-2].

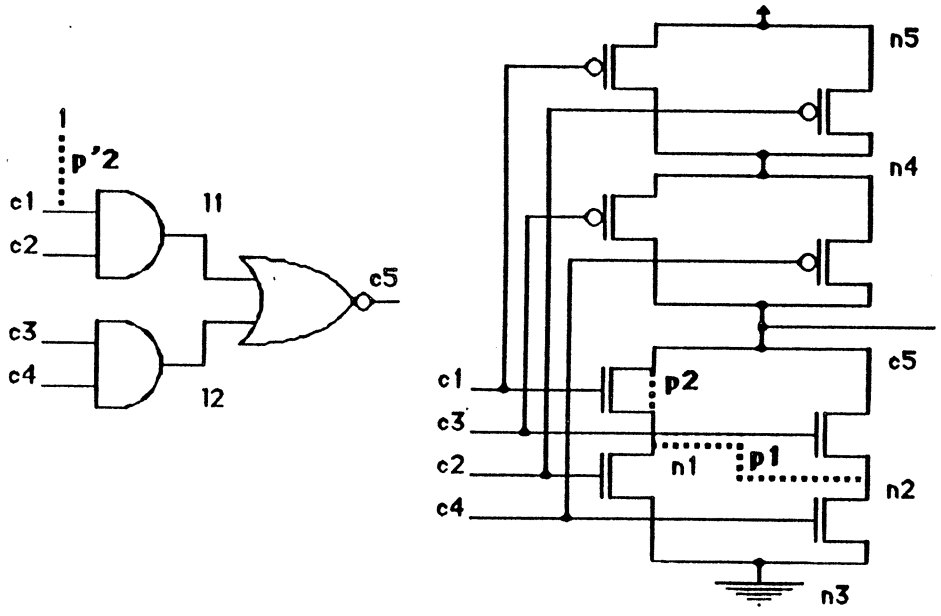


figure 3-2 : un exemple de pannes et leurs correspondances aux deux niveaux.

c1	c2	c3	c4	c5	c5 (p1)
0	1	1	0	1	Ind.
1	0	0	1	1	Ind.

∄p1'

figure 3-3 : mise en évidence de la panne p1 au niveau interrupteur uniquement.

c1	c2	c3	c4	c5	c5 (p2)	c5 (p2')
0	1	0	0	1	Ind.	0
0	1	0	1	1	Ind.	0
0	1	1	0	1	Ind.	0

figure 3-4 : mise en évidence des pannes p2 et p'2.

Le court-circuit entre n1 et n2 (panne p1) ne peut être ni représenté, ni couvert par aucune séquence de test obtenue au niveau logique. Par contre le collage-passant du transistor T1 (panne p2) peut être représenté au niveau logique par un collage à 1 de l'entrée c1 (panne p'2). Les figures 3-3 et 3-4 donnent les séquences de test qui mettent ces trois pannes en évidence.

Une autre différence entre les deux niveaux réside dans la "précision" des pannes. Par exemple, le collage d'un transistor peut être éventuellement simulé au niveau logique [fig. 3-4], mais il ne pourra être fait aucune variation sur cette conduction anormale. Alors qu'au niveau logique, la modélisation de cette panne ne permet que deux états pour les transistors (passant ou bloqué modélisés par le collage à 0 ou à 1 des lignes correspondantes, suivant le type du transistor), il est possible de moduler le collage au niveau interrupteur avec un modèle de résistance, comme il est exposé par la suite.

Toutes ces différences justifient pleinement qu'un modèle de circuit soit construit pour le niveau interrupteur, en vue de la simulation de pannes. Le modèle est une extension du modèle précédent[(1)], certaines caractéristiques étant modifiées pour qu'il s'adapte à la simulation de pannes.

3, 2 Modélisation des circuits pour la simulation de pannes

En fonction de la précision et de la correction désirées pendant la simulation de pannes, on peut utiliser trois modèles de réseaux d'interrupteurs, depuis l'adaptation du modèle défini jusque-là au calcul des valeurs électriques d'un circuit à l'aide d'équations caractéristiques. Ces trois méthodes sont décrites dans ce chapitre.

(1) ce modèle est exposé dans le chapitre 2.

3, 21 Adaptation du modèle multivalué

Ce modèle comprend une algèbre multivaluée de 21 états décrite précédemment :

$$\{1, 0, 1^-, 0^+, I\}x\{1, 2, 3, 4\} \cup \{X\}.$$

L'algorithme de calcul des signaux traite les transistors un par un, en déterminant pour chacun les nouveaux états des deux sources (opérateur &). Cette méthode évite tout traitement préliminaire du réseau visant à déterminer les boucles et les dispositifs bi-directionnels. Ceux-ci sont traités naturellement et restent transparents pour l'algorithme de calcul. Pour la simulation de pannes, ce modèle possède un inconvénient majeur : il ne prend pas en compte les dispositions en parallèle et en série des transistors. Il ne permet donc pas de simuler efficacement les réseaux pour lesquels la résistance des transistors doit être prise en compte.

3, 211 Transistors en parallèle et en série

Cette limitation est rédhibitoire pour la simulation des circuits dans lesquels existe, à cause d'une panne, un chemin de conduction entre les alimentations.

Exemple :

soit le circuit de la figure 3-5, dans lequel un collage-passant sur un transistor N peut créer un chemin de conduction entre Vdd et Gnd. Dans ce cas, la tension de sortie dépend en grande partie des résistances présentes sur ce chemin.

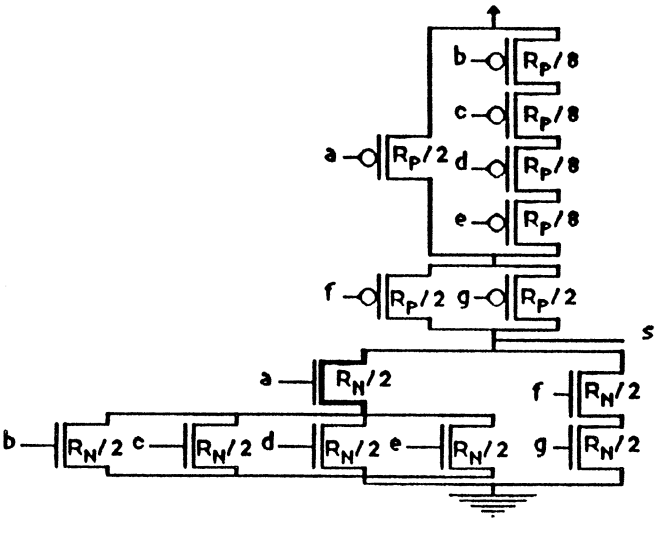


figure 3-5 : le transistor Na est collé-passant.

La panne du circuit est d'abord traitée d'une manière théorique, de façon que son effet soit évalué. Ensuite, lui est comparé le résultat d'une simulation utilisant le modèle précédent, de manière à mettre en valeur son manque de précision. Suivant le vecteur d'entrée affecté à la porte, il peut y avoir une différence de résistance entre le réseau N et le réseau P. Si le réseau N est bien moins résistif que le réseau P, la sortie s peut prendre une valeur proche de 0, au lieu de la valeur 1 qu'un circuit sans panne fournirait. Parmi tous les vecteurs qui peuvent être affectés aux entrées, deux seulement sont étudiés ici ; ce sont les deux vecteurs qui induisent dans la porte le plus grand et le plus petit rapport $R_{charge}/R_{décharge}$ *(ils peuvent être appelés vecteurs "extrêmes") :*

- *(a) (abcdefh) = (0111101),*

- *(b) (abcdefg) = (0100000).*

Le rapport $R_{charge}/R_{décharge}$ *, pour chacun des deux vecteurs, est égal à :*

- *(a)* $R_P/(R_N/2 + R_N/8) = 8/5\ R_P/R_N$

- *(b)* $(R_P/2 + R_P/4)/R_N = 3/4\ R_P/R_N$

où R_N *(resp.* R_P*) est la résistance d'un transistor N (resp. P) dans un inverseur standard. On suppose que le dimensionnement des portes est fait de telle manière que les réseaux N et P n'aient jamais une résistance supérieure à* R_N *et* R_P*. Les tailles des transistors sont ainsi adaptées à partir de leur valeur standard, selon que les transistors sont en série ou en parallèle [fig. 3-6].*

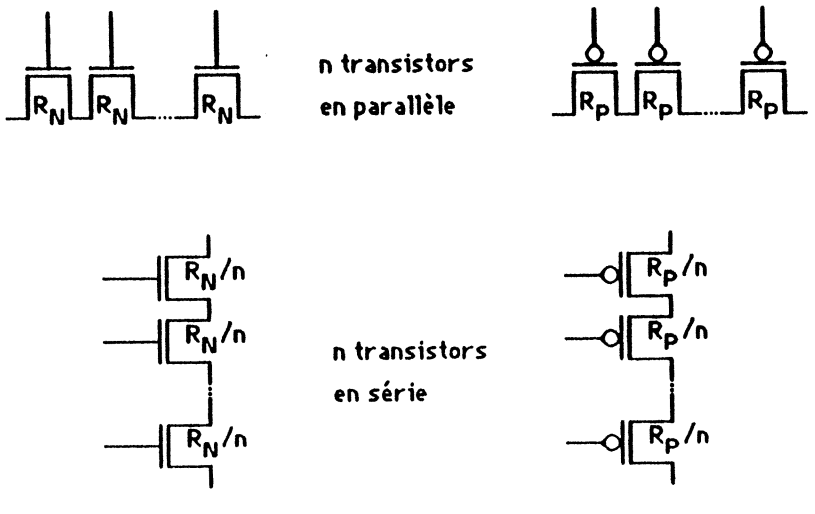


fig. 3-6 : adaptation du dimensionnement des transistors en fonction du réseau.

Le rapport R_P/R_N est fixé à 1 de façon que les temps de montée et de descente d'un inverseur soient égaux (résultat SPICE). Le niveau théorique en sortie est donc :

$$\text{- (a) } V_s = \frac{1}{1+R_{dt}/R_{déch}} V_{dd} = \frac{1}{1+8/5\,R_P/R_N} V_{dd} = 5/13\ V_{dd} = 1,9\ V ;$$

$$\text{- (b) } V_s = \frac{1}{1+R_{dt}/R_{déch}} V_{dd} = \frac{1}{1+3/4\,R_P/R_N} V_{dd} = 4/7\ V_{dd} = 2,86\ V.$$

Ces deux valeurs sont calculés avec l'hypothèse que les transistors se comportent comme des résistances pures, ce qui n'est pas tout-à-fait exact : notamment, les transistors saturés ne relèvent pas de cette hypothèse. Quoiqu'il en soit, une simulation SPICE a permis de retrouver approximativement ces valeurs, et surtout de les vérifier par rapport à leur effet sur un inverseur placé sur la sortie de la porte. Ainsi, la première valeur est reconnue par cet inverseur comme un 0 (sortie anormale), alors que la seconde est reconnue comme un 1 (sortie normale).

Ici intervient l'importance du modèle. En effet, en appliquant le modèle déjà défini, on trouve que dans les deux cas, la sortie s est au niveau indéterminé I (superposition de 1_I et de 0_I). Cette valeur ne permet pas de décider si les deux vecteurs choisis sont vecteurs de test (c'est-à-dire, d'être sûr qu'ils mettront la panne en évidence au cours du test) ou non, puisqu'en réalité, seul le premier vecteur est un vecteur de test ; au cours du test, il devra être seul utilisé pour tester la panne. Donc la valeur I, et plus globalement l'algèbre et l'algorithme définis jusqu'à présent, sont insuffisants pour une bonne simulation de panne.

Plus précisément, la différence essentielle entre une simulation normale (de circuits sans panne) et une simulation de pannes réside dans la valeur indéterminée I. Dans le cas d'un court-circuit créant un chemin de conduction entre les alimentations dans un circuit normal (créé non par une panne mais par une erreur de conception), il est souhaitable que la simulation normale donne en résultat la valeur I pour montrer qu'il y a une erreur. Par contre, cette valeur I devient indésirable dans le type de simulation décrit ici, puisqu'il ne s'agit plus de détecter une erreur inconnue mais de mettre en évidence une panne connue. Cet essai de mise en évidence permet alors de choisir parmi les vecteurs d'entrée ceux qui deviennent vecteurs de test. Il faut donc construire un nouveau modèle qui permette d'obtenir des résultats, non pas indéterminés (valeur I), mais bien déterminés (qui reflètent la "vision" qu'à le reste du circuit vis-à-vis de ce résultat).

Exemple :

Pour le circuit, les résultats doivent être (a) valeur(s) = 0 et (b) valeur(s) = 1.

Dans le but d'améliorer dans le sens voulu le modèle existant, un outil de reconnaisance des structures parallèles est associé à l'opérateur &, modifiant la représentation de la résistance des transistors lors des calculs. Cet outil affecte ainsi à chaque transistor un facteur de "parallélisme" p adapté, en fonction du réseau dynamique auquel il appartient (les transistors bloqués ne sont pas inclus dans cette recherche dynamique). Ces facteurs sont inclus dans le calcul des signaux, en remplaçant les facteurs de résistance r des transistors[1].

3, 212 Enrichissement de l'algèbre d'états

3, 2121 Calcul des forces

La résistance des transistors est prise en compte dans le modèle, selon la configuration de la figure 3-7. Le nœud n2 est isolé (il n'a pas de connexion vers Vdd ou Gnd), et le signal se propage de n1 vers n2. Le nœud n1 impose donc son état à n2, à des pertes de force et de valeur près.

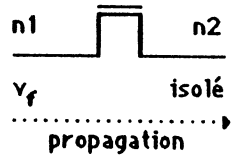


figure 3-7 : configuration de base pour la prise en compte des résistances.

Si la force de n1 est f_1, celle de n2 est après la propagation :

$$f'_2 = (f_1) = f_1 + r,$$

où r est le facteur de résistance du transistor, et dépend du type, de la taille, et de la valeur de grille de celui-ci. On rappelle que la force d'un signal est inversement proportionnel à la valeur de la force : par exemple, la plus grande force dans l'ensemble initial $F_4 = \{1, 2, 3, 4\}$ est 1. Cet ensemble est bien sûr modifié dans le modèle : l'opération d'addition ci-dessus n'est cohérente que dans un ensemble possédant suffisamment d'éléments, et qui est fermé quels que soient les opérandes impliqués. L'ensemble des forces est donc :

$$F = \mathbb{N}.$$

(1) la prise en compte de p est décrite dans le chapitre 3, 2121.

Dans cet ensemble, la force 0 est la plus grande, par exemple celle des alimentations Vdd et Gnd ; cette force diminue donc à chaque passage du signal "provenant" de Vdd ou de Gnd dans un transistor, car le facteur de résistance r est ajouté à chaque fois.

Cet effet cumulatif des facteurs de résistance autorise la prise en compte naturelle des séries de transistors. Par contre, les dispositions en parallèle des transistors demandent une modélisation particulière, en dehors de l'outil servant à les détecter. Cette modélisation consiste en un facteur de parallélisme p, qui est introduit dans le calcul comme suit :

$$f'_2 = (f_1) = f_1 + p.$$

Exemple :

le facteur de parallélisme de trois transistors en parallèle ayant un facteur de résistance r, est égal à p = [r/3] (la notation entre crochets indique que le résultat est arrondi à l'entier le plus proche).

Le calcul des forces doit également permettre d'inclure la taille des transistors. Le facteur de résistance r dépend de celle-ci de la façon suivante :

- un facteur de résistance r_s est affecté à un transistor dont la taille est standard (notée W_s/L_s);

- un facteur de résistance $[r_s/n]$ est affecté à un transistor dont la taille W/L est égale à n fois W_s/L_s (n pouvant être plus petit que 1) ; dans ce cas, la nouvelle force de n2 est :

$$f'_2 = (f_1) = f_1 + [r_s/n],$$

Enfin, le facteur r d'un transistor dépend de la valeur de la grille du transistor. Cela concerne les cas où un transistor N (resp. P) est commandé par un 0^+ (resp. 1^-). Le choix du facteur multiplicatif sur r_s qui traduit l'augmentation de résistance du transistor dans ces deux cas, est déduit de simulations SPICE. Ainsi, on prend un facteur égal à 24[1]. Le nombre r_s est choisi selon des critères de précision : il doit pouvoir être divisé suffisamment au gré des dispositions parallèles, sans pour autant se rapprocher exagérément de 0 ; le nombre maximal de transistors en parallèle que l'on rencontre dans un circuit est de l'ordre de 10 ; le nombre $r_s/10$ obtenu doit pouvoir encore être divisé en fonction du rapport W/L du transistor et néanmoins garder une utilité arithmétique. On prend ainsi $r_s = 100$.

(1) les simulations donnent comme résultat les tensions 0, 2 V et 4, 8 V en sortie d'un inverseur commandé par un 1^- et par un 0^+. Ces valeurs, après la résolution de l'équation donnant la tension de sortie d'un pont diviseur entre Vdd et Gnd, induisent les rapports 24/1 et 1/24 entre les deux résistances du pont.

3, 2122 Calcul des états

Le calcul des états utilise toujours l'opérateur & décrit précédemment. Son application est précédée des mêmes vérifications sur l'isolation ou la connexion aux alimentations des deux nœuds impliqués. Le calcul exposé ici, parce qu'il est le plus intéressant, concerne le cas où les deux nœuds sont reliés aux alimentations [fig. 3-8].

figure 3-8 : disposition avant l'application de &.

3, 21221 Exposé de la solution retenue

Dans le modèle défini précédemment, les nouveaux états $v_f'_1$ et $v_f'_2$ sont calculés en même temps, en fonction des états précédents v_{f1} et v_{f2}, au moyen de tableaux décrivant l'opération & pour chaque cas, et n'utilisant qu'implicitement l'opérateur #. Ce qui était possible grâce au nombre limité d'états ne l'est plus à présent : l'ensemble des forces n'autorise plus l'utilisation de tableaux, mais oblige à calculer les nouveaux états par l'utilisation explicite de #. Cette utilisation est adaptée au nouveau modèle, avec notamment l'introduction de la notation $\{v_f\}$, qui caractérise un état modifié lors de son passage dans un transistor. Ainsi, si le nœud n2 est isolé, l'état de n1 s'impose et le nouvel état de n2 est calculé selon la règle suivante :

$$v_f'_2 = \{v_{f1}\} = (\{v_1\}, \{f_1\}),$$

où $\{v_1\}$ est la valeur de n1, après les ajustements éventuels (perte de seuil, grille à un niveau altéré, ...) exigés par le type, la taille et la grille du transistor[1], et où $\{f_1\}$ est calculée selon les règles exposées précédemment.

Dans le cas où les deux nœuds sont reliés à une alimentation, les nouveaux états sont calculés comme suit :

$$v_f'_1 = \#(v_{f1}, \{v_{f2}\}),$$

$$v_f'_2 = \#(v_{f2}, \{v_{f1}\}).$$

(1) voir le chapitre 2, 311.

L'opération #, qui normalement est appliquée dans les treillis, est ici adaptée au nouveau modèle. En effet, le critère principal de décision dans un treillis est la différence de force : un état de force i est plus fort qu'un état de force i+1. Dans le cas présent, cette simple différence doit être remplacée par une autre opération plus complexe, qui prenne en compte le nouvel ordre de grandeur des forces (r_s = 100) qui ne peut se satisfaire d'une précision à l'unité. En outre, cette opération ne doit pas se contenter de la différence absolue entre les forces pour décider de la prépondérance de tel ou tel état ; elle doit utiliser comme critère la différence relative entre les forces, qui est plus précise (par exemple, la différence relative est moins grande entre 1000 et 990 qu'entre 20 et 10, bien que la différence absolue soit la même).

C'est pourquoi les règles de décision utilisent le rapport entre les forces. Les nouveaux états sont calculés de la manière suivante :

- si $\{f_2\}/f_1 \geq q$, alors $v_f'_1 = v_{f1}$; *(1)*

- si $\{f_2\}/f_1 \leq 1/q$, alors $v_f'_1 = \{v_{f2}\}$; *(2)*

- si $1/q < \{f_2\}/f_1 < q$, alors $v'_1 = \#(v_1, \{v_2\})$ et $f'_1 = f_1$; *(3)*

- si $\{f_1\}/f_2 \geq q$, alors $v_f'_2 = v_{f2}$; *(4)*

- si $\{f_1\}/f_2 \leq 1/q$, alors $v_f'_2 = \{v_{f1}\}$; *(5)*

- si $1/q < \{f_1\}/f_2 < q$, alors $v'_2 = \#(v_2, \{v_1\})$ et $f'_2 = f_2$, *(6)*

avec $\{f_1\} = f_1 + r$ et $\{f_2\} = f_2 + r$ (r est le facteur de résistance du transistor).

Le paramètre q représente le rapport minimal qui doit exister entre les deux forces pour que l'un des nœuds impose son état sur l'autre. Si le rapport entre les forces est trop proche de 1 (compris entre 1/q et q), le nouvel état du nœud concerné est calculé en utilisant la hiérarchie au sein de l'algèbre.

Le choix de la valeur de q est assez délicat : si q est trop grand, l'opérateur # (cas (3) et (6)) sera appliqué souvent, et trop de valeurs indéterminées I pourront être propagées, nuisant à l'utilité de la simulation ; si q est trop petit, les conflits réels seront mal modélisés. On a choisi de considérer une tension comme indéterminée lorsqu'elle se situe entre 2, 2 V et 2, 8 V. Ainsi, la tension 2, 8 V doit être modélisée par un 1 et la tension 2, 2 V doit être modélisée par un 0. La tension 2, 8 V est créée par un pont diviseur de deux résistances dont le rapport est 5/2, 8 - 1 = 0, 79 (1/q), et la tension 2, 2 V est créée par un pont diviseur de

deux résistances dont le rapport est 5/2,2 - 1 = 1, 27 (q). C'est pourquoi on pose dans la suite q = 1, 30 (cette valeur induit comme niveau haut minimal la tension 2, 83 V et comme niveau bas maximal la tension 2, 17 V) ; cette valeur remplit parfaitement les deux impératifs ci-dessus.

Exemple :

soient $v_{f1} = 1_{1000}$, $v_{f2} = 0_{800}$, $r = 100$, *et* $q = 1, 30$ $(1/q = 0,77)$. *On a :*

- $(f_2)/f_1 = (800 + 100)/1000 = 0, 9$ *(cas (3)),*
- $(f_1)/f_2 = (1000 + 100)/800 = 1, 375$ *(cas (4)), donc :*

- $v'_1 = \#(1, 0) = I$ *et* $f_1 = 1000$;
- $v'_2 = 0$ *et* $f_2 = 800$.

Les nouveaux états obtenus I_{1000} *et* 0_{800} *sont définitifs, car ce calcul converge en une étape ; si l'on applique l'opération ci-dessus à ces deux états, le résultat est le même.*

L'opérateur # utilisé dans les cas (3) et (6) réalise la hiérarchie entre les valeurs de l'algèbre. Il n'est utilisé que pour la superposition d'états ayant la même force, c'est-à-dire pour lesquels le rapport entre les forces est compris entre 1/q et q. Cette hiérarchie est donnée par la figure 3-9.

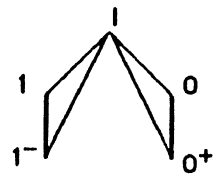


figure 3-9 : hiérarchie entre les valeurs à une même force.

Ce treillis est légèrement différent de ceux décrivant précédemment cette algèbre[1] : la valeur X n'est pas représentée ici, car elle n'est pas concernée par le cas présent ; la valeur I de la force inférieure n'est pas non plus représentée, pour la même raison.

3, 21222 Discussion sur la convergence des calculs

La convergence rapide du calcul décrit dans l'exemple ci-dessus est nécessaire dans ce modèle, dont l'objectif est de représenter les tensions, les courants et leurs interactions de façon discrète.

(1) voir les chapitres 2, 3113 et 2, 32.

Il est intéressant de noter que seul un petit détail permet cette convergence rapide : si, dans les cas (3) et (6), on définissait les nouvelles forces comme suit :

- si $1/q < \{f_2\}/f_1 < q$, alors $v'_1 = \#(v_1, \{v_2\})$ et $f'_1 = \min(f_1, \{f_2\})$; *(7)*

- si $1/q < \{f_1\}/f_2 < q$, alors $v'_2 = \#(v_2, \{v_1\})$ et $f'_2 = \min(f_2, \{f_1\})$, *(8)*

il y aurait une possibilité pour que le rapport entre les deux nouvelles forces soit inférieur à ce qu'il était avant le calcul, autorisant une nouvelle itération.

Exemple :

pour les mêmes données que dans l'exemple précédent, on aurait :

- $v'_1 = \#(1, 0) = 1$ et $f'_1 = \min(1000, 800 + 100) = 900$; $v'_2 = 0$ et $f'_2 = 800$.

Si on répète le calcul, on obtient :

- $\{f_2\}/f_1 = (800 + 100)/900 = 1$ (cas (7)) ;
- $\{f_1\}/f_2 = (900 + 100)/800 = 1,25$ (cas (7)) ;
- $v'_1 = \#(1, 0) = 1$ et $f'_1 = \min(900, 800 + 100) = 900$ (pas de changement) ;
- $v'_2 = \#(1, 0) = 1$ et $f'_2 = \min(800, 900 + 100) = 800$.

Pour obtenir cet état stable, qui d'ailleurs est inexact, il a donc fallu au moins deux itérations. Il faut ainsi éviter de "rapprocher" les deux forces l'une de l'autre dans les cas (3) et (6).

La convergence des calculs est aisément démontrable. Dans la solution retenue, les nouvelles forces ne peuvent avoir qu'un nombre limité de valeurs :

- $f'_1 = f_1$ et $f'_2 = f_2$ (cas (1), (3), (4), (6)) ; *(9)*

- $f'_1 = f_1$ et $f'_2 = f_1 + r$; $v'_2 = \{v_1\}$ (cas (5)) ; *(10)*

- $f'_1 = f_2 + r$ et $f'_2 = f_2$; $v'_1 = \{v_2\}$ (cas (2)) ; *(11)*

Les deux cas ((2) et (5)) où les nouvelles forces peuvent se "rapprocher" l'une de l'autre (leur quotient diminue) ne posent pas de problème car alors les valeurs v'_1 et v'_2 sont égales (à un ajustement près) et une nouvelle itération ne produirait aucun changement.

Les deux forces ne peuvent changer en même temps (les cas (2) et (5) sont exclusifs), car on aurait simultanément :

- $(f_2 + r)/f_1 \leq 1/q$
- $(f_1 + r)/f_2 \leq 1/q$,

ce qui est impossible :

- $(f_2 + r)/f_1 \leq 1/q \Rightarrow f_1 \geq q(f_2 + r) \Rightarrow q(f_1 + r) \geq q^2 f_2 + q^2 r + qr > f_2$ $(q>1, r>0)$
- $(f_1 + r)/f_2 \leq 1/q \Rightarrow q(f_1 + r) \leq f_2$ (les deux termes son exclusifs).

De l'énoncé des trois cas ci-dessus, on peut déduire immédiatement que :

- le calcul converge en une itération dans le cas (9) car les nœuds ne changent pas de force : un second test donnerait le même quotient. De plus, en ce qui concerne les cas (3) et (6), l'opération # est conçue pour être "idempotente"[1] :

$$\#(v_1, \#(v_1, \{v_2\})) = \#(v_1, \{v_2\}), \quad \#(v_2, \#(v_2, \{v_1\})) = \#(v_2, \{v_1\}).$$

- le calcul converge en une itération pour les cas (10) et (11) car bien que le rapport des nouvelles forces soit égal à 1 (cas (3) et (6)), l'application de # n'amène aucun changement :

$$\#(\{v_2\}, \{v_2\}) = \{v_2\}, \quad \#(\{v_1\}, \{v_1\}) = \{v_1\}.$$

Ainsi, le calcul des nouveaux états par l'application de & converge toujours en une itération, grâce au choix des cas (3) et (6) au lieu des cas (7) et (8). Ce choix induit d'ailleurs quelques imprécisions (négligeables) dans la résolution des nouveaux états.

L'imprécision de la méthode retenue est suffisamment petite pour être négligée, d'autant qu'elle ne survient que dans les cas où les deux forces en présence sont "proches" l'une de l'autre : qu'on prenne l'une ou l'autre pour le résultat est sans importance. La méthode retenue est plus avantageuse, car le calcul converge toujours en une étape.

Exemple :

La circuit décrit par la figure 3-10 contient des transistors de type P et N ; les états hauts qui sont créés sont donc égaux à 1 puis à 1. Le facteur de résistance des transistors

(1) la notion d'idempotence d'une fonction d'un ensemble E vers le même ensemble E est ici élargie aux fonctions de ExE vers E et de ExE vers ExE.

est égal à 100. Pendant la première phase de l'évaluation (jusqu'à ce que le dernier transistor devienne passant), les états s'imposent de haut en bas.

Ensuite, le dernier transistor devient passant, induisant une évaluation du rapport des forces en jeu. Les calculs successifs des états (de bas en haut) sont indiqués sur la figure. L'évaluation s'arrête au nœud n6, qui ne change pas d'état.

Avec la méthode prévoyant les cas (7) et (8), l'état de n7 et de n8 est I_{700}.

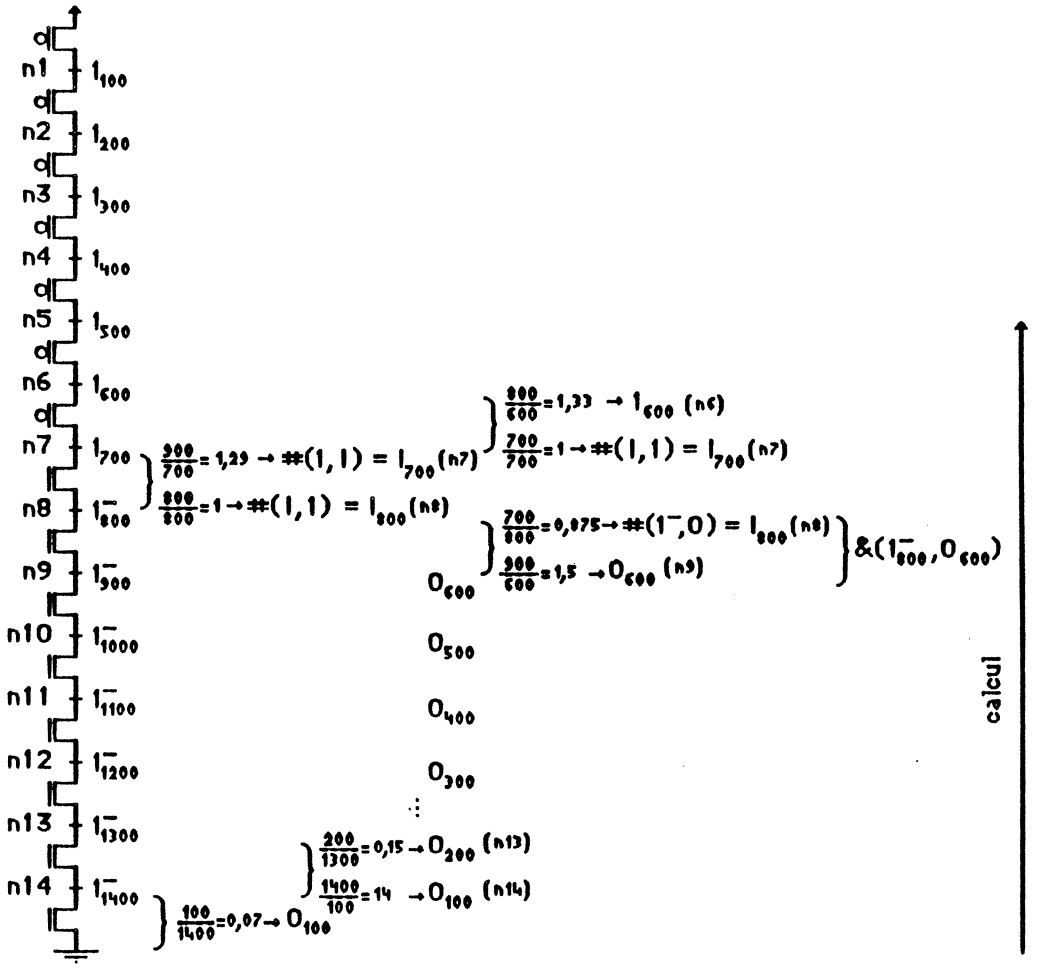


figure 3-10 : exemple de détermination des états.

3, 21223 Discussion sur la précision des calculs

Contrairement à la différence entre les solutions incluant les cas (3)-(6) et (7)-(8), qui est négligeable, l'écart entre les résultats obtenus avec le modèle et ceux obtenus par des simulations électriques du type SPICE est parfois conséquent. Cette différence, qui a son importance, est en partie due au comportement des transistors qui ne peut plus se comparer à celui d'une résistance pure dès qu'ils deviennent sources de courant. On ne peut cependant

supprimer, dans ce type de modèle, l'approximation qui en résulte, car elle est inhérente à l'utilisation d'une algèbre et d'un algorithme discrets.

Exemple :

les résultats obtenus par une simulation SPICE pour le même circuit que précédemment sont légèrement différents de ceux donnés sur la figure 3-10 ; le seuil d'inversion se situe sur les nœuds n5 et n6, au lieu d'être sur les nœuds n7 et n8 :

-V(n1) = 4, 65 V , donnant une fois inversé 0 V ;
-V(n2) = 4, 25 V , donnant une fois inversé 0 V ;
-V(n3) = 3, 8 V , donnant une fois inversé 0, 05 V ;
-V(n4) = 3, 35 V , donnant une fois inversé 0, 2 V ;
-V(n5) = 2, 85 V , donnant une fois inversé 0, 6 V ;
-V(n6) = 2, 2 V , donnant une fois inversé 4, 2 V ;
-V(n7) = 0, 6 V , donnant une fois inversé 5 V ;
-V(n8) = 0 V , donnant une fois inversé 5 V ;
-V(n9) = 0 V , donnant une fois inversé 5 V ;
-V(n10) = 0 V , donnant une fois inversé 5 V ;
-V(n12) = 0 V , donnant une fois inversé 5 V ;
-V(n13) = 0 V , donnant une fois inversé 5 V ;
-V(n14) = 0 V , donnant une fois inversé 5 V.

Toutefois, ce modèle reste assez précis pour la simulation de nombreuses pannes, notamment celles du type de la panne du circuit décrit par la figure 3-5, pour lesquelles il faut décider de la capacité d'un vecteur d'entrée à mettre la panne en évidence en fonction de la résistance du chemin conducteur entre les alimentations.

Exemple :

le circuit de la figure 3-5 est simulé pour les deux vecteurs d'entrée "extrêmes" (abcdefg) = (0111101) et (abcdefg) = (0100000), correspondant respectivement au plus grand et au plus petit rapport $R_{charge}/R_{décharge}$. On suppose que le facteur de résistance correspondant à R_N et à R_P est r = 100.

- la figure 3-11 donne le chemin de conduction associé au premier vecteur. On suppose pour la simulation que tous les transistors sont d'abord bloqués, puis qu'ils deviennent passants de haut en bas. Au début, les nœuds x, s, et y sont à l'état X (initialisation), puis leur nouvel état est calculé d'une manière simple, chacun s'imposant sur le suivant :

- état(x) := 1_{50} (50 = 0 + [100/2] ; il n'y a pas de perte de seuil) ;
- état(s) := 1_{100} (100 = 50 + [100/2]) ;
- état(y) := 1^{-}_{150} (perte de seuil).

Ensuite, les quatre transistors parallèles deviennent passants. Ils ont dans ce cas un facteur de résistance égal à [r/2/4] = 13 (chaque transistor a une résistance égale à $R_N/2$; il y a quatre transistors en parallèle) :

- 13/150 = 0, 09 ≤ 0, 77, donc état(y) := 0_{13} ;
- (50 + 13)/100 = 0, 63 ≤ 0, 77 donc état(s) := 0_{63} ;
- (63 + 50)/50 = 2, 26 ≥ 1, 30 donc état(x) = 1_{50}

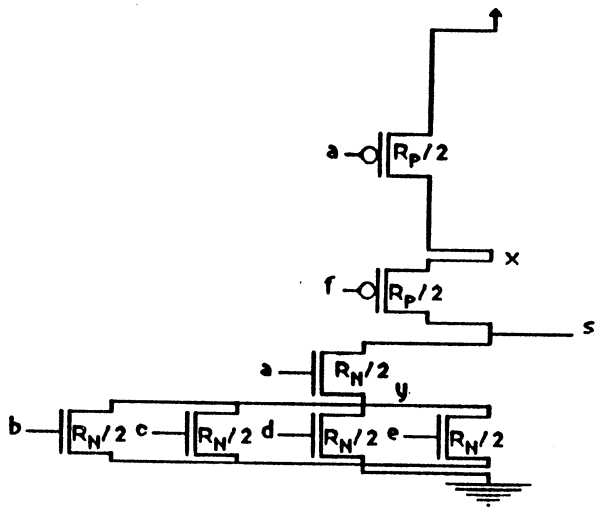


figure 3-11 : chemin de conduction pour le plus grand rapport $R_{charge}/R_{décharge}$

- la figure 3-12 donne le chemin de conduction associé au second vecteur.

- état(x) = 1_{50} ;
- état(s) = 1_{75} (75 = 50 + [100/2/2]) ;
- état(y) = 1^{-}_{125} (125 = 75 + 50 ; perte de seuil).

- 50/125 = 0, 4 ≤ 0, 77 donc état(y) := 0_{50} ;
- (50 + 50)/75 = 1, 33 ≥ 1, 30 donc état(s) = 1_{75} ;
- état(x) = 1_{50}

La valeur de la sortie s est 0 dans le premier cas et 1 dans le second cas. Ces résultats sont cohérents avec les prévisions théoriques. Les tensions obtenues en sortie par une simulation SPICE sont égales à 1, 2 V et à 3 V respectivement.

Le fait que ces valeurs ne soient pas parfaites ne les empêche pas d'être reconnues comme un 0 et un 1 par un inverseur attaqué par la sortie s. Ce dernier point confirme l'intérêt d'avoir en résultat des valeurs entières, et non une valeur indéterminée[1].

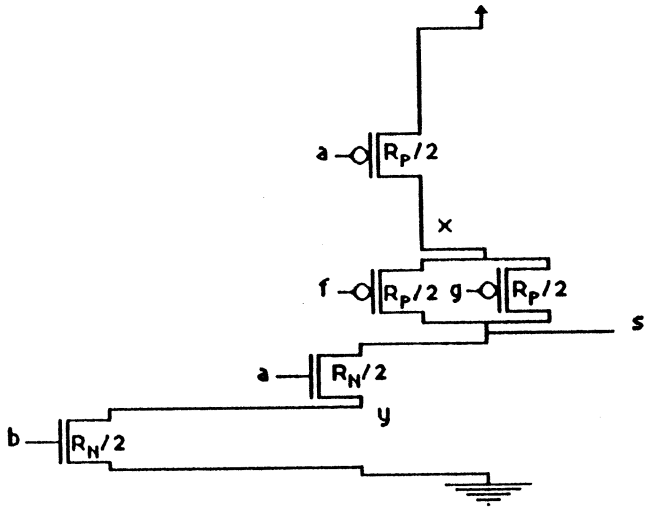


figure 3-12 : chemin de conduction pour le plus petit rapport $R_{charge}/R_{décharge}$

Afin que la précision de ce modèle soit plus grande, on peut également utiliser une autre fonction pour déterminer la prépondérance des forces. On définit par exemple deux variables $diff_1$ et $diff_2$ qui servent à comparer les forces de chaque côté du transistor considéré :

$$- diff_1 = 2\frac{\{f_2\} - f_1}{\{f_2\} + f_1} ;$$

$$- diff_2 = 2\frac{\{f_1\} - f_2}{\{f_1\} + f_2} ;$$

Les différents cas (1) à (6) deviennent alors :

- si $diff_1 \geq d$, alors $v'_{f1} = v_{f1}$; *(12)*

- si $diff_1 \leq (-d)$, alors $v'_{f1} = \{v_{f2}\}$; *(13)*

- si $|diff_1| < d$, alors $v'_1 = \#(v_1, \{v_2\})$ et $f'_1 = f_1$; *(14)*

(1) voir la remarque sur la valeur I dans le chapitre 3, 211.

- si $diff_2 \geq d$, alors $v_{f\,2}' = v_{f2}$; *(15)*

- si $diff_2 \leq (-d)$, alors $v_{f\,2}' = \{v_{f1}\}$; *(16)*

- si $|diff_2| < d$, alors $v'_2 = \#(v_2, \{v_1\})$ et $f'_2 = f_2$. *(17)*

La valeur de d, pour être cohérente avec celle de q (q = 1, 30) doit être égale à 0, 26.

Cette méthode n'a pas été choisie pour des raisons de rapidité de calcul. En effet, le nombre d'opérations nécessaires pour évaluer $diff_1$ et $diff_2$ est bien supérieur à celui que demande l'évaluation d'un simple quotient. Comme cette évaluation est réalisée très souvent (lors de la propagation unidirectionnelle d'un signal, elle est réalisée deux fois par transistor), il était préférable de choisir une opération simple.

3, 21224 Discussion sur l'algèbre de valeurs

L'algèbre initiale à cinq valeurs n'a pas été modifiée, par rapport à sa définition originale[(1)]. Pourtant, il peut sembler intéressant d'augmenter le nombre de valeurs, pour mieux modéliser les tensions. Cette extension a fait l'objet d'une étude qui est rapportée ici.

Malheureusement, de graves problèmes de convergence se posent dès que l'on abandonne la structure de treillis, dont une des particularités est que toute paire d'éléments possède une borne supérieure. En effet, cela permet d'avoir, pour les calculs décrits ici, la propriété d'idempotence, qui est fondamentalement liée à la convergence des calculs en une itération. Ainsi, dans tout treillis, et pout tout paire (a, b) du treillis, on a :

- #(a, #(a, b)) = #(a, b) ;
- #(#(a, b), b) = #(a, b).

Cette propriété implique la présence dans le treillis d'un élément absorbant (ici, I), situé à la pointe supérieure du treillis. Dès que l'algèbre choisie n'a plus une structure de treillis, cet élément disparaît. Pour cette raison, la solution attrayante décrite ci-après n'a pas été retenue.

On définit deux opérations, notées $\oplus$ et $\ominus$, qui sont associées à l'algèbre d'états :

- $v \oplus x$ (x entier relatif) indique la valeur obtenue par un déplacement de x rangs dans l'algèbre à partir de v ;

(1) voir le chapitre 2.

- $v \ominus v'$ est un entier relatif donnant la différence de rang entre v et v' dans l'algèbre.

Exemple :

on donne quelques résultats avec l'algèbre initiale (1, 1^-, I, 0^+, 0).

- $0^+ \oplus 2 = 1^-$;
- $1 \oplus (-3) = 0^+$;
- $1 \ominus 0 = 4$;
- $0^+ \ominus 1 = (-3)$.

Avec ces deux opérations, les nouvelles valeurs v'_1 et v'_2 de part et d'autre du transistor sont calculées ainsi :

$$- v'_1 = v_1 \oplus \left[(\{v_2\} \ominus v_1) \frac{f_1}{f_1 + \{f_2\}} \right] ;$$

$$- v'_2 = v_2 \oplus \left[(\{v_1\} \ominus v_2) \frac{f_2}{f_2 + \{f_1\}} \right].$$

Les forces sont calculées de la manière suivante :

- $f'_1 = \min(f_1, \{f_2\})$;

- $f'_2 = \min(f_2, \{f_1\})$.

Exemple :

le circuit de la figure 3-11 est simulé avec l'algèbre $\mathcal{A}''$ suivante :

$$\mathcal{A}'' = (1, 1^-, 1^{--}, 1^{---}, 1^{----}, M \text{ (milieu)}, 0^{++++}, 0^{+++}, 0^{++}, 0^+, 0).$$

Dans cette algèbre, les valeurs 1^- et 0^+ (pertes de seuil) définies dans l'algèbre initiale correspondent à 1^{--} et 0^{++} (les valeurs de $\mathcal{A}''$ sont associées à des tensions éloignées les unes des autres de 0, 5 V).

- $état(x) = 1_{50}$;
- $état(s) = 1_{100}$;
- $état(y) = 1^{- -}{}_{150}$.

- $valeur(y) := 1^{- -} \oplus [(0 \ominus 1^{- -})150/(150 + 13)] = 1^{- -} \oplus [(-8).0, 92] = 0^{+}$;
- $force(y) := 13$;
- $valeur(s) := 1 \oplus [(0^{+} \ominus 1)100/(100 + 63)] = 1 \oplus [(-9).0, 614] = 0^{+ + + +}$;
- $force(s) := 63$;
- $valeur(x) := 1 \oplus [(0^{+ + + +} \ominus 1)50/(50 + 113)] = 1 \oplus [(-6).0, 31] = 1^{- -}$;
- $force(x) = 50$.

Ces valeurs sont plus proches de la réalité que celles obtenues précédemment. Toutefois, la conservation n'est pas réalisée. En effet, on peut déduire de ces calculs que :

- $\&(1_{100}, 0^{+}{}_{13}) = (0^{+ + + +}{}_{63}, 0^{+}{}_{13})$.

Or :

- $\&(0^{+ + + +}{}_{63}, 0^{+}{}_{13}) = (0^{+ +}{}_{63}, 0^{+}{}_{13}) \neq (0^{+ + + +}{}_{63}, 0^{+}{}_{13})$.

On doit donc opérer de nouvelles itérations sur le même transistor avant d'arriver à l'état stable, ce qui est prohibitif pour la simulation désirée. Si l'on s'arrête à la première itération, le résultat est inexact et dépend de l'ordre dans lequel sont traités les transistors.

En conclusion, le modèle retenu exposé dans le chapitre 3, 21221 est satisfaisant à plusieurs points de vue :

- il utilise une algèbre multivaluée, et est compatible avec le simulateur spécifié dans le chapitre 2 ;

- cette algèbre peut encore être représentée sous forme de treillis, même si l'on ne peut plus concaténer les treillis des différentes forces : les forces sont maintenant des éléments de $\mathbb{N}$; les calculs de prépondérance des forces ne sont plus résolues par l'emploi d'un treillis, mais par des opérations arithmétiques ;

- l'opération & garde sa simplicité d'emploi, bien que l'usage de tableaux soit devenu impossible. De plus, elle répond au critère impératif d'idempotence, défini de manière générale comme suit :

- $\&(v_{f1}, v_{f\,2}') = \&(v_{f1}, v_{f2})$;

- $\&(v_{f\,1}', v_{f2}) = \&(v_{f1}, v_{f2})$;

avec $(v_{f\,1}', v_{f\,2}') = \&(v_{f1}, v_{f2})$.

La convergence des calculs en une itération, comme cela est expliqué plus haut, est due au fait que l'algèbre est un treillis, et au fait que le calcul des nouvelles forces des nœuds est essentiellemment idempotent[1].

3, 22 Modélisation de réseaux de résistances

Ce modèle diffère radicalement du précédent pour ce qui concerne le calcul des états des nœuds : l'algorithme ne traite pas les transistors un par un, mais d'une manière globale, en résolvant les équations linéaires décrivant un réseau de résistances. La force d'un signal est ainsi sans signification, puisque les signaux ne sont plus propagés de nœud en nœud. Ils sont définis directement en fonction de leur contexte en termes de résistances. Les courants sont pris en compte explicitement lors de la résolution des système d'équations, au lieu d'être traités de façon implicite par la propagation de signaux pondérés (forces).

Cette technique présente deux inconvénients majeurs : d'une part, la résolution de telles équations est complexe et longue, d'autre part, elle ne permet pas a priori de prendre en compte les pertes de seuil éventuelles des signaux. Si ce dernier point peut être amélioré, la complexité des calculs n'en semble pas moins excessive devant leur précision. Celle-ci est obligatoirement imparfaite car on ne peut toujours assimiler un transistor à une résistance pure. La technique de modélisation précédente[2] présente le même type d'imprécision, mais est beaucoup plus simple et rapide.

3, 221 Calcul de la résistance des transistors

Ce calcul utilise les caractéristiques en courant des transistors, qui sont adaptées pour avoir un caractère général. Ces caractéristiques se présentent sous deux formes (les tensions sont en valeur absolue) :

$$- I_{DS} = K\gamma|2(V_{GS} - V_{th})V_{DS} - V_{DS}^{2}|, \text{ avec } K = 1/2\mu c_{ox} \text{ et } \gamma = W/L. \qquad (18)$$

Cette expression peut être simplifiée lorsque VDS << VGS - Vth, et devient :

(1) voir le chapitre 3, 21222.

(2) voir le chapitre 3, 21.

- $I_{DS} = 2K\gamma(V_{GS} - V_{th})V_{DS}$, qui est la caractéristique du régime ohmique ;

- $I_{DSat} = K\gamma(V_{GS} - V_{th})^2$, qui est la caratéristique du régime saturé. *(19)*

Toutefois, ces relations ne permettent pas de déduire directement la valeur de la résistance du transistor, la première parce que le courant dépend de V_{DS}, la seconde parce que le transistor lorsqu'il est saturé devient une source de courant ce qui exclut toute notion de résistance. Néanmoins, plusieurs techniques peuvent être employées pour construire un modèle plus simple et plus utilisable.

3, 2211 Résistance dynamique

Pour obtenir une approximation de la résistance du transistor, la relation (18) est dérivée par rapport à V_{DS}, en supposant que le transistor est entre son régime ohmique et son régime saturé, c'est-à-dire que, par exemple, $V_{DS} = V_{dd}/2$ ($V_{GS} = V_{dd}$) [fig. 3-13] :

$$- R = 1/(1/R) = 1/(\partial I_{DS}/\partial V_{DS}) = \frac{1}{K\gamma(V_{GS} - V_{th} - V_{DS})} = \frac{1}{K\gamma(V_{dd} - 2V_{th})}.$$

L'hypothèse sur la valeur de V_{DS} est paradoxale, car elle permet justement, en calculant R, de calculer la valeur de V_{DS} lors du calcul des signaux dans le réseau de résistances. Cette valeur permet de déterminer l'état des deux nœuds de chaque côté du transistor. Mais cette hypothèse est nécessaire car le modèle doit rester suffisamment simple.

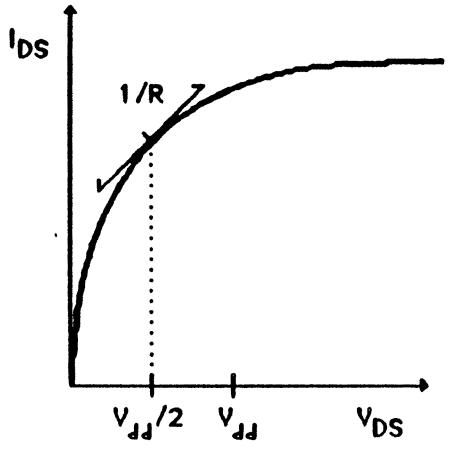


figure 3-13 : détermination de la résistance dynamique.

Exemple :

calcul de la résistance R_N d'un transistor N (paramètres de la technologie CMP CMOS MIIS 2mm 2 alus).

- $\mu_N cox = 60.10^{-6} A/V^2$, $(V_{GS})_N = 5$ V, $V_{thn} = 0,75$ V, $(V_{DS})_N = 2,5$ V,

- $R_N = 9,5/\gamma_N$ KΩ.

3,2212 Résistance statique

L'hypothèse qui est faite dans ce cas est que I_{DS} est proportionnel à V_{DS} jusqu'à ce que V_{DS} atteigne $V_{dd}/2$ [fig. 3-14] :

$$- R = 1/(1/R) = 1/(I_{DS}/V_{DS}) = \frac{1}{K\gamma[2(V_{GS} - V_{th}) - V_{DS}]} = \frac{1}{K\gamma(3/2V_{dd} - 2V_{th})}.$$

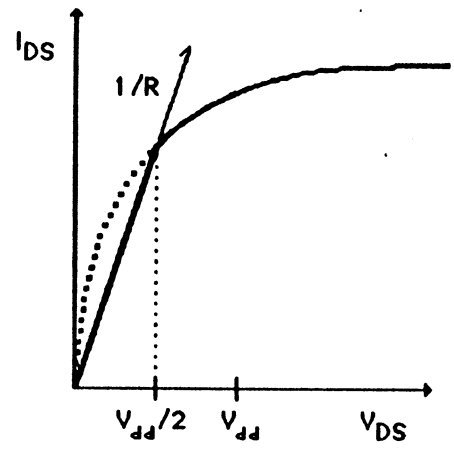


figure 3-14 : détermination de la résistance statique.

Pour les mêmes paramètres que précédemment, la résistance d'un transistor N est égale à $R_N = 5,6/\gamma_N$ KΩ.

Plusieurs simulations SPICE ont montré que la première méthode est plus proche de la réalité, mais reste assez approximative. Ce calcul peut être rendu plus précis par l'utilisation d'une valeur de V_{GS} plus précise, mais cela ralentit la transformation du réseau de transistors en un réseau de résistances "équivalent".

3, 222 Pertes de seuil

Les équations ci-dessus ne permettent pas de prendre en compte les seuil des transistors, et donc les pertes de seuil éventuelles sur les signaux. Cet inconvénient peut être atténué en imposant les pertes de seuil aux états concernés par ce phénomène. Si la tension calculée est V_0, la nouvelle tension est :

$$V'_0 = \min(V_0, V_0 - V_{th}).$$

Cette technique, utilisée dans plusieurs simulateurs, est assez rudimentaire.

En conclusion, le modèle de résistances équivalentes, bien qu'il puisse être convaincant pour une simulation de circuits sans panne, est trop imprécis vis-à-vis de la complexité de l'algorithme, pour être d'une grande efficacité dans le cadre d'une simulation de pannes.

3, 23 Utilisation des équations caractéristiques complètes

Dans cette méthode, il n'est fait aucune hypothèse : les tensions et les courants sont calculés en utilisant les équations (18) et (19) ou les suivantes :

$$- I_{DS} = K\gamma[2(V_{GS} - V_{th})V_{DS} - (1 + \beta)V_{DS}^2], \text{ avec } K = 1/2\mu c_{ox} \text{ et } \gamma = W/L. \quad (20)$$

$$- V_{th} = V_{th0} + \beta.V_S = V_{th0} + K_B[(2\Phi_F + V_S)^{1/2} - (2\Phi_F)^{1/2}], \quad (21)$$

$$- I_{DSat} = K\gamma/(1 + \beta)\,(V_{GS} - V_{th})^2, \quad (22)$$

$$- V_{DSat} = (V_G - V_{th0})/(1 + \beta) = (V_{GS} - V_{th})/(1 + \beta), \quad (23)$$

où β est l'effet de substrat et Φ_F est le potentiel de Fermi.

Ce modèle permet une simulation plus rapide qu'avec le programme SPICE, mais est moins précis. Toutefois, il convient pour la simulation de pannes, lorsqu'une plus grande précision que celle obtenue avec le modèle multivalué est demandée.

Ce modèle est le seul des trois exposés ici qui permette une bonne gestion des tensions de seuil et une détermination correcte de l'état des transistors : saturé ou en mode ohmique. Ainsi, il permet par exemple de simuler la configuration de la figure 3-15, dans laquelle plusieurs transistors N sont en série entre Vdd et Gnd.

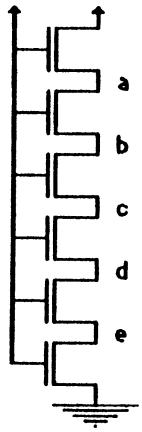


figure 3-15 : configuration dont la simulation est délicate.

A titre indicatif, une simulation SPICE donne les résultats suivants[1] :

- V(a) = 2, 3 V ; V(b) = 1, 65 V ; V(c) = 1, 15 V ; V(d) = 0, 7 V ; V(e) = 0, 35 V.

3, 24 Choix du modèle

Si l'on se réfère à un critère de précision pure, la troisième méthode est la meilleure. Mais les spécifications initiales et les objectifs de départ incitent à choisir une solution utilisant un contexte multivalué. De fait, les deux dernières méthodes ne permettent pas d'utiliser une algèbre multivaluée, ou du moins ne procèdent pas d'un environnement multivalué.

Pour cette raison, on retient le premier modèle, en gardant la possibilité de faire appel aux deux autres si une plus grande précision est désirée.

3, 3 Simulation des pannes

Ce chapitre donne la liste des pannes que l'on peut simuler au niveau interrupteur, en fonction du type de la conception. Ensuite, la modélisation de ces pannes est décrite pour l'environnement multivalué spécifié jusqu'à présent. Si toutes les pannes ne peuvent être envisagées, notamment celles demandant une algèbre de valeurs différentes (couplages capacitifs, ...), le modèle exposé ici permet de simuler de nombreuses pannes inaccessibles au niveau logique.

(1) avec le premier modèle (algèbre multivaluée), les résultats sont :
état(a) = 1^-_{100} ; état(b) = 1^-_{200} ; état(c) = 1_{300} ; état(d) = 0_{200} ; état(e) = 0_{100}.

3, 31 Liste des pannes

De façon générale, on peut considérer deux classes de pannes, suivant que la simulation est effectuée dans une démarche de conception descendante (du niveau fonctionnel au niveau des masques) ou lors d'une vérification ascendante (du niveau des masques au niveau fonctionnel, par des extractions successives).

3, 311 Conception descendante

Dans une démarche descendante, aucune information topologique n'est disponible au niveau interrupteur, puisqu'il est impossible de connaître par avance l'implantation d'un réseau de transistors. Par conséquent, les pannes qui peuvent être traitées sont celles sur lesquelles le réseau donne assez d'informations par lui-même. Par exemple, les courts-circuits entre lignes (alu, poly, diff, ...) ne peuvent être envisagées ici, mais les courts-circuits entre nœuds peuvent l'être. Toutefois, l'intérêt de simuler des pannes dont l'occurence au niveau des masques est inconnue peut être discutable : par exemple, deux nœuds pour lesquels on simule un court-circuit peuvent être trop éloignés en réalité pour pouvoir faire l'objet d'une telle panne.

La classe des pannes concernées dans le cadre d'une conception descendante englobe donc principalement les pannes des transistors, dont la localisation est la même au niveau interrupteur et au niveau des masques :

- collage-passant d'un transistor ("stuck-on") ;

- collage-bloqué d'un transistor ("stuck-off" ou parfois "stuck-open") ;

- court-circuit entre grille et drain ou grille et source ;

- modification du seuil d'un transistor ;

- modification de la résistance du canal d'un transistor ;

Elle englobe aussi les pannes ne demandant pas d'informations topologiques, avec une réserve sur l'intérêt de leur simulation :

- court-circuit entre deux nœuds ;

- collage d'un nœud.

Les coupures de nœuds ne sont pas simulées à ce niveau, car un nœud (niveau interrupteur) peut être composé de plusieurs lignes (niveau des masques), sans aucune prévision possible. Ces pannes sont simulées plus efficacement lors d'une démarche ascendante. De même, les effets des couplages capacitifs ne sont pas concernés par cette étude car d'une part ils demandent pour être modélisés des valeurs plus grandes que 1 et plus petites que 0, d'autre part l'algorithme de calcul doit être modifié pour que les états des deux nœuds couplés tiennent compte de l'état de charge de la capacité qui les sépare[1].

3, 312 Vérification ascendante

Le réseau de transistors extrait du dessin des masques doit être beaucoup plus complet qu'un réseau simple d'interrupteurs. C'est pourquoi, afin de posséder toutes les informations nécessaires pour la simulation des pannes, ce réseau reproduit la composition des équipotentielles du dessin des masques. Ainsi, les bifurcations des lignes, les contacts, et les trois extrémités des transistors deviennent autant de nœuds, tandis que les lignes sont transformées en objets [fig. 3-16] dont le comportement est modélisé comme celui d'une résistance.

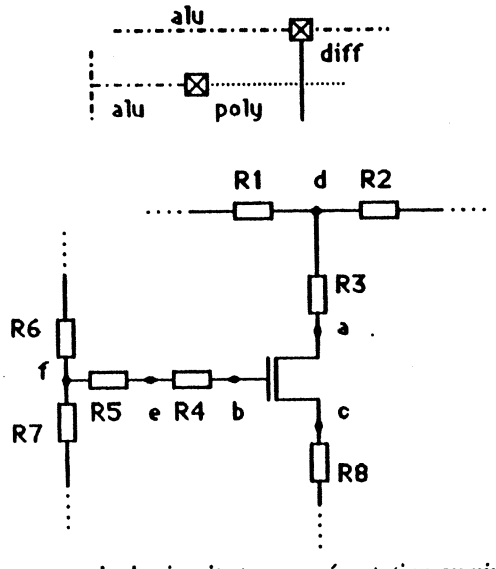


figure 3-16 : un exemple de circuit et sa représentation au niveau interrupteur.

Les objets R_i modélisent la résistance des lignes. Ils sont simulés de la même manière que les transistors : un facteur de résistance leur est affecté, en fonction de la résistance réelle des lignes. L'opérateur & utilise aussi les états $\{v_f\} = (\{v\}, \{f\})$. La valeur $\{v\}$ peut être modifiée en fonction du facteur de résistance[2]. Les pannes qui peuvent être simulées lors d'une démarche ascendante sont les suivantes :

(1) la modélisation des couplages capacitifs et de certains dispositifs est abordée dans le chapitre 2, 72.

(2) cette modification peut s'inspirer de celle qui est décrite dans le chapitre 2, 72.

- collage-passant d'un transistor ("stuck-on") ;

- collage-bloqué d'un transistor ("stuck-off" ou parfois "stuck-open") ;

- court-circuit entre grille et drain ou grille et source ;

- modification du seuil d'un transistor ;

- modification de la résistance du canal d'un transistor ;

- court-circuit entre deux nœuds ;

- collage d'un nœud ;

- modification de la résistance d'une ligne ;

- coupure sur une ligne.

Les effets de couplage capacitifs ne sont pas traités ici. En outre, les défauts des contacts ne peuvent pas être modélisés, car un contact est représenté par un nœud. Toutefois, on peut approcher ces défauts en agissant sur la résistance des lignes connectées aux contacts.

3, 32 Modélisation des pannes

3, 321 Conception descendante

- collage-passant d'un transistor ("stuck-on").

Une panne de collage-passant sur un transistor est simulé de deux manières : soit l'état du transistor, par un accès direct, est forcé à "passant", soit le facteur de résistance r du transistor est modifié. Cette dernière solution permet de moduler le collage. L'intérêt d'une telle modulation est montré par l'étude suivante sur l'effet d'un collage-passant sur un transistor P dans une chaîne d'inverseurs. Cet effet varie en fonction de la résistance du collage. L'accent est mis sur la recherche de la résistance de "basculement". Le circuit de la figure 3-17 est simulé avec le programme SPICE.

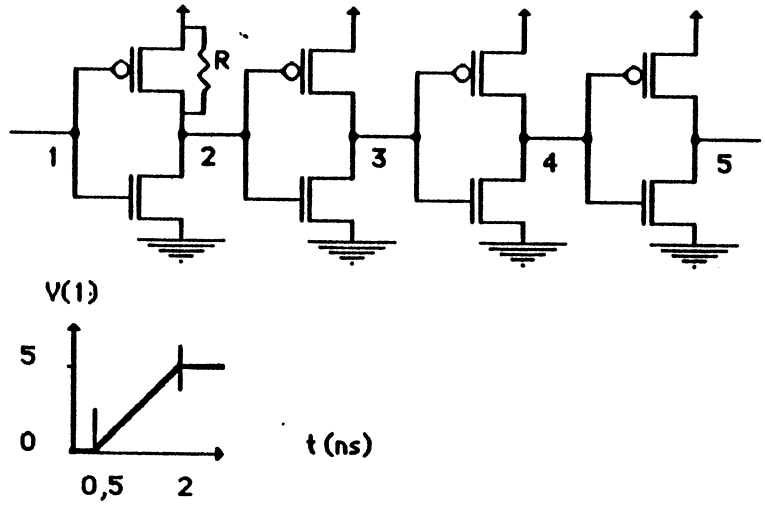


figure 3-17 : chaîne d'inverseurs et tension d'entrée.

Un premier tableau donne la valeur des tensions finales des nœuds de la chaîne. Ces résultats sont représentés sous forme de courbes dans le figure 3-18. Un second tableau exprime le retard sur chaque nœud, toujours en fonction de la résistance du collage-passant. La courbe 3-19 représente la courbe des retards sur le nœud 5.

R (Ω)	$V_f(2)$	$V_f(3)$	$V_f(4)$	$V_f(5)$ (Volts)
1000	4,6	0	5	0
2000	4,3	0	5	0
5000	3,3	0,3	5	0
7000	2,75	1,5	5	0
7200	2,7	2,3	4,3	0
7300	2,7	2,7	1,5	4,9
7400	2,6	3,2	1,4	5
7500	2,6	3,6	0,2	5
8000	2,5	4	0	5
8500	2,4	4,2	0	5
9000	2,2	4,4	0	5
10000	2	4,6	0	5
15000	1,2	4,9	0	5
25000	0,75	5	0	5
50000	0,4	5	0	5
∞	0	5	0	5

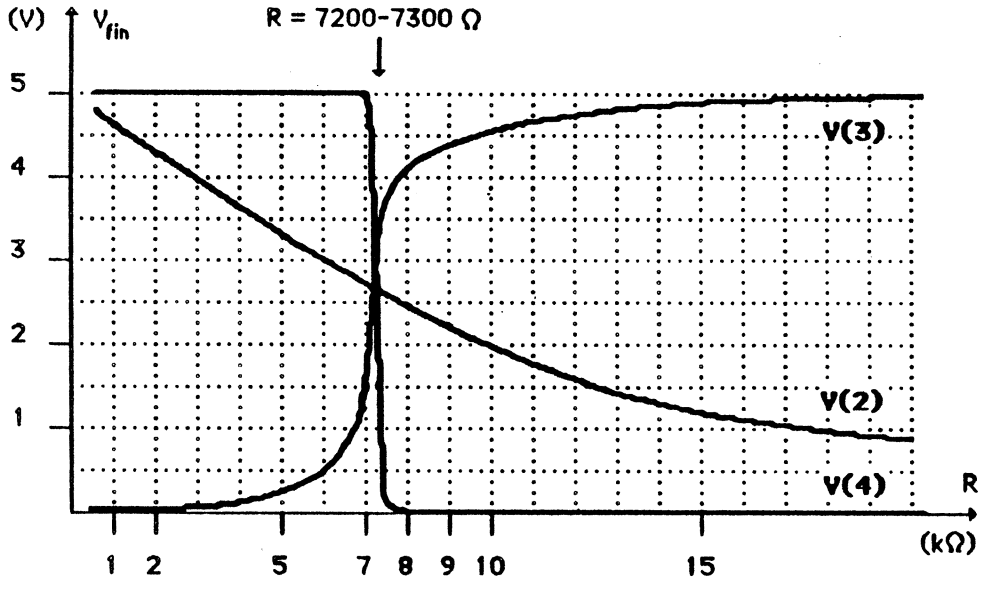


figure 3-18 : courbe des tensons finales.

R (Ω)	$t_{deb}(2)$	$t_{fin}(2)$	$t_{deb}(3)$	$t_{fin}(3)$	$t_{deb}(4)$	$t_{fin}(4)$	$t_{deb}(5)$	$t_{fin}(5)$	ret(5) (ns)
1000	0,8	2,1	-	-	-	-	-	-	-
2000	0,8	2	-	-	-	-	-	-	-
5000	0,8	2,4	1,8	2,7	-	-	-	-	-
7000	0,8	3,2	1,7	4,5/8,5	-	-	-	-	-
7200	0,8	3/3,5	2	17	4	17	-	-	-
7300	0,8	3/3,5	2	20	4	50	14	40/60	35/55
7400	0,8	3,5	2,3	15/20	3,5	20/25	9	17	12,5
7500	0,8	3	1,5	15/20	3,5	13/17	7,5	13	8,5
8000	0,8	2,7	1,7	8/9	3	8/9	5,1	8	3,5
8500	0,8	2,7	1,7	6/7	2,7	6/7	4,3	6,8	2,3
9000	0,8	2,7	1,7	5,4	2,6	5,7	3,8	6,2	1,7
10000	0,8	2,8	1,7	5/6	2,4	5,2	3,4	5,6	1,1
15000	0,8	2,8/3,8	1,6	4	2,3	4,5	2,9	5	0,5
25000	0,8	2,5/3,8	1,6	3,8	2,2	4,2	2,7	4,7	0,2
50000	0,8	3,1	1,6	3,6	2,2	4,2	2,6	4,6	0,1
∞	0,8	3,1	1,6	3,5	2,1	4,1	2,5	4,5	0

avec ret(5) = $t_{fin}(5)_R - t_{fin}(5)_\infty$.

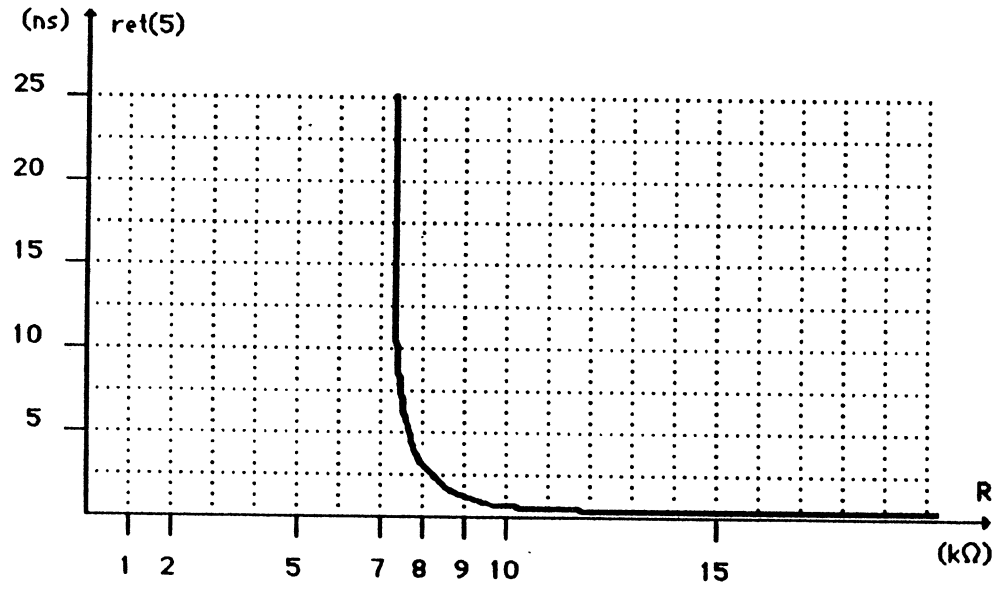


figure 3-19 : courbe des retards.

Les conditions de simulation sont les suivantes : pour R = 7200 Ω et R = 7300 Ω, le pas de simulation est de 1 ns ; pour R = 7400 Ω, le pas de simulation est de 0,5 ns ; pour les autres valeurs de R, le pas de simulation est de 0,1 ns.

Il ressort de cette étude que cette résistance a une influence importante sur l'effet de la panne, et qu'il peut être utile de pouvoir l'inclure dans la simulation.

Il faut noter que la résistance réelle du transistor doit être connue pour que la modélisation de la résistance du collage ait une signification. Si la résistance d'un transistor est inconnue, le calcul est impossible.

Exemple :

la résistance de "basculement" pour une chaîne d'inverseur de taille standard et pour la technologie CMP CMOS 85 est de l'ordre de 7200-7300 Ω. Pour affecter au transistor collé un facteur de résistance qui soit compatible avec l'ordre de grandeur de ce résultat, il faut ramener la valeur de la résistance à un facteur de résistance adéquat.

Autrement dit, si la résistance d'un transistor standard est connue (par exemple 10000 Ω), et que l'on veut simuler un collage-passant de résistance 5000 Ω, il faut affecter au transistor un facteur de résistance de 100(5000/10000) = 50.

- collage-bloqué d'un transistor ("stuck-off" ou parfois "stuck-open").

La modélisation de cette panne suit le même principe que celle du collage-passant d'un transistor. Soit l'état du transistor est forcé à "bloqué", soit on affecte au transistor un facteur de résistance approprié.

- court-circuit entre grille et drain ou grille et source.

Un tel court-circuit peut être modélisé par l'équivalence en termes d'état des nœuds concernés : une variation de l'état et des connexions aux alimentations de l'un est immédiatement répercutée sur l'autre. Le court-circuit peut aussi être représenté par un objet modélisant une résistance, ce qui permet de simuler des courts-circuits plus ou moins francs[1].

- modification du seuil d'un transistor.

Cette modification se traduit soit par une augmentation de la tension de seuil, soit par sa diminution (en valeur absolue). Dans le premier cas, le transistor conduit "plus tard", dans le second cas, il conduit "plus tôt". Cela se traduit dans la réalité par des changements dans les tensions finales d'une part (car la résistance du transistor est modifiée), et dans les temps de montée et de descente d'autre part (car les délais de commutation sont différents).

Une série de simulations SPICE a été effectuée dans le but de définir une loi sur la variation de la résistance d'un transistor en fonction de sa tension de seuil. Le circuit de base ayant été simulé est un simple inverseur dont les deux transistors conduisent : le transistor P est commandé par un 0 et le transistor N est commandé par un 1. Les simulations donnent comme résultat la tension de sortie V_S de l'inverseur en fonction de la variation de la tension de seuil du transistor N et du transistor P (un seul transistor est modifié à la fois dans l'inverseur). Cette tension de sortie permet de calculer (en supposant que les transistors se comportent comme des résistances pures) le nouveau rapport $(R_P/R_N)'$ associé grâce à la formule du pont diviseur :

$$V_S = \frac{V_{dd}}{1 + (R_P/R_N)'} \quad \Rightarrow \quad (R_P/R_N)' = V_{dd}/V_S - 1.$$

Comme le rapport normal R_P/R_N est égal à 1, on peut déduire du nouveau rapport $(R_P/R_N)'$ les rapports R_N'/R_N et R_P'/R_P. Ces calculs sont regroupés dans le tableau suivant :

(1) voir l'utilisation des objets R_i dans le chapitre 3, 312.

V_{thn}	V_{thp}	V_S	$(R_P/R_N)'$	R_N'/R_N	R_P'/R_P
0, 125	-0, 75	1, 9	1, 63	0, 61	
0, 75	-0, 125	2, 85	0, 75		0, 75
0, 15	-0, 75	1, 9	1, 63	0, 61	
0, 75	-0, 15	2, 85	0, 75		0, 75
0, 185	-0, 75	1, 95	1, 56	0, 64	
0, 75	-0, 185	2, 8	0, 79		0, 79
0, 25	-0, 75	2	1, 5	0, 67	
0, 75	-0, 25	2, 8	0, 79		0, 79
0, 375	-0, 75	2, 15	1, 33	0, 75	
0, 75	-0, 375	2, 75	0, 82		0, 82
0, 75	-0, 75	2, 5	1	1	1
1, 5	-0, 75	3, 15	0, 59	1, 7	
0, 75	-1, 5	1, 4	2, 58		2, 58
2, 25	-0, 75	3, 7	0, 35	2, 86	
0, 75	-2, 25	0, 85	4, 88		4, 88
3	-0, 75	4, 2	0, 19	5, 26	
0, 75	-3	0, 5	9		9
3, 75	-0, 75	4, 6	0, 09	11, 11	
0, 75	-3, 75	0, 25	19		19
4, 5	-0, 75	4, 9	0, 02	50	
0, 75	-4, 5	0, 05	99		99

La figure 3-20 représente sous forme de courbes les résultats du tableau. Les courbes en trait épais concernent les coefficients R_N'/R_N et R_P'/R_P, celles en trait fin décrivent la tension de sortie V_S.

Le point (0, 75 ; 2, 5) est un point d'inflexion pour les deux courbes en trait fin. Une dissemblance entre le comportement des transistors N et P est apparente sur les courbes en trait épais.

Cette série de courbes peut être utilisée pour affecter un nouveau facteur de résistance r' au transistor dont le seuil est modifié. Ce facteur r' est égal au produit du facteur normal r multiplié par le coefficient correspondant au nouveau seuil.

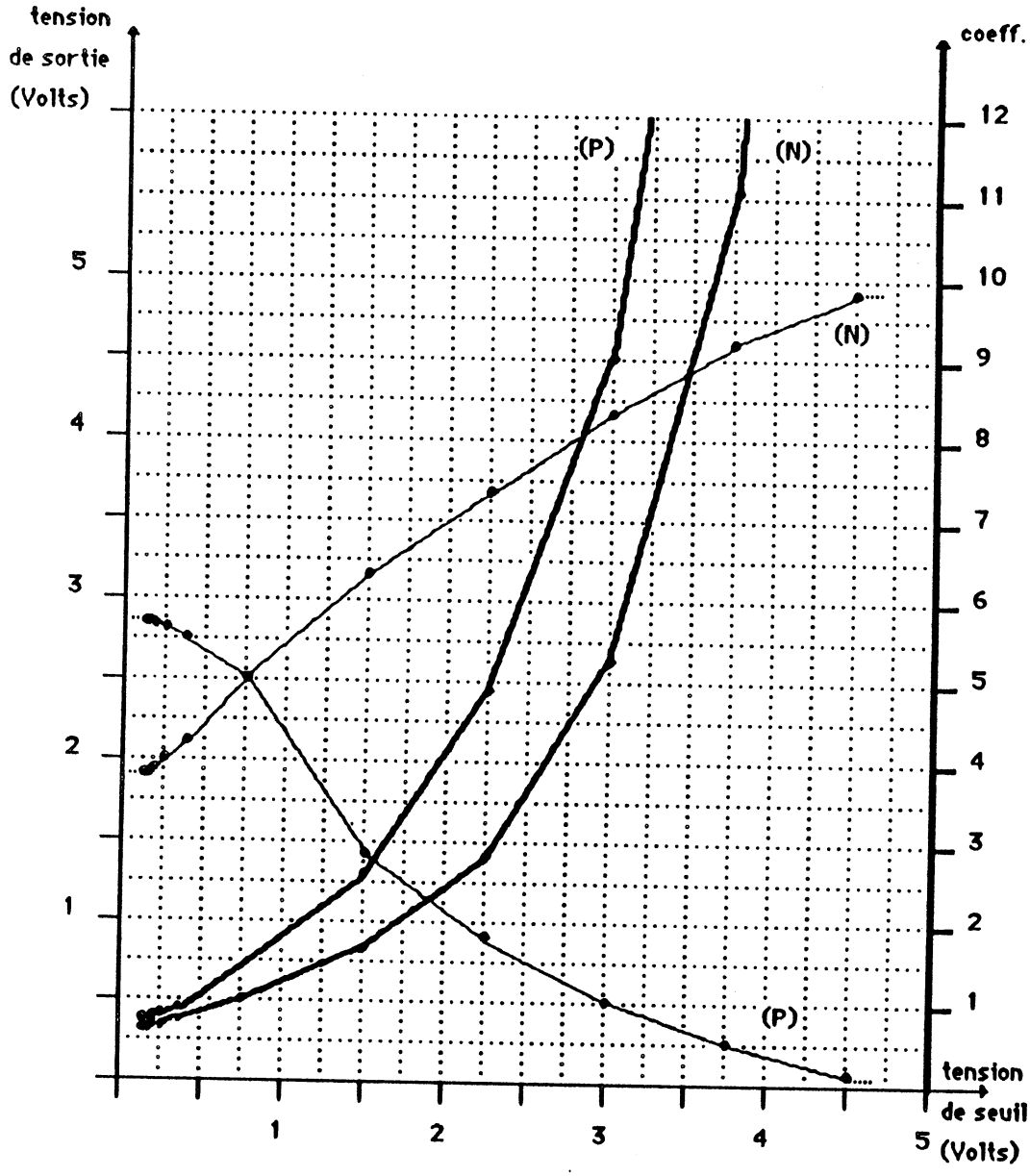


figure 3-20 : courbes décrivant le changement de résistance des transistors N et P.

Une autre série de simulations SPICE a été effectuée pour déterminer la variation des temps de montée et de descente d'un inverseur en fonction de la tension de seuil du transistor N et du transistor P. Les résultats de ces simulations sont regroupés dans un tableau, puis représentés sous forme de courbe [fig. 3-21]. Les temps de montée qui sont modifiés ne concernent que les transistors P (qui servent à la charge), et les temps de descente ne concernent que les transistors N (qui servent à la décharge).

Comme la variation du temps de montée est la même que celle du temps de descente, une seule courbe suffit pour décrire les deux phénomènes. Comme précédemment, les tensions de seuil sont en valeur absolue. Cette courbe, bien que grossière, permet d'affecter

un nouveau délai[1] à la commutation passant-bloqué et bloqué-passant des transistors : au lieu d'être égal à 1, il peut prendre les valeurs 2, 3, ... selon le cas. Cette modification ne concerne que les tensions de seuil supérieures à 2 V. Au-dessous de cette valeur, la variation est négligeable devant la précision du modèle à délai unitaire. En outre, les tensions de seuil proches de 5 V ne peuvent être modélisées de façon précise.

V_{thn}	V_{thp}	t_m'/t_m	t_d'/t_d
0, 125	-0, 75	1	1
0, 75	-0, 125	1	1
0, 75	-0, 75	1	1
1, 5	-0, 75	1	1, 25
0, 75	-1, 5	1, 25	1
2, 25	-0, 75	1	1, 5
0, 75	-2, 25	1, 5	1
3	-0, 75	1	2
0, 75	-3	2	1
3, 75	-0, 75	1	3, 5
0, 75	-3, 75	3, 5	1
4, 5	-0, 75	1	9
0, 75	-4, 5	9	1

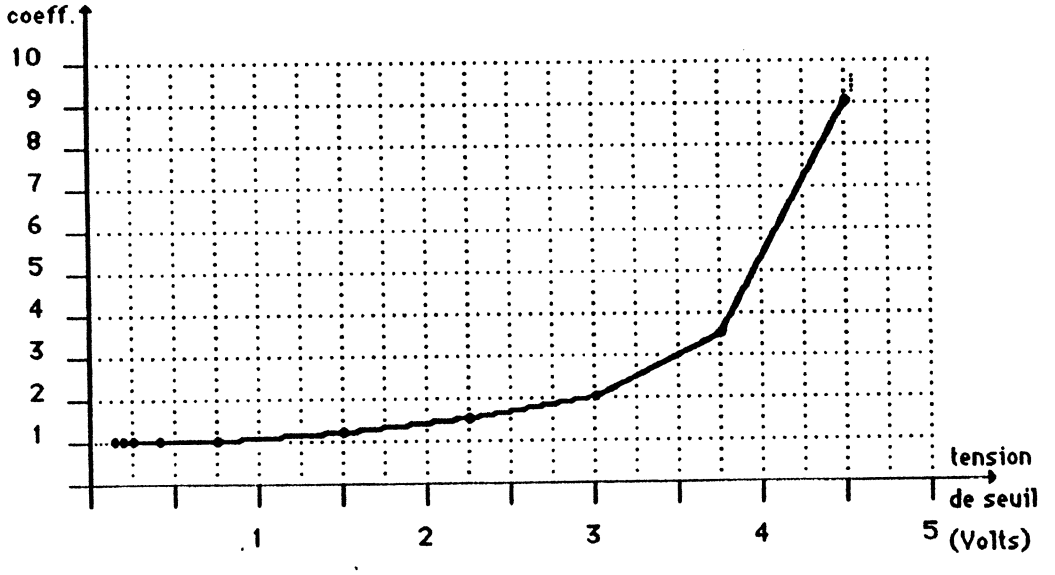


figure 3-21 : courbe déterminant la variation du temps de montée et de descente.

(1) voir la modélisation temporelle utilisant un délai unitaire décrite dans le chapitre 2, 54.

- modification de la résistance du canal d'un transistor.

Cette panne se modélise par l'affectation d'un nouveau facteur de résistance r'. Toutefois, la précision de cette solution reste liée à celle du calcul de la résistance d'un transistor. Dans tous les cas, les effets de cette panne sont similaires à ceux des pannes de collage-passant et collage-bloqué d'un transistor.

- court-circuit entre deux nœuds.

La modélisation de cette panne suit les mêmes principes que celle d'un court-circuit drain-grille et drain-source.

- collage d'un nœud.

Cette panne est modélisée par un accès direct à l'état d'un nœud : celui-ci est imposé en fonction du collage considéré. Seuls les collages à 1 et à 0 sont traités. Tous les autres cas de figure (collages à des valeurs altérées, collages résistifs, ...) sont traités comme des courts-circuits avec une alimentation, la résistance de ceux-ci étant choisie en fonction de la valeur du collage désiré. L'intérêt de prévoir les collages à 0 et à 1 en plus des courts-circuits (un collage à 1 ou à 0 provient dans la réalité d'un court-circuit), réside dans la rapidité du traitement : plutôt que de calculer l'effet d'un court-circuit, l'état du nœud est directement imposé.

3, 322 Vérification ascendante

L'utilisation des objets R_i représentant la résistance des lignes fait surgir une difficulté notable : s'il est assez aisé de calculer la résistance d'une ligne à l'aide des caractéristiques technologiques de fabrication, il est très difficile de calculer la résistance d'un transistor[1].

Si l'on veut affecter aux objets R_i un facteur de résistance non nul, celui-ci doit être cohérent avec le facteur de résistance des transistors.

Exemple :

si, pour un transistor, un facteur de résistance r = 100 est associé à une résistance de 10 KΩ, alors une ligne de résistance égale à 5 KΩ a un facteur de résistance r' = 50.

(1) le calcul approché de la résistance d'un transistor est décrit dans le chapitre 3, 221.

Toutefois, l'intérêt des objets R_i n'étant pas seulement de prendre en compte la résistance des lignes, mais avant tout de pouvoir simuler des pannes imprévisibles dans une démarche descendante, il est possible d'affecter aux lignes un facteur de résistance nul. Ainsi, la résistance absolue des transistors peut rester inconnue.

La modélisation des sept premières pannes est la même que lors d'une conception descendante, à un détail près pour les courts-circuits : un court-circuit entre deux lignes est modélisé par un court-circuit entre deux nœuds, mais ceux-ci ne représentent pas exactement les lignes initiales [fig. 3-22] ; cette approximation n'est jamais gênante, mais demande de choisir les nœuds concernés. Ce point représente la limite du modèle : si la résistance des lignes n'est pas nulle, la résistance du chemin incluant le court-circuit dépend du siège de celui-ci ; il est préférable de ne pas tenir compte de cette dépendance.

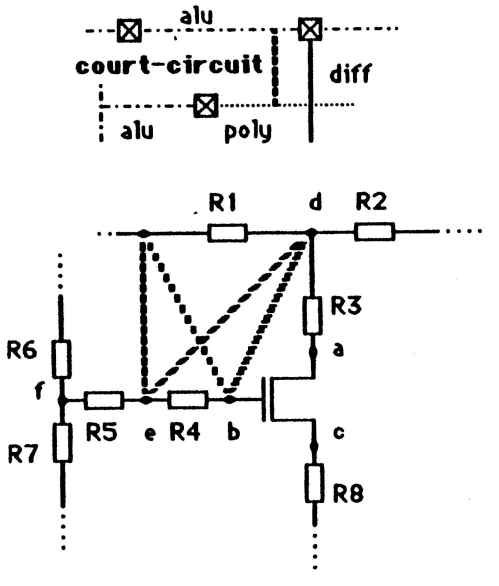


figure 3-22 : un court-circuit et ses différentes représentations.

Les deux pannes nouvelles sont celles concernant les lignes.

- modification de la résistance d'une ligne.

Cette panne est modélisée par le changement du facteur de résistance associé à la ligne.

- coupure sur une ligne.

La modélisation de cette panne implique l'association d'un état aux objets R_i, qui indique une coupure éventuelle sur la ligne. Cet état, similaire à celui des transistors, peut prendre les deux valeurs passant et bloqué. Le traitement de l'état bloqué est le même que

pour un transistor. On pourrait aussi définir un facteur de résistance infini, avec un traitement particulier, mais la gestion d'un état est préférable à cause de son homologue pour les transistors.

En conclusion, la représentation des différentes pannes qui peuvent être simulées avec le modèle est suffisamment souple et précise pour que les résultats de simulation soient utiles pour la phase de test. Que ce soit dans une démarche ascendante ou descendante, le modèle de pannes et le calculs des états permettent une évaluation des états intéressante des pannes envisagées. Cette étude n'aborde pas les problèmes liés à la simulation concurrente des circuits, ni à la gestion des listes de pannes.

CHAPITRE 4

GENERATION DE SEQUENCES DE TEST AU NIVEAU "SWITCH"

4 Génération de vecteurs de test au niveau "switch"

Ce chapitre décrit l'utilisation du modèle défini dans le précédent chapitre pour la génération automatique de vecteurs de test. Celle-ci doit s'effectuer dans l'environnement multivalué spécifié pour la simulation de pannes, car les résultats de celle-ci sont indispensables au test. Cette partie est une étude préliminaire destinée à définir l'orientation des spécifications finales, et se contente de décrire la recherche d'un vecteur de test pour une panne. Les retours en arrière imposés lors de cette recherche ("backtracking") n'y sont pas traités. De même, les actions à effectuer lorsqu'une panne possède plusieurs vecteurs de test (intersection des ensembles de vecteurs de test, recherche de couverture...), et les problèmes créés par l'équivalence de pannes (un même vecteur peut tester plusieurs pannes à la fois), n'y sont pas abordés.

D'une manière générale, la génération automatique de vecteurs de test consiste à trouver le ou les vecteurs qui injectés en entrée d'un circuit peuvent mettre en évidence une panne donnée. Cette mise en évidence se traduit par une différence entre l'ensemble des sorties du circuit sans panne et celui du circuit contenant la panne. Pendant le test du circuit, on n'a accès en principe qu'à la sortie du circuit. Comme la sortie normale est connue, une anomalie observée permet, compte-tenu des vecteurs d'entrée qui l'ont créée, de déduire la ou les pannes présentes dans le circuit : ce sont celles que ces mêmes vecteurs d'entrée ont pu mettre en évidence durant la simulation de pannes ou la génération de vecteurs de test.

La recherche des vecteurs de test s'opère en trois phases principales :

- mise en évidence locale de la panne ;

- propagation vers les sorties de l'anomalie qui en résulte ; cette propagation impose la valeur d'un certain nombre de nœuds ;

- remontée vers les entrées pour la déduction des vecteurs de test.

Exemple :

génération des vecteurs de test pour la panne du circuit de la figure 4-1.

- mise en évidence locale de la panne : la grille du transistor collé-passant doit être mise à 0, donc valeur(n1) = 0 ; il existe du fait de la conduction du transistor collé un chemin de conduction possible entre n4 et Gnd ; ce chemin doit être unique, donc le deuxième transistor N ne doit pas conduire : valeur(n2) = 0 ; l'anomalie créée est la valeur 0 sur le nœud n4 ;

- propagation vers les sorties de l'anomalie : le premier transistor P de la deuxième porte conduit ; il faut donc achever ce chemin de conduction vers la sortie s, donc le deuxième transistor P doit conduire : valeur(n3) = 0 ; la sortie est au niveau 1 ;

- remontée vers les entrées : on trouve ainsi que valeur(n1) = valeur(n2) = valeur(n3) = 0.

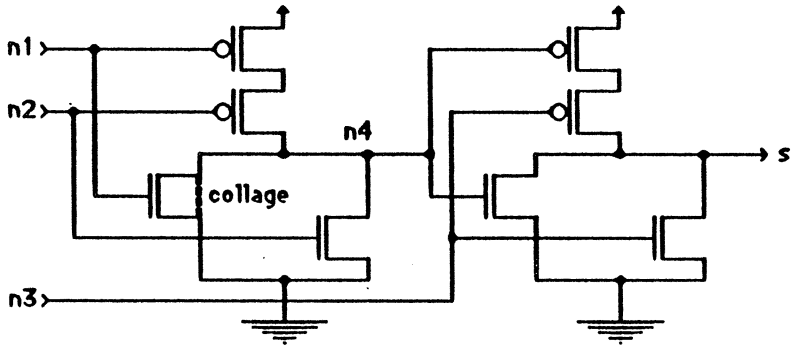


figure 4-1 : un transistor N est collé-passant.

Cette technique de génération est utilisée largement au niveau logique par la famille des "D-algorithmes", sur des modèles de circuits possédant une algèbre ternaire {0, 1, X}. La description des circuits sous forme de portes permet d'écrire des règles qui rendent possible l'automatisation de la troisième phase : pour chaque porte, on peut déduire de la sortie désirée l'ensemble des entrées adéquates, car la fonction de celle-là est connue. La description des circuits sous forme de réseaux d'interrupteurs bruts ne permet pas d'automatiser la génération de cette manière : la fonction d'un réseau est a priori inconnue. Certains algorithmes de génération de vecteurs de test au niveau interrupteur traitent les réseaux tels quels ([PRA87]), mais la plupart ne les traitent pas directement. Certains les transforment d'abord en groupes de transistors ([CHE84], [RED85], [VIV86]) afin de reconstituer la fonction des circuits. D'autres les transforment en réseaux de portes logiques ([AGR84], [CHI83], [JAI85], [SHI86]), de façon à utiliser facilement les algorithmes du niveau logique. Ces deux méthodes utilisent les résultats sur le test de portes logiques NMOS et CMOS, qui permettent de connaître le comportement des portes logiques en présence de certaines fautes, en particulier les collages-passant et les collages-bloqué des transistors ([BAS86], [CHE84], [MAL84], [RAJ86]).

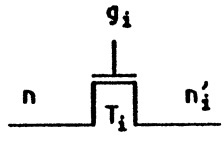
Ce chapitre expose une technique de génération de vecteurs de test adaptée au modèle "switch" multivalué. On choisit d'effectuer la génération sans traitement préliminaire de regroupement sur le réseau, et utiliser l'algèbre décrite dans le chapitre 3. Les motivations de la génération de vecteurs de test au niveau interrupteur sont les mêmes que celles de la simulation de pannes à ce niveau[1].

(1) voir le chapite 3, 1.

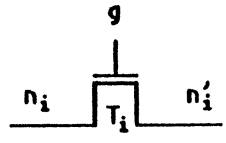
4, 1 Sensibilisation des chemins

4, 11 Description des outils nécessaires

La sensibilisation des chemins concerne la propagation de la panne vers les sorties et la remontée vers les entrées. Dans un premier temps, un chemin est défini depuis le siège de la panne vers les sorties ; une technique particulière doit être utilisée pour que la propagation se fasse uniquement vers les sorties. Pour cela, deux listes sont créées qui décrivent les connexions d'un nœud n aux sorties du circuit. Ces listes sont construites sur le modèle des listes de base voisinage(n) et influence(n) utilisées dans la simulation [fig. 4-2].



$$\text{voisinage}(n) = \{(\ldots\ ,\ (n,\ g_i,\ n'_i,\ T_i),\ \ldots\ \}$$



$$\text{influènce}(g) = \{\ldots\ ,\ (n_i,\ g,\ n'_i,\ T_i),\ \ldots\ \}$$

figure 4-2 : listes de voisinage et d'influence.

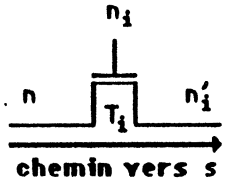
Ces deux listes ne permettent pas de "viser" les sorties ; les deux autres listes sont les suivantes :

- la première liste, notée vois_sort(s, n), décrit, pour chaque nœud n et chaque sortie s, les nœuds n'_i voisins de n et connectés de près ou de loin à s [fig. 4-3] ;

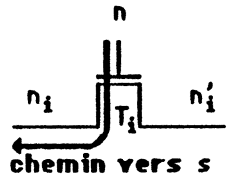
- la deuxième, notée infl_sort(s, n), décrit les nœuds n_i qui sont une source d'un transistor commandé par n et qui sont connectés de près ou de loin à s [fig. 4-3].

Ces deux listes introduisent un terme o_i, qui est un entier indiquant le nombre d'étapes nécessaires pour atteindre la sortie s depuis le nœud n. Il permet de choisir le plus court chemin possible. Cela ne garantit d'ailleurs pas que le chemin choisi soit possible, mais que s'il est possible, c'est le plus court. Le choix du chemin utilise le fait que la propagation de source à source d'un transistor est plus rapide que la propagation de grille à source : si

deux chemins ont le même terme o_i, la préférence va à celui qui permet le premier type de propagation.



$$vois\_sort(s, n) = \{\dots , (n, n_i, n'_i, T_i, o_i), \dots \}$$



$$infl\_sort(s, n) = \{\dots , (n_i, n, n'_i, T_i, o_i), \dots \}$$

figure 4-3 : listes permettant de descendre vers les sorties.

Les listes vois_sort et infl_sort sont construites avant la génération, à partir des sorties s_i du circuit, par un processus récursif utilisant les listes voisinage et influence des nœuds :

$voisinage(s_i) = \{(s_i, g_i, n_i, T_i)\} \Rightarrow$
  $vois\_sort(s_i, n_i) = \{(n_i, g_i, s_i, T_i, 1)\}$;
    $voisinage(n_i) = \{(n_i, g'_i, n'_i, T'_i)\} \Rightarrow$
      $vois\_sort(s_i, n'_i) = \{(n'_i, g'_i, n_i, T'_i, 2)\}$
      ... ;
    $voisinage(n_i) = \{(n_i, g'_i, n'_i, T'_i)\} \Rightarrow$
      $infl\_sort(s_i, g'_i) = \{(n_i, g'_i, n'_i, T'_i, 2)\}$
      ... ;
$voisinage(s_i) = \{(s_i, g_i, n_i, T_i)\} \Rightarrow$
  $infl\_sort(s_i, g_i) = \{(s_i, g_i, n_i, T_i, 1)\}$
  ... ;

Exemple :

les listes décrites ci-dessus sont construites pour un circuit [fig. 4-4] qui n'a pas de fonction particulière, mais qui possède des boucles et des éléments bi-directionnels. Ce circuit est décrit de la manière suivante :

T1 n1 e1 Vdd N

T2 n1 e1 n3 P

T3 n3 n1 Gnd P

T4 n1 e1 n2 N

T5 e1 e2 s N

T6 s n2 e2 N

T7 s e2 Gnd P.

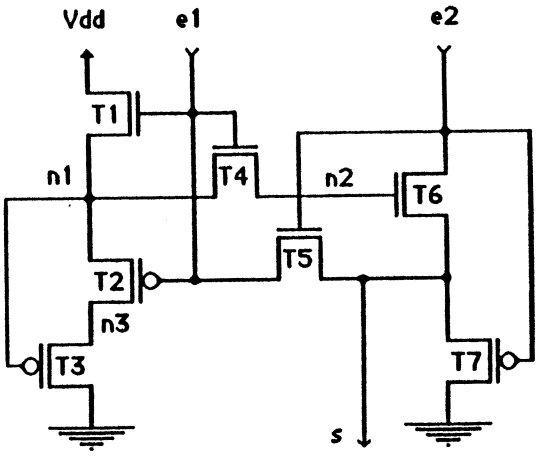


figure 4-4 : circuit donné en exemple pour la construction des listes.

D'abord, les listes de voisinage et d'influence sont construites à partir de la description du circuit :

(1) voisinage(e1) = {(e1, e2, s, T5)}

(2) influence(e1) = {(n1, e1, Vdd, T1), (n1, e1, n3, T2), (n1, e1, n2, T4)}

(3) voisinage(e2) = {(e2, n2, s, T6)}

(4) influence(e2) = {(e1, e2, s, T5), (s, e2, Gnd, T7)}

(5) voisinage(s) = {(s, e2, e1, T5), (s, n2, e2, T6), (s, e2, Gnd, T7)}
influence(s) = {}

(6) voisinage(n1) = {(n1, e1, Vdd, T1), (n1, e1, n3, T2), (n1, e1, n2, T4)}

(7) influence(n1) = {(n3, n1, Gnd, T3)}

(8) voisinage(n2) = {(n2, e1, n1, T4)}

(9) influence(n2) = {(s, n2, e2, T6)}

(10) voisinage(n3) = {(n3, e1, n1, T2), (n3, n1, Gnd, T3)}
influence(n3) = {}

Ensuite, les listes sont construites pour chaque sortie ; les listes de voisinage et d'influence utilisées ne sont pas réécrites ; leur numéro seul est indiqué.

(5) ⇒ vois_sort(s, e1) = {(e1, e2, s, T5, 1)}

(5) ⇒ vois_sort(s, e2) = {(e2, n2, s, T6, 1)}

(5) ⇒ infl_sort(s, e2) = {(s, e2, e1, T5, 1), (s, e2, Gnd, T7, 1)}

(5) ⇒ infl_sort(s, n2) = {(s, n2, e2, T6, 1)}

(8) ⇒ vois_sort(s, n1) = {(n1, e1, n2, T4, 2)}

(6) ⇒ vois_sort(s, n2) = {(n2, e1, n1, T4, 3)}

(8) ⇒ vois_sort(s, n1) = {(n1, e1, n1, T4, 4)} ; stop, car vois_sort(s, n1) a déjà été définie par la même implication.

(8) ⇒ infl_sort(s, e1) = {(n2, e1, n1, T4, 4)}

(6) ⇒ vois_sort(s, n3) = {(n3, e1, n1, T2, 3)}

(10) ⇒ vois_sort(s, n1) = {(n1, e1, n3, T2, 4)}

(6) ⇒ vois_sort(s, n2) = {(n2, e1, n1, T4, 5)}
stop...

(6) ⇒ vois_sort(s, n3) = {(n3, e1, n1, T2, 5)}
stop....

(6) ⇒ infl_sort(s, e1) = {(n1, e1, Vdd, T1, 5), (n1, e1, n3, T2, 5), (n1, e1, n2, T4, 5)}

(10) ⇒ infl_sort(s, e1) = {(n3, e1, n1, T2, 4)}

(10) ⇒ infl_sort(s, n1) = {(n3, n1, Gnd, T3, 4)}

(6) ⇒ vois_sort(s, n2) = {(n2, e1, n1, T4, 5)}
stop...

(6) ⇒ vois_sort(s, n3) = {(n3, e1, n1, T2, 5)}
stop...

(6) ⇒ infl_sort(s, e1) = {(n1, e1, Vdd, T1, 5), (n1, e1, n3, T2, 5), (n1, e1, n2, T4, 5)}

(6) ⇒ infl_sort(s, e1) = {(n1, e1, Vdd, T1, 3), (n1, e1, n3, T2, 3), (n1, e1, n2, T4, 3)} remplacementdes anciens ordres plus petits.

(8) ⇒ infl_sort(s, e1) = {(n2, e1, n1, T4, 2)}.

En conclusion :

vois_sort(s, e1) = {(e1, e2, s, T5, 1)}

infl_sort(s, e1) = {(n2, e1, n1, T4, 2), (n1, e1, Vdd, T1, 3), (n1, e1, n3, T2, 3), (n1, e1, n2, T4, 3), (n3, e1, n1, T2, 4)}

vois_sort(s, e2) = {(e2, n2, s, T6, 1)}

infl_sort(s, e2) = {(s, e2, e1, T5, 1), (s, e2, Gnd, T7, 1)}

vois_sort(s, n1) = {(n1, e1, n2, T4, 2), (n1, e1, n3, T2, 4)}

infl_sort(s, n1) = {(n3, n1, Gnd, T3, 4)}

vois_sort(s, n2) = {(n2, e1, n1, T4, 3)}

infl_sort(s, n2) = {(s, n2, e2, T6, 1)}

vois_sort(s, n3) = {(n3, e1, n1, T2, 3)}.

Certains chemins sont redondants, car ils donnent lieu à des va-et vient inutiles. Mais cela est sans importance, car il existe toujours des chemins plus courts (avec un ordre plus petit) ; en outre, des tests d'arrêts peuvent être facilement introduits dans la construction des listes.

Ces deux listes permettent de déterminer les chemins du siège de la panne vers les sorties. Mais ces différents chemins doivent être sensibilisés, c'est-à-dire que les entrées les rendant actifs doivent être déterminées. La sensibilisation est l'étape la plus délicate de la génération de vecteurs de test ; de fait, cet aspect difficile est la raison première de la rareté des générateurs spécifiés pour les réseaux bruts d'interrupteurs. En effet, aux niveaux supérieurs, la sensibilisation des chemins peut être régie par des lois ayant un caractère mathématique, car la fonction du circuit est connue. Par contre, un réseau d'interrupteurs traité tel quel ne permet pas a priori l'application de telles lois. La voie suivie pour résoudre ce problème est la recherche d'une correspondance entre le raisonnement au niveau logique et celui qui doit être tenu au niveau interrupteur.

Exemple :

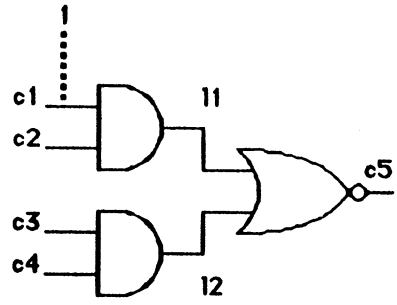


figure 4-5 : la ligne c1 est collée à 1.

soit le circuit de la figure 4-5, déjà abordé dans le chapitre 3, 1.

(a) la panne qui colle la ligne c1 à 1 demande pour être mise en évidence, d'une part que c1 soit mise à 0, d'autre part que l'autre entrée c2 de la porte soit à 1 (porte ET).

(b) le chemin vers la sortie c5 est sensibilisé si la ligne l2 est à 0 (porte OU).

(c) ce qui demande que c3 ou c4 soit à 0.

Or, en ce qui concerne le réseau d'interrupteurs équivalent [fig. 4-6], l'assertion (a) signifie en termes de conduction qu'un chemin de conduction est artificiellement créé par la panne entre la sortie c5 et le nœud n1 ; ce chemin doit être achevé pour connecter c5

à Gnd, ce qui impose que c2 soit à 1. L'assertion (b) signifie que ce chemin ne doit pas être doublé par un autre reliant c5 à Gnd, ce qui impose que c3 ou c4 soit à 0 (assertion (c)).

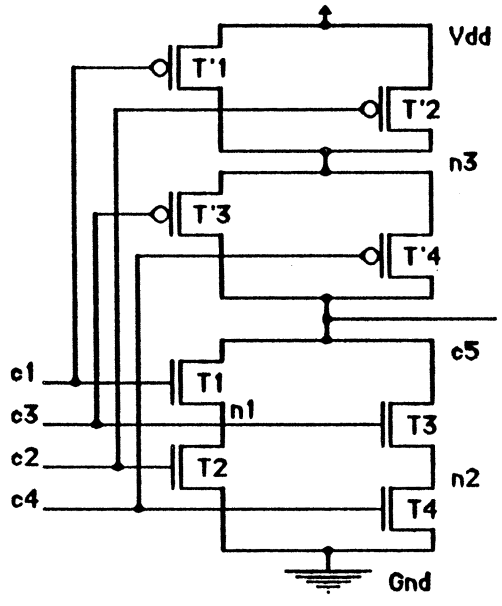


figure 4-6 : réseau de transistors équivalent.

La liste infl_sort(c5, c1) permet de commencer le chemin de sensibilisation :

- infl_sort(c5, c1) = {(n1, c1, c5, T1, 1)}.

Ensuite, puisque c5 est atteint, on descend depuis n1 vers les alimentations.

A ce niveau, d'autres listes sont nécessaires, pour définir les connexions de source à source vers les alimentations et les entrées[1]. Ces listes sont définies de la façon suivante :

- connex_1(n) = {... , (n, g_i, n'_i, T_i, o_i), ... } ;

- connex_1$^-$(n) = {... , (n, g_i, n'_i, T_i, o_i), ... } ;

- connex_0(n) = {... , (n, g_i, n'_i, T_i, o_i), ... } ;

- connex_0$^+$(n) = {... , (n, g_i, n'_i, T_i, o_i), ... },

où n'_i est le nœud qui débute le chemin vers l'alimentation ou l'entrée et où o_i est un entier qui indique le nombre d'étapes nécessaires pour l'atteindre.

(1) le fait qu'un nœud puisse être directement influencé par une entrée doit être pris en compte. Dans ce cas, une entrée pouvant amener un 0, un 0$^+$, un 1 ou un 1$^-$, quatre listes sont créées pour ce nœud.

Le nom de la liste indique la valeur du signal en n'_i (les valeurs 1^- et 0^+ sont des valeurs altérées créées par le "passage" d'un 1 à travers un transistor N et d'un 0 à travers un transistor P).

Exemple :

en reprenant le circuit précédent, on obtient :

- *connex_0(c5) = {(c5, c1, n1, T1, 2, 200), (c5, c3, n2, T3, 2)} ;*
- *connex_0(n1) = {(n1, c2, Gnd, T2, 1)} ;*
- *connex_0(n2) = {(n2, c4, Gnd, T4, 1)} ;*
- *connex_0^+(n3) = {(n3, c3, c5, T3, 3), (n3, c4, c5, T4, 3)}...*

A partir de n1, on sensibilise le chemin vers Gnd en imposant c2 à 1 (T2 est de type N). A partir de c5, on trouve un autre chemin qui passe par T3, ce qui impose c3 à 0 (T3 est de type N).

Deux solutions sont possibles à ce stade : soit on se contente de ce résultat, auquel cas on n'obtiendra pas tous les vecteurs possibles, mais seulement les premiers trouvés, soit on effectue une recherche approfondie sur les chemins pour trouver tous les vecteurs de sensibilisation. La première solution, bien qu'elle soit moins précise, peut être choisie dans le contexte présent. La deuxième solution sera appliquée par la suite pour améliorer le modèle.

4, 12 Sensibilisation des chemins vers les sorties

4, 121 Le siège de la panne est un nœud

Avec ces nouveaux outils, on peut formaliser la sensibilisation du chemin depuis le nœud n dont la valeur v est imposée par la panne vers une sortie s_i.

(0)

- vois_sort(s_i, n) = {... , (n, g_j, n_j, T_j, o_j), ... } ;

- infl_sort(s_i, n) = {... , (n_k, n, n_l, T_k, o_k), ... }.

Soit o_j le plus petit ordre des éléments de vois_sort(s_i, n), et o_k le plus petit ordre des éléments de infl_sort(s_i, n) dont le transistor peut conduire compte-tenu de la valeur v imposée en grille :

(i) si $o_j \leq o_k$, alors on continue en (1) [cf. fig. 4-7] ;

(ii) si $o_j > o_k$, alors on continue en (2) [cf. fig. 4-8] ;

(iii) si vois_sort(s_i, n) est vide ou ne contient plus d'éléments utilisables (c'est-à-dire qui n'ont pas de marque signifiant qu'ils conduisent à une impossibilité), et si les seuls éléments utilisables de infl_sort(s_i, n) ont un transistor qui ne peut pas conduire avec la valeur v en grille, alors la sensibilisation déterminera un chemin "passant" de n vers s_i, qui sera coupé par la panne ; la sortie peut alors prendre la valeur de haute impédance, auquel cas une phase d'initialisation est nécessaire avant la phase de mise en évidence (on envoie sur la sortie la valeur opposée à celle déterminée par la sensibilisation). On choisit de traiter l'élément ayant le plus petit ordre o_k, et on continue en (2), avec une mention particulière indiquant que le chemin sensibilisé sera coupé par la panne.

(1) on affecte à n_j la valeur de n : $v_j = \{v\}$. On impose sur le nœud g_j la valeur permettant à T_j de conduire ; grâce aux listes connex_...(g_j) , on peut d'une part savoir si g_j peut prendre la valeur désirée, d'autre part rendre passant un chemin vers l'alimentation donnant lieu à cette valeur, par l'utilisation des listes connex_... sur chacun des nœuds le constituant ; cette action impose la valeur des grilles des transistors constituant le chemin ; il reste à remonter de ces grilles vers les entrées, toujours en utilisant les listes connex_... , pour finalement trouver un ensemble d'entrées rendant passant le chemin depuis l'alimentation adéquate jusqu'à g_j. Il faut enfin couper tous les chemins qui peuvent amener sur g_j une valeur qui l'empêche de conduire (car cela créerait, outre une indétermination sur la grille, un court-circuit entre les deux alimentations, car la valeur qui rend passant un transistor est opposée à celle qui le rend bloqué), en amenant sur le premier transistor de ces chemins la valeur qui le rend bloqué (si une impossibilité survient à cause de ce choix, on peut être amené à ne bloquer que le deuxième transistor, ou le troisième, ...).

Ensuite, on recommence l'opération (0) avec n := n_j ; lorsque $n_j = s_i$, la sensibilisation est terminée : un chemin est sensibilisé depuis n jusqu'à s_i, c'est-à-dire que :

- un chemin unique est déterminé de n à s_i,

- les entrées activant ce chemin sont définies,

- ainsi que celles qui désactivent les chemins pouvant doubler celui-ci.

Ainsi, on est sûr que la panne aura un effet tendant à modifier la valeur normale de la sortie s_i.

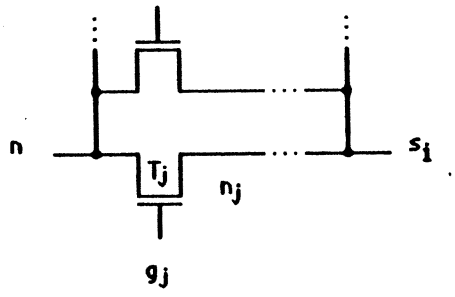


figure 4-7 : utilisation des listes vois_sort.

(2) on cherche à l'aide des listes connex_...(n_l) quelles valeurs n_l peut prendre ; on choisit une valeur v_l, et on affecte à n_k la valeur de n_l : $v_k := \{v_l\}$. En utilisant les listes connex_... , on rend passant un chemin de n_l vers l'alimentation donnant lieu à cette valeur v_l, en imposant à chaque fois la valeur de la grille des transistors le constituant et en remontant vers les entrées, toujours en utilisant les listes connex_... . Les autres chemins qui peuvent amener la valeur v_k sur n_k doivent être désactivés : on impose sur la grille du premier transistor de chaque chemin donné par la liste connex_...(n_k), la valeur qui le rend bloqué, et on remonte vers les entrées (en cas d'impossibilité, on imposera la valeur du second transistor, ou du troisième, ...).

On recommence l'opération (0) avec $n := n_k$; lorsque $n_k = s_i$, la sensibilisation est terminée ;

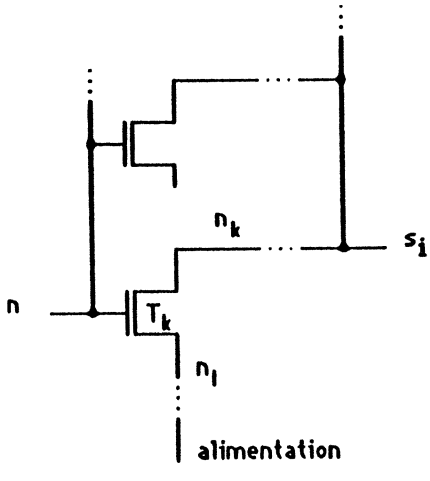


figure 4-8 : utilisation des listes infl_sort.

Dès que le processus se bloque, on revient au dernier choix : choix de la valeur du nœud n_l, choix du transistor T_j ou T_k ... S'il ne peut être débloqué, alors la panne n'a pas d'influence sur la sortie.

4, 122 Le siège de la panne est un transistor

Le processus commence à l'étape (2). Si l'environnement du transistor T en panne est (n_k, g, n_l, T), on sensibilise, comme cela est décrit dans le cas (2), un chemin amenant une valeur v_l sur n_l, on affecte à n_k la valeur de n_l, aux ajustements près, et on désactive les chemins y amenant la même valeur. Ensuite, on continue au cas (0), avec n := n_k. Si la panne est un collage-bloqué, une mention particulière indique que le chemin de sensibilisation sera coupé par la panne. La mise en évidence locale de la panne consiste à amener la valeur adéquate sur la grille g.

4, 13 Informations sur la valeur de la sortie

Une fois que le chemin est sensibilisé, la valeur finale prévue par la sensibilisation du chemin, notée valeur_test(s_i), est la suivante :

(4) si le chemin de n vers la sortie s_i n'a été défini qu'avec les cas (1), alors valeur_test(s_i) = valeur(n), aux ajustements près à travers les transistors (notation { }) ;

(5) sinon (le cas (2) a été utilisé une fois au moins), valeur_test(s_i) est celle qui a été choisie parmi les valeurs que peut prendre le dernier nœud appelé n_l dans la description du cas (2), aux ajustements près ;

(6) si une mention particulière indique que le chemin de sensibilisation est coupé à un endroit par la panne, alors on doit, au cours de la phase d'initialisation, amener sur s_i la valeur opposée à valeur_test(s_i) telle qu'elle a été définie par la sensibilisation[1]. L'absence de changement de la sortie lors de la phase de mise en évidence permet de détecter la panne ;

(7) les autres chemins possibles de s_i vers une alimentation, qui peuvent donner la même valeur valeur_test(s_i), ont été désactivés[2] ;

(8) valeur_test(s_i), lorsqu'elle existe, n'est pas forcément la véritable valeur finale observable, notée valeur_réelle(s_i) : il peut exister un chemin passant vers l'autre alimentation qui entraîne en sortie une autre valeur. On choisit d'opérer une simulation de pannes avec les vecteurs de test déterminés par la génération, plutôt que de rechercher tous les chemins possibles vers l'autre alimentation et de calculer les valeurs finales résultantes : ce calcul

(1) le fait d'amener la valeur d'initialisation sur la sortie, par un chemin qui peut être indépendant du chemin de sensibilisation, est plus général et moins restrictif que la création de cette valeur au siège-même de la panne (par exemple, sur la grille d'un transistor collé-bloqué).

(2) si la valeur de la sortie est 0 (resp. 1), les chemins pouvant créer un 0^+ (resp. 1^-) doivent aussi être désactivés, et inversement.

obligerait à tenir compte de la force des signaux, ce qui est complexe dans le cadre d'une génération de vecteurs de test ; en outre, le calcul des signaux effectué lors de la simulation de pannes prend en compte de manière précise la résistance des transistors.

En conclusion, on obtient les cas suivants :

- valeur_réelle(s_i) = valeur_test(s_i) ; dans ce cas, la panne est bien testable, et le vecteur d'entrée déterminé par la génération est bien un vecteur de test ;

- valeur_réelle(s_i) = opposée[valeur_test(s_i)] ; dans ce cas, il existe un chemin de la sortie s_i vers l'autre alimentation, qui est beaucoup moins résistif ; le vecteur d'entrée est mauvais ;

- valeur_réelle(s_i) = I ; il existe un chemin de s_i vers l'autre alimentation, qui est aussi résistif ; le vecteur d'entrée est mauvais ;

- lorsqu'il y a une phase d'initialisation (coupure du chemin), et si la valeur de la sortie ne change pas, le vecteur d'entrée est un vecteur de test.

4, 14 Mise en évidence locale de la panne depuis les entrées

Les listes connex_... sont examinées une fois avant la sensibilisation, pour déterminer si la valeur qui met la panne en évidence peut être créée. Il ne faut pas confondre à ce sujet ce qui est appelé plus haut "valeur imposée" avec la valeur mettant la panne en évidence : la première est la valeur que "crée" la panne et qui est "propagée" vers les sorties (un collage à 1 crée la valeur imposée 1) ; la seconde est la valeur qu'il faut "amener" au nœud depuis les entrées pour que la panne puisse être mise en évidence (un collage à 1 demande pour être mis en évidence la valeur 0). Cette vérification n'assure pas que la panne puisse être effectivement détectée.

Exemple :

pour le circuit de la figure 4-4, on effectue une génération de vecteurs de test pour la panne "n2 collé à 0". La valeur qui la met en évidence est 1 : connex_1(n2) = {(n2, e1, n1, T4, 2)}, donc "il se peut" que la valeur 1 puisse être appliquée en n2.

La sensibilisation vers s suit le déroulement suivant :

- vois_sort(s, n2) = {(n2, e1, n1, T4, 3)} ;
- infl_sort(s, n2) = {(s, n2, e2, T6, 1)} ;

On choisit de propager la panne en utilisant vois_sort(s, n2) (cas (1)), car le transistor T6 ne conduit pas avec la valeur imposée 0 de n2.

- valeur(e1) := 1 ; valeur(n1) := (valeur(n2)) = 0 ; on recommence avec n1 :

- vois_sort(s, n1) = ((n1, e1, n2, T4, 2), (n1, e1, n3, T2, 4)) ;

- infl_sort(s, n1) = ((n3, n1, Gnd, T3, 4)) ;

on ne peut pas choisir le premier élément de vois_sort(s, n1) car le transistor T4 a déjà été rencontré. On prend donc le deuxième :

- valeur(e1) := 0 ; c'est incompatible, donc on revient au dernier choix ; on prend donc l'élément de infl_sort(s, n1) (cas (2)), car T3 conduit si valeur(n1) = 0 :

- valeur(n3) := (Gnd) = 0^+ ; on recommence avec n3 :

- vois_sort(s, n3) = ((n3, e1, n1, T2, 3)) ; le nœud n1 a déjà été rencontré, donc on revient en arrière, pour finalement choisir d'utiliser la liste infl_sort(n2) ; comme le transistor T6 ne conduit pas avec un 0 en grille, on mentionne que la panne coupera le chemin de sensibilisation ; la valeur_test de la sortie est donc (valeur(e2)), et une valeur d'initialisation doit être déterminée ; la phase d'initialisation consistera à amener en s une valeur opposée, par un chemin qui n'inclut pas e2 ; or, l'examen des listes connex_...(s) montre qu'un tel chemin n'existe pas. Donc la panne n'est pas testable.

Bien que la panne ne soit pas testable, on décrit la mise en évidence locale de la panne, afin d'illustrer sa description. Elle utilise connex_1(n2) = (n2, e1, n1, T4, 2)) :

- e1 := 1 ; on recommence avec n1 :

- connex_1(n1) = ((n1, e1, Vdd, T1, 1)), donc e1 := 1.

La solution utilisant les listes connex_... peut être comparée à celles décrites dans [AMA85] et [PRA87], où les chemins de conduction sont analysés. De fait, l'analyse des chemins de conduction est tout-à-fait cohérente avec une description au niveau interrupteur, puisqu'elle ne s'apparente en rien à une tentative de regroupement sur le réseau d'interrupteurs. Dans cette technique, les chemins analysés sont ceux reliant les nœuds "importants" aux nœuds "source" ; pour chaque nœud important, les chemins de conduction sont classés par ordre de force ; celle-ci permet lors de la simulation de déterminer le chemin dominant et par suite l'état final du nœud. Ici, les chemins de conduction servent uniquement à savoir créer telle ou telle valeur sur un nœud. La simulation utilise une autre méthode, décrite dans les précédents chapitres.

4, 2 Etude de la génération pour les pannes envisagées

Cette étude se borne à la description de l'étape de mise en évidence locale, qui est la première effectuée. Celle-ci consiste, comme cela est expliqué précédemment, à déterminer la valeur qui met la panne en évidence sur le nœud où elle se trouve. Les pannes traitées sont celles qui sont exposées dans les chapitres 3, 311 et 3, 312.

4, 21 Conception descendante

- collage-passant d'un transistor.

La valeur à amener sur la grille est un 0 pour un transistor N et un 1 pour un transistor P. Le fait que le collage puisse créer un court-circuit et donc induire une valeur différente de celle prévue par la génération n'est pris en compte que dans la simulation de pannes qui vérifie l'efficacité des vecteurs.

- collage-bloqué d'un transistor.

La valeur à amener sur la grille est un 1 pour un transistor N et un 0 pour un transistor P. La valeur d'initialisation à appliquer sur la sortie est l'opposée de celle trouvée par la sensibilisation du chemin.

- court-circuit entre deux nœuds n1 et n2.

Dans ce cas, chacun des nœuds a une valeur mettant la panne en évidence. Ces deux valeurs doivent être "opposées" :

- valeur(n1) $\in \{0, 0^+\}$ et valeur(n2) $\in \{1, 1^-\}$, ou l'inverse.

La valeur imposée sur les deux nœuds est délicate à définir, car la valeur créée en réalité par le court-circuit dépend de la résistance des réseaux concernés. De manière à opérer le plus généralement possible, on choisit de propager la valeur I sur les deux nœuds..

- modification du seuil d'un transistor ; modification de la résistance du canal.

Ces pannes sont testées comme un collage-passant si le seuil descend ou si la résistance diminue, et comme un collage-bloqué si c'est l'inverse.

- collage d'un nœud.

La valeur qui met la panne en évidence est l'opposée de la valeur de collage, qui est la valeur à propager.

4, 22 Vérification ascendante

Dans ce cas, le réseau est composé de transistors et de résistances (objets R_i). Les listes diverses définies pour la simulation et la génération de vecteurs de test doivent intégrer ces dernières, pour que la méthode soit inchangée. Les deux pannes spécifiques de ce type de réseau, par rapport à celles d'un réseau de transistors, sont celles touchant les lignes.

- coupure sur une ligne.

Cette panne est très différente dans son traitement d'un collage-bloqué sur un transistor. En effet, on peut commander un transistor, et donc trouver une valeur à appliquer à sa grille, alors qu'on ne peut commander une résistance avec le reste du réseau. Le seul moyen de générer un vecteur de test pour ce type de panne est de considérer que la coupure se ramène à un collage sur un des côtés de la coupure. Le nœud dont la valeur est collée est le premier dont la liste vois_sort est vide parmi ceux qui se trouvent du côté de la sortie (un tel nœud est la grille d'un ou de plusieurs transistors). On génère pour ce nœud les vecteurs de test pour un collage à 1, à 1^-, à 0, et à 0^+.

- modification de la résistance d'une ligne.

On ne peut générer des vecteurs de test pour cette panne que si elle équivaut à une coupure de ligne, ce qui ramène au cas précédent. En effet, si la ligne n'est pas coupée, il ne peut y avoir de collage sur un côté.

La couverture des pannes n'a pas été traitée ici, l'étude se bornant à déterminer une technique de génération de vecteurs de test pour un certain nombre de pannes. Toutefois, celle-ci n'exclut pas l'utilisation de techniques de "détection multiple" : par exemple, le test d'un collage à 1 sur la grille d'un transistor inclut le test du collage-passant de celui-ci, le test des collages d'une grille inclut le test de la coupure sur la ligne correspondante... L'aspect multivalué de la méthode intervient uniquement pour ce qui concerne les valeurs des signaux, avec les listes connex_1, connex_1^-, connex_0, et connex_0^+. Les forces ne sont calculées que dans la simulation de pannes qui vérifie les vecteurs de test déterminés par la génération.

CONCLUSION

CONCLUSION

Le choix d'un modèle multivalué adapté à la vérification d'un circuit VLSI décrit au niveau "interrupteur" doit suivre une démarche logique. Les états de l'algèbre et l'algorithme de calcul ne peuvent être définis qu'après une étude précise des besoins de ce type de vérification. En effet, un réseau "brut" d'interrupteurs étant par essence bi-directionnel, le calcul des signaux doit nécessairement résoudre ce problème, sans avoir recours à des techniques artificielles transformant le réseau. Celles-ci permettent en général d'accélérer le traitement, et sont en cela très positives. Mais le but sous-jacent des spécifications proposées est de créer un modèle qui accepte directement en donnée un réseau d'interrupteurs, de construire logiquement une algèbre de travail, et de spécifier une méthode de calcul d'états entièrement adaptée plutôt que de rechercher un algorithme rapide. Les circuits concernés par cette étude sont a priori anarchiques et de petite taille.

*
* *

Il a ainsi été nécessaire de créer une opération &, calculant les nouveaux états de part et d'autre d'un transistor devenant passant, dont l'utilisation soit en harmonie avec l'algèbre d'états choisie. Celle-ci est composée d'une valeur d'initialisation et de vingt couples (valeur, force) notés v_f, pour la technologie CMOS ; ces états permettent de modéliser avec une précision acceptable le fonctionnement d'un circuit VLSI :

$$\mathcal{A} = \{1_1, 0_1, 1^-_1, 0^+_1, 1_2, 0_2, 1^-_2, 0^+_2, 1_3, 0_3, 1^-_3, 0^+_3, 1_4, 0_4, 1^-_4, 0^+_4, I_1, I_2, I_3, I_4, X\}.$$

L'algorithme traite les boucles et les portes de transmission de la même façon que les portes unidirectionnelles, sans opérer de compilation préliminaire. Pour cela, à chaque nœud n sont associées quatre listes. VDD(n) et GND(n) décrivent ses connexions dynamiques avec les alimentations, voisinage(n) et influence(n) décrivent son environnement statique.Le calcul des états répond ainsi entièrement à l'esprit général de la méthode, qui reste constamment en contact avec le fonctionnement réel d'un circuit VLSI. Pour le calcul temporel des signaux, un modèle à délai unitaire analogue à celui décrit par R. E. Bryant [BRY84] a été choisi pour sa simplicité. Néanmoins, d'autres modèles plus précis peuvent être utilisés pour la simulation de parties "à risque".

*
* *

Le modèle présenté est ensuite enrichi par des spécifications pour la simulation de pannes, à la lumière des derniers développements concernant leur modélisation, en particulier pour la technologie CMOS. En effet, les résultats obtenus par D. Baschiera [BAS86] soulèvent de nouveaux problèmes, qui concernent notamment le niveau en sortie créé par certaines pannes (collages-passant, ...), dont il doit être tenu compte en simulation. Pour cela, l'algèbre est modifiée pour que la résistance des transistors soit prise en compte plus précisément. Les forces des signaux sont ainsi calculées de manière cumulative, pour rendre compte de la résistance des réseaux conducteurs en jeu en cas de panne : certaines pannes créent un chemin conducteur entre l'alimentation Vdd et la masse Gnd, et le niveau de sortie dépend essentiellement de la résistance de ce chemin. Le modèle garde néanmoins son aspect multivalué pour rester compatible avec le premier. En particulier, l'ensemble des valeurs garde sa structure de treillis, ce qui est une condition nécessaire pour que l'application de l'opérateur & converge en une itération ; la valeur indéterminée I est conservée en tant qu'élément absorbant : bien que cette valeur ne soit pas souhaitée comme résultat de simulation - elle ne permet pas de décider de l'aptitude d'un vecteur à mettre une panne en évidence -, le traitement est beaucoup plus rapide que le calcul précis des valeurs des signaux, avec par exemple une algèbre représentant toutes les tensions entre 0 V et 5 V, espacées de 1/2 V - ce calcul ne converge pas en une itération.

Les pannes concernées par cette étude sont regroupées en deux classes : d'une part, les pannes qui peuvent être simulées lors de la conception descendante du circuit - du niveau fonctionnel au niveau des masques, d'autre part, celles qui peuvent être simulées lors de la vérification ascendante du circuit - du niveau des masques au niveau fonctionnel, par des extractions successives. Dans le premier cas, aucune information ne permet de prévoir la composition topologique des nœuds. Par conséquent, certaines pannes (court-circuit entre deux niveaux de masques, coupure sur un nœud, ...) ne peuvent être simulées. En outre, la simulation d'un court-circuit entre deux nœuds n'a pas forcément d'intérêt, dans la mesure où la proximité réelle de ces deux nœuds est inconnue. Toutes les pannes spécifiques aux transistors (collage-passant, collage-bloqué, variation de la tension de seuil, ...) restent accessibles, et le modèle permet de les simuler assez finement. Dans le deuxième cas, le réseau brut est légèrement modifié lors de l'extraction du niveau des masques vers le niveau des transistors : les lignes deviennent des objets qui peuvent représenter leur résistance, et qui sont traités comme des transistors (application de l'opérateur &, ...) ; les nœuds regroupent les contacts, les bifurcations, et les extrémités des transistors. Avec une telle représentation, les pannes qui ne sont pas accessibles dans une phase descendante, peuvent être simulées de manière satisfaisante.

*
* *

La méthode de génération de vecteurs de test décrite dans ce travail utilise le modèle spécifié pour la simulation de pannes. En effet, les résultats de celle-ci devant être utiles à la génération, l'environnement doit rester le même. La génération des vecteurs de test suit trois étapes principales : la panne doit être mise en évidence localement ; l'anomalie qui est créée doit être propagée vers les sorties ; les chemins définis doivent être sensibilisés, par une remontée vers les entrées. La propagation de l'anomalie vers les sorties demande d'autres outils que ceux définis précédemment. Pour chaque nœud n et pour chaque sortie s, deux listes statiques sont construites : vois_sort(s, n) donne la liste des voisins de n qui sont connectés de loin en loin à s, infl_sort(s, n) donne la liste des nœuds influencés par n qui sont connectés de loin en loin à s. Ces listes permettent de trouver un chemin de n à s. La détermination du niveau final de la sortie, ainsi que la sensibilisation des chemins exigent également d'autres outils. Pour chaque nœud n, des listes statiques indiquent la valeur que peut prendre n et par quel chemin. Ces listes, appelées connex_1, connex_1^-, connex_0, connex_0^+, ne doivent pas être confondues avec les listes VDD et GND utilisées par la simulation, qui sont dynamiques et permettent de définir la prépondérance d'un nœud sur un autre. Elles sont utilisées pour déterminer quels doivent être les états des grilles des transistors du chemin de n vers les alimentations pour que celui-ci ait la valeur 1, 1^-, 0, ou 0^+. Elles permettent, de loin en loin, de remonter jusqu'aux entrées et de définir les vecteurs de test. La mise en évidence locale de la panne est décrite pour chaque panne envisagée ; elle consiste à générer une valeur ayant l'effet inverse de celui de la panne : par exemple, un collage à 1 demande la valeur 0. Certaines pannes (collage-bloqué, ...) demandent une phase d'initialisation avant la phase de mise en évidence, à cause de la valeur de haute impédance qui est générée, et qui transforme un circuit combinatoire en un circuit séquentiel.

*
* *

La simulation, la simulation de pannes et la génération de vecteurs de test des circuits VLSI décrits sous forme de réseaux d'interrupteurs, ne sauraient être dissociées d'un certain nombre d'outils de génération et de manipulation pour la conception de circuit, tels ceux de description et de compaction. Ces trois applications doivent faire partie d'un ensemble complet de conception, qui comprend :

- deux outils de description symbolique hiérarchiques, l'un métrique comme celui de [JER85], l'autre à grille virtuelle. Les deux doivent posséder des mécanismes d'assemblage automatique, similaires à ceux du système LUBRICK [SCH83], et admettre une entrée graphique et textuelle. Les outils de compilation doivent fournir de telles descriptions ;

- un simulateur électrique du type de SPICE, associé aux deux outils de description ;

- un compacteur et un décompacteur, qui sont des outils permettant de passer d'une description à l'autre, pour la description finale des masques ;

- un vérificateur de règles de dessin ("DRC") hiérarchique, associé à l'outil de description métrique ;

- un processeur d'expansion, générant des formats de description classiques (comme GDS-II, CIF, EDIF, ...), à partir d'une description métrique ;

- un outil permettant de nommer les nœuds de la description métrique. Ces noms sont utilisés par la suite, notamment lors de la mise au point de prototypes ("debugging") avec un faisceau d'électrons ("electron-beam testing"). De cette manière, le concepteur peut comparer la valeur logique d'un nœud pour une certaine entrée, à la valeur attendue calculée auparavant.

Cet ensemble complet d'outils de conception et de mise au point, décrit dans la figure c-2, est développé dans l'Equipe d'Architecture des Ordinateurs du laboratoire TIM3, sous le nom de NAUTILE ("New AUtomatic and Technology-Independent Layout Environment") [BON87].

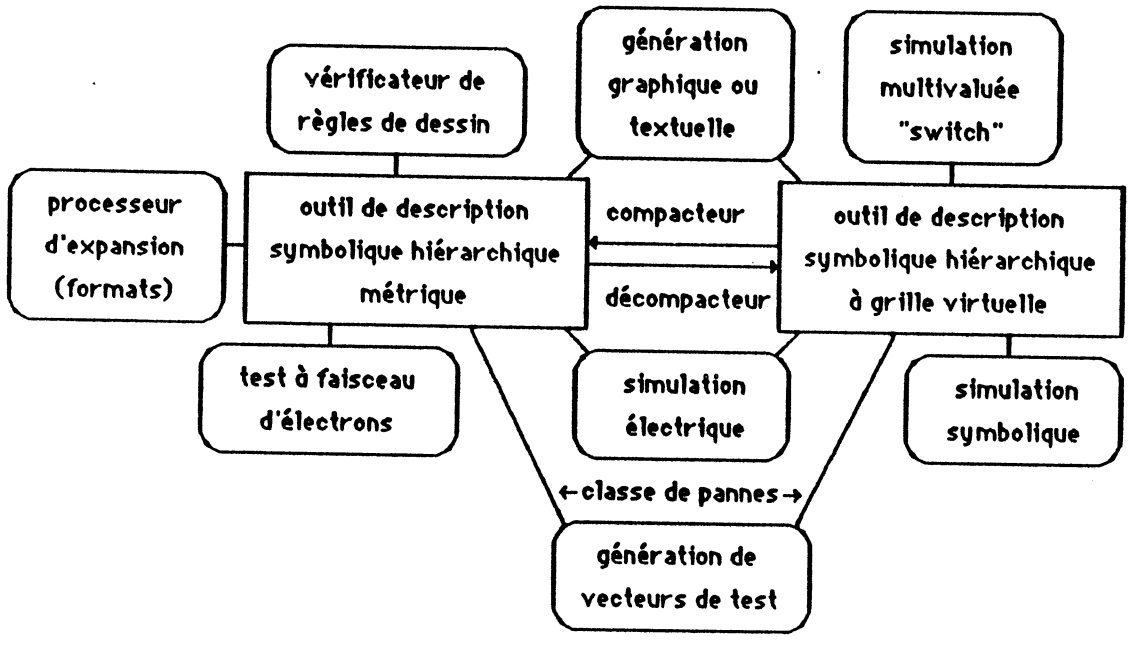


figure c-1 : ensemble complet d'outils de conception.

*
* *

Une extension intéressante du modèle de simulation concerne la simulation symbolique. Celle-ci consiste à représenter un réseau d'interrupteurs par un ensemble de systèmes d'équations linéaires, dont la résolution donne le nouvel état stationnaire de chaque nœud. Cette résolution utilise l'élimination gaussienne qui permet de résoudre un système d'équations linéaires grâce à une série de substitutions employant un pivot, qui transforment la matrice du système en une matrice diagonale. Ce type de simulation a été étudié pour une algèbre de valeurs ternaire {0, 1, X} [BRa87][BRb87]. Le passage des valeurs aux variables booléennes suit le principe suivant : on associe à chaque nœud n un suffixe, qui le transforme en une variable booléenne ; celle-ci indique la possibilité pour n de prendre la valeur indiquée par le suffixe. Par exemple, n.1 = 1 indique que n peut prendre la valeur 1, n.0 = 0 indique que n ne peut pas prendre la valeur 0, n.1 = n.0 = 1 indique un conflit. Brièvement, la détermination de l'état stationnaire des nœuds d'un réseau consiste en la résolution des équations n.0 = 1 et n.1 = 1 pour chaque nœud n.

Exemple :

les équations de la sortie s du circuit de la figure c-2 sont les suivantes :

s.1 = a.0 b.0 (opérateur "et") ;
s.0 = a.1+b.1 (opérateur "ou").

Il serait intéressant d'étendre cette technique de simulation à l'algèbre définie pour la simulation de pannes, qui est plus à même de représenter les états au sein d'un réseau de transistors. Pour que l'algèbre {1, 0, 1^-, 0^+, I} puisse être utilisée, il faut vérifier que les opérations de description des chemins dans le réseau puissent y être adaptées, et que l'élimination gaussienne puisse encore être employée. A priori, les variables booléennes, indépendamment de toute notion de force, pourraient être utilisées de la manière suivante :

n.1	1	0	0	0	1	ϕ	1	ϕ
$n.1^-$	ϕ	1	0	0	ϕ	1	ϕ	1
n.0	0	0	0	1	1	1	ϕ	ϕ
$n.0^+$	0	0	1	ϕ	ϕ	ϕ	1	1
valeur(n)	1	1^-	0^+	0	I	I	I	I

Exemple :

les équations de la sortie s du circuit de la figure c-3 devraient être les suivantes :

$s.0 = (b.1+b.1^-)[(a.1c.0)+(a.1^-c.0)+(a.1^-c.0^+)+(a.1c.0^+)]$;

$s.1 = (a.0+a.0^+)[(d.1b.0)+(d.1^-b.0)+(d.1^-b.0^+)+(d.1b.0^+)]$;

$s.1^- = [(c.1a.1)+(c.1^-a.1)+(c.1^-a.1^-)+(c.1a.1^-)][(d.1b.0)+(d.1^-b.0)+(d.1^-b.0^+)+(d.1b.0^+)]$;

$s.0^+ = [(d.0b.0)+(d.0^+b.0)+(d.0^+b.0^+)+(d.0b.0^+)][(c.0a.1)+(c.0^+a.1)+(c.0^+a.1^-)+(c.0a.1^-)]$.

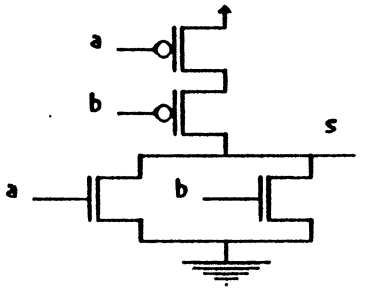


figure c-2 : porte NOR CMOS.

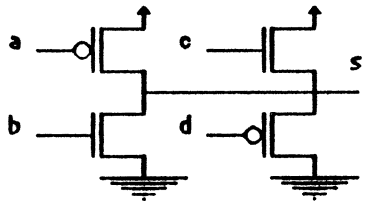


figure c-3 : la sortie s peut prendre les quatre valeurs déterminées de l'algèbre.

L'adaptation du modèle décrit dans ce document à la simulation symbolique serait d'un grand intérêt, du fait de l'aspect attrayant de la description d'un réseau de transistors par des équations linéaires, et de sa grande souplesse d'utilisation.

BIBLIOGRAPHIE

BIBLIOGRAPHIE

[AGR84] **P. Agrawal**
Test generation at the switch level.
International Conference on Computer-Aided Design, November 1984.

[ALA84] **A. Al-Arian, D. P. Agrawal**
Modeling and testing of CMOS circuits.
International Conference on Computer Design, October 1984.

[ALM84] **C. B. Almeida, M. J. A. Lança**
An eight-value modelling technique for logic simulation of transmission gates and buses.
E. D. A. 1984.

[AMA85] **P. Amadou, C. Durante, P. Guiraudou, C. Landrault, M. Tabusse**
Moslam : a switch-level simulator based on an efficient analysis of conducting paths.
ESSIRC, September 1985.

[AMB83] **P. Amblard, M. Crastes de Paulet, J. Rarivomanana, G. Saucier**
CADOC : a functionnal specification and simulation tool.
ICCAD, Santa Clara, 1983.

[AUV87] **D. Auvergne, D. Deschacht, M. Robert**
Explicit formulation of delays in CMOS VLSI.
Electronics letters, vol. 23 n° 14, July 1987.

[BAN82] **P. Banerjee, J. A. Abraham**
Fault characterization of VLSI MOS circuits.
ICCD, 1982.

[BAN83] **P. Banerjee, J.A. Abraham**
MURPHY : a logic simulator for MOS VLSI circuits.
Coordinated Science Laboratory, Illinois, Urbana, 1983.

[BAS84] **D. Baschiera, B. Courtois**
Testing CMOS : a challenge.
VLSI Design, vol. 10, October 1984.

[BAS86] **D. Baschiera, B. Courtois**
Advances in fault modelling and test pattern generation for CMOS.
ICCD, New-York, October 1986.

[BAS86] **D. Baschiera**
Modélisation de pannes et méthodes de test de circuits intégrés CMOS.
Thèse de Doctorat, spécialité Informatique, I.N.P.-G. , Mars 1986.

[BAY84] **R. J. Bayruns et al.**
Delay analysis of Si NMOS Gbits/s logic circuits.
IEEE Journal of Solis-State Circuits, Vol. sc. 19, October 1984.

[BEI74] **J. Beister**
An unified approach to combinational hazards.
IEEE Transactions on Computers, June 1974.

[BON87] **P. Bondono, J. -P. Caisso, A. Hornik, A. A. Jerraya**
Définition préliminaire d'un système d'aide à la conception automatique de circuits V.L.S.I. : NAUTILE.
Rapport interne IMAG/TIM3, Juin 1987.

[BRE74] **M. A. Breuer, R. L. Harrison**
Procedures for eliminating static and dynamic hazards in test generation.
IEEE Transactions on Computers, October 1974.

[BRO85] **J. A. Brzozowski, C. -J. Seger**
A characterization of ternary simulation of gate networks.
Department of Computer Science, VLSI Group, Waterloo, report CS-85-37.

[BRY83] **R. E. Bryant, M. D. Schuster**
Fault simulation of MOS digital circuits.
VLSI Design, October 1983.

[BRY84] **R. E. Bryant**
Race detection in MOS circuits by ternary simulation.
California Institute of Technology, 1984.

[BRa87] **R. E. Bryant**
Algorithmic aspects of symbolic switch network analysis.
IEEE Transactions on Computer-Aided Design, July 1987.

[BRb87] **R. E. Bryant**
Boolean analysis of MOS circuits.
IEEE Transactions on Computer-Aided Design, July 1987.

[BUX86] **J. Buxo**
A model for the delay analysis of CMOS VLSI circuits.
Private communication, 1986.

[CHE83] **M. C. Chen, C. A. Mead**
A hierarchical simulator based on formal semantics.
3rd Caltech Conference on VLSI, March 1983.

[CHE84] **H. H. Chen, R. G. Matthews, J. A. Newkirk**
Test generation for MOS circuits.
International Test Conference, 1984.

[CHI83] **K. W. Chiang, Z. G. Vranesic**
On fault detection in CMOS logic networks.
20th Design Automation Conference, June 1983.

[CHI83] **K. W. Chiang, Z. G. Vranesic**
On fault detection in CMOS logic networks.
20th Design Automation Conference, June 1983.

[COU81] **B. Courtois**
Test et LSI.
Thèse de Doctorat d'Etat ès Sciences U.S.M.-G-I.N.P.-G., Juin 1981.

[DOS84] **M. H. Doshi, R. B. Sullivan, D. M. Schuler**
THEMIS - a mix mode, multilevel, hierarchical interactive digital circuit simulator.
21st DAC, 1984.

[EIC65] **E. B. Eichelberger**
Hazard detection in combinational and sequential switching circuits.
IBM J. Res. & Dev., Vol. 9, March 1965.

[FAN74] **G. Fantauzzi**
An algebraic model for the analysis of logical circuits.
IEEE Transactions on Computers, June 1974.

[FLA83] **P. L. Flake, P. R. Moorby, G. Musgrave**
An algebra for logic strength simulation.
20th DAC, 1983.

[FUJ85] **H. Fujiwara**
Logic testing and design for testability.
Computer Systems, the MIT Press, 1985.

[GIA82] **M. Giambiasi, M. Sigal, J. C. Rault**
A deductive method for non classical logic fault simulation.
International Conference on Circuits and Computers, 1982.

[GON84] **M. Gonauser, F. Egger, D. Frantz**
SMILE - a multilevel simulation system.
ICCD, 1984.

[HAJ83] **I. Hajj, D. Saab**
Fault modeling & logic simulation of MOS VLSI circuits based on logic expression extraction.
ICCAD, 1983.

[HAa82] **J. P. Hayes**
A fault simulation methodology for VLSI.
19th DAC, 1982.

[HAb82] **J. P. Hayes**
A unified switching theory with applications to VLSI design.
Proceedings of the IEEE, October 1982.

[HAY83] **J. P. Hayes**
Fault modeling for digital MOS integrated circuits.
IEEE Transactions on Computers, June 1983.

[HAY84] **J. P. Hayes**
A systematic approach to multivalued simulation.
ICCD, 1984.

[HEY83] **M. H. Heydeman**
A survey of MOS logic simulation tools.
ESSIRC, 1983.

[HOD84] **S. Hodgson**
A multilevel, mixed state simulator for hierarchical design verification.
ICL, U. K. , 1984.

[HUR84] **S. L. Hurst**
Multiple-valued logic - Its status and its future.
IEEE Transactions on Computers, December 1984.

[JAI85] **S. K. Jain, V. D. Agrawal**
Modeling and test generation algorithms for MOS circuits.
IEEE Transactions on Computers, May 1985.

[JER85] **A. A. Jerraya, E. Rosier, F. R. Rougeaux, B. Courtois**
A hierarchical symbolic design layout tool : STYX.
International Conference on VLSI, Tokyo, 1985.

[KAW84] **M. Kawai, J. P. Hayes**
An experimental MOS fault simulation program CSASIM.
21st DAC, 1984.

[KEA84] **M. A. Kearney**
DECSIM : a multilevel simulation system for digital design.
ICCD, 1984.

[KIM84] **Y. H. Kim, J. E. Kleckner, R. A. Saleh, A. R. Newton**
Electrical-logic simulation.
ICCAD, 1984.

[LEW72] **D. W. Lewis**
Hazard detection by quinary simulation of logic devices with bounded propagation delay.
Proc. Design Automation Conference, June 1972.

[LIN84] **T.-M. Lin, C. A. Mead**
Signal delay in general RC networks with applications to timing simulation.
Conference on Advanced Research in VLSI, M.I.T., January 1984.

[MAL84] **Y. K. Malaiya**
Testing stuck-on faults in CMOS integrated circuits.
ICCAD, 1984.

[McC62] **E. J. McCluskey**
Transients in combinational logic circuits.
Redundancy Techniques for Computing Systems, Spartan Book, 1962.

[McC81] **E. J. McCluskey**
Logic simulation.
19th of May 1981.

[MO 84] **Z. L. Mo, M. R. Lightner**
A two parameter delay model for switch-level simulation.
ICCD, 1984.

[OKA83] **K. Okazaki, T. Moriya, T. Yahara**
A multiple media delay simulator for MOS LSI circuits.
20th DAC, 1983.

[OUS83] **J. K. Ousterhout**
Crystal : a timing analyzer for NMOS VLSI circuits.
3rd Caltech Conference on VLSI, March 1983.

[OUS84] **J. K. Ousterhout**
Switch-level delay models for digital MOS VLSI.
21st DAC, 1984.

[PRA87] **S. Pravossoudovitch**
Contribution au test des circuits intégrés MOS : génération automatique de vecteurs de test au niveau transistors interrupteurs.
Thèse Doctorat d'Etat ès Sciences, U. S .T. du Languedoc, Septembre 1987.

[RAJ86] **J. Rajski, H. Cox**
Stuck-open fault testing in large CMOS networks by dynamic path tracing.
ICCD 1986.

[RAM83] **V. Ramachandran**
An improved switch-level simulator for MOS circuits.
20th DAC, 1983.

[RAM86] **V. Ramachandran**
Algorithmic aspects of MOS switch-level simulation with race detection.
IEEE Transactions on Computers, May 1986.

[RED85] **M. K. Reddy, S. M. Reddy, P. Agrawal**
Transistor level test generation for MOS circuits.
22nd DAC, 1985.

[REN85] **M. Renovell, G. Cambon, D. Auvergne**
FSPICE : a tool for fault modelling in MOS circuits.
INTEGRATION, the VLSI journal 3, Elsevier Science Publ. B. V., 1985.

[ROY85] **C. Roy, L. -P. Demers, E. Cerny, J. Gecsei**
An object-oriented switch level simulator.
22nd DAC, 1985.

[SAA84] **D. Saab, I. Hajj**
Parallel and concurrent fault simulation of MOS circuits.
ICCD, 1984.

[SCH83] **J. P. Schoellkopf**
Lubrick : a silicon assembler and its application to data-path design for FISC.
VLSI 83, edited by F. Anceau and L. J. Aas, Elsevier Science Publishers.

[SCH84] **G. G. Schrooten**
Switch, a switch level logic simulator with unique element models.
ICCAD, 1984.

[SCa84] **M. D. Schuster, R. E. Bryant**
Concurrent fault simulation of MOS digital circuits.
Conference on Advanced Research in VLSI, January 1984.

[SCb84] **M. Schuster**
Switch-level fault simulation of MOS digital circuits.
Part of "Master of Science" Degree , 5132:TR:84, May 1984.

[SHI86] **H. -C. Shih, J. A. Abraham**
Transistor-level test generation for physical failures in CMOS circuits.
23rd DAC, 1986.

[SIL84] **A. L. Silburt, R. C. Foss, W. F. Petrie**
An efficient MOS transistor model for Computer-Aided Design.
IEEE Transactions on Computer-Aided Design, January 1984.

[STE83] **P. Stevens, G. Arnout**
BIMOS, a MOS oriented multilevel logic simulator.
20th DAC, 1983.

[TAH84] **H. El. Tahawy, G. Mazaré, M. Polze, A. Puissochet**
Functionnal modeling for logic simulation.
ICCD, 1984.

[THO75] **E. W. Thompson, S. A. Szygenda**
Three levels of accuracy for the simulation of different fault types in digital circuits.
12th DAC, 1975.

[VIV86] **C. Vivier, G. Fournie**
Automatic modelling of MOS transistor networks for test pattern generation.
ITC, September 1986.

[WAD78] **R. L. Wadsack**
Fault modelling and logic simulation of CMOS and MOS integrated circuits.
The Bell System Technical Journal, May-June 1978.

[WAT80] **J. Watanabe, J. Miura, T. Kurashi, I. Suetsugu**
Seven value logic simulation for MOS LSI circuits.
ICCC, 1980.

[WHI82] **M. H. White**
Characterization of CMOS devices for VLSI.
IEEE Transactions on Electron Devices, Vol. ED-29, N° 4, April 1982.

[YOE84] **M. Yoeli, J. A. Brzozowski**
A mathematical model for digital CMOS networks.
Rapport CS-84-22, University of Waterloo, Canada, August 1984.

TABLE DES MATIERES

TABLE DES MATIERES

CHAPITRE 2

CHAPITRE 3

CHAPITRE 4

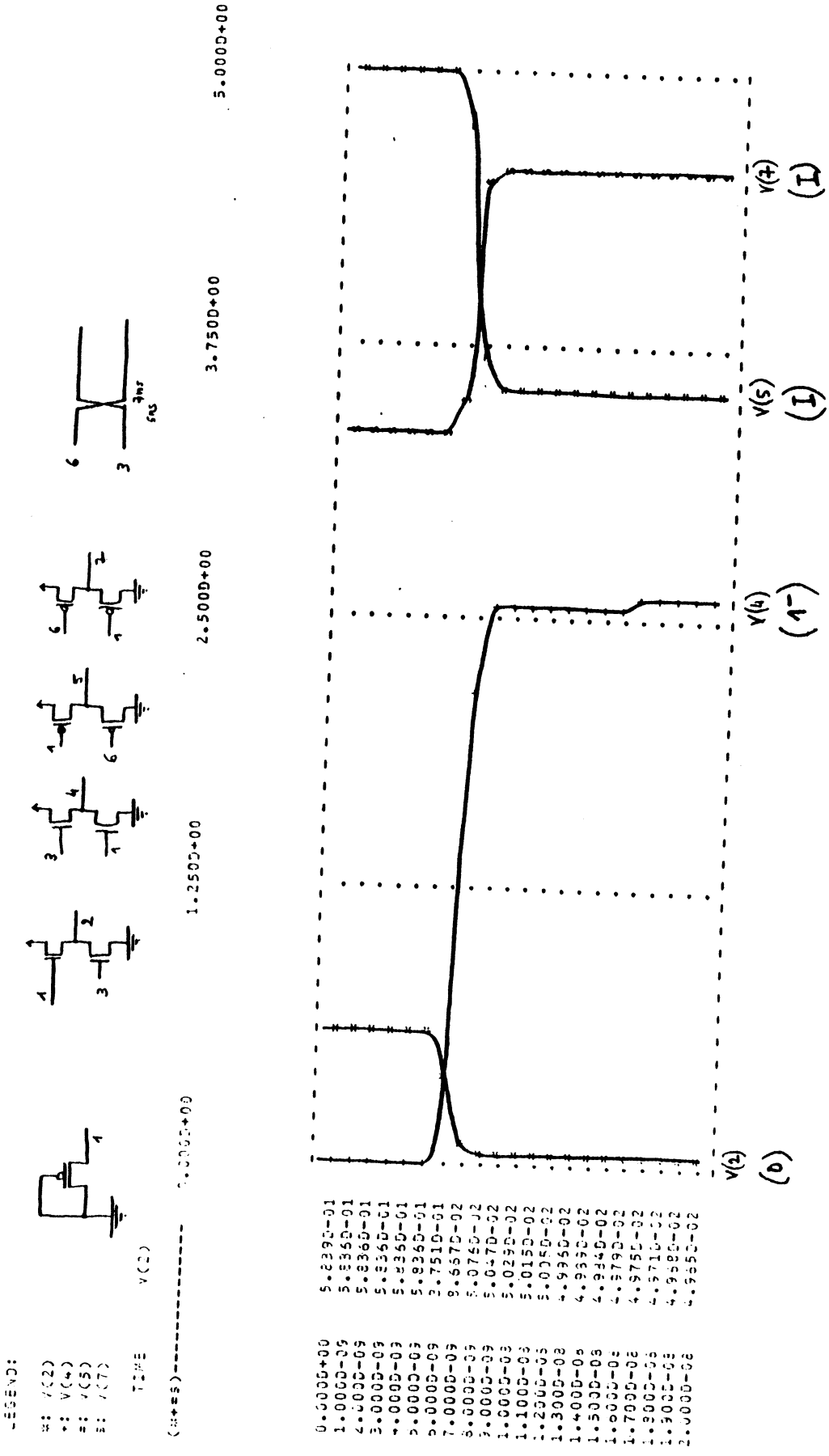
ANNEXE 1

Simulations SPICE de circuits

déterminant l'augmentation éventuelle de résistance

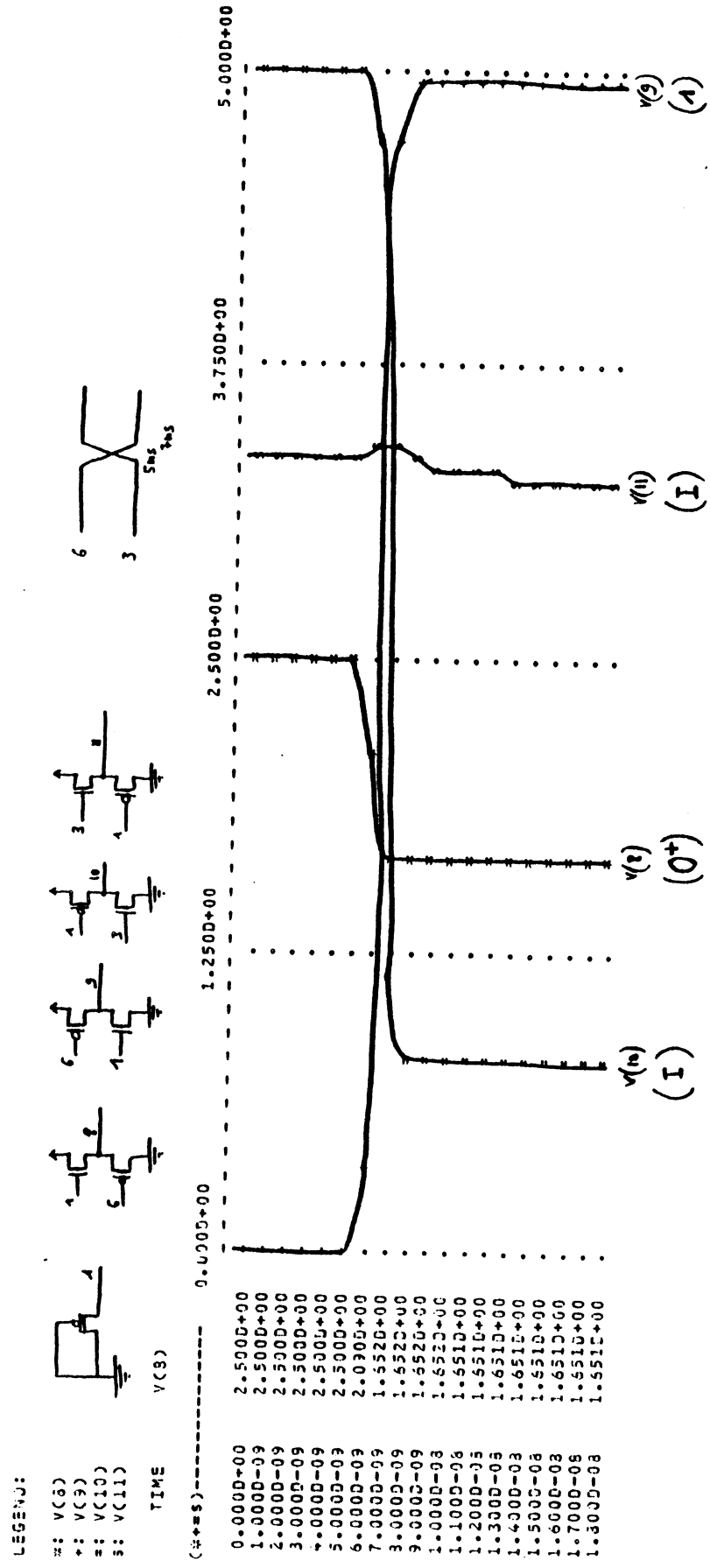
pour un transistor dont la grille est à un niveau altéré

Augmentation de la résistance d'un transistor de taille Standard dont la grille a la valeur 0^+

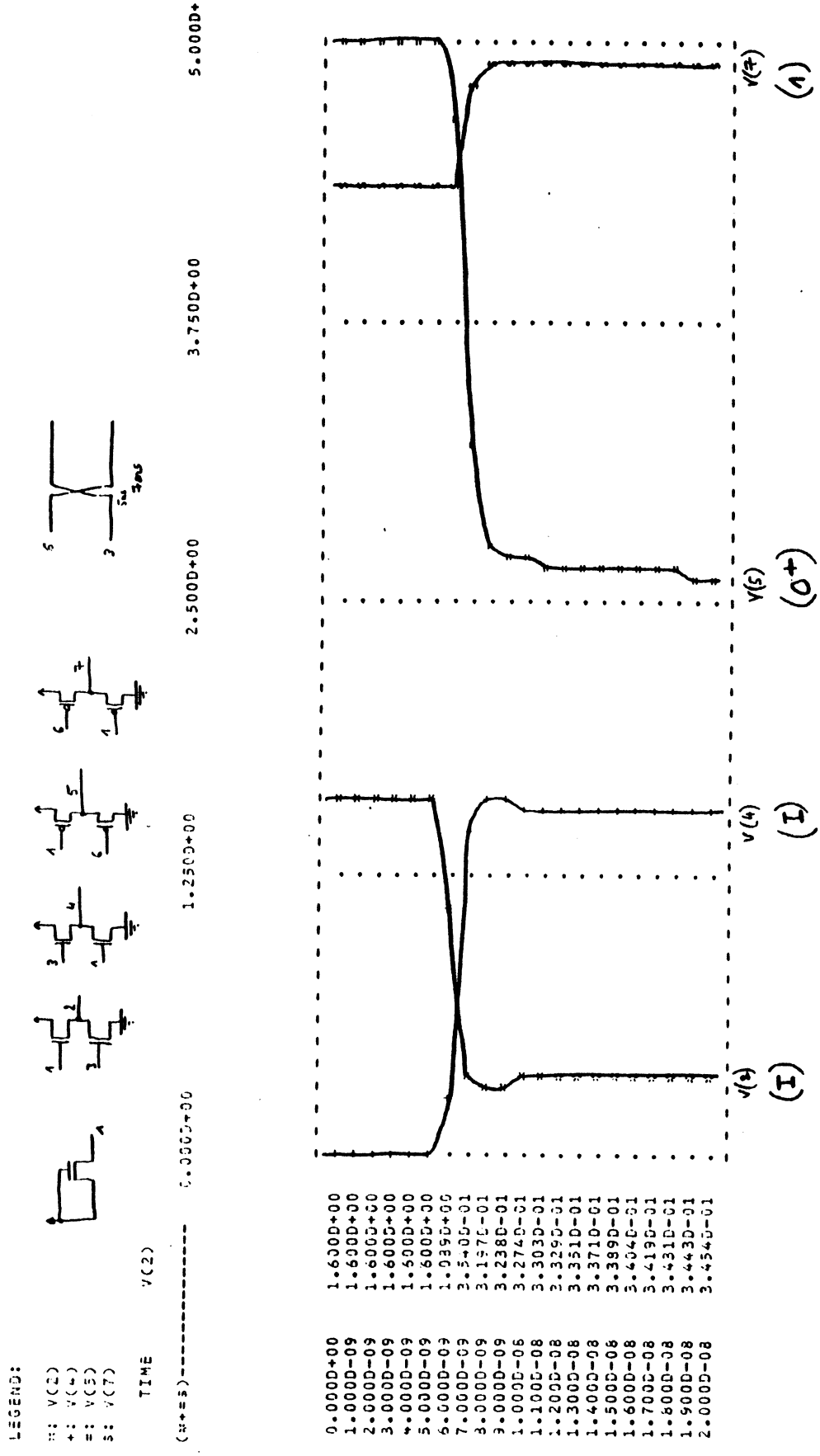


Annexe 1

Augmentation de la résistance d'un transistor de taille Standard dont la grille a la valeur 0^+

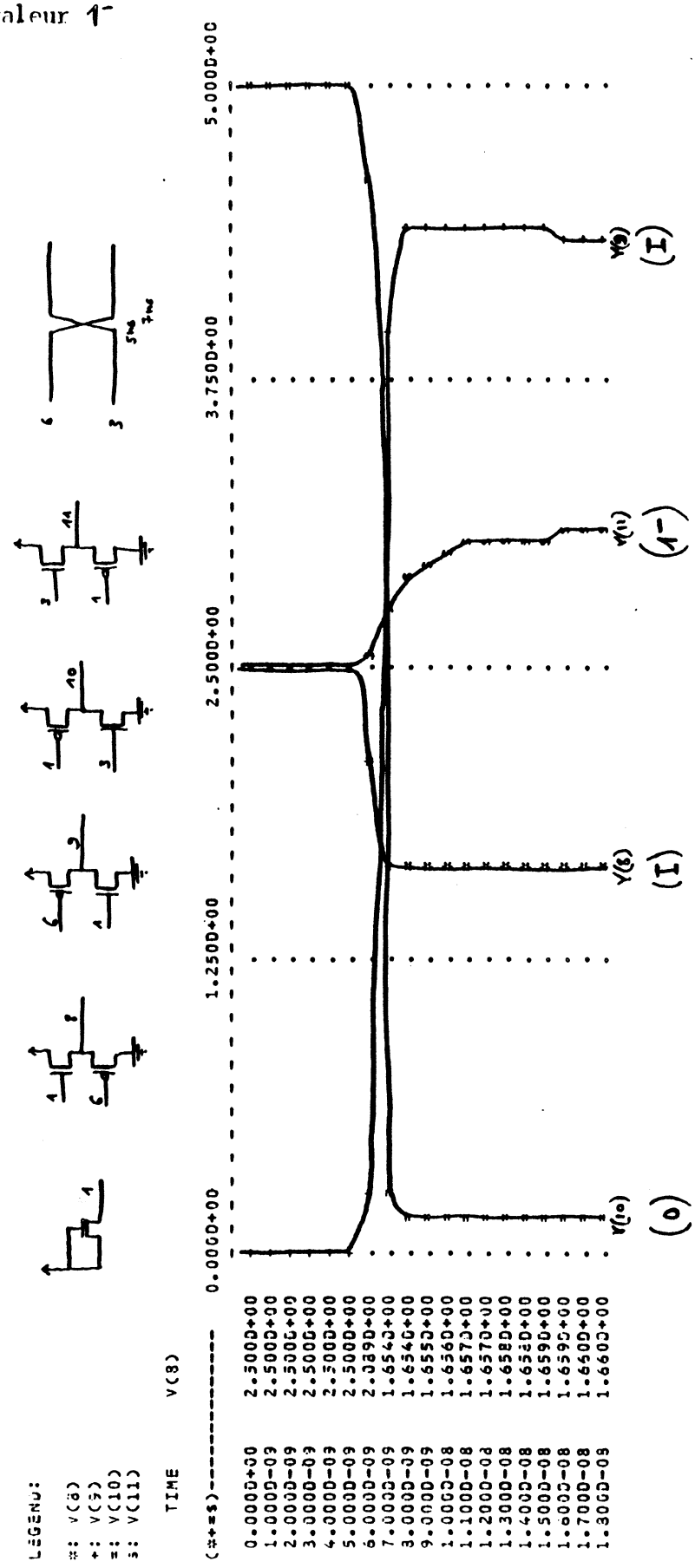


Augmentation de la résistance d'un transistor de taille Standard dont la grille a la valeur 1-



Annexe 1

Augmentation de la résistance d'un transistor de taille Standard dont la grille a la valeur 1



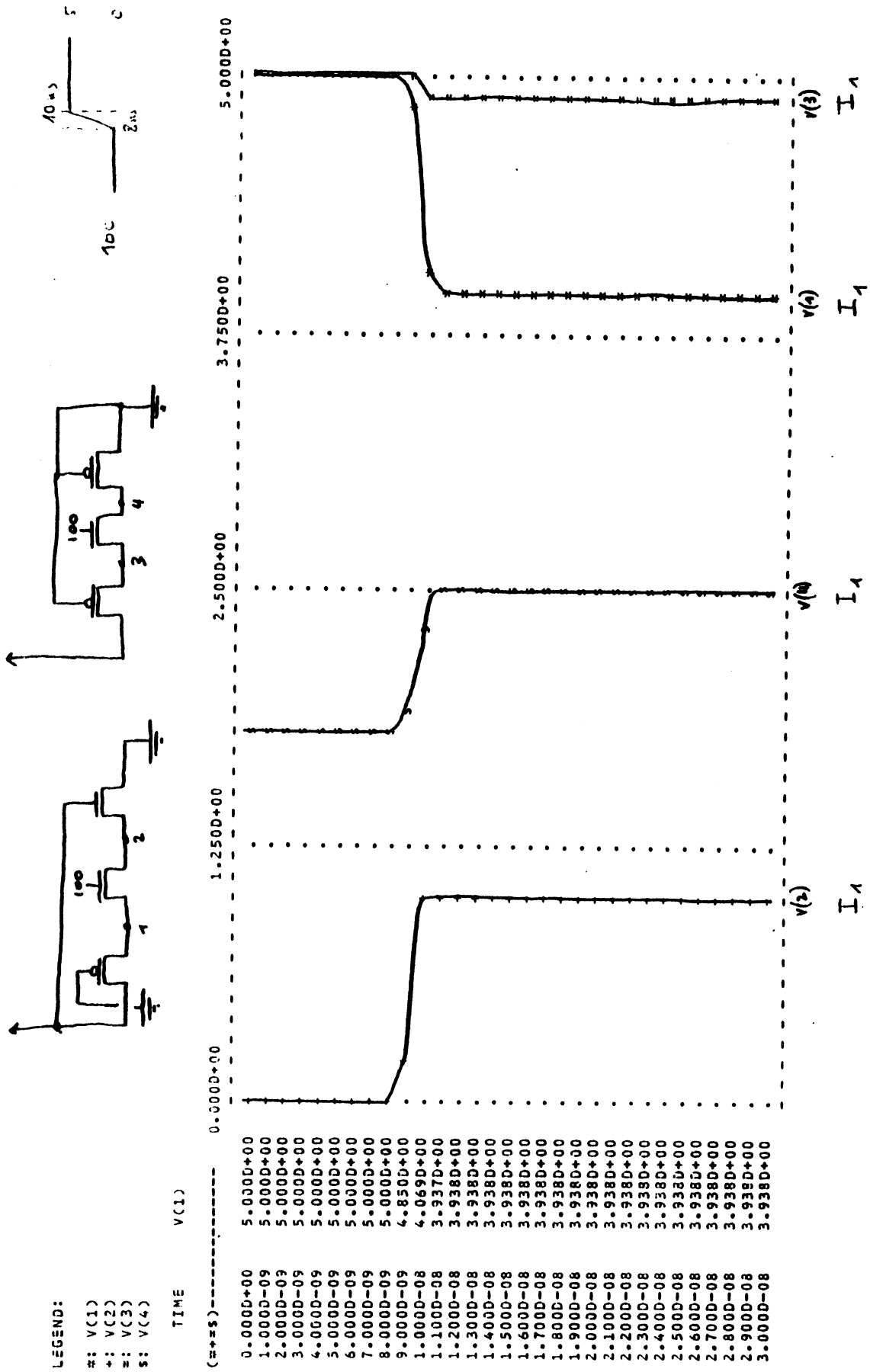
Annexe 1

ANNEXE 2

Simulations SPICE de circuits

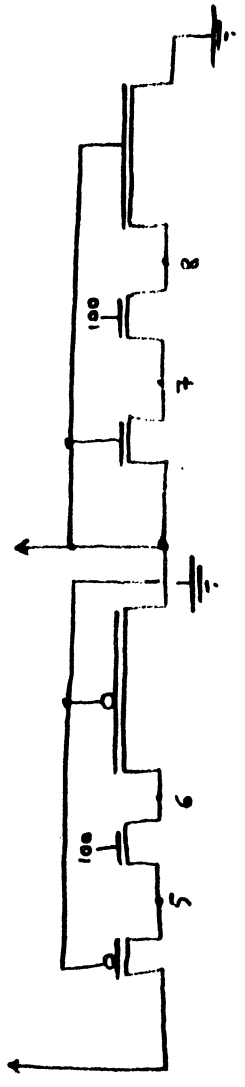
définissant l'opération & (transistor de type N et de taille Standard)

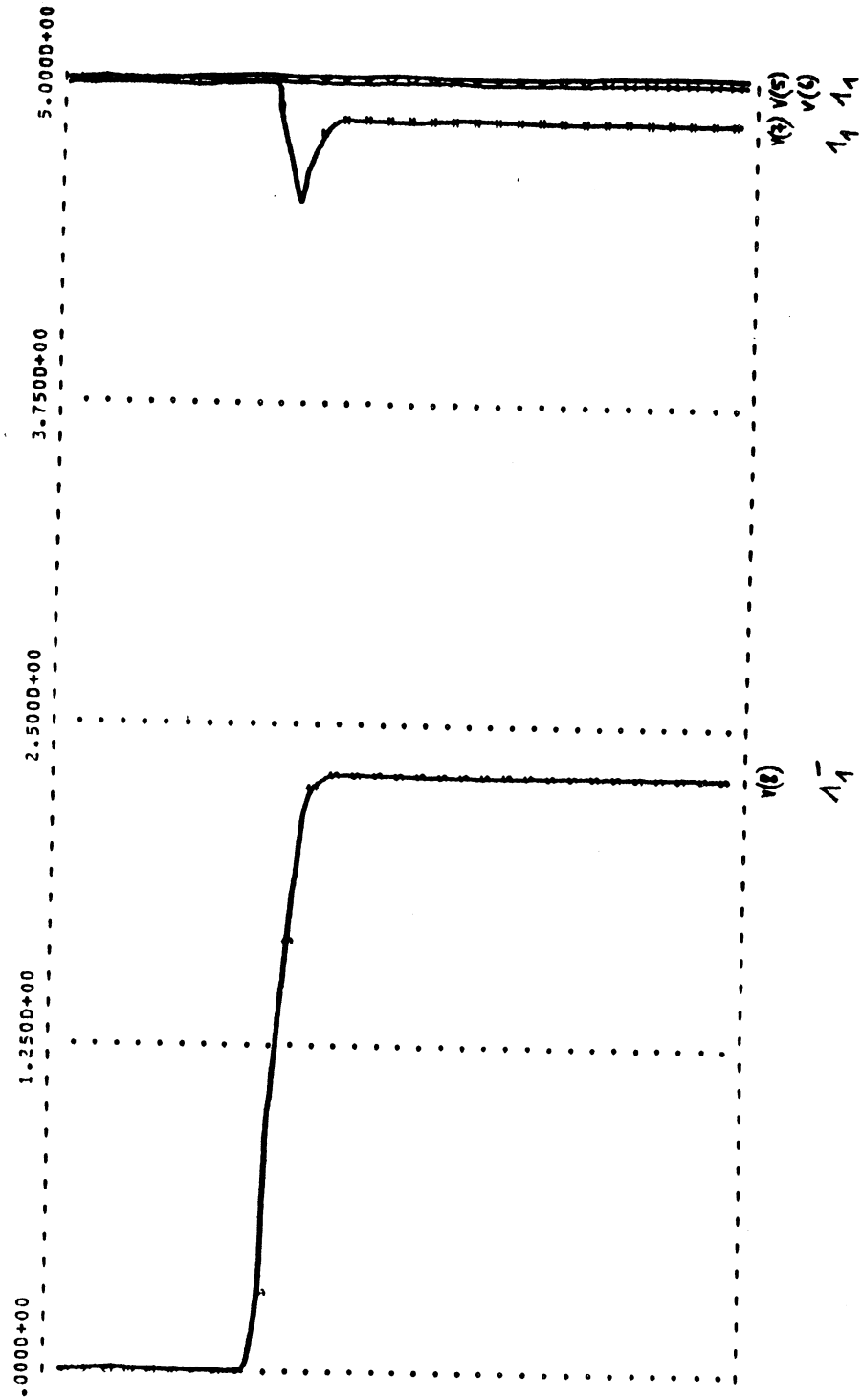
&(1_1 , 0_1 , N, S, 1) et &(1_1 , 0_1^+ , N, S, 1)



Annexe 2

&(1_1 , 1_2 , N, S, 1) et &(1_1 , 0_2 , N, S, 1)

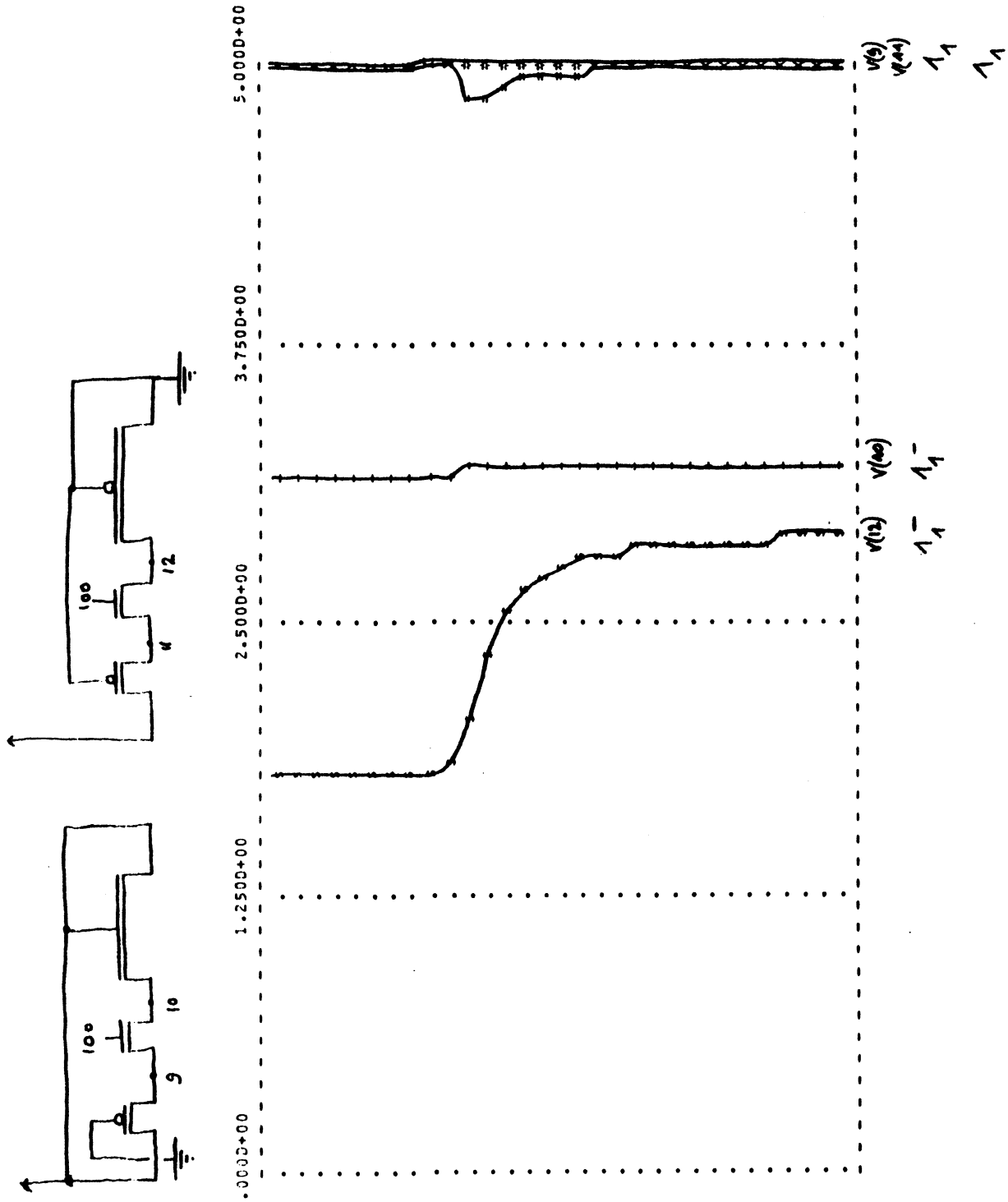




LEGEND:
*: V(5)
+: V(6)
=: V(7)
$: V(8)

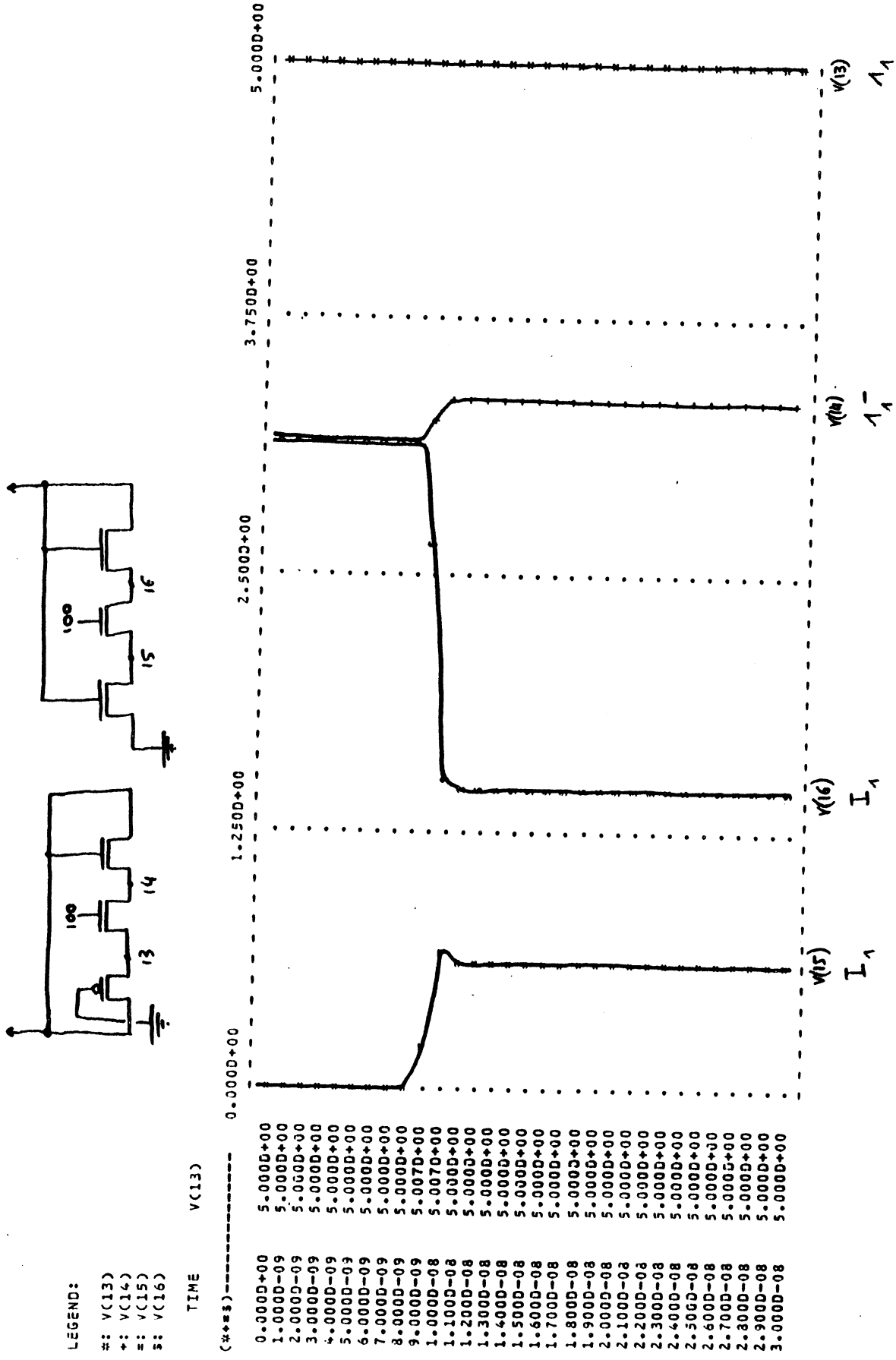
TIME	V(5)
0.0000D+00	5.0000D+00
1.0000D-09	5.0000D+00
2.0000D-09	5.0000D+00
3.0000D-09	5.0000D+00
4.0000D-09	5.0000D+00
5.0000D-09	5.0000D+00
6.0000D-09	5.0000D+00
7.0000D-09	5.0000D+00
8.0000D-09	5.0000D+00
9.0000D-09	5.0070D+00
1.0000D-08	5.0070D+00
1.1000D-08	5.0000D+00
1.2000D-08	5.0000D+00
1.3000D-08	5.0000D+00
1.4000D-08	5.0000D+00
1.5000D-08	5.0000D+00
1.6000D-08	5.0000D+00
1.7000D-08	5.0000D+00
1.8000D-08	5.0000D+00
1.9000D-08	5.0000D+00
2.0000D-08	5.0000D+00
2.1000D-08	5.0000D+00
2.2000D-08	5.0000D+00
2.3000D-08	5.0000D+00
2.4000D-08	5.0000D+00
2.5000D-08	5.0000D+00
2.6000D-08	5.0000D+00
2.7000D-08	5.0000D+00
2.8000D-08	5.0000D+00
2.9000D-08	5.0000D+00
3.0000D-08	5.0000D+00

&(1_1 , 1_2^- , N, S, 1) et &(1_1 , 0_2^+ , N, S, 1)



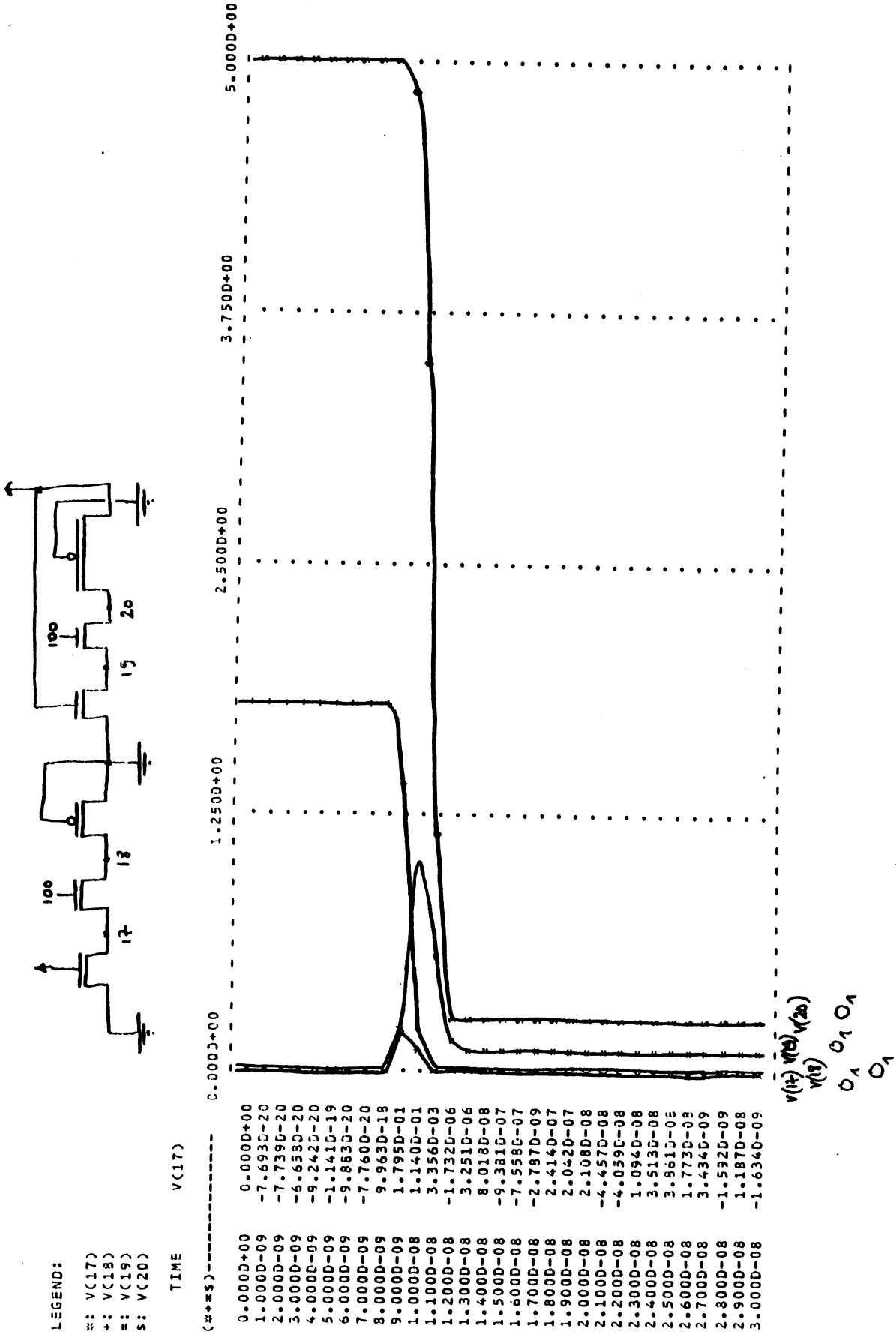
Annexe 2

&(1_1 , 1_1^- , N, S, 1) et &(0_1 , 1_1^- , N, S, 1)



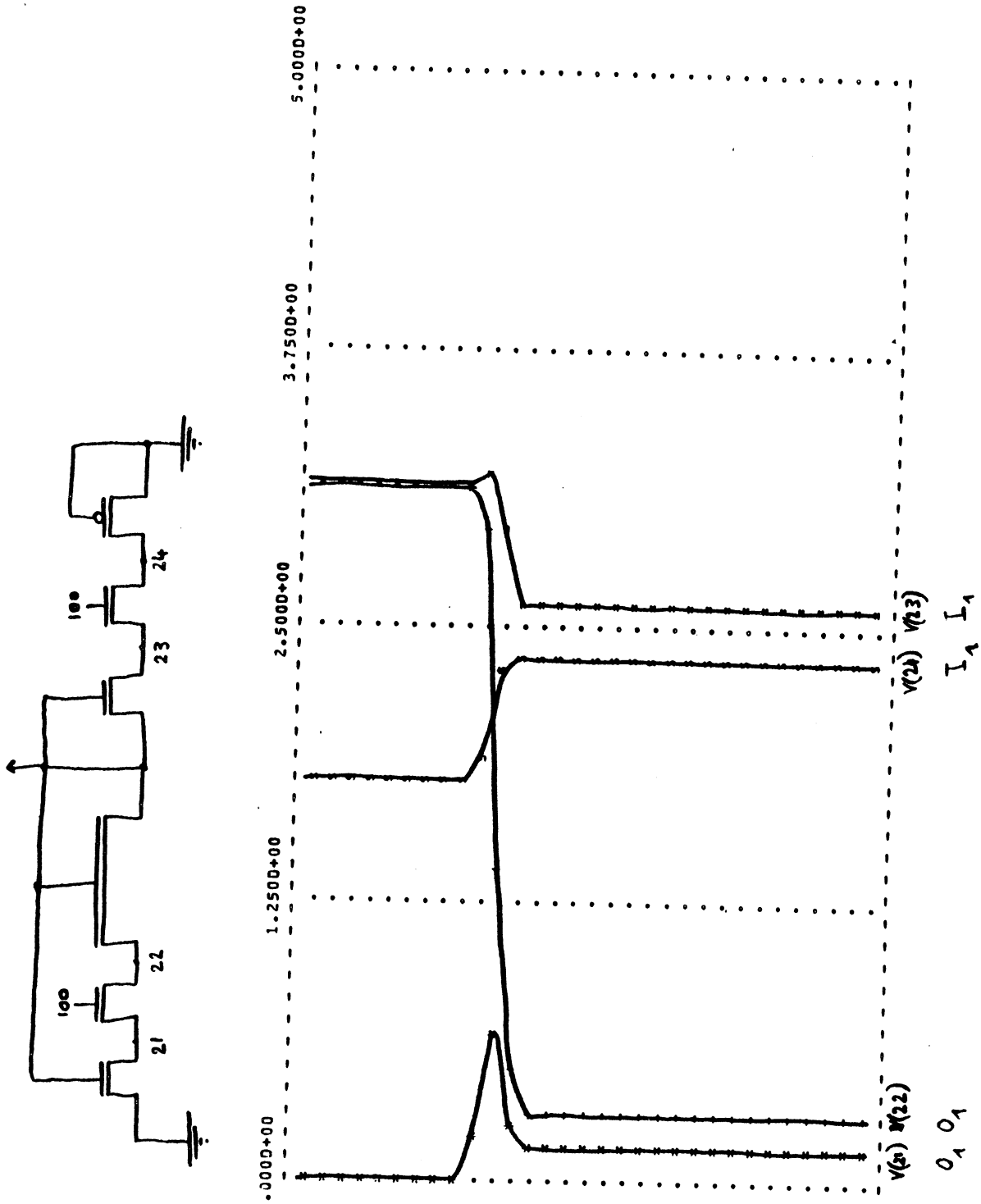
Annexe 2

&(0_1 , 0_1^+ , N, S, 1) et &(0_1 , 1_2 , N, S, 1)



Annexe 2

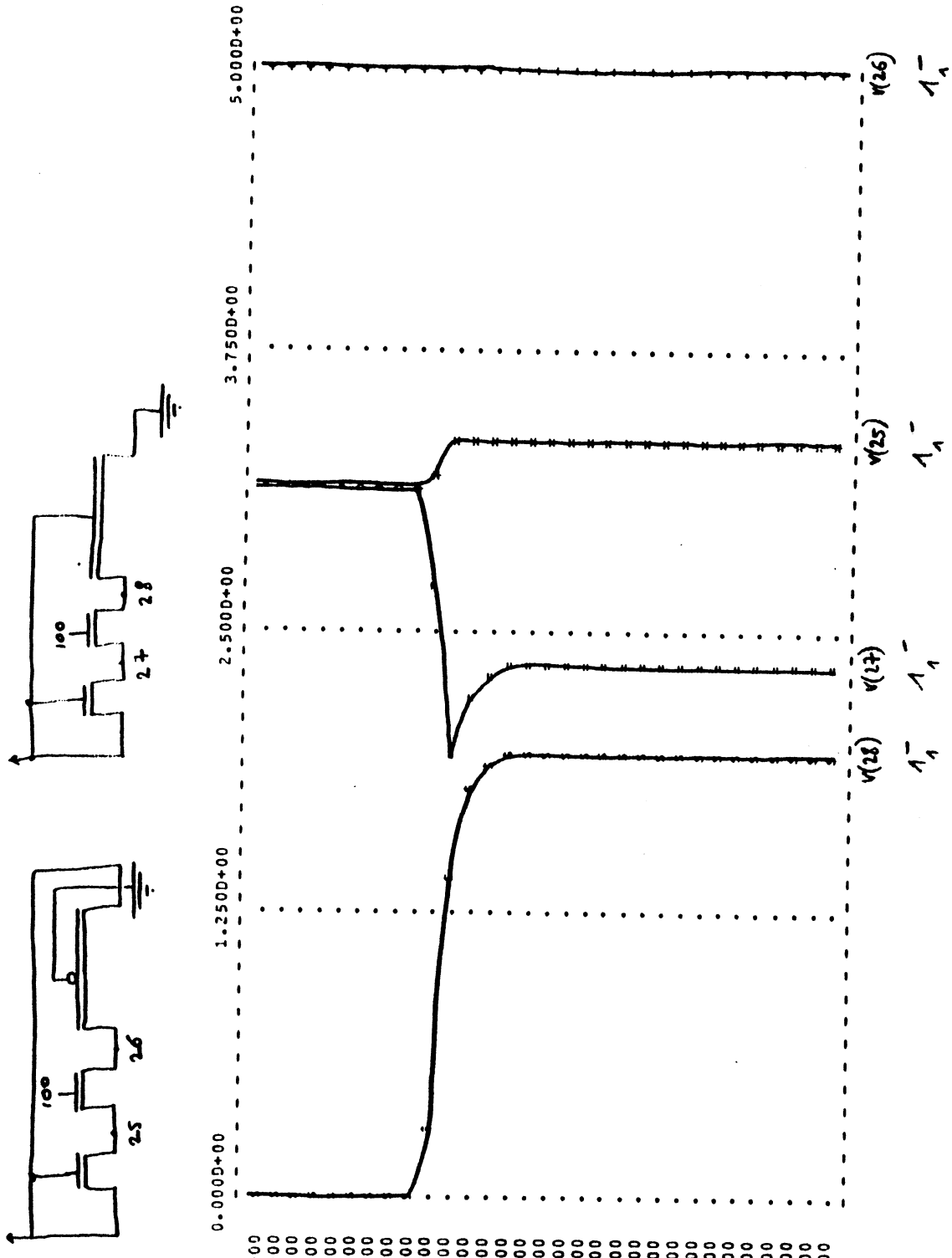
&(0_1 , 1_2^- , N, S, 1) et &(1_1^-, 0_1^+, N, S, 1)



LEGEND:
*: V(21)
+: V(22)
=: V(23)
$: V(24)

TIME	V(21)
0.0000D+00	0.0000D+00
1.0000D-09	-7.6333D-20
2.0000D-09	-7.7390D-20
3.0000D-09	-6.6580D-20
4.0000D-09	-9.2420D-20
5.0000D-09	-1.1410D-19
6.0000D-09	-9.8330D-20
7.0000D-09	-7.7600D-20
8.0000D-09	9.9630D-18
9.0000D-09	2.0410D-01
1.0000D-08	6.4000D-01
1.1000D-08	2.4310D-01
1.2000D-08	1.5760D-01
1.3000D-08	1.4360D-01
1.4000D-08	1.4180D-01
1.5000D-08	1.4150D-01
1.6000D-08	1.4150D-01
1.7000D-08	1.4150D-01
1.8000D-08	1.4150D-01
1.9000D-08	1.4150D-01
2.0000D-08	1.4150D-01
2.1000D-08	1.4150D-01
2.2000D-08	1.4150D-01
2.3000D-08	1.4150D-01
2.4000D-08	1.4150D-01
2.5000D-08	1.4150D-01
2.6000D-08	1.4150D-01
2.7000D-08	1.4150D-01
2.8000D-08	1.4150D-01
2.9000D-08	1.4150D-01
3.0000D-08	1.4150D-01

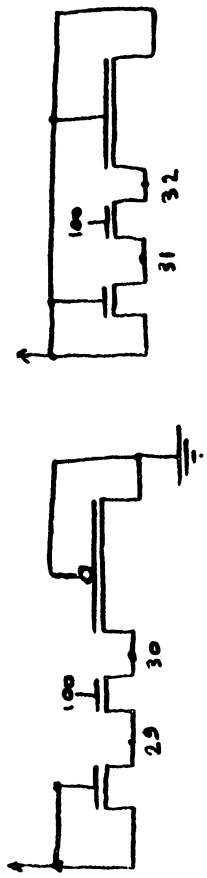
&(1_1^- , 1_2 , N, S, 1) et &(1_1^- , O_2 , N, S, 1)

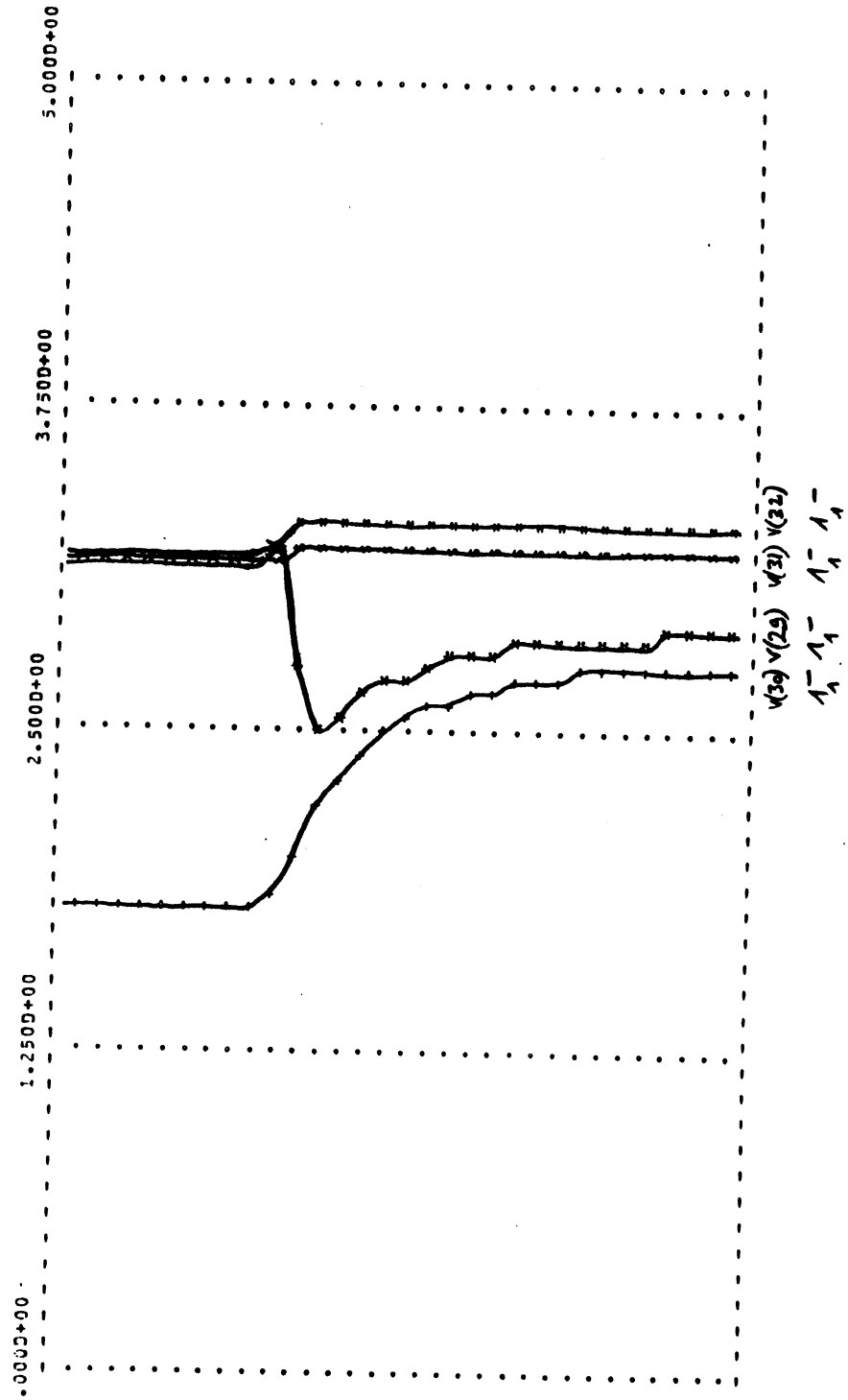


LEGEND:
*: V(25)
+: V(26)
=: V(27)
$: V(28)

TIME	V(25)
0.000D+00	3.132D+00
1.000D-09	3.131D+00
2.000D-09	3.131D+00
3.000D-09	3.131D+00
4.000D-09	3.131D+00
5.000D-09	3.131D+00
6.000D-09	3.131D+00
7.000D-09	3.131D+00
8.000D-09	3.131D+00
9.000D-09	3.224D+00
1.000D-08	3.344D+00
1.100D-08	3.344D+00
1.200D-08	3.344D+00
1.300D-08	3.344D+00
1.400D-08	3.344D+00
1.500D-08	3.344D+00
1.600D-08	3.344D+00
1.700D-08	3.344D+00
1.800D-08	3.344D+00
1.900D-08	3.344D+00
2.000D-08	3.344D+00
2.100D-08	3.344D+00
2.200D-08	3.344D+00
2.300D-08	3.344D+00
2.400D-08	3.344D+00
2.500D-08	3.344D+00
2.600D-08	3.344D+00
2.700D-08	3.344D+00
2.800D-08	3.344D+00
2.900D-08	3.344D+00
3.000D-08	3.344D+00

&(1_1^- , O_2^+ , N, S, 1) et &(1_1^- , 1_2^- , N, S, 1)



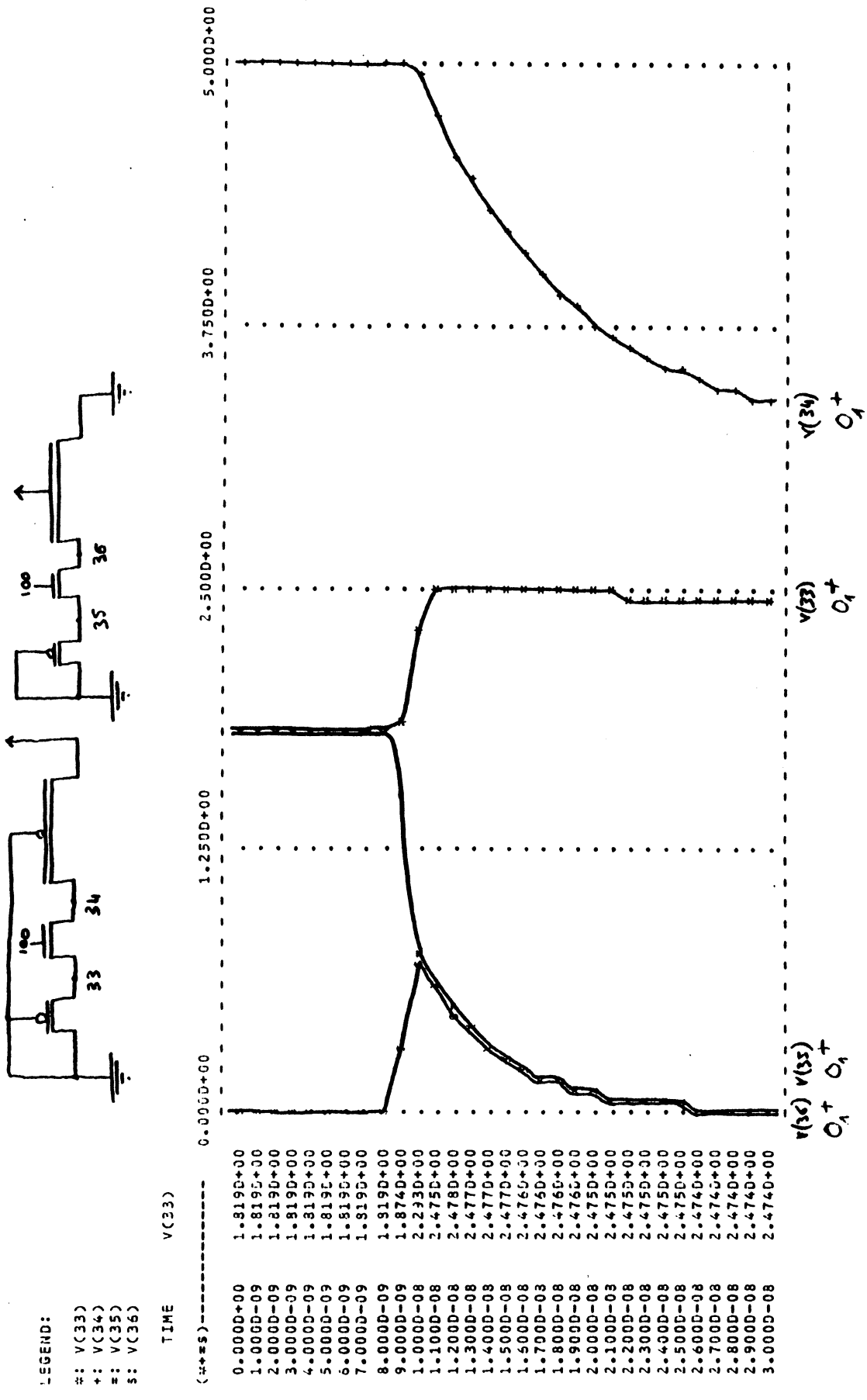


LEGEND:
*: V(29)
+: V(30)
=: V(31)
$: V(32)

TIME	V(29)
0.000D+00	3.132D+00
1.000D-09	3.131D+00
2.000D-09	3.131D+00
3.000D-09	3.131D+00
4.000D-09	3.131D+00
5.000D-09	3.131D+00
6.000D-09	3.131D+00
7.000D-09	3.131D+00
8.000D-09	3.131D+00
9.000D-09	3.224D+00
1.000D-08	2.727D+00
1.100D-08	2.486D+00
1.200D-08	2.573D+00
1.300D-08	2.637D+00
1.400D-08	2.685D+00
1.500D-08	2.723D+00
1.600D-08	2.754D+00
1.700D-08	2.779D+00
1.800D-08	2.798D+00
1.900D-08	2.815D+00
2.000D-08	2.828D+00
2.100D-08	2.840D+00
2.200D-08	2.849D+00
2.300D-08	2.857D+00
2.400D-08	2.864D+00
2.500D-08	2.870D+00
2.600D-08	2.875D+00
2.700D-08	2.879D+00
2.800D-08	2.882D+00
2.900D-08	2.885D+00
3.000D-08	2.888D+00

Annexe 2

&(Φ_1^+, 1_2, N, S, 1) et &(Φ_1^+, O_2, N, S, 1)

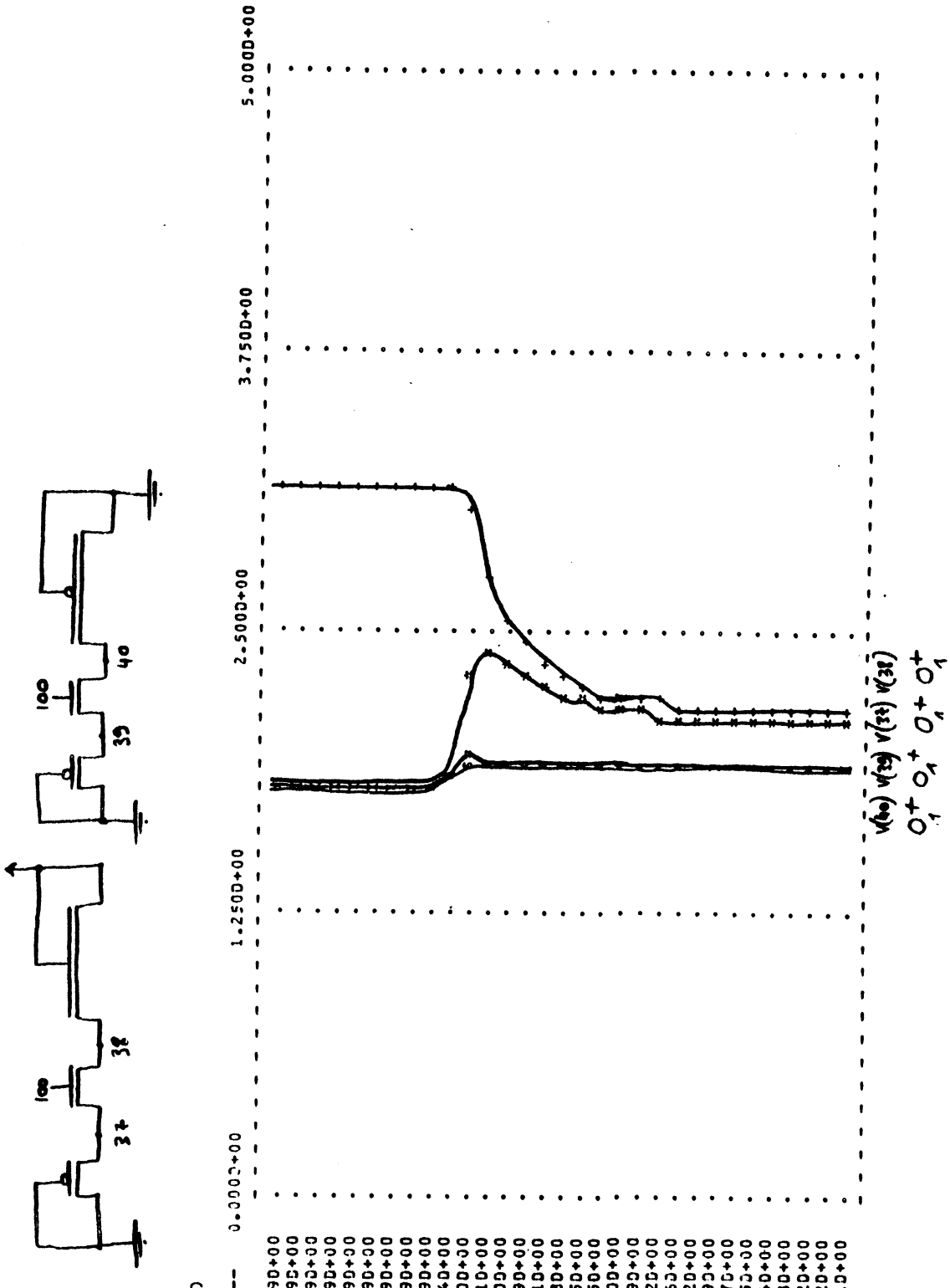


LEGEND:
*: V(33)
+: V(34)
=: V(35)
s: V(36)

TIME	V(33)
0.000D+00	1.819D+00
1.000D-09	1.819D+00
2.000D-09	1.819D+00
3.000D-09	1.819D+00
4.000D-09	1.813D+00
5.000D-09	1.819D+00
6.000D-09	1.819D+00
7.000D-09	1.819D+00
8.000D-09	1.819D+00
9.000D-09	1.874D+00
1.000D-08	2.233D+00
1.100D-08	2.475D+00
1.200D-08	2.478D+00
1.300D-08	2.477D+00
1.400D-08	2.477D+00
1.500D-08	2.477D+00
1.600D-08	2.476D+00
1.700D-08	2.476D+00
1.800D-08	2.476D+00
1.900D-08	2.476D+00
2.000D-08	2.475D+00
2.100D-08	2.475D+00
2.200D-08	2.475D+00
2.300D-08	2.475D+00
2.400D-08	2.475D+00
2.500D-08	2.475D+00
2.600D-08	2.474D+00
2.700D-08	2.474D+00
2.800D-08	2.474D+00
2.900D-08	2.474D+00
3.000D-08	2.474D+00

Annexe 2

&(0_1^+, 1_2^-, N, S, 1) et &(0_1^+, 0_2^+, N, S, 1)

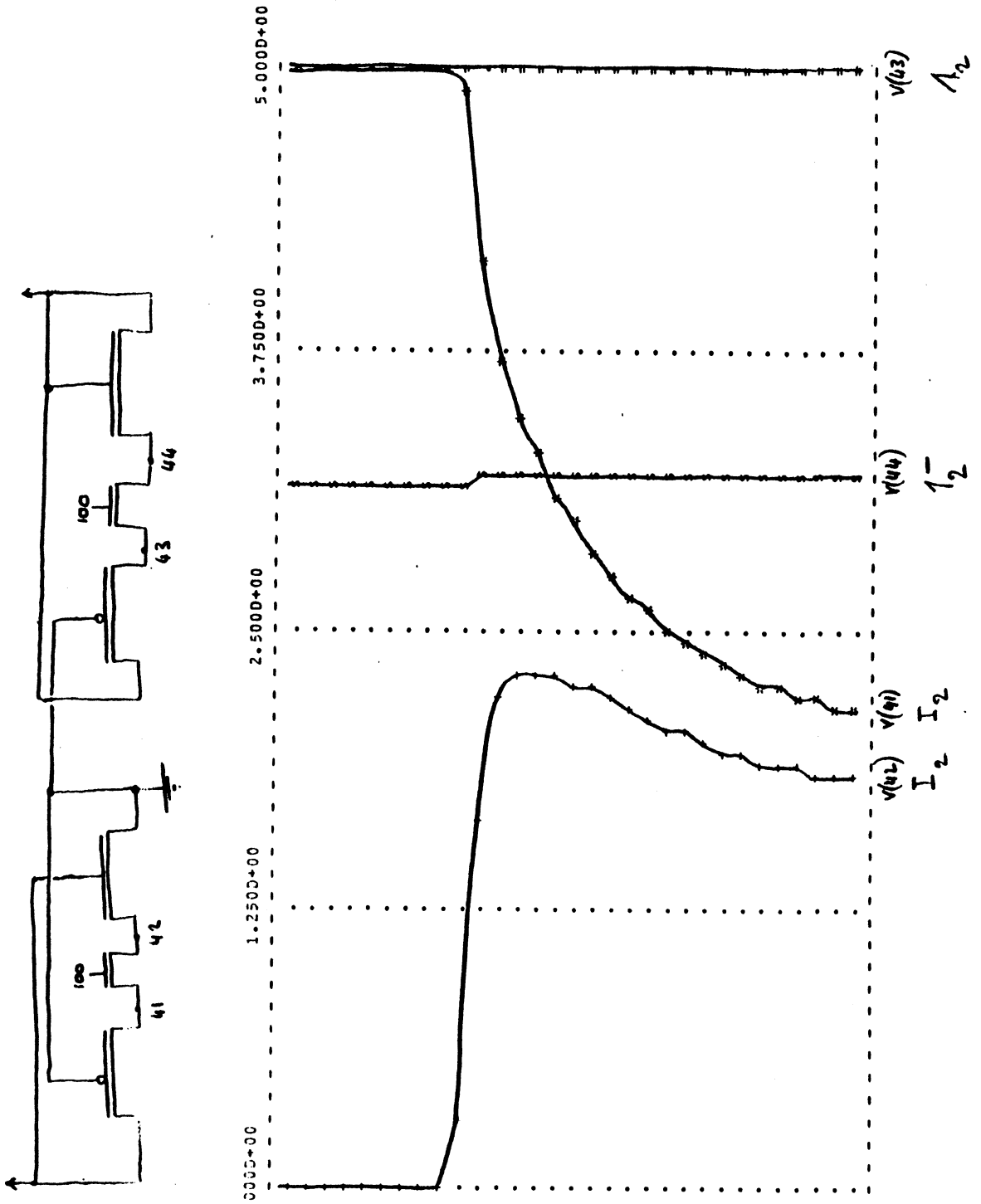


LEGEND:
#: V(37)
+: V(38)
=: V(39)
S: V(40)

TIME	V(37)
0.000D+00	1.819D+00
1.000D-09	1.819D+00
2.000D-09	1.819D+00
3.000D-09	1.819D+00
4.000D-09	1.819D+00
5.000D-09	1.819D+00
6.000D-09	1.819D+00
7.000D-09	1.819D+00
8.000D-09	1.819D+00
9.000D-09	1.974D+00
1.000D-08	2.280D+00
1.100D-08	2.411D+00
1.200D-08	2.350D+00
1.300D-08	2.229D+00
1.400D-08	2.241D+00
1.500D-08	2.203D+00
1.600D-08	2.175D+00
1.700D-08	2.155D+00
1.800D-08	2.140D+00
1.900D-08	2.123D+00
2.000D-08	2.122D+00
2.100D-08	2.116D+00
2.200D-08	2.112D+00
2.300D-08	2.109D+00
2.400D-08	2.107D+00
2.500D-08	2.105D+00
2.600D-08	2.104D+00
2.700D-08	2.103D+00
2.800D-08	2.102D+00
2.900D-08	2.102D+00
3.000D-08	2.101D+00

Annexe 2

&(1_2 , 0_2 , N, S, 1) et &(1_2 , 1_2^- , N, S, 1)

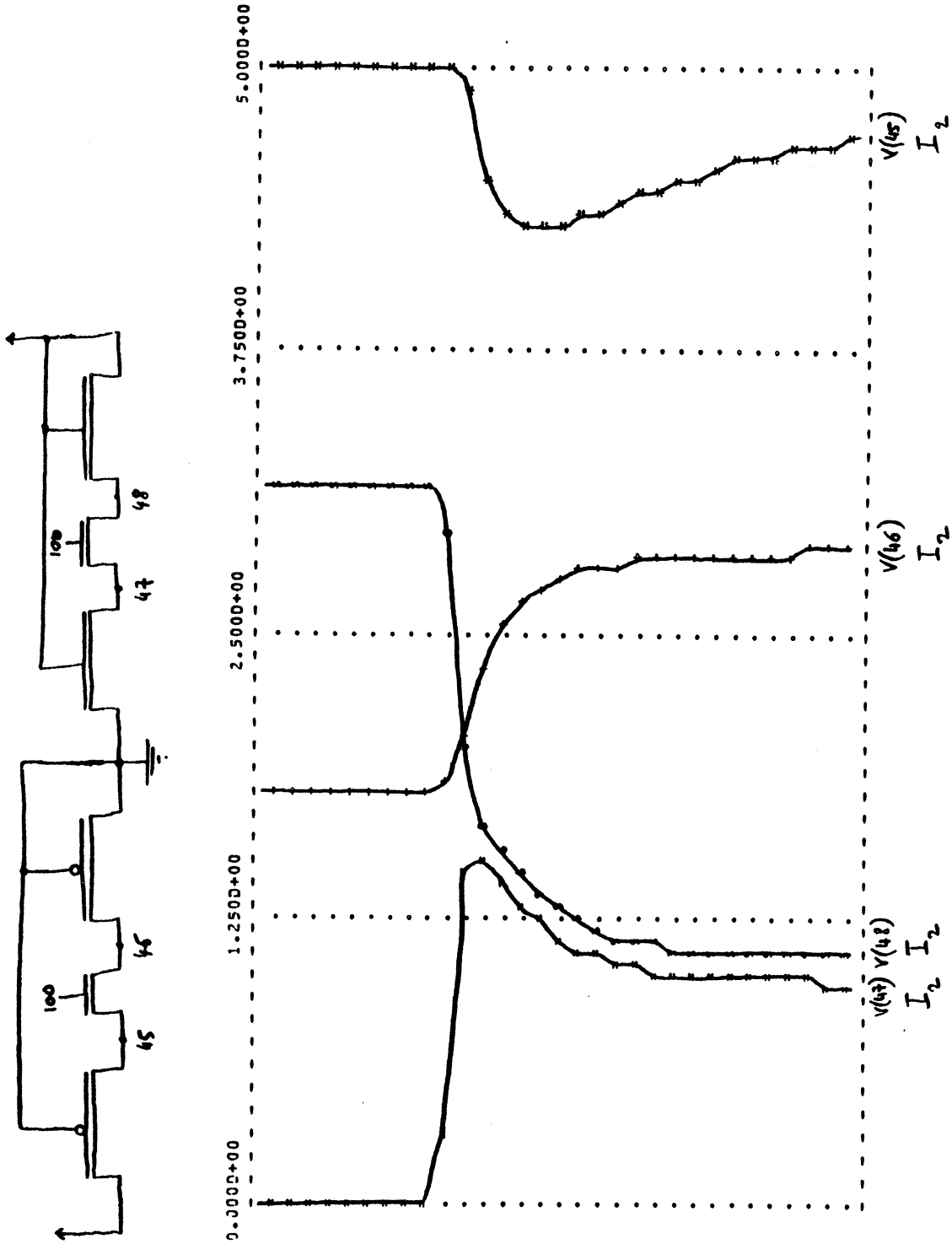


LEGEND:
=: V(41)
+: V(42)
=: V(43)
s: V(44)

TIME	V(41)
0.000D+00	5.000D+00
1.000D-09	5.000D+00
2.000D-09	5.000D+00
3.000D-09	5.000D+00
4.000D-09	5.000D+00
5.000D-09	5.000D+00
6.000D-09	5.000D+00
7.000D-09	5.000D+00
8.000D-09	5.000D+00
9.000D-09	4.897D+00
1.000D-08	4.162D+00
1.100D-08	3.689D+00
1.200D-08	3.468D+00
1.300D-08	3.266D+00
1.400D-08	3.124D+00
1.500D-08	2.982D+00
1.600D-08	2.859D+00
1.700D-08	2.754D+00
1.800D-08	2.661D+00
1.900D-08	2.530D+00
2.000D-08	2.509D+00
2.100D-08	2.447D+00
2.200D-08	2.393D+00
2.300D-08	2.345D+00
2.400D-08	2.303D+00
2.500D-08	2.266D+00
2.600D-08	2.234D+00
2.700D-08	2.206D+00
2.800D-08	2.181D+00
2.900D-08	2.160D+00
3.000D-08	2.141D+00

Annexe 2

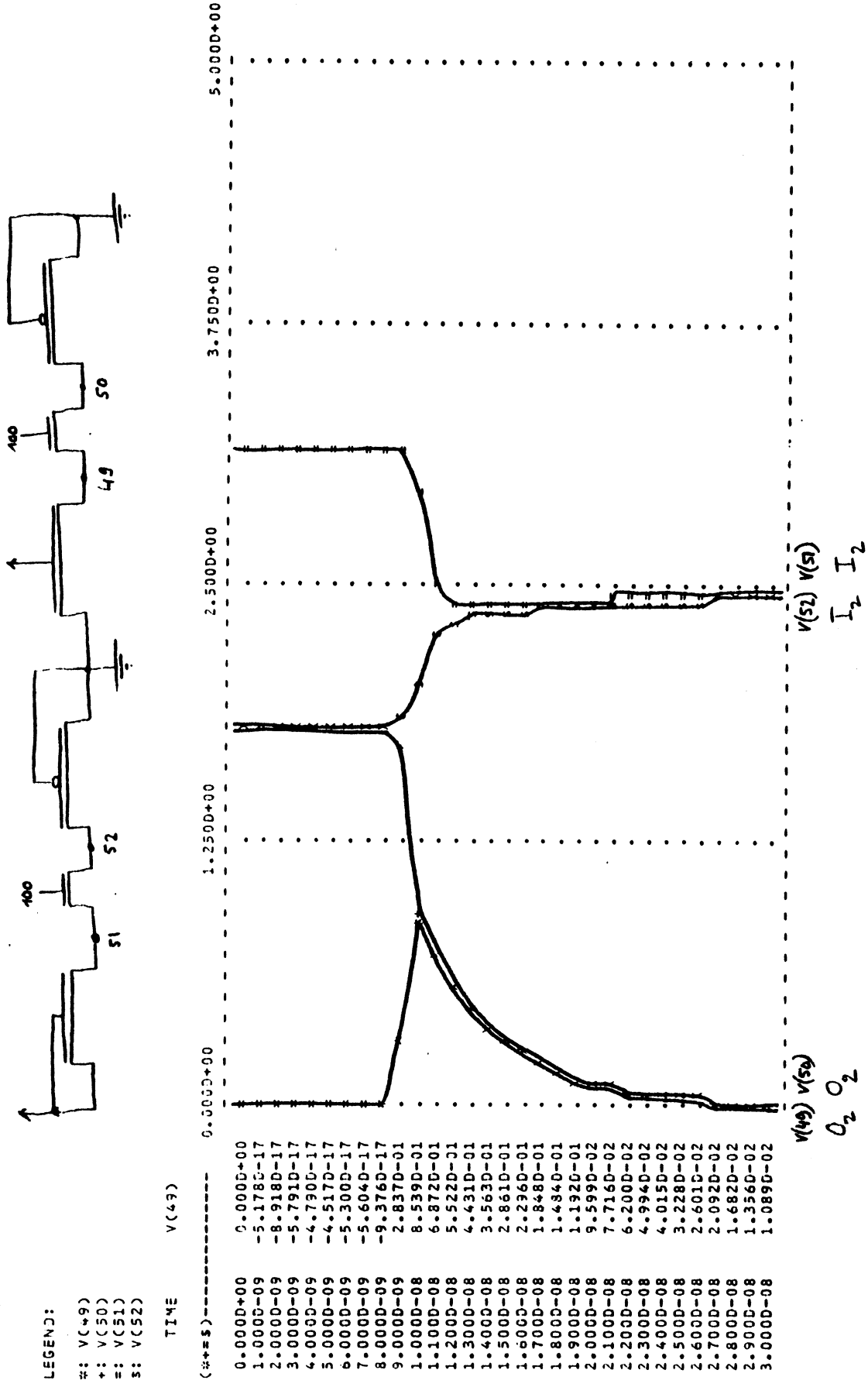
&(1_2 , 0_2^+ , N, S, 1) et &(0_2 , 1_2^- , N, S, 1)



LEGEND:
#: V(45)
+: V(46)
=: V(47)
$: V(48)

TIME	V(45)
0.000D+00	5.000D+00
1.000D-09	5.000D+00
2.000D-09	5.000D+00
3.000D-09	5.000D+00
4.000D-09	5.000D+00
5.000D-09	5.000D+00
6.000D-09	5.000D+00
7.000D-09	5.000D+00
8.000D-09	5.000D+00
9.000D-09	5.015D+00
1.000D-08	4.875D+00
1.100D-08	4.514D+00
1.200D-08	4.359D+00
1.300D-08	4.300D+00
1.400D-08	4.290D+00
1.500D-08	4.303D+00
1.600D-08	4.329D+00
1.700D-08	4.361D+00
1.800D-08	4.395D+00
1.900D-08	4.430D+00
2.000D-08	4.463D+00
2.100D-08	4.495D+00
2.200D-08	4.525D+00
2.300D-08	4.552D+00
2.400D-08	4.577D+00
2.500D-08	4.599D+00
2.600D-08	4.620D+00
2.700D-08	4.638D+00
2.800D-08	4.655D+00
2.900D-08	4.669D+00
3.000D-08	4.682D+00

&(0_2 , 0_2^+ , N, S, 1) et &(1_2^- , 0_2^+ , N, S, 1)

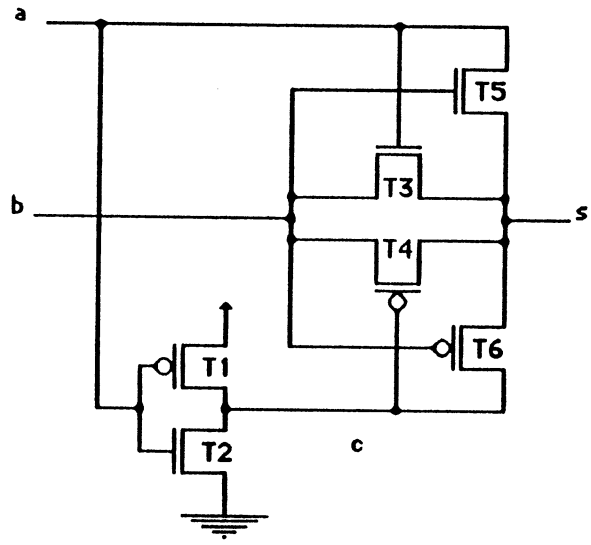


Annexe 2

ANNEXE 3

Exemples du déroulement de l'algorithme sur des circuits contenant des boucles et des éléments bi-directionnels

EXEMPLES DE DEROULEMENT DE L'ALGORITHME



porte de conjonction CMOS : s = conjonction(a, b).

- description du circuit

T1 c a Vdd P Standard
T2 c a Gnd N S
T3 s a b N S
T4 s c b P S
T5 s b a N S
T6 s b c P S

- listes de voisinage et d'influence

voisinage(a) = {(a, b, s, T5)}
influence(a) = {(b, a, s, T3), (c, a, Gnd, T2), (c, a, Vdd, T1)}

voisinage(b) = {(b, a, s, T3), (b, c, s, T4)}
influence(b) = {(a, b, s, T5), (c, b, s, T6)}

voisinage(c) = {(c, b, s, T6), (c, a, Gnd, T2), (c, a, Vdd, T1)}
influence(c) = {(b, c, s, T4)}

voisinage(s) = {(s, b, a, T5), (s, a, b, T3), (s, c, b, T4), (s, b, c, T6)}
influence(s) = {}

- initialisation

Tous les nœuds à X, tous les transistors à X

- déroulement du calcul des états

t0 : état(a) := 0_1 ; état(b) := 0_1 ;
empiler(propager_état_infl(a), t0+1) ;
empiler(propager_état_infl(b), t0+1) ;
propager_état_vois(a) :
voisinage(a) = {(a, b, s, T5)}
état(T5) = X donc rien ;
propager_état_vois(b) :
voisinage(b) = {(b, a, s, T3), (b, c, s, T4)}
état(T3) = état(T4) = X donc rien ;

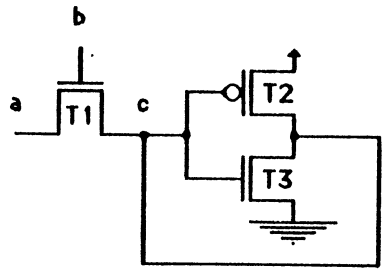
t0+1 : **propager_état_infl**(a) :
influence(a) = {(b, a, s, T3), (c, a, Gnd, T2), (c, a, Vdd, T1)}
on traite d'abord les transistors qui deviennent bloqués (transistors N) :
type(T3) = N donc état(T3) := bloqué ;
type(T2) = N donc état(T2) := bloqué ;
ensuite on traite les transistors qui deviennent passants (transistors P) :
type(T1) = P donc état(T1) := passant ;
appliquer &(c, Vdd)
VDD(c) = { }, GND(c) = { } donc Vdd s'impose sur c :
état(c) := 1_1 ; VDD(c) := {(c, Vdd, T1)}
empiler(propager_état_infl(c), t0+2) ;
propager_état_vois(c) :
voisinage(c) = {(c, b, s, T6), (c, a, Gnd, T2), (c, a, Vdd, T1)}
état(T6) = X donc rien ;
état(T2) = bloqué donc rien ;
on est en train de traiter T1, donc rien ;
propager_état_infl(b) :
influence(b) = {(a, b, s, T5), (c, b, s, T6)}
on traite d'abord les transistors qui deviennent bloqués (transistors N) :
type(T5) = N donc état(T5) := bloqué ;
ensuite on traite les transistors qui deviennent passants (transistors P) :
type(T6) = P donc état(T6) := passant ;
appliquer &(c, s) :
VDD(c) = {(c, Vdd, T1)} ; GND(c) = { } ;

VDD(s) = GND(s) = { } donc c s'impose sur s :
état(s) := 1_1 ; VDD(s) := {(s, c, Vdd, T6, T1)} ;
empiler(propager_état_infl(s), t0+2) ;
propager_état_vois(s) :
voisinage(s) = {(s, b, a, T5), (s, a, b, T3), (s, c, b, T4),
(s, b, c, T6)}
état(T5) = bloqué donc rien ;
état(T3) = bloqué donc rien ;
état(T4) = X donc rien ;
on ne traite pas T6 ;

t0+2 : **propager_état_infl**(c) :
influence(c) = {(b, c, s, T4)}
type(T4) = P donc état(T4) := bloqué ;
propager_état_infl(s) :
infl(s) = { } donc rien.

- états stationnaires :

état(a) = (0_1, t0) ;
état(b) = (0_1, t0) ;
état(s) = (1_1, t0+1) ;
état(c) = (1_1, t0+1).



oscillateur CMOS.

- description du circuit

T1 a b c N Long
T2 c c Vdd P Standard
T3 c c Gnd N S

- listes de voisinage et d'influence

voisinage(a) = {(a, b, c, T1)}
influence(a) = { }

voisinage(b) = { }
influence(b) = {a, b, c, T1)}

voisinage(c) = {(c, b, a, T1), (c, c, Vdd, T2), (c, c, Gnd, T3)}
influence(c) = {(c, c, Vdd, T2), (c, c, Gnd, T3)}

- initialisation

Tous les nœuds à X, tous les transistors à X

- déroulement du calcul des états

t0 : état(a) := 1_1 ; état(b) := 1_1 ;
empiler(propager_état_infl(a), t0+1) ;
empiler(propager_état_infl(b), t0+1) ;
propager_état_vois(a) :
voisinage(a) = {(a, b, c, T1)}
état(T1) = X donc rien ;
propager_état_vois(b) :
voisinage(b) = { } donc rien ;

t0+1 : **propager_état_infl**(a) :
influence(a) = { } donc rien ;
propager_état_infl(b) :
influence(b) = {a, b, c, T1)}
type(T1) = N donc état(T1) := passant ;
appliquer &(a, c) :
VDD(c) = GND(c) = { } donc a s'impose sur c :
état(c) := 1^{-}_2 (transistor N Long), VDD(c) := {(c, a, T1)}
empiler(propager_état_infl(c), t0+2) ;
propager_état_vois(c) :
voisinage(c) = {(c, b, a, T1), (c, c, Vdd, T2), (c, c, Gnd, T3)}
on ne traite pas T1 ;
état(T2) = X donc rien ;
état(T3) = X donc rien ;

t0+2 : **propager_état_infl**(c) :

influence(c) = {(c, c, Vdd, T2), (c, c, Gnd, T3)}

on traite d'abord les transistors qui deviennent bloqués (transistors P) :

type(T2) = P donc état(T2) := bloqué ;

ensuite on traite les transistors qui deviennent passants (transistors N) :

type(T3) = N donc état(T3) := passant ;

appliquer &(c, Gnd) :

VDD(c) = {(c, a, T1)}

état(c) := #(1^-_2, 0_1) = 0_1 ; GND(c) := {(c, Gnd, T3)} ;

empiler(propager_état_infl(c), t0+3) ;

propager_état_vois(c) :

voisinage(c) = {(c, b, a, T1), (c, c, Vdd, T2), (c, c, Gnd, T3)}

état(T1) = passant donc

appliquer &(c, a) :

&(0_1, 1_1, N, L, 1) = &(0_1, 1_1) (pas de changement) ;

GND(a) := {(a, c, Gnd, T1, T3)} ;

état(T2) = bloqué donc rien ;

on ne traite pas T3 ;

t0+3 : **propager_état_infl**(c) :

influence(c) = {(c, c, Vdd, T2), (c, c, Gnd, T3)}

on traite d'abord les transistors qui deviennent bloqués (transistors N) :

type(T3) = N donc état(T3) := bloqué ;

GND(c) = {(c, Gnd, T3)} donc GND(c) := { } (on y supprime tous les éléments contenant Gnd et T3) ;

l'état de c n'est donc plus 0_1 mais 0_3 ;

on propage cette suppression aux voisins de c :

voisinage(c) = {(c, b, a, T1), (c, c, Vdd, T2), (c, c, Gnd, T3)}

état(T1) = passant donc

GND(a) := { } ; état(a) ne change pas, car c'est une entrée ;

appliquer &(c, a) :

&(0_3, 1_1, N, L, 1) = (1^-_2, 1_1) ;

empiler(propager_état_infl(c), t0+4) ;

propager_état_vois(c) :

voisinage(c) = {(c, b, a, T1), (c, c, Vdd, T2), (c, c, Gnd, T3)}

on ne traite pas T1 ;

état(T2) = état(T3) = bloqué donc rien ;

état(T2) = bloqué donc rien ;

on ne traite pas T3 ;

ensuite on traite les transistors qui deviennent passants (transistors P) :
type(T2) = P donc état(T2) := passant ;
appliquer &(c, Vdd) :
VDD(c) = {(c, a, T1)} ; GND(c) = { } ;
état(c) := #(1^-_2, 1_1) = 1_1 ; VDD(c) := {(c, a, T1), (c, Vdd, T2)} ;
empiler(propager_état_infl(c), t0+4) ;
propager_état_vois(c) :
voisinage(c) = {(c, b, a, T1), (c, c, Vdd, T2), (c, c, Gnd, T3)}
état(T1) = passant donc
appliquer &(c, a) :
&(1_1, 1_1, N, L, 1) = &(1_1, 1_1) ; pas de changement ;
VDD(a) := {(a, c, Vdd, T1, T2)} ;
on ne traite pas T2 ;
état(T3) = bloqué donc rien ;

t0+4 : **propager_état_infl**(c) :
...

- états stationnaires

état(c) = (1^-_2, t0+1), (0_1, t0+2), (1_1, t0+3) ...

RESUME

Le but de cette thèse est de spécifier des outils de simulation, de simulation de pannes et de génération de vecteurs de test, utilisables sur des circuits V. L. S. I. décrits sous forme de réseaux de transistors. Un transistor MOS (interrupteur) est par nature bi-directionnel, et il est impossible de prévoir le sens des courants qui le traversent sans appliquer aux réseaux de transistors un traitement préliminaire, qui reconnaît les boucles et les transistors de transmission, et définit le sens de propagation des signaux. Ce traitement préliminaire, bien qu'il permette à la vérification proprement dite d'être plus rapide, ne respecte pas le concept de réseau bi-directionnel. On a donc choisi de traiter les réseaux de transistors de façon directe, en créant des outils qui pallient l'ignorance des courants. En outre, l'algèbre des états représentant les signaux qui circulent dans les réseaux, doit être choisie de façon à pouvoir modéliser tous les comportements spécifiques de ce niveau de description. Cette algèbre est multivaluée, et comporte des couples (valeur, force) décrivant la tension et l'intensité des signaux.

MOTS-CLES *: simulation, simulation de pannes, génération de vecteurs de test, réseaux de transistors, réseaux d'interrupteurs, algèbres multivaluées.*

ABSTRACT

The aim of this thesis is the specification of simulation, fault simulation and test pattern generation tools, for V. L. S. I. transistor-level circuits. A MOS (switch) is naturally bidirectional, and the direction of currents flowing through a switch network cannot be determined without any preliminary treatment, which recognizes the loops and the transmission switches, and defines the signal propagation directions. Such a treatment, although it allows the actual verification to be faster, does not respect the bidirectional-network concept. Therefore, the verification of switch networks is performed directly, with the help of special tools which palliate the current ignorance. Moreover, the states which represent the signal characteristics, must be chosen in order to model all the specific behaviours of the switch level. The state algebra is multiple-valued, and contains (value, strength) pairs, where the value represents the voltage, and the strength represents the current of the signals.

KEY_WORDS *: simulation, fault simulation, test pattern generation, transistor networks, switch networks, multiple-valued algebras.*