Conception temporellement sûre de circuits intégrés complexes

Paul Amblard

THESE

présentée à

l'Institut National Polytechnique de Grenoble

pour obtenir le grade de

DOCTEUR DE 3ème CYCLE
«Informatique»

par

Paul AMBLARD

CONCEPTION TEMPORELLEMENT SURE DE CIRCUITS INTEGRES COMPLEXES.

Thèse soutenue le 15 juin 1984 devant la commission d'examen.

Madame	G. SAUCIER	Président
Messieurs	P. CASPI	Examinateurs
	F. DEVOS	
	G. MAZARÉ	
	G. MICHEL	

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Année universitaire 1982-1983

Président de l'Université : D. BLOCH

Vice-Président : René CARRE
Hervé CHERADAME
Marcel IVANES

PROFESSEURS DES UNIVERSITES :

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

.../...

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

PROFESSEURS ASSOCIES

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

PROFESSEURS UNIVERSITE DES SCIENCES SOCIALES (Grenoble II)

BOLLIET Louis
Chatelin Françoise

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

RIEU Jean
SOUSTELLE Michel

CHERCHEURS DU C.N.R.S.

FRUCHART Robert	Directeur de Recherche
VACHAUD Georges	Directeur de Recherche

.../...

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

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

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

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

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

.../...

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

.../...

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

LABORATOIRES EXTERIEURS

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

Dans la religion hindouiste, l'un des noms du seigneur VISHNOU est "HARI" et "dwar" signifie "porte". Hardwar, c'est donc "la porte du Seigneur".

Rapporté par Robert ARNAUD
dans " L'arbre à deux branches
la grande aventure du C.N.R.S."
Ed. Presses de la cité, C.N.R.S.
France Inter, Paris, 1979.

Je tiens à remercier Gabrièle SAUCIER qui m'a accueilli dans son équipe de recherche et qui m'a fait une confiance exigeante.

Je remercie Guy MAZARE qui accepte de juger ce travail malgré de nombreuses autres activités.

Francis DEVOS a manifesté beaucoup de curiosité à l'égard de ce travail; je tiens à l'en remercier ainsi que de la chaleur de son accueil lors de ma visite dans son équipe.

Je profite de l'occasion qui m'est donnée ici de remercier les concepteurs du Centre Norbert Segard avec qui j'ai eu l'occasion de travailler, pour le réalisme de leurs remarques. Gérard MICHEL, membre de ce jury voudra bien se considérer comme leur représentant.

Paul CASPI m'a donné de précieuses indications dans le travail qui a conduit à cette thèse. C'est, à travers lui, à tous mes camarades du laboratoire que j'exprime ici mon amitié.

Je voudrais remercier aussi les personnes de l'IMAG dont la tache est de permettre à " ceux qui font des thèses " de le faire dans de bonnes conditions.

Dans la première partie, on pose le problème suivant : comment mener de front plusieurs descriptions d'un meme circuit intégré? Différentes façons de décrire un circuit sont d'abord envisagées (1.1 ; 1.2) suivant

* leur niveau (architectural , logique , électrique , implanté)
* leur mode (structurel , fonctionnel) .

L'accent est mis sur les descriptions prenant en compte l'aspect temporel du fonctionnement, spécialement au niveau architectural (1.3)

La question de la sureté d'une description, ou de sa cohérence avec une autre est traitée à part (1.4).Dans la deuxième partie, on rappelle deux modèles :

* Le modèle par événements, développé pour spécifier le comportement de systèmes temporisés en général, est présenté partiellement, (2.1)

* Le modèle classique d'automate d'état fini est rappelé (2.1) ainsi que les difficultés de son utilisation pour la description de systèmes temporisés (2.2). Cela conduit à proposer une définition d'automate d'état fini temporisé combinant les deux approches (2.2.5).

Cette définition permet de caractériser, au moyen d'un calcul algébrique, le fonctionnement de différentes réalisations (3) . On étudie d'abord deux réalisations de base, classiques, (3.1), des variantes de l'une d'entre elles sont ensuite passées en revue. Le calcul conduit à leur équivalence du point de vue du délai de réponse (3.2) . On met enfin en évidence les hypothèses temporelles nécessaires au bon fonctionnement d'une troisième

famille de réalisation (3.3)

Les résultats donnés dans la troisième partie, concernant un automate, permettent d'étudier des machines plus complexes (4.2) réalisées, par exemple, au moyen d'une partie controle et d'une partie opérative. On montre que dans ce contexte, partie controle, partie opérative, on peut caractériser par le calcul la synchronisation entre les deux machines, c'est à dire le nombre d'états d'attente entre l'émission d'une commande par la partie controle et la prise en compte possible des comptes rendus correspondant à cette commande. (4.3)

Quelques exemples simples, illustrant les modes de descriptions utilisés dans les parties précédentes, sont regroupés dans la cinquième partie.

PREMIERE PARTIE

DESCRIPTION D'UN CIRCUIT INTEGRE INFORMATIQUE

1.1 NIVEAUX ET MODES DE DESCRIPTION D'UN CIRCUIT

1.1.1 Niveaux de description
1.1.2 Modes de description
1.1.3 Formalismes de description
1.1.4 langages de description

1.2 NIVEAUX LOGIQUE ,ELECTRIQUE, IMPLANTE

1.2.1 Description en termes logiques
 1.2.1.1 Choix des opérateurs élémentaires
 1.2.1.2 Logique temporisée
 1.2.1.3 Logique multivaluée
1.2.2 Description en termes électriques
1.2.3 Description des plans du circuit
1.2.4 Descriptions mixtes

1.3 NIVEAU ARCHITECTURAL

1.3.1 Description structurelle
1.3.2 Description fonctionnelle en termes de coopération de primitives fonctionnelles
1.3.3 Synchronisation et référence temporelle
1.3.4 Quel temps ?

1.4 SURETE D'UNE DESCRIPTION

1.4.1 Passage automatique d'une description à une autre
 1.4.1.1 Changement de niveau
 1.4.1.2 Changement de mode
 1.4.1.3 Optimiseurs
 1.4.1.4 Chaine intégré de conception automatique
1.4.2 Outils de vérification
 1.4.2.2 Vérification d'invariants
 1.4.2.2 Preuve de conformité entre deux descriptions
 1.4.2.3 Simulation d'une description structurelle

DEUXIEME PARTIE

MODELES TEMPORISES POUR UNE DESCRIPTION ARCHITECTURALE

2.1 DEFINITION DU MODELE PAR EVENEMENTS

2.1.1 Définitions de base
2.1.2 Propriétés de base
2.1.3 Valeur des variables
2.1.4 Opérations de base
2.1.5 Exemples d'utilisation
2.1.6 Relations d'ordre entre événements
2.1.7 Conditions et filtrage
2.1.8 Exemple 4

2.2 AUTOMATES D'ETAT FINI

2.2.1 Définitions classiques d'automate d'état fini
2.2.1 Interprétation
2.2.3 Réalisations synchrones ou asynchrones
2.2.4 Definition-réalisation
2.2.5 Définition à temporisation explicite

TROISIEME PARTIE

3.1 REALISATION A ENTREES SORTIES ECHANTILLONNEES A BASE DE REGISTRES A FRONT

3.1.1 Introduction
3.1.2 Caractérisation de la réalisation 1
 3.1.2.1 Fonction de transition
 3.1.2.2 Fonction de sortie
 3.1.2.3 Table des résultats
3.1.3 Conformité à la définition d'un automate
 3.1.3.1 Correspondance réalisation 1 Automate de Moore
 3.1.3.2 Réalisation 1 Automate de Mealy
 3.1.3.3 Réalisation 2 Automate de Mealy
 3.1.3.4 Réalisation 2 Automate de Moore
3.1.4 Expression des délais

3.2 QUELQUES VARIANTES A LA REALISATION 1

3.2.1 Introduction
3.2.2 Alternative dans la synchronisation entrée/état

PREMIERE PARTIE

DESCRIPTIONS D'UN CIRCUIT INTEGRE INFORMATIQUE

1.1 NIVEAUX ET MODES DE DESCRIPTION D'UN CIRCUIT

1.1.1 NIVEAUX DE DESCRIPTION

On distingue classiquement quatre niveaux de description d'un circuit intégré, ou d'un ensemble informatique matériel en général. Ces niveaux correspondent à différents types de grandeurs traitées par le circuit.

Au niveau architectural,on considère que le circuit manipule des informations, dans le sens le plus général : nombres, instructions d'un processeur, adresse dans une mémoire, sont des exemples de telles informations.

Au niveau logique, on s'intéresse de plus au codage de ces informations. Dans la plupart des circuits actuels, les informations sont codées selon un vecteur booléen dont les composantes sont notées , selon les cas, 0 et 1 ou faux et vrai.

Au niveau électrique, on considère que le circuit traite des grandeurs de nature proprement électriques c'est à dire des intensités, des différences de potentiel.

Au niveau implanté n'apparait plus que la description des opérations nécessaires à la fabrication du circuit. Les techniques de fabrication étant standardisées, cette description se fait au moyen d'un schéma normalisé du circuit dit schéma des masques.

Les frontières entre les différents niveaux présentés ci-dessus ne sont pas définies de manière stricte. Par exemple une information qui ne peut prendre que deux valeurs sera tantot considérée comme étant de niveau architectural, tantot comme étant de niveau logique. De meme, un dispositif matériel qui laisse passer ou non une information selon que sa commande dépasse ou non un certain seuil de tension sera parfois décrit en termes logiques, parfois en termes électriques.

Il est clair, par ailleurs, que la description complète du circuit , que ce soit dans le cahier des charges avant conception, ou dans les documents de présentation après conception, nécessite de faire appel aux quatre niveaux de description.

1.1.2 MODES DE DESCRIPTIONS

A chacun des niveaux précédemment mentionnés, le circuit est décrit en termes de primitives ou blocs de base. Ces primitives peuvent etre des blocs matériels identifiés physiquement dans le circuit ou dans une bibliothèque de blocs précaractérisés. Il s'agit, par exemple de résistances, capacités, diodes ou

transistors au niveau électrique, de portes ou relais au niveau logique, d'opérateurs de mémorisation ou de transformation d'information au niveau architectural. Le circuit peut alors etre vu comme une interconnexion de ces blocs et on parlera de description selon un mode structurel.

Ces blocs peuvent également n'etre décrit qu'en termes de fonctions réalisées sur leurs entrées et le circuit est alors vu comme une coopération ou une composition de ces fonctions élémentaires. La description est alors selon un mode fonctionnel. Le niveau implanté a une particularité puisqu'on n'y rencontre que des descriptions structurelles.

Là aussi, il est difficile de définir une frontière stricte entre les deux modes de description puisqu'il est fréquent que les dispositifs matériels construits soient faits pour accomplir exactement une fonction au niveau considéré. On assimile alors la primitive matérielle et la fonction qu'elle réalise.

1.1.3 FORMALISMES DE DESCRIPTION

A chaque niveau de description peut exister un modèle de référence, c'est à dire une théorie mathématique avec ses axiomes, ses techniques de calcul et ses résultats propres. Les relations entre ces théories et les objets matériels à décrire sont l'objet de convention généralement admises.

Ainsi la théorie générale des fonctions de variables réelles (voire complexes) est réputée prendre bien en compte les

préoccupations des concepteurs pour une description au niveau électrique.

De meme, l'algèbre booléenne est adaptée aux descriptions au niveau logique. Les descriptions au niveau implanté font appel à une simple géometrie (plane si l'on considère un seul niveau de masques).

La meme unanimité n'existe pas au niveau architectural, c'est pourquoi les différents formalismes de description utilisés à ce niveau seront présentés plus loin de façon plus détaillée.

1.1.4 LANGAGES DE DESCRIPTION

La tache de conception d'un circuit n'étant pas dévolue à une personne seule, le besoin de communiquer a amené à fixer des règles, des normes de description. Ces règles, fixées à priori ou introduites par habitudes, concernent soit des communications entre individus, soit des communications homme/machine dans le cas des systèmes de conception assistée par ordinateur. Ces langages de description ont une syntaxe qui prend des formes bien différentes selon qu'il s'agit d'une description textuelle, graphique ou les deux à la fois. La sémantique de ces langages est dans tous les cas définie en référence au(x) modèle(s) propre(s) au niveau considéré. Des langages multi-niveau peuvent etre définis en référence à plusieurs modèles.

I.2 NIVEAUX LOGIQUE, ELECTRIQUE, IMPLANTE

I.2.1 DESCRIPTION EN TERMES LOGIQUES

Le formalisme permettant la description au niveau logique est né bien avant la possibilité de construire du matériel informatique (au sens actuel). Les " Lois pour la pensée logique " données par les logiciens anglais du XIX ème siècle BOOLE, de MORGAN, ou " Lewis CARROLL " étendues depuis en " algebre booléenne " permettent de calculer sur l'ensemble < vrai, faux > et donc d'y exprimer des fonctions dites booléennes.
Un circuit réalisant une ou plusieurs fonctions booléennes peut etre décrit en mode fonctionnel par le tableau de valeur des fonctions qu'il réalise, et conduisant aux expressions canoniques ou minimisées de ces fonctions <KUNT>.
Le meme circuit peut etre décrit en mode structurel si les primitives utilisées correspondent à des opérateurs matériels effectivement présents dans le circuit.

I.2.1.1 CHOIX DES OPERATEURS ELEMENTAIRES

Les opérateurs élémentaires ou primitives logiques dépendent essentiellement de la technologie utilisée pour réaliser les dispositifs . En technologie à base de transistors bipolaires, les primitives matérielles classiques correspondent aux fonctions de base NON-ET ou NON-OU qui serviront donc à exprimer les

fonctions booléennes.

Une expression , très classique , en deux etages NON-OU de plusieurs fonctions booléennes à l'aide d'un ensemble de monomes communs est aussi utilisée pour les réalisations sur des structures matérielles préparées à l'avance appelées réseaux logiques programmables ou P.L.A.

Dans les technologies à relais <CALD> et dans les technologies à transistors à effet de champ (M.O.S.F.E.T. ou M.E.S.F.E.T.) , la variable booléenne est associée à un relais, un monome à un chemin, le ET et le OU cablé réalisant le produit ou la somme de monomes. <MEAD> <MURO> <CARR>.

Une mention particulière doit etre faite de certaines organisations internes particulières des constituants des portes logiques comme l' I2L ou le MD.MOS <BERI> <MAJO>. On considère dans ces cas que le circuit est constitué d'opérateurs inverseurs dont les sorties sont en ET cablé ce qui amène à une description fonctionnelle à base de portes NON-ET alors que le circuit est effectivement réalisé à base de portes NON-OU.

1.2.1.2 LOGIQUE TEMPORISEE

La nécessité, primordiale, de prendre en compte non seulement le fonctionnement logique mais aussi le fonctionnement dans le temps a amené à développer une "algèbre booléenne temporisée" dans laquelle une variable booléenne dans l'ensemble < vrai, faux > cède la place à une fonction d'un ensemble d'instants

(typiquement l'ensemble N des naturels ou l'ensemble R+ des réels positifs), vers l'ensemble < vrai, faux >. On trouve de telles études dans <BARR> , <HAVI>, <SIFA>, <VITT>, par exemple.
Une extension peut etre faite aux ensembles de valeurs possibles pour prendre en compte des phénomènes temporels. Une valeur à un instant pouvant alors appartenir à l'ensemble

< vrai, faux, indéfini, pas encore défini (non initialisé) , front montant (en train de passer de faux à vrai) , front descendant >.

1.2.1.3 LOGIQUE MULTIVALUEE

La possibilite technique de réaliser des opérateurs élémentaires reconnaissant plus de deux valeurs différentes des entrées dans un circuit a nécessité d'établir des relations entre les travaux des concepteurs réalisant de tels dispositifs et ceux des logiciens qui explorent des logiques formelles basées sur des ensembles à plus de deux valeurs.
Une extension au domaine booléen est parfois faite pour considérer des phénomènes électriques d'impédance, <BRYA> ou <LEWK>, par exemple utilisent des valeurs telles que vrai_fort, vrai_faible, etc. Il est à remarquer que de telles extensions ne sont en général faite que dans les langages de description au niveau logique en vue d'une communication homme machine.
Les conférences annuelles sur les logiques multivaluées présentent de tels travaux.

I.2.2 DESCRIPTION EN TERMES ELECTRIQUES

La description du circuit en termes électriques suppose choisie une technologie de réalisatiuon des opérateurs logiques. Les techniques basées sur des phénomènes non purement électriques (Bio-transistors <dROS>, opto-électronique <BARC>) ne sont pas envisagées ici.

Les opérateurs de type portes logiques ou interrupteurs sont réalisés au moyen de dispositif de type électrique ou électronique (résistances, capacités, diodes, transistors,) Les équations décrivant la fonction de ces dispositifs sont données avec une précision plus ou moins importante pour les dispositifs complexes. Par exemple un transistor M.O.S. peut etre décrit de façon plus ou moins approchée suivant que l'on considère, par exemple, sa tension de seuil comme fixe ou variable, ses capacités "parasites " comme plus ou moins négligeables, son expression plus ou moins détaillée du courant dans le cas des transistors à canal court notamment.

Une différence importante avec le niveau logique est que les équations décrivant le fonctionnement du circuit en termes électriques sont nécessairement liées entre elles par le temps. Le fait que les grandeurs manipulées (intensité de courant, charge d'une capacité) fassent intervenir le temps sous forme différentielle ne facilite pas la résolution des systèmes d'équations décrivant au niveau électrique le fonctionnement d'un ensemble de dispositifs. Pour connaitre le " comportement

solution " on ne cherche, en général, pas à résoudre de tels systèmes d'équations, mais on calcule les valeurs successives de tensions ou d'intensités en certains points du circuit.

1.2.3 DESCRIPTIONS DES PLANS DU CIRCUIT

La réalisation éffective d'un circuit se fait par une succession d'opérations physico-chimiques en des zones bien déterminées d'un substrat. La conception du circuit nécessite donc de fournir une description des zones du circuit où chaque opération doit etre faite (ou pas faite).

Cette description est le plan d'un masque du circuit.

Certaines opérations se faisant soit sur tout le substrat, soit aux memes endroits qu'une autre, les ordres de grandeur sont d'une dizaine de plans de masques pour une quarantaine d'opérations différentes.

Des procédés de photogravure permettent de réaliser les opérations dans les zones du substrat correspondant aux zones du plan de masques. Les opérations ne se faisant dans les zones prévues qu'avec une certaine précision, des règles, habituellement nommées règles de dessin, précisent les précautions à prendre pour éviter les effets indésirables dus à ces imprécisions.

La grande répétitivité des motifs des plans a amené les concepteurs à utiliser pour travailler, des représentations symboliques de certains ensembles de motifs. Cela diminue la

quantité d'informations à manipuler pendant la conception, mais nécessite une traduction au moment de faire le schéma réel des masques.

Le traducteur peut alors plus facilement respecter les règles de dessin.

1.2.4 DESCRIPTIONS MIXTES

Aux cotés des descriptions bien caractéristiques d'un niveau donné ou d'un mode donné, on trouve des descriptions mixtes qui regroupent des informations de plusieurs niveaux. On trouve, par exemple de telles descriptions :

* logico-électriques dans un catalogue de constructeur :

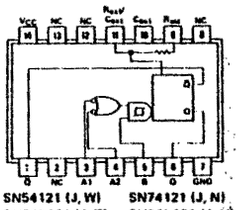
MONOSTABLE MULTIVIBRATORS

121

FUNCTION TABLE

INPUTS			OUTPUTS	
A1	A2	B	Q	$\bar{Q}$
L	X	H	L	H
X	L	H	L	H
X	X	L	L	H
H	H	X	L	H
H	↓	H	⊓	⊔
↓	H	H	⊓	⊔
↓	↓	H	⊓	⊔
L	X	↑	⊓	⊔
X	L	↑	⊓	⊔

NOTES: 1. An external capacitor may be connected between C_{ext} (positive) and R_{ext}/C_{ext}.
2. To use the internal timing resistor, connect R_{int} to V_{CC}. For improved pulse width accuracy and repeatability, connect an external resistor between R_{ext}/C_{ext} and V_{CC} with R_{int} open-circuited.



SN54121 (J, W) SN74121 (J, N)
SN54L121 (J, T) SN74L121 (J, N)
'121 . . . R_{int} = 2 kΩ NOM
'L121 . . . R_{int} = 4 kΩ NOM
NC—No internal connection

* logico-implantées dans les catalogues des constructeurs de réseaux prédiffusés où un opérateur logique est caractérisé par sa fonction logique, ses

caractéristiques temporelles, sa forme et les positions de ses connexions avec l'extérieur.

La description d'un additionneur faite dans <BREN> ou celle d'un multiplieur dans <VUIL> procedent de ces approches puisqu'un schéma logique de circuit y est considéré comme meilleur qu'un autre parce que conduisant à une implantation plus "facile ".
De meme, certains symbolismes de descriptions de circuits (les "stick- diagram " de <MEAD> , par exemple) contiennent a la fois des informations électriques et des informations sur la topologie du circuit.
Il est clair que toute approche de conception nécessitant une définition nette des niveaux de description considérés suppose de résoudre le probleme que pose la prise en compte de telles descriptions.

1.3 NIVEAU ARCHITECTURAL

Les informations considérées au niveau architectural sont, par exemple les nombres, les chaines de caractères et les propositions. Les modèles permettant d'exprimer les traitements subis par ces informations sont donc respectivement l'arithmétique, la théorie des langages et le calcul des propositions. La liaison avec le niveau logique se fait via une règle de codage. Par exemple le codage des naturels suppose l'introduction d'un numéro d'ordre aux composantes d'un vecteur booléen, et la transformation élémentaire suivante :

valeur logique " 1 " nombre 1 en numération binaire

valeur logique " 0 " nombre 0 en numération binaire

1.3.1 DESCRIPTION STRUCTURELLE

Les descriptions structurelles au niveau architectural décrivent les circuits comme une interconnexion de blocs combinatoires qui effectuent des transformations de données et de points de mémorisation servant à stocker l'information et comme barrière de synchronisation. L'extension de ces descriptions vers des structures plus complexes en maintenant le mode de description structurel conduit à une description en terme d'interconnexion de processeurs, de mémoires, et de connecteurs. Un ensemble de définitions précises permettant une description formalisée à ce niveau est proposée dans <HAFE>.

Des langages de description à ce niveau , souvent connus sous le nom de langages au niveau transfert de registres (R.T.L.) sont par exemple C.D.L., Cassandre ou KARL , ce dernier existant sous forme textuelle ou graphique.

1.3.2 DESCRIPTION FONCTIONNELLE EN TERMES DE COOPERATION DE PRIMITIVES FONCTIONNELLES

La complexité des circuits amène à considérer des primitives de plus haut niveau pouvant etre décrites par leur fonction. Le bloc élémentaire de base est alors le circuit séquentiel considéré comme une primitive et non comme l'interconnexion du circuit combinatoire et des points de mémorisation. Le circuit séquentiel sera décrit par l'automate d'état fini qu'il implémente. La description la plus fréquente est le graphe d'état et représente les transitions entre états. Les descriptions par fonctions donnant l'état suivant et les sorties en fonction de l'état présent seront longuement discutées dans cette thèse et l'on s'intéressera essentiellement au problème de la temporisation d'une telle description.

Une mention particulière doit etre faite des automates de controle qui sont des circuits séquentiels dont les sorties sont appelées des signaux de controle et dont les entrées sont en général non pas entièrement décodées sur l'ensemble < 0, 1 > mais données en termes de prédicat sur ces entrées. De meme les sorties ne seront parfois pas exprimées sous forme de " valeur "

des sorties mais sous forme d'action qu'ont ces sorties sur la partie "controlée" par cet automate de controle.

C'est le cas par exemple dans les
" algorithmic state machine " <WINK>
" control graphs " <AMB1>
" flowcharts for hardware " <TRED>, <JOHN>
" hardware flocharts " <GUST>
" organigrammes " <MANG>
" transitions graphs " <WALD>

et la liste n'est sans doute pas exhaustive.

De façon plus générale, le système CADOC <AMB2>, permet de décrire une ressource fonctionnelle par les transformations temporisées effectuées sur ses entrées. Une telle définition banalise les primitives du circuit. Un bloc combinatoire est une ressource fonctionnelle toujours activée, un bloc composé d'un circuit combinatoire entre deux registres peut etre facilement décrit par une ressource fonctionnelle. La notion de ressource générique permet de retrouver les facilités habituelles des langages de niveau R.T.L.

COOPERATION DE RESSOURCES FONCTIONNELLES

La description de circuits en terme de ressources fonctionnelles coopérantes peuvent s'inspirer

* des descriptions d'automates coopérants <MILI>, <LEIN>, <BONO>
* des descriptions de systèmes à transition avec parallélisme apparentées aux réseaux de PETRI. Ces systèmes à transitions acceptent l'expression des entrées sous forme de prédicat et peuvent faciliter les descriptions. <PETE>, <GRAF>, <MOAL>

De telles descriptions sont à la base de langages comme CAP_DSDL <RAMM>, Modlan <PAWL>, et CADOC <AMB2>

1.3.3 SYNCHRONISATION ET REFERENCE TEMPORELLE

Ces différents modèles sont plus ou moins adaptés à décrire différents schémas de synchronisation , réalisables sur un circuit, lorsqu'on veut que plusieurs dispositifs matériels échangent des informations :

schéma synchrone : les différents dispositifs reçoivent une meme référence temporelle physique, l'horloge, et n'évoluent qu'à des instants connus par rapport à cette horloge. Les échanges d'information n'ont lieu aussi qu'en référence à cette horloge.

schéma asynchrone : chaque dispositif évolue de manière autonome, on connait les délais d'évolution de chacun de manière suffisante pour garantir par construction la bonne " co-évolution " de l'ensemble. Il est clair que dans ce cas, les échanges se font " naturellement " aux moments où les deux dispositifs sont prets à échanger.

schéma endochrone (ou asynchrone synchronisé) Chaque dispositif évolue de manière autonome, mais lorsque deux d'entre eux ont à échanger une information, un troisième système vient les synchroniser. Ce système, spécialisé, peut etre soit réparti entre les deux dispositifs, soit centralisé.

Des éléments de comparaison enre ces différents schémas se trouvent notamment dans <LOOM>, <SEIT>, <WANN>.

1.3.4 QUEL TEMPS ?

Les différentes façons de décrire un circuit, outre leur aspect structurel ou fonctionnel, seront caractérisées par la " richesse " de l'ensemble considéré comme ensemble d'instants.

Certains modeles n'admettent sur les instants qu'une relation d'ordre, éventuellement non totale, (réseaux de PETRI) ; si la relation d'antériorité est totale on parle de première date, deuxième date, ... etc (Logique temporelle) .

On peut enrichir le modèle du temps en assimilant cette suite de dates à l'ensemble des naturels. On peut alors calculer des délais entre des dates et comparer ces délais.

Un enrichissement peut encore etre ajouté en assimilant le temps à un des ensembles tels qu'entre deux instants quelconques, il y ait un instant.

1.4 SURETE D'UNE DESCRIPTION

Les différentes façons de décrire un circuit qui ont été évoquées amènent à poser deux problèmes :

Etant données deux descriptions, décrivent-elles le meme circuit ?

Etant donnée une description , comment en obtenir un autre ?

Des procédés " manuels " existent qui permettent de résoudre ces deux problèmes de manière sure. Il se trouve que leur application est très couteuse en moyens humains:

soit qu'il s'agisse de tache très répétitive, demandant une grande attention pour ne pas commettre d'erreur,

soit qu'il s'agisse de tache très créative demandant à des spécialistes beaucoup d'intuition et de " métier " .

Habituellement la tache demandant beaucoup d'expertise est faite par des équipes de concepteurs, la partie répétitive du travail étant confiée à un système informatique.
Si l'on veut confier au systeme informatique l'aspect intelligent du travail il faut tenter de modéliser ce comportement d'un expert en conception de circuits . C'est l'objectif poursuivi dans le systeme PALLADIO par exemple <STE1>, <BRO1> ou dans <AGRE>.

Nous envisageons ici différentes façons de résoudre

automatiquement les deux problemes évoqués plus haut, en nous limitant à l'aspect répétitif et méticuleux du travail.

Le deuxième probleme est résolu par les outils de passage automatique d'une description à une autre (d'un niveau et/ou d'un mode différent) .

Le premier problème est résolu par les outils de vérification.

Il est clair que les deux approches sont opposées. Il est inutile, à priori , de vérifier la cohérence de deux descriptions dont l'une a été obtenue automatiquement à partir de l'autre.

Ces deux approches sont aussi complémentaires puisque, si l'on veut bien considérer que la conception d'un circuit consiste à mener de front plusieurs descriptions, on procèdera, suivant les cas soit par transformation automatique, soit par transformation manuelle et vérification automatique.

1.4.1 PASSAGE AUTOMATIQUE D'UNE DESCRIPTION A UNE AUTRE

1.4.1.1 CHANGEMENT DE NIVEAU

Dans les outils permettant le passage automatique d'un niveau de description à un autre, on distinguera les simples traducteurs qui n'envisagent qu'une seule solution d'arrivée possible et les outils plus généraux et plus ambitieux qui, parmi plusieurs solutions possibles, construisent " la meilleure " pour certains critères.

Des exemples de traducteurs sont par exemples les programmes qui passent directement d'une description architecturale à la

description implantée d'un circuit combinatoire réalisé à base de mémoire morte (calcul par tabulation de valeur) La réalisation peut aussi etre à base de P.L.A. , dont on peut, bien que ce ne soit guère astucieux, obtenir un schéma sans passer par l'intermédiaire des fonctions booléennes.

On considèrera aussi comme traducteurs les systèmes qui transforment un schéma mixte logico-implanté ou électrico-implanté en schéma implanté à partir d'une bibliothèque de cellules prédéfinies par exemple.

Les extracteurs qui donnent le schéma électrique d'un circuit à partir de son schéma implanté sont aussi des traducteurs.

Par contre , des outils comme ceux utilisés en placement ou en routage dans les systèmes d'implantation automatique ne sont pas seulement des traducteurs puisqu'ils tiennent compte de plusieurs critères d'évaluation des solutions possibles et ne retiennent que la meilleure de celles qu'ils ont construites.

On peut en rapprocher les systèmes où l'on cherche les meilleures solutions à des problèmes de niveau logique compte-tenu de critères tels que le nombre de portes, le nombre de niveaux de portes, le nombre de connexions internes, ...

1.4.1.2 CHANGEMENT DE MODE

L'usage a consacré le terme de synthèse de circuit à un certain niveau pour désigner à ce niveau le passage d'une description fonctionnelle à une description structurelle (et le terme d'analyse pour l'opération réciproque). Ces transformations sont

faites automatiquement par exemple dans les systèmes qui permettent , à partir d'une description d'une fonction booléenne sous forme de table de valeurs d'en donner une expression booléenne sous une forme voulue (somme de monomes, produit de monaux...) De manière analogue, des systèmes envisagent, au niveau architectural, l'organisation possible d'une partie opérative en termes de registres, opérateurs et bus, en fonction d'une structure visée (machine à un, deux, ... bus) et d'une description fonctionnelle du circuit sous forme d'algorithme faisant la meme transformation que le circuit.

1.4.1.3 OPTIMISEURS

Les résultats obtenus automatiquement par l'un des outils évoqués ci-dessus ou les résultats manuels sont parfois jugés peu performant pour un critère donné.

Une gamme d'outils de pure optimisation permet , en général, d'améliorer pour ces critères , une description pour fournir une description de meme niveau et de meme mode.

C'est le cas pour les outils qui visent à diminuer le nombre de monomes utilisés pour l'expression de plusieurs fonctions booléennes données comme somme de monomes.

C'est aussi le cas pour certains optimiseurs topologiques qui re-traitent une implantation pour en diminuer la surface mais sans " remonter " au schéma qui avait conduit à cette implantation .

I.4.1.4 CHAINE INTEGREE DE CONCEPTION AUTOMATIQUE

L'ensemble des outils mentionnés ci-dessus étant supposés disponibles , il est tentant d'organiser un système permettant à ces différents outils de prendre la suite les uns des autres. On trouve des descriptions de tels systèmes dans <KATZ> ou <BEYL>. Par exemple l'obtention du schéma implanté d'une machine spécialisée dans l'exécution d'un algorithme peut etre considéré comme entièrement automatisable. On parle alors de compilation de silicium.

Etant donné un algorithme décrit en terme de coopération de ressources fonctionnelles, la conception du circuit peut etre assistée, ou automatisée dans les étapes suivantes :

* Evaluer la complexité et les performances de différentes "parties opératives" possibles . <COFF> <HAFE>

**** En choisir une.

* Rechercher la " meilleure " synthèse logique des opérateurs arithmétiques . <LIU >.

* Faire l'implantation de la partie opérative. <SISK>

* Faire la synthèse de la partie controle

 - Pour un controleur decrit par un automate

  * à partir de mémoire morte R.O.M. <WALD>

  * ou à partir de P.L.A. <GRAS>

  * ou à partir de logique "aléatoire" <BROZ>

 - Pour un controleur décrit par un réseau de PETRI

  * à partir de P.L.A. <OLIV>

* ou à partir de "logique aléatoire " répétitive <DAVI>

- Pour un automate de controle décrit par une expression régulière

* à partir de logique aléatoire régulière <FOST> <CULI>

* ou à partir de P.L.A. <FLOY> <KARL> <TRIC>

**** Dans le cas d'une synthèse sur P.L.A. :

* Trouver un " bon " codage des états <SAUC>

* Construire les équations booléennes de la partie combinatoire <KANG> <TEEL>

* Faire le schéma du P.L.A. et l'améliorer si besoin .

**** Dans le cas d'une synthèse aléatoire, utiliser un outil d'implantation automatique <MALL>

Un problème complexe est alors de savoir quels optimiseurs choisir, sachant que les résultats d'un optimiseurs peuvent compliquer la tache des outils qui travaillent après lui.
L'utilisation successive de tels outils fournit un schéma juste par construction , qu'il n'y a pas besoin de vérifier, si tant est, naturellement, que les différents outils soient certifiées.
On trouve dans <MIL2> une proposition de preuve d'un tel outil.

1.4.2 OUTILS DE VERIFICATION

On distinguera les outils de vérification qui permettent de s'assurer que le circuit dont on n'a qu'une description possède des propriétés invariantes, et les outils qui vérifient la cohérence entre deux descriptions.

1.4.2.1. VERIFICATION D'INVARIANTS

La vérification de propriétés invariantes s'applique en général à des descriptions structurelles obtenues à la main ; on cherche alors à vérifier qu'elle respecte ceratines règles d'assemblage propre au niveau considéré.

* vérification des règles de dessin au niveau implanté,
* vérification de bonnes connexions au niveau électrique (court-circuits, entrée acceptable de certains dispositifs...)
* vérification d'entrance ou de sortance , de rebouclage , au niveau logique.

On peut aussi chercher, dans une description fonctionnelle au niveau architectural, que les types d'informations obtenues dans une transformation soient compatibles avec les définitions de ces informations.

1.4.2.2 PREUVE DE CONFORMITE ENTRE DEUX DESCRIPTIONS

Lorsque deux descriptions reposent sur un meme formalisme, il est envisageable de prouver, dans la théorie mathématique adéquate, que ces deux descriptions sont " équivalentes " au sens d'une relation d'équivalence qu'il faut préciser. <GORD>
On dispose, pour faire ces preuves, de calculs des systèmes communicants, <MILI>, utilisé dans <CARD>, <GORD>, <McFA> <SUBR>, de calculs de prédicats tempoisés <MANN>, utilisés dans <BOCH>, <MALA>, <PNUU>, d'algèbre booléenne temporisée utilisée dans

<DARR>, d'algèbre des événements <CAS1> utilisée dans <CAS2> et dans cette thèse, de calcul sur les langages formels , ou de modèles développés spécialement comme arrière plan formel à un langage de type R.T.L. . La preuve d'un receveur transmetteur universel asynchrone décrit en A.H.P.L. <HILL>, est par exemple fournie dans <UMRI>.

Dès que le circuit devient complexe, il devient nécessaire d'automatiser cette preuve d'équivalence comme <SHOS> ou <FUJI>. Une alternative à la technique de preuve d'équivalence étant la manipulation symbolique des informations traitées dans le circuit. <CART>, <WAGN>.

1.4.2.3 SIMULATION D'UNE DESCRIPTION STRUCTURELLE

Lorsqu'une description structurelle devient complexe, il est fréquent qu'on n'arrive pas à prouver ni à la main, ni automatiquement, l'équivalence avec une description fonctionnelle de meme niveau (Non qu'on ne sache pas, mais en raison de la comlexité) . En pratique, on se contente alors , en général, d'une sous-équivalence pour un ensemble d'entrées judicieusement choisies par le concepteur mais dont on sait qu'elles ne couvrent pas exhaustivement l'ensemble des fonctionnements possibles. Cette sous-équivalernce peut etre obtenue par simulation soit au niveau logique, soit au niveau électrique, soit au niveau architectural.

DEUXIEME PARTIE

MODELES POUR UNE DESCRIPTION ARCHITECTURALE <CASI> <HALB>

2.1 DEFINITION DU MODELE PAR EVENEMENTS

Conventions typographiques

N désigne l'ensemble des naturels
Z ..... entiers
R ..... réels
N* ..... naturels strictement positifs
% désigne l'infini à la fois dans TT, N, N*.

! remplace les flèches en bas (Front descendant)
^ ..... haut (..... montant)
* est le symbole de la composition des fonctions
=/ remplace le signe d'inégalité

qqst remplace le quantificateur universel,
ilex remplace le quantificateur existentiel
apra remplace le symbole d'appartenance

2.1.1 DEFINITIONS DE BASE

TT est le temps, ensemble des instants, ou dates.

Il est totalement ordonné.

Muni de l'addition et de la soustraction usuelles, l'ensemble des instants TT peut aussi etre considéré comme l'ensemble des délais, c'est à dire des longueurs d'intervalles entre instants.

Dans la plupart des cas, TT est assimilé soit à l'ensemble Z des entiers relatifs, soit à l'ensemble R des réels.

L'etre de base du modèle est un événement. Intuitivement un événement correspond à " Il s'est passé quelquechose dans un système, et je sais à quelle date."

Un événement, e, est appréhendé par sa suite de dates, D.e .
D.e est une application croissante de N vers TT .
D.e(n) désigne la date de la n-ième occurence de l'événement e.
Un événement peut etre à occurences simples si
qqst n apra N*, D.e(n+1) > D.e(n)

ou à occurences multiples si
ilex n0 apra N*, D.e(n0+1) = D.e(n0)

Pour des raisons algébriques explicitees dans <HALB>, les trois valeurs aux limites suivantes seront prises :
D.e(0) = - %
D.e(1) > - %
D.e(+ %) = + %

Une autre façon de caractériser un événement e est son compteur d'occurences noté C.e :

C.e est une application de TT vers N U {-1} définie par
C.e(-%) = -1

$C.e(t) = Sup \{ n \text{ apra } N^* , D.e(n) < t \}$

avec $Sup\ N^* = +\infty$

C.e(t) est le nombre d'occurences de e survenues strictement avant t

De manière analogue, on définit C+.e(t) nombre d'occurences de e survenues avant t au sens large.

$C+.e(-\infty) = 0$

$C+.e(t) = Sup \{ n \text{ apra } N^* , D.e(n) <= t \}$

Par composition fonctionelle, on forme

$L.e = D.e * C.e$

$L.e(t) = D.e * C.e(t)$

$= D.e(C.e(t))$

L.e(t) désigne la date de la dernière occurence de e survenue avant t.

2.1.2 PROPRIETES DE BASE

qqst t apra IT, $C.e(t) <= C+.e(t)$

Ce que l'on notera

$C.e <= C+.e$

Si e est à occurences simples, on a de plus

qqst t apra TT, C+.e(t) <= C.e(t) + 1

on notera

C+.e <= C.e + 1

Remarquons que dans cette deuxième écriture, 1 désigne l'application constante 1(t) = 1. Cette convention d'écriture sera prise , le contexte permettant de lever l'ambiguité.

qqst n apra N, C.e * D.e (n) = C.e(D.e(n))

<= n-1

Ce que l' on notera

(I+1) * C.e * D.e <= I

I désignant l'identité dans l'ensemble des entiers.

qqst n apra N, C+.e * D.e (n) >= n

On écrira

C+.e * D.e >= I

REMARQUE

Pour un événement à occurences simples on a :

C+.e * D.e = I

et

(I+1) * C.e * D.e = I

2.1.3 VALEUR DES VARIABLES

Une variable X est décrite au moyen :

* du domaine $\mathcal{X}$ des valeurs possibles

* de la séquence V.x des valeurs apparaissant dans l'évolution de X. V.x est une application de N vers $\mathcal{X}$ et V.x(n) est la n-ième valeur survenant dans l'histoire de la variable X.

* de l'événement D.x des changements de valeurs ; par convention, X prend la valeur V.x(n) à la date D.x(n).

V.x(0) est donc indéterminée.

Par composition , on forme

W.x = V.x * C.x

(et, de meme, W+.x = V.x * C+.x)

W.x(t) désigne alors la valeur " courante " de x à la date t.

2.1.4 OPERATIONS DE BASE

SOMME

On souhaite pouvoir décrire l'événement x qui se produit à chaque fois qu'un des deux (au moins) événements a ou b survient. La suite des dates de x résultant alors de la fusion, ordonée, de la suite des dates de a et de la suite des dates de b.

Cette opération se décrit aisément au moyen des compteurs d'occurences :

On a alors :

qqst t apra TT : C.x(t) = C.a(t) + C.b(t)

on notera alors

x = a + b

Remarquons que des occurences multiples peuvent apparaitre meme si a et b sont à occurences simples.

SUPPRESSION DE LA (DES) PREMIERE(S) OCCURENCE(S)

x étant un événement, on définit :

C.x'(t) = C.x(t) - 1 si C.x(t) >= 1

= C.x(t) sinon

Alors C.x' est un compteur. C'est à dire qu'il existe un événement x' tel que C.x' soit son compteur d'occurences. La suite de dates de x' est alors donnée par :

D.x'(0) = -%

D.x'(1) = D.x(2)

qqst n apra N*, D.x'(n) = D.x(n+1)

On écrira alors :

x' = x - 1

et, de proche en proche,

x - (n+1) = (x - n) - 1

Remarquons que la notation soustractive ne doit pas faire penser à une " réciproque " de la somme. On n'a pas

$(x-1) + (y-1) = x+y - 2$.

RETARD

On souhaite pouvoir exprimer qu'un événement y est " le meme que x " mais retardé d'un certain délai d (d apra ZZ).

On a ainsi :

qqst n apra N , D.y(n) = D.x(n) + d

C'est à dire

qqst t apra ZZ , C.y(t) = C.x(t - d)

on notera

C.y = C.x * (I-d)

ou

y = R_d x

Cet opérateur de retard a les propriétés suivantes :

R_0 x = x

R_d (R_d' x) = R_(d + d')

R_d (x + y) = R_d x + R_d y

R_d (x - 1) = R_d x - 1

2.1.5 EXEMPLES D'UTILISATION

EXEMPLE 1

Description d'un circuit combinatoire à délai nul

Un circuit combinatoire à deux entrées (qui sont des variables), A et B, et une variable de sortie S.

On a

$s = a + b$

Ce qui signifie que la variable de sortie change à chaque fois qu'une des deux variables d'entrée change.

Si le circuit combinatoire réalise une fonction f, on a :

$W.s(t) = f(W.a(t) , W.b(t))$

Ce que l'on notera

$W.s = f (W.a , W.b)$

EXEMPLE 2

Description d'un circuit combinatoire à délai fixe d.

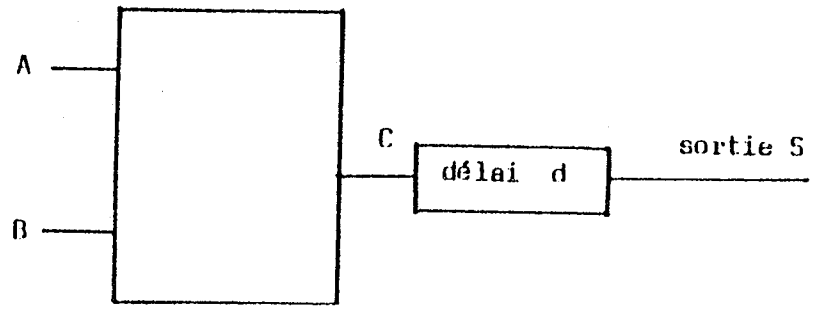
On prendra, pour décrire le fonctionnement d'un tel dispositif une hypothèse simplificatrice : on le considèrera comme constitué:

* d'un circuit combinatoire à délai nul analogue à celui de

l'exemple 1

suivi

* d'un retardateur pur.



Comme dans l'exemple 1 :

c = a + b

W.c = f (W.a , W.b)

Pour le retardateur, on a :

s = R_d c

c'est à dire

C.s = C.c * (1-d)

L'hypothèse d'intègrité du retardateur (aucune valeur n'apparait ou ne disparait) étant :

V.s = V.c

On a alors :

$W.s = V.s * C.s$ par définition

$= V.s * C.c * (I-d)$

$= V.c * C.c * (I-d)$

$= W.c * (I-d)$

$= f(W.a , W.b) * (I-d)$

C'est à dire

$W.s = f(W.a * (I-d) , W.b * (I-d))$

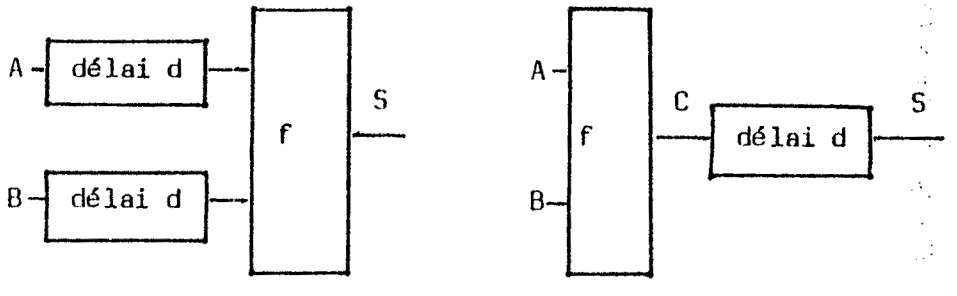
et

$C.s = C.c * (I-d)$

$= (C.a + C.b) * (I-d)$

$= C.a * (I-d) + C.b * (I-d)$

On note au passage l'équivalence (sous l'hypothèse du délai fixe) entre les deux compositions figurées ci dessous



Le besoin d'exprimer la notion de délai maximal au lieu de ce faible délai fixe nécessite de pouvoir comparer, de pouvoir ordonner deux événements.

2.1.6 RELATIONS D'ORDRE ENTRE EVENEMENTS

Deux façons d'ordonner les événements sont intéressantes:

Un ordre strict peut etre défini entre des événements e et f par:
qqst n apra N*, D.e(n) < D.f(n)

Un ordre peut etre défini entre deux événements x et y par :
qqst t apra TT , C.x(t) <= C.y(t)

Il est à remarquer que cet ordre strict n'est pas associe à cet ordre large, on a, en fait :
qqst n apra N*, D.e(n) > D.f(n) <=> qqst t apra TT, C+.e(t) <= C.f(t)

Démonstration

Supposons

qqst n apra N*, D.e(n) > D.f(n) P1

Soit t apra TT (t =/ -%)

Soit k = C+.e(t) donc D.e(k) <= t 1

Soit l = C.f(t)

Raisonnant par l'absurde, supposons l < k

Comme l = Sup {n apra N , D.f(n) < t}

D.f(k) >= t 2

P1, 1 et 2 donnent alors

t <= D.f(k) < D.e(k) <= t

On a donc bien k <= l, c'est à dire C+.e(t) <= C.f(t)

réciproquement,

posons

qqst t apra TT (t =/ -%) , C+.e(t) <= C.f(t) P2

Soit n apra N*

et t0 = D.e(n) donc n <= C+.e(t0) 3

et t1 = D.f(n) donc C.f(t1) < n 4

Raisonnant par l'absurde, supposons t0 <= t1

C+.e est croissante donc C+.e(t0) <= C+.e(t1) 5

Les équations 3, 4, 5, et p2 donnent

n <= C+.e(t0) <= C+.e(t1) <= C.f(t1) < n

On a donc bien D.e(n) > D.f(n) et cela pour tout n strictement positif

Nous noterons

e < f pour qqst t apra TT C+.e(t) <= C.f(t)

et

e <= f pour qqst t apra TT C.e(t) <= C.f(t)

en gardant toutefois présent à l'esprit que

e<f implique

e<=f et e =/f

mais que la réciproque est fausse

Il faut noter que ces deux ordres ont un "bon" comportement par rapport à l'addition, au retard et à la suppression de la première occurence. On a notamment :

$e \leq f$ implique $e + g \leq f + g$

$R\_d\ e \leq R\_d\ f$

$e - 1 \leq f - 1$

et

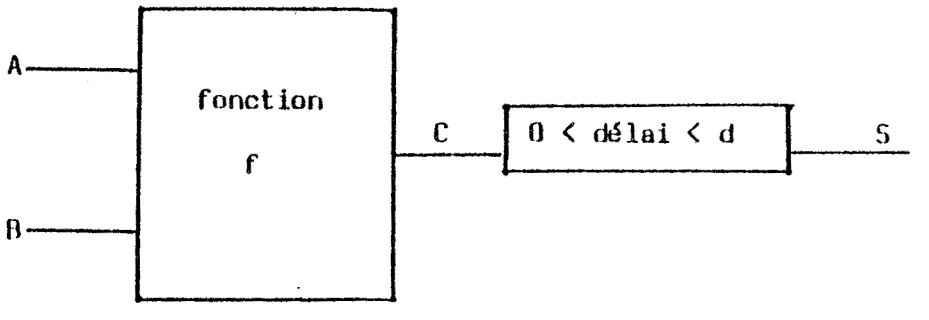
$e < f$ implique $e + g < f + g$

$R\_d\ e < R\_d\ f$

$e - 1 < f - 1$

EXEMPLE 3

Circuit combinatoire à délai non nul et inférieur à d strictement.



Comme dans l'exemple 2, on a :

$c = a + b$

$W.c = f (W.a , W.b)$

$V.s = V.c$

mais, à la différence de l'exemple 2 :

R_d c < s < c

Ce qui signifie que

C+.c * (I-d) <= C.s

et C+.s <= C.c

On peut aussi introduire ce retard inconnu en supposant l'existence d'une fonction r, de TT dans TT , telle que

I - d < r < I

et de r' ayant la meme propriété et poser

W.c = f (W.a * r , W.b * r')

2.1.7 CONDITIONS ET FILTRAGE

CONDITIONS

Une variable dont le domaine de valeur est booléen sera appelée une condition. A une condition A ($\{0,1\}$; V.a ; D.a) on peut associer les deux événements notés a^ et a! correspondant aux fronts montants et descendants de A. a^ et a! ont les propriétés suivantes :

V.a * D.a! = 1

V+.a * D.a! = 0

V.a * D.a^ = 0

V+.a * D.a^ = 1

FILTRAGE

L'opération de filtrage d'un événement par une condition permet de ne considérer que les occurences d'un événement qui ont lieu pendant que la condition est vraie (vaut 1).

Ainsi, si e désigne un événement et a une condition,

f = e / a désigne l'événement e filtré par la condition a.

On a alors

ilex n tel queD.f(n) = t <=> ilex p tel que D.e(p) = t et W.a(t) = 1

2.1.8 EXEMPLE 4

Le dispositif matériel décrit dans cet exemple récapitulatif est constitué d'un circuit combinatoire ensérré entre deux étages de mémorisation.

Le circuit combinatoire est à délai non nul, mais inférieur à une valeur maximale d.

Les points de mémorisation sont des registres sensibles au front parfaits.

L'échantillonnage d'une valeur E dans un registre statique, sensible au front d'une horloge h , supposé parfait, c'est à dire à temps d'établissement nul sera décrit ainsi :

E est une variable,

h est un événement,

S est une variable définie sur le meme domaine que E.

On a s = h

C'est à dire que les dates de changement de la sortie S sont les dates d' occurence de l'horloge h.

et

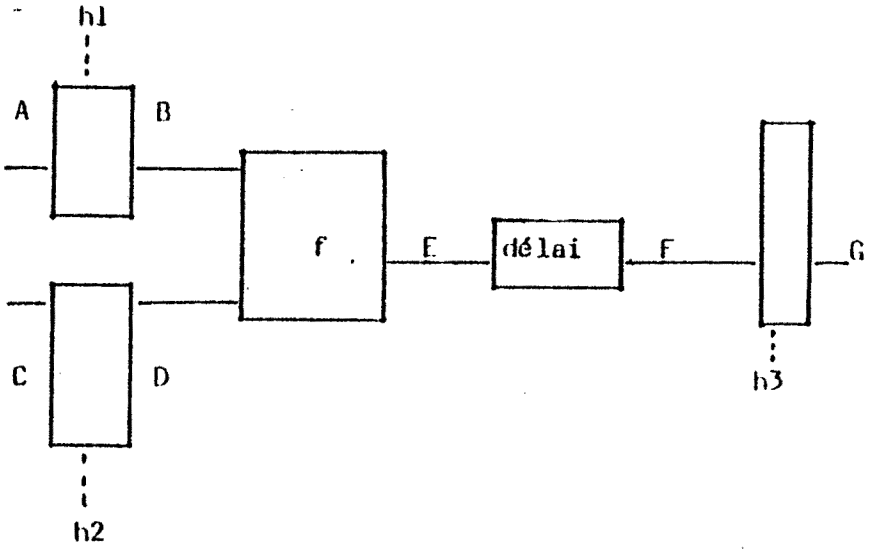
V.s = W.e * D.h

C'est à dire que la n-ième valeur de la sortie S est la valeur " courante " de l'entrée E à la date de la n-ième occurence de l'horloge E.

REMARQUES

Le choix de V.s = W.e * D.h au lieu de V.s = W+.e * D.h précise de plus qu'en cas d'occurence de l'horloge pendant un changement de l'entrée E c'est l'ancienne valeur de l'entrée qui est mémorisée . Il y a la un parti pris, discutable, de modélisation, qui ne présage pas de la plus ou moins grande difficulté qu'il y a à réaliser un dispositif qui ait ce fonctionnement.
Aucune hypothèse n'est faite ici sur la nature physique du front d'horloge provoquant l'échantillonnage. Il peut tout aussi bien s'agir d'un front montant, que d'un front descendant.

STRUCTURE DU DISPOSITIF



* Fonctionnement des trois registres.

V.b = W.a * D.h1 ; b = h1 1

V.d = W.c * D.h2 ; d = h2 2

V.g = W.f * D.h3 ; g = h3 3

* Fonctionnement du circuit combinatoire.

(D'après l'exemple 3)

e = b + d = h1 + h2 4

W.e = f(W.b , W.d) 5

V.f = V.e 6

R_d e < f < e 7

* Hypothèses de fonctionnement temporel .

h3 < R_d Inf(h1 , h2) 8

Sup(h1 , h2) - 1 <= h3 9

ou, ce qui est équivalent,

C.sup(h1 , h2) <= C.h3 +1

****
* Propriété : V.g = f(V.b , V.d) *
****

Avant d'établir cette propriété, précisons la portée des hypothèses sur l'échantillonnage. Ces deux hypothèses sur le fonctionnement temporel , donc sur l'échantillonage, sont assez " naturelles ".

8 traduit que l'échantillonnage des sorties a lieu au moins d après le dernier échantillonnage d'une entrée. Cela correspond à laisser le délai d d'établissement des sorties du circuit combinatoire.

9 demande que cet échantillonnage des sorties n'ait pas lieu après le prochain premier échantillonnage de l'une des entrées, le délai combinatoire pouvant etre arbitrairement proche de zero. Une conséquence , non évidente de ces deux hypothèses temporelles est qu'aucune des deux horloges h1 ou h2 ne peut " prendre deux coups d'avance " sur l'autre.

Deux lemmes sont nécessaires à l'établissement de la propriété annoncée :

Lemme 1 : C.e * D.h3 = C.f * D.h3

Lemme 2 : C.h1 * D.h3 = C.h2 * D.h3 = I

Démonstration

D'après 4 ,

e = h1 + h2

on tire, en notant 2x pour x + x,

2 Inf(h1 , h2) <= e <= 2 Sup (h1 , h2)

et

2 R_d Inf (h1 , h2) <= R_d e

En tenant compte, de plus de 7, 8 et 9 , il vient :

2 h3 <= 2 R_d Inf(h1,h2) <= R_d e < f < e <= 2 Sup(h1,h2)

et

2 (Sup (h1,h2) - 1) <= 2 h3

c'est à dire, en particulier

2 (h3 - 1) < f - 1 < e - 1 <= 2 h3

c'est à dire

2C+.h3 <= C.f <= C+.f <= C.e <= 2(C.h3 + 1)

Par composition fonctionnelle à droite par D.h3, il vient

2 C+.h3 * D.h3 <= C.f * D.h3 <= C.e * D.h3

C.e * D.h3 <= 2(I+1) * C.h3 * D.h3

Comme

I <= C+.h3 * D.h3

et

(I+1) * C.h3 * D.h3 <= I

on a

2 I <= C.f * D.h3 <= C.e * D.h3 <= 2 I

D'ou le lemme 1 : C.f * D.h3 = C.e * D.h3

De manière analogue, on a, (pour i = 1 ou 2),

h3 < R_d Inf(h1,h2) < Inf(h1,h2) <= hi <= Sup(h1,h2)

et

Sup(h1,h2) - 1 <= h3

D'où

C+.h3 <= C.hi <= (I+1) * C.h3

et

C.hi * D.h3 = I

D'où le lemme 2 : C.h1 * D.h3 = C.h2 * D.h3 = I

On a alors

V.g = W.f * D.h3 d'après 3

= V.f * C.f * D.h3

= V.e * C.f * D.h3 d'après 6

= V.e * C.e * D.h3 d'après le lemme 1

= W.e * D.h3

= f(W.b , W.d) * D.h3 d'après 5

= f(W.b * D.h3 , W.d * D.h3)

= f(V.b * C.b * D.h3 , V.d * C.d * D.h3)

= f(V.b * C.h1 * D.h3 , V.d * C.h2 * D.h3)

d'après 1 et 2

soit, finalement, d'après le lemme 2

V.g = f (V.b , V.d)

2.2 DEFINITION D'AUTOMATE D'ETAT FINI

2.2.1 DEFINITIONS CLASSIQUES D'AUTOMATES D'ETAT FINI

Un automate d'état fini est défini classiquement de l'une des deux façons suivantes :

D1 DEFINITION PAR FONCTIONS

Un automate d'état fini est la donnée d'un quintuplet $(\mathcal{L}, \mathcal{S}, \mathcal{O}, f, g)$ où :

* $\mathcal{L}$, $\mathcal{S}$, $\mathcal{O}$ sont des ensembles finis non vides respectivement appelés ensembles d'entrées, d'états, de sorties.
* f est une application de $\mathcal{L} \times \mathcal{S}$ vers $\mathcal{S}$

  f s'appelle la fonction de transition.
* g est une application

  de $\mathcal{L} \times \mathcal{S}$ vers $\mathcal{O}$ pour un automate dit de MEALY

  on la note alors ge

  de $\mathcal{S}$ vers $\mathcal{O}$ pour un automate dit de MOORE

  on la note alors go

  g s'appelle la fonction de sortie.

D2 DEFINITION PAR SUITE DE VALEURS

Un automate d'état fini est la donnée d'un quintuplet (L,S,O,f,g) où :

* L = < l1, l2, ... , ln , .. >

* S = < s1, s2, ... , sn , .. >

* O = < o1, o2, ... , on , .. >

sont des suites de valeurs appelées respectivement suites d'entrées, d'états, de sortie,

* s1 s'appelle l'état initial,

* f et g sont deux relations entre ces suites telles que :

qqst n >= 1 s n+1 = f (In,sn)

on = ge(In,sn) pour un automate de MEALY

= go(sn) pour un automate de MOORE

La correspondance entre les deux définitions se faisant en supposant que les Ii sont éléments de $\mathcal{I}$, les si de $\mathcal{S}$ les oi de $\mathcal{O}$

2.2.2 INTERPRETATION

L'inteprétation de cette deuxième définition peut se faire de deux points de vue : l'un , très " algébrique ", considère ces suites de valeurs sans notion de temps. Cette interprétation est celle habituellement retenue en théorie des langages réguliers. L'autre interprétation, plus " physique " considère les suites de valeurs comme des suites de valeurs observées sur un dispositif matériel. Le temps y est en général considéré comme un espace discret, totalement ordonné, < t1, t2, ..., tn, ..> et l'interprétation la plus courante est de considérer que
si est la valeur de " l'état " à la date ti.
A l'intérieur de l'interprétation physique,on rencontre toutefois, entre différents auteurs, des divergences concernant

la signification des li et des oi. La question est en général de savoir si li désigne l'entrée

* à la date ti
* à la date ti-1, ti+1, ...
* entre les dates ti-1 et ti mais on peut alors se demander ce que signifie l'expression " entre " deux dates si le temps est considéré comme espace discret.

Le meme probleme se pose pour les sorties.

* Soit oi est la sortie à la date ti mais alors l'équation oi = go(si) ou o(ti)=go(s(ti)) ne peut décrire une réalisation matérielle car elle suppose un délai nul de calcul de la fonctin go
* Soit oi est la sortie à la date ti+1, strictement postérieure à ti, ce qui est physiquement réalisable, mais, écrivant oi = go (si), c'est à dire o(ti+1)=go(s(ti)), on introduit une surspécification de réalisation en sous-entendant que les variations de sortie sont synchronisées sur les variations d'état avec exactement un retard d'une date.
* Soit oi est la sortie entre des dates mais là aussi se pose le problème du sens de cette expression.

2.2.3 REALISATIONS SYNCHRONES OU ASYNCHRONES

Toutes les réalisations habituellement décrites d'un automate sont organisées autour d'un circuit combinatoire de calcul de la fonction de transition f et d'un circuit combinatoire de calcul

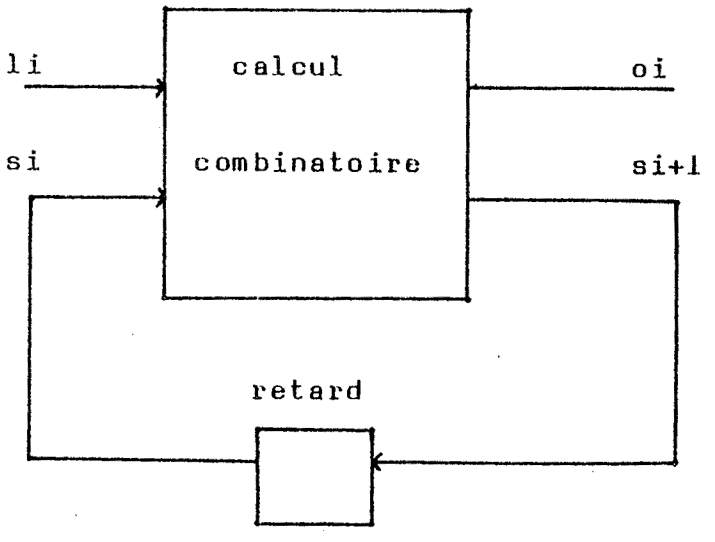
de la fonction de sortie g .

Dans les réalisations séquentielles, le " nouvel état " est la sortie du circuit de calcul de f. Il est rebouclé, en entrée du meme circuit , avec un certain retard. Ce "certain " retard, d'une date, assure la transformation mentionnée plus haut :

s(ti+1) = f (s(ti), li)

ou

si+1 = f (si , li)



On distingue deux familles de réalisations :

* les réalisations asynchrones où le retard introduit ne l'est que par le délai de travail du circuit de calcul de f. Dans ce cas, les dates d'evolution de l'état sont fixées par les dates d'évolution des entrées.

* les réalisations synchrones où le retard est fixé en référence à un signal physique d'horloge qui le controle. Dans presque tous les cas, ce signal est périodique, c'est à dire que, si la notion de délai a un sens dans l'espace d'instants considéré, délai(ti+2;ti+1) = délai(ti+1;ti).

Il est clair qu'une façon de voir la différence entre les deux réalisations revient à se demander si le signal d'horloge est ou non une entrée de l'automate.

Les réalisations asynchrones présentent des difficultés particulières. Un changement spécifié comme simultané de deux variables internes codant les états ou de deux entrées n'est en fait pas simultané dans la réalité et peut conduire à un état incorrect. (Phénomène appelé course critique, ou aléa) Ce problème peut etre résolu par un codage spécial des variales internes pour ce qui concerne les aléas. <UNGE>, <SAUC> . Les variations simultanées des entrées sont des configurations interdites.

2.2.4 DEFINITION-REALISATION

On a vu la difficulté qu'il y a à donner une définition temporisée d'automate d'état fini surtout si l'on veut que cette définition traduisent les différentes possibilités de réalisations.

Si on reste au niveau des suites de valeurs si, si+1, li, li+1, les problèmes temporels (retard, synchronisation) sont mal pris en compte.

Si on choisit une définition très proche de la réalisation

qqst t apra R $s(t) = f (s(t- \Delta t), 1(t - \Delta' t))$

on a du mal à exprimer la notion de succession d'état.

Nous proposons de donner ici une définition à temporisation

explicite basée sur le modèle par événements précédemment décrit et qui permettra d'avoir les avantages des deux approches.

Dans cette définition, on utilisera les trois variables L, S, O, définies sur les trois domaines $\mathcal{L}$, S, O, correspondant aux domaines d'entrées, états, sorties.
Aux trois événements l, s, o, correspondant à ces trois variables, on ajoutera deux événements l' et o' correspondant à des observations des entrées ou des sorties. l' correspond aux dates où l'automate " observe l'extérieur, o' aux dates où l'extérieur observe l'automate. On distingue donc les instants de changement des entrées, et les dates où des changements d'entrées sont perçus par l'automate.

Dans les cas de réalisation à entrées et sorties échantillonnées, les dates d'observation et les dates de variation des entrées seront considérées comme identiques.

2.2.5 DEFINITION A TEMPORISATION EXPLICITE

Un automate d'état fini est la donnée

* de trois ensembles finis , non vides, $\mathcal{L}$, S, O, d'entrées, d'états et de sorties.
* de deux événements l' et o'
* de deux applications f et g définies sur

$f: \mathcal{L} \times S \longrightarrow S$

$g: \mathcal{L} \times S \longrightarrow O$ pour les automates de MEALY

g est alors notée ge

$g: \mathcal{S} \longrightarrow \mathcal{O}$ pour les automates de MOORE

g est alors notée go

* de deux délais dmin et dmax, éléments de TT, positifs, tels que :

R_dmax l' <= o' -l < R_dmin l'

Il élabore, à partir d'une variable L= ($\mathcal{L}$, V.l , l) deux variables S = ($\mathcal{S}$, V.s , s) et O = ($\mathcal{O}$, V.o , o) tels que:

V.s(0) = Sinit

V.s * (I+1) = W.s * D.s * (I+1) = f (W.l * D.l' , W.s * D.s)

W.o * D.o' = ge (W.l * D.l' , W.s * D.s) pour les automates de MEALY

W.o * D.o' = go (W.s * D.s) pour les automates de MOORE

COMMENTAIRE

Cette défnition englobe les réalisations asynchrones dans ce cas, l = l', et toute variation dans l'histoire de l'entrée L provoque une variation dans l'histoire de l'état S, et les réalisations synchrones dans ce cas s est l'horloge et l peut etre différent de l'. Tout changement d'état est alors précédé d'une observation l' de l'entrée; mais certaines variations des entrées peuvent avoir été perdues.

TROISIEME PARTIE

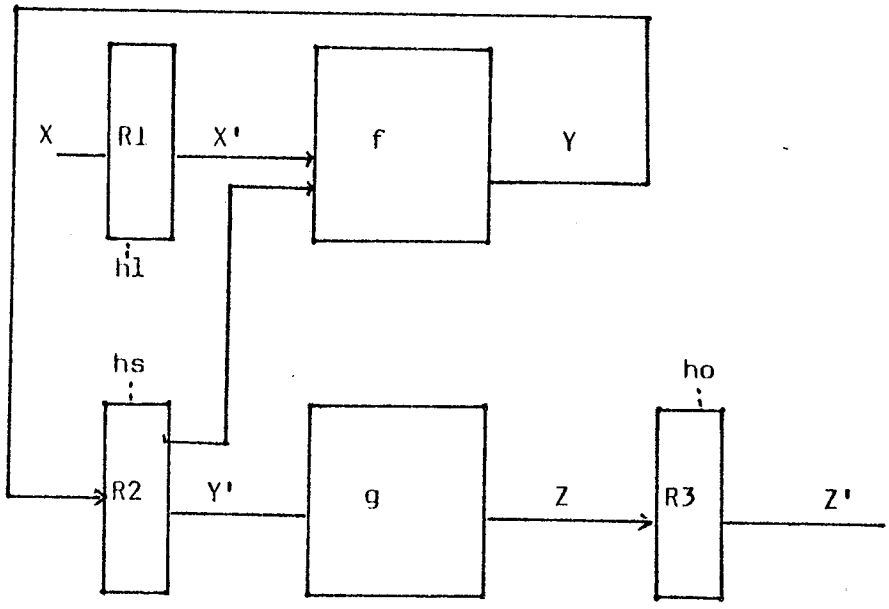
DIFFERENTES REALISATIONS

3.1 REALISATION A ENTREES SORTIES ECHANTILLONNEES A BASE DE REGISTRES A FRONT

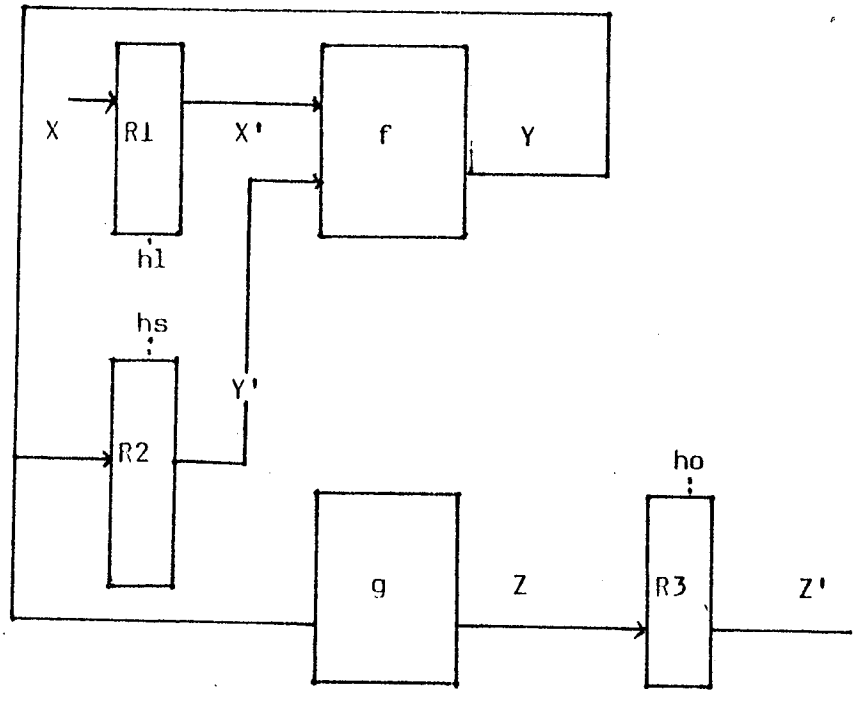
3.1.1 INTRODUCTION

On présente habituellement deux façons de réaliser des automates à base de registres sensibles au front d'horloge, avec échantillonnage des entrées et des sorties.
Ces deux réalisations sont le plus souvent décrites par leur schéma fonctionnel, comme ci-dessous, où R1, R2, R3 désignent trois registres à front, h1, h2, h3 les trois horloges qui les commandent.

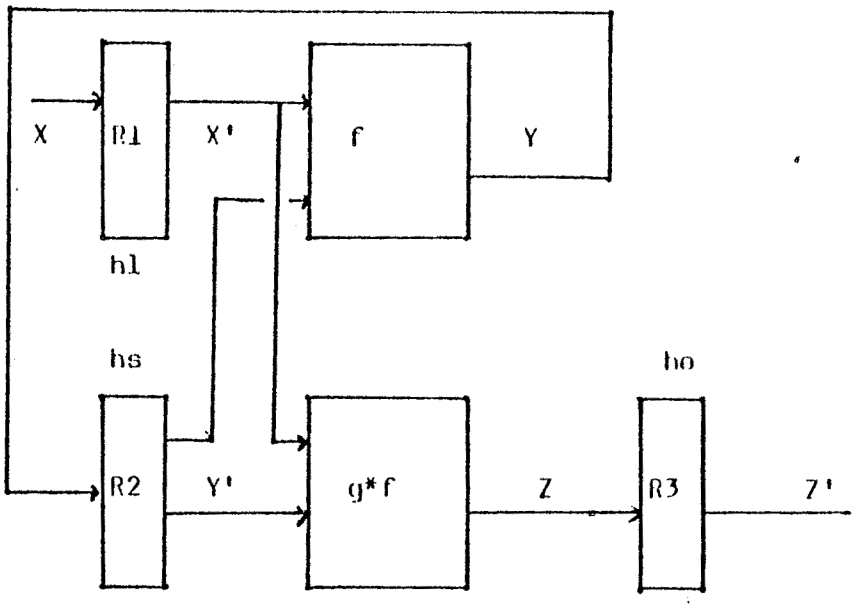
RFALISATION 1



RFALISATION 2



La réalisation 2 peut aussi etre décrite par le schéma fonctionnel suivant :



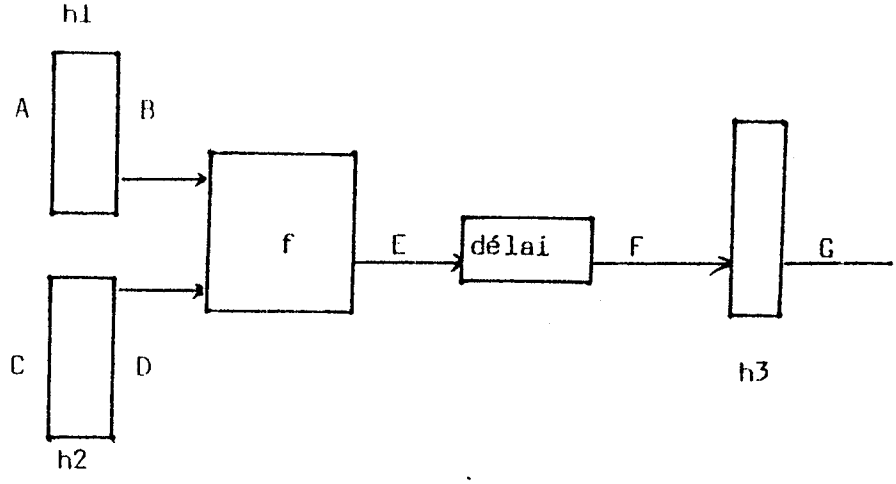
où l'on a fait apparaitre la composée de g et f au lieu de séparer ces deux fonctions. Dans le paragraphe 2.2.5, nous avons proposé une définition d'automate temporisé "abstrait ". Nous nous proposons de montrer ici

* sous quelles hypothèses temporelles on sait donner le comportement de la réalisation 1 et de la réalisation 2 ,
* à quel comportement cela conduit,
* que ce comportement est conforme à la définition d'un automate abstrait,
* que, sous certaines hypothéses concernant h1, h2, h3, on sait calculer les délais dmin et dmax qui figurent dans la définition d'un automate.

Nous nous appuierons pour cela sur les résultats donnés dans le paragraphe EXEMPLE 4 où était décrite un dispositif constitué d'une partie combinatoire entre des étages de registres à front.

3.1.2 CARACTERISATION DE LA REALISATION 1

On rappelle que dans l'exemple 4 , en 2.1.8, pour une réalisation décrite par le schéma suivant:



les hypothèses temporelles

Sup (h1,h2) -1 <= h3 < R_d Inf (h1,h2)

conduisaient à

V.g = f (V.b , V.d)

3.1.2.1 FONCTION DE TRANSITION

Si, dans les réalisations 1 et 2 on pose

hs - 1 = h2 - 1 = h3 et h1 = h1 ,

les hypothèses temporelles deviennent :

Sup (h1,hs) -1 <= hs -1 < R_d Inf (h1, hs)

où d est le délai maximal de calcul de la fonction de transition.

Remarquant que cette hypothèse est vérifiée si hl <= hs

En effet

Sup (hl , hs) -1 <= hs -1

⇑

Sup (hl , hs) <= hs

⇕

Sup (hl , hs) = hs

⇕

hl <= hs

on choisira comme hypothèse de fonctionnement hl <= hs , ce qui conduit à

hs - 1 < R_d hl < hl <= hs

Par la simple identification des variables entre la réalisation et le dispositif de l'exemple 4,

V.x' = V.b

V.q = V.y' * (I+1) = V.d * (I+1) ,

on a

V.y' * (I+1) = f (V.x', V.y')

3.1.2.2 FONCTION DE SORTIE

On distinguera ici les deux réalisations 1 et 2.

Réalisation 1

On appelle d1 le délai maximal de calcul de g. On est ramené à une partie co mbinatoire entre deux registres :

Sous condition que

hs - 1 <= ho < R_d1 hs

on a :

V.z' = g (V.y')

Réalisation 2

On appelle d2 le délai maximal de calcul de g*f . Il faut remarquer qu'on sait construire un circuit combinatoire qui calcule g*f autrement qu'en cascadant un circuit f et un circuit g et d2 est certainement inférieur à d + d1.

On est de nouveau dans une configuration simple , analogue à celle de l'exemple 4, et, si

Sup (hl , hs) -1 <= ho < R_d2 Inf (hl , hs)

C'est à dire, puisque hl <= hs

hs - 1 <= ho < R_d2 hl

on a

V.z' = g*f (V.x' , V.y')

c'est à dire

V.z' = g (f (V.x' , V.y')

= g (V.y' * (I+1))

3.1.2.3 TABLE DE RESULTATS

X désignant à chaque fois la variable d'entrée, on peut résumer les caractéristiques de ces deux réalisations par le tableau suivant.

A V.x' = W.x * D.hl

B z' = ho

Réalisation 1	Réalisation 2
hs-1 < R_dhl < hl <= hs	hs-1 < R_dhl < hl <= hs
hs-1 <= ho < R_dlhs	hs-1 < ho < R_d2hl
C1 V.y'*(I+1) = f(V.x',V.y')	C2 V.y'*(I+1) = f(V.x',V.y')
D1 V.z' = g(V.y')	D2 V.z' = g(V.y'*(I+1))

3.1.3 CONFORMITE A LA DEFINITION D'UN AUTOMATE

Il s'agit d'établir la correspondance entre les définitions d'automate de Moore ou de Mealy et les deux réalisations décrites.

Dans tous les cas on identifie :

T1 * la fonction combinatoire f de la réalisation et la fonction de transition f qui figure dans la définition.

T2 * la variable X de la réalisation et la variable I d'entrée.

3.1.3.1 CORRESPONDANCE REALISATION 1 AUTOMATE DE MOORE

	definition		realisation
E1		l' =	hl
E2		S =	Y'
E3		O =	Z'
E4		o' =	ho
E5		go =	g

on a alors

```
W.s * D.s * (I+1) = W.y' * D.y' * (I+1)          d'après E2
                  = V.y' * (I+1)                 définition de V
                  = f ( V.x' ,V.y' )             d'après C1
                  = f ( W.x * D.hl , V.y' )      d'après A
                  = f ( W.l * D.hl , V.y' )      d'après T2
                  = f ( W.l * D.l' , V.y' )      d'après E1
```

soit, d'après E2

```
W.s * D.s * (I+1) = f ( W.l * D.l' , W.s * D.s )
```

```
W.o * D.o' = W.z' * D.o'                         d'après E3
           = W.z' * D.ho                         d'après E4
           = W.z' * D.z'                         d'après B
           = V.z'                                definition de V
           = g ( V.y' )                          d'après D1
           = g ( V.s )                           d'après E2
           = go ( V.s )                          d'après E5
```

soit, par définition de V

```
W.o * D.o' = go ( W.s * D.s )
```

3.1.3.2 REALISATION 1 AUTOMATE DE MEALY

On a, comme dans l'identification avec l'automate de MOORE,

```
E1          l' = hl
E2          S  = Y'
```

donc W.s * D.s * (I+1) = f (W.l * D.l' , W.s * D.s)
comme dans la correspondance avec l'automate de Moore

```
E3           O = Z'
E'4         o' = ho - 1
E'5         ge = g*f
```

On a alors :

```
W.o * D.o' = W.o * D.(h0-1)                    d'après F'4
           = W.z' * D.(ho-1)                   d'après E3
           = W.z' * D.(z'-1)                   d'après B
           = W.z' * D.z' * (I+1)
           = V.z' * (I+1)
           = q ( V.y' * (I+1) )                d'après D1
           = q*f ( V.x' , V.y' )               d'après C1
           = qe  ( V.x' , V.y' )               d'après E'5
```

C'est à dire, la fin du calcul étant identique à celui fait pour l'automate de Moore,

W.o * D.o' = qe (W.1 * D.1' , W.s * D.s)

L'identification o' = ho-1 trouve sa justification dans le fait que la première sortie, qui apparait à la premiere occurence de ho, peut n'etre pas significative puisqu'aucune hypothèse n'est faite concernant l'ordre entre ho et hl. Il est clair qu'on ne peut décrire par une machine de Mealy une réalisation qui délivre une sortie avant d'avoir reçu la première entrée. Ceci pouvait, bien sur etre acceptable dans une machine de Moore, la première sortie correspondant à l'état initial.

3.1.3.3 REALISATION 2 AUTOMATE DE MEALY

Comme dans les deux identifications précédentes :

```
E1      1' = hl
E2      S  = Y'
E3      0  = Z'
et, ici,
E''4    o' = ho
E''5    qe = q*f
```

On a alors

```
W.o * D.o' = W.o * D.ho                        d'après E''4
           = W.o * D.z'                        d'après B
           = W.z' * D.z'                       d'après E3
```

```
            = V.z'
            = q ( V.y' *(I+1) )                    d'après D2
            = g*f ( V.x' , V.y' )                  d'après C
            = ge ( V.x' , V.y' )                   d'après E''5
```

La suite du calcul est identique aux précédents et conduit à :

W.o * D.o' = ge (W.1 * D.1' , W.s * D.s)

3.1.3.4 REALISATION 2 AUTOMATE DE MOORE

Ici encore

```
E1      1' = h1
E2      S  = Y'
E3      0  = Z'
mais

E'''4    o' - 1 = ho
E'''5   go = q
```

On obtient :

```
W.o * D.o' * (I+1) = W.o * D.(o'-1)
                   = W.o * D.ho                    d'après E'''4
                   = W.z' * D.z'                   d'après E3 et B
                   = V.z'
                   = q ( V.y'  * (I+1) )           d'après D
```

et, d'après E2 et E5
W.o * D.o' * (I+1) = go (W.s * D.s * (I+1))

On aimerait alors conclure que
W.o * D.o' = go (W.s * D.s)
Il n'en est rien car les hypothèses temporelles
hs - 1 < ho <= hs
permettent, compte tenu de E'''4 , d'écrire
s-1 < o'-1 <= s
mais ne permettent pas de comparer o' et s.

La première sortie, correspondant à l'état initial n'est pas significative si elle est émise avant que l'état initial ne soit forcé.

Les deux remarques restrictives faites en établissant les

correspondances

Réalisation 1 automate de Mealy

Réalisation 2 automate de Moore

ont amené beaucoup d'auteurs à adopté le vocabulaire, à notre avis facheux :

Réalisation 1 = réalisation de Moore

Réalisation 2 = réalisation de Mealy

Il faut noter, enfin, que ces restrictions sont inhérentes à l' " équivalence " entre automate de Moore et automate de Mealy. Meme dans <BENZ> où un point de vue totalement algébrique est pris, sans aucune considération de temps, la non-équivalence machine de Moore machine de Mealy comme reconnaisseurs de langages dans un monoide est mentionnée pour le cas de la chaine vide.

Cela correspond bien aux remarques ci-dessus concernant la " synchronisation " entre la première entrée et la première sortie.

3.1.4 EXPRESSION DES DELAIS

Dans les quatre cas considérés, on supposera l'horloge hl périodique et de période T. Ceci facilite grandement l'expression de dmax qui , sans cette hypothèse dépend du plus grand délai qui sépare deux occurences de hl.

On a donc hl - 1 = R_T hl

3.1.4.1 REALISATION 1 AUTOMATE DE MOORE

On a
hs - 1 < R_hl < hl <= hs
hs - 1 <= ho < R_dl hs
l' = hl
o' = ho
et l'on cherche dmin et dmax tels que
R_dmax l' <= o' -l <= R_dmin l'

hs - 2 <= o' - 1 < R_dl hs - 1
R_dl hs - 1 < hs - 1
hs - 1 < R_d hl

d'où
o' - 1 < R_(dl+d) l'
et dmin = dl+d
hs - 1 < o'
hs - 2 < o' - 1
l' <= hs
l' - 2 <= hs - 2 < o' - 1
R_2T l' <= o' - 1
et dmax = 2T

3.1.4.2 REALISATION 1 AUTOMATE DE MEALY

On a
hs - 1 < R_d hl < hl <= hs
hs - 1 <= ho < R_dl hs
l' = hl
o' = ho - 1
On veut dmin et dmax tels que :
R_dmax l' <= o' - 1 <= R_dmin l'

hs - 2 <= ho - 1 < R_dl hs - 1
hs - 2 <= o' < R_dl hs - 1
hs - 3 <= o' - 1 < R_dl hs - 2
hs - 2 < R_d hl - 1

d'où :
o' - 1 <= R_(dl+d) hl - 1
<= R_(dl+d+T) l'
et dmin = dl + d + T
de plus,
hl - 3 <= hs - 3 <= o' - 1
d'où
dmax = 3T

3.1.4.3 REALISATION 2 AUTOMATE DE MOORE

on a
hs - 1 < R_d hl < hl <= hs
hs - 1 < ho < R_d2 hl
l' = hl
o' - 1 = ho
On veut calculer dmin et dmax pour que
R_dmax l' <= o' - 1 <= R_dmin l'

l'-1 <= hs - 1 <= o' - 1 <= R_d2 l'
d'où
dmax = T
dmin = d2

3.1.4.4 REALISATION 2 AUTOMATE DE MEALY

hs - 1 < R_d hl < hl <= hs
hs - 1 < ho < R_d2 hl
l' = hl
o' = ho
On veut calculer dmin et dmax pour que
R_dmax l' <= o' - 1 <= R_dmin l'

ho < R_d2 hl - 1
o' <= R_(d2+T) l'
d'où
dmin = T+d2
hl - 2 <= hs - 2 < ho - 1
ou
R_2T l' <= o' - 1
et dmax = 2T

3.2 QUELQUES VARIANTES A LA REALISATION 1

3.2.1 INTRODUCTION

Les hypothèses temporelles que nous avons faites dans la description de la réalisation type numéro 1 sont très restrictives. Nous montrerons ici quelques variantes reposant sur des hypothèses plus faibles.

En nécessitant, comme ce fut le cas, que hl <= hs, on choisit implicitement une réalisation dans laquelle l'état initial de l'automate est indépendant des entrées de cet automate. On est là très proche des modèles abstraits d' automate reconnaisseur de langage où existe un état initial préalable à " l'apparition de la première entrée ".

Une solution matérielle de réalisation d'automate, que nous allons détailler, procède de façon non conforme à ce principe. Dans cette solution, on s'" arrange " pour que la première valeur d'entrée soit la valeur FORCAGE. La partie combinatoire qui calcule l'état suivant est alors telle que f(FORCAGE, w) soit égal à l'état initial pour toute valeur de w.

La valeur de w est réellement indéterminée puisqu'elle résulte de la seule mise sous tension du circuit. On est alors incapable de dire quelle valeur est affichée en sortie d'une bascule fabriquée par rétrocouplage de deux portes NON-OU ou NON-ET.

Si cette valeur FORCAGE est échantillonnée, on a bien, dans ce

cas, hs < hl et cette possibilité doit etre prise en considération.
Ce sera l'objet de la première distinction entre deux cas de réalisation.

De la meme façon, en nécessitant ho < R_d hs , on s'interdit la possibilité d'utiliser une seule horloge pour un échantillonnage simultané de l'état et de la sortie, c'est à dire hs = ho. Cette facilité est pourtant largement utilisée meme si elle produit au début quelque(s) sortie(s) sans signification. L'introduction de cette possibilité amènera une deuxième distinction entre deux cas de réalisation.

Ces deux modifications dans la synchronisation
entrée/état
état/sortie
étant indépendantes, ce sont quatre cas que nous allons considérer, c'est à dire :

	cas A	cas B
cas a	hl <= hs et ho < R_d hs	hl <= hs et hs <= ho
cas b	hs < R_d hl et ho < R_d hs	hs < R_d hl et hs <= ho

On ne s'attend, bien entendu, pas à ce que ces modifications possibles n' entrainent une modification entre les parties signifiantes des variables d'entrée et de sortie (Valeurs FORCAGE et indeterminée mises à part.) On montrera donc l'équivalence des quatre réalisations sous le point de vue du délai entre entrée et sortie qui lui correspond. On se limitera dans cette partie à des machines décrites selon le modèle de Moore.

3.2.2 ALTERNATIVE DANS LA SYNCHRONISATION ENTREE / ETAT

Dans la construction précédemment envisagée, on avait :

hl - 1 <= hs - 1 < R_d hl < hl <= hs et

V.s (n+1) = f (V.l (n+1) , V.s (n+1))

Introduisons la variable ENTREE, dérivée de la variable L définie par

Un événement h'l tel que

h'l - 1 = hl

Une suite de valeurs telles que
V.entrée (1) = FORCAGE
V.entrée (2) = V.l (1)
. .
. .
. .
V.entrée (n+1) = V.l (n)

On a alors

h'l - 2 <= hs - 1 < R_d h'l - 1 < h'l - 1 <= hs

C'est à dire
h'l-1 <= hs < R_d h'l
et

V.s (1) = f (FORCAGE , indeterminé)
V.s (n+1) = f (V.entrée (n+1) , V.s (n))

On obtient donc deux cas de réalisation , le cas a construit précédemment et le cas b que l'on vient de construire.

cas a : hs - 1 < R_d hl < hl <= hs
conduisant à
V.s (n + 1) = f (V.l (n) , V.s (n))
et nécessitant une initialisation de l'état par
V.s (1) = Sinit

cas b : h'1 - 1 <= hs < R_d h'1
conduisant à
V.s (n + 1) = f (V.entrée (n + 1) , V.s (n))
où l' initialisation est faite par
V.entrée (1) = FORCAGE

On peut remarquer que l'on retrouve là l'analogue de la distinction faite dans <AISE> entre automate Passé-passé et automate Passé-présent. L'automate ,défini par suite de valeurs, peut selon les auteurs etre défini sous la forme passé-passé

s n+1 = f (l n , s n)

ou sous la forme passé-présent

s n+1 = f (l n+1 , s n)

3.2.3 ALTERNATIVE DANS LA SYNCHRONISATION ETAT / SORTIE

On avait, dans la construction type :

hs - 1 <= ho < R_d' hs

conduisant à

V.o (n) = g (V.s (n))

Introduisons, comme précédemment une variable SORTIE, dérivée de O définie par

L'événement h'o de sortie caractérisé par

h'o -1 = ho

La suite de valeurs donnée par
V.sortie (1) = indéterminé
V.sortie (n+1)= V.o (n)

On a alors

hs - 1 <= h'o - 1 < R_d' hs < hs <= h'o

et V.sortie (n+1) g (V.s (n))

Comme pour l'alternative de synchronsation état / entrée , on obtient ici deux cas :

cas A : hs - 1 <= ho < R_d' hs
et, alors
V.o (n) = g (V.s (n))

cas B :
h'O - 1 < R_d' hs < hs < h'o
et
V.sortie (1) = indéterminé
V.sortie (n+1) = g (V.s (n))

3.2.4 RECAPITULATIF ET COMPARAISON DES QUATRE SOLUTIONS

CALCUL DES DELAIS MINIMAUX ET MAXIMAUX DES DIFFERENTES SOLUTIONS

On souhaite ici resituer ces différentes réalisations par rapport à la définition donnée en 2.2.5 d'un automate avec temporisation explicite. En particulier, on calculera pour ces quatres variantes les délais minimaux et maximaux et l'on montrera qu'ils sont égaux.

Ceci suppose d'établir les correspondances nécéssaires entre

variables observées sur les réalisations matérielles et les variables données dans la définition. On prendra tout naturellement l'hypothèse que la fonction f qui calcule matériellement l'état suivant est bien identique à la fonction f qui figure dans la définition. De meme pour la fonction de sortie g.

Afin de mettre en évidence les différences entre les organisations temporelles envisagées et les conséquences qu'elles ont sur le fonctionnement, on notera ici, dans tous les cas :

A la variable d'entrée dans la définition,
B la variable d'état dans la définition,
C la variable de sortie dans la définition,
L la variable d'entrée observée sur la réalisation,
S la variable d'état observée,
O la variable de sortie observée.

Pour pouvoir comparer les réalisations entre elles, on choisira toujours d'établir la correspondance avec les variables A B C de telle façon que

V.c (n+1) = g*f (V.a (n) , V.b (n))

Le détail des calculs conduisant à l'évaluation de dmin et de dmax pour chacune des quatre solutions est présenté sous l'hypothèse commune d'une périodicité de l'horloge hl , la période T étant dans tous les cas supposée supérieure strictement au délai d . Il est clair qu'on retiendra chaque fois

* comme délai minimal dmin la plus grande valeur possible
* comme délai maximal dmax la plus petite valeur possible

On ne retiendra de plus comme délai que d, maximum des délais de calcul de f ou de g.

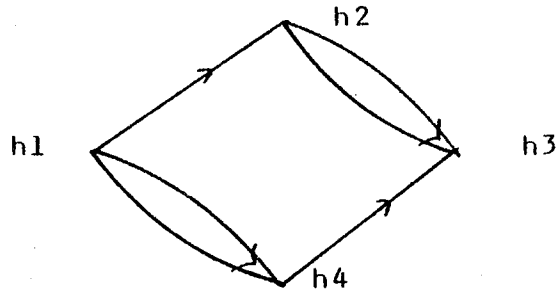
Les hypothèses de synchronisation entre hl, hs et ho seront représentées graphiquement avec les conventions suivantes :

Pour la lisibilité du dessin, au lieu de représenter la relation d'ordre entre les événements, on ne représente que la relation de précédence qui en est déduite par

x précède y <=> x < y et il n'existe pas d'autre élément différent de x et de y compris entre x et y.

Le sens gauche-droite est privilégié sur le dessin, pour figurer l'infériorité

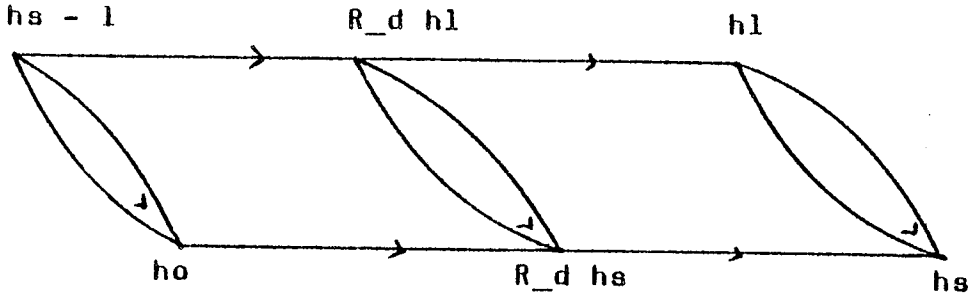
Une relation stricte x < y est figurée par un trait simple,
Une relation x <= y est figurée par un trait double,
EXEMPLE
Les relations
hl < h2 <= h3
et
hl <= h4 < h3
sont figurées par



Pour simplifier l'écriture des équations, on notera dans cette partie sn (resp. ln+1) au lieu de V.s (n) (resp. V.l (n)) le n-ième état (resp. la n+1-ième entrée).

REALISATION aA

Hypothèses temporelles



Equations de fonctionnement

$$s_{n+1} = f(l_n, s_n)$$
$$o_n = g(s_n)$$

soit,

$$o_{n+1} = g*f(l_n, s_n)$$

Les hypothèses temporelles donnent

$$h_o < R\_d\ h_s$$
$$h_s - 1 < R\_d\ h_l$$
$$h_s - 1 \leq h_o$$
$$h_l - 1 \leq h_s - 1$$
$$h_l - 2 \leq h_s - 2 \leq h_o - 1 < R\_d\ h_s - 1 < R\_2d\ h_l$$

Soit, en posant $h_l - 2 = R\_2T\ h_l$:

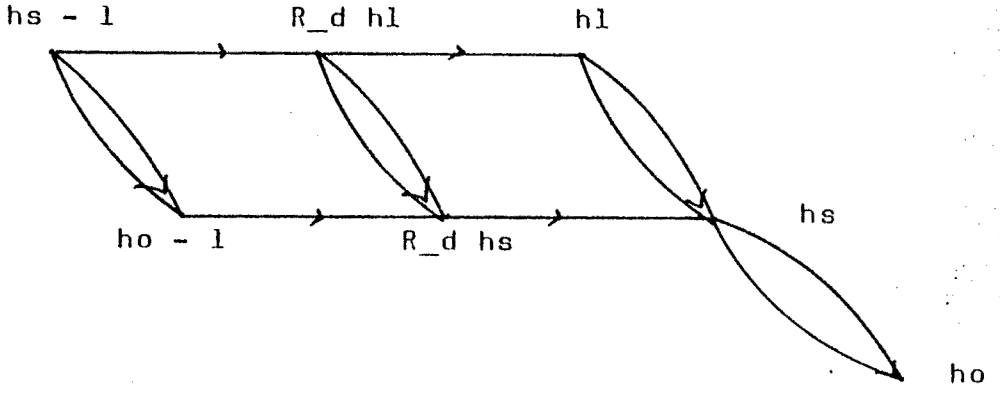
$$R\_2T\ h_l \leq h_o - 1 < R\_2d\ h_l$$

La correspondance avec la définition d'automate s'établit alors simplement par

A = L
B = S
C = 0
dmin = 2d
dmax = 2T

REALISATION aB

Hypothèses temporelles



Equations de fonctionnement
sl = sinit sn+1 = f (sn , ln)
ol = indeterminé on+1 = g (sn)
soit
on+2 = g*f (ln, sn)

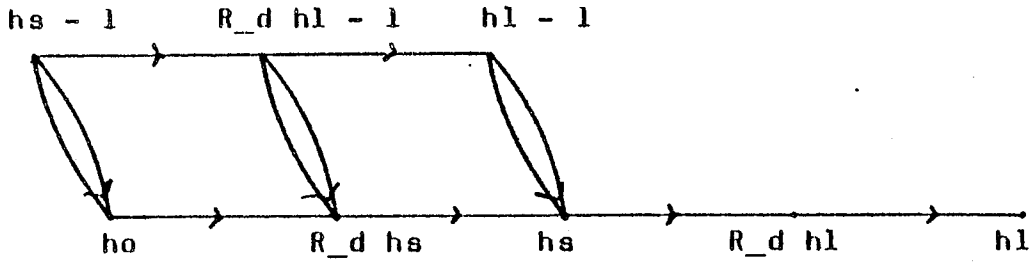
Les hypothèses temporelles donnent

hs <= h0
hl <= hs
ho - 1 < R_d hs
R_d hs < hs
hs - 1 < R_d hl
hl - 2 <= hs - 2 <= ho - 2 < R_d hs -1 < R_2d hl
soit,
R_2T hl <= ho - 2 < R_2d hl

La correspondance avec la définition s'établit alors par
A = L
B = S
C est obtenue à partir de 0 en posant
c = ho - 1
V.c (n) = V.o (n+ 1)
On a bien alors
V.c (n+1) = g*f (V.a (n) , V.b (n))
L'inéquation
R_2T hl <= ho - 2 < R_2d hl
devient alors R_2T a <= c - 1 < R_2d a
et : dmin = 2d et dmax = 2T .

REALISATION bA

Hypothèses temporelles



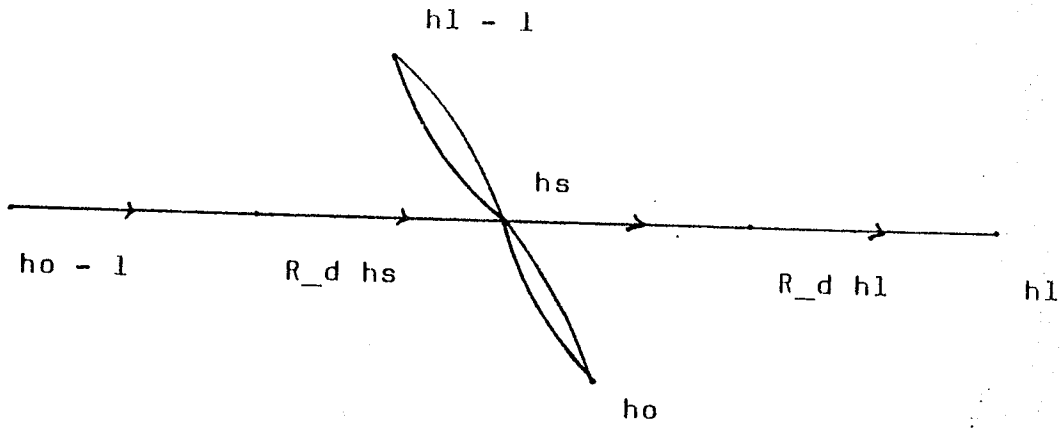
Equations de fonctionnement
sn+1 = f (ln+1 , sn)
f (FORCAGE , ineterminé) = sinit = s1
on = g (sn)
soit
on+1 = g*f (ln+1 , sn)

Les hypothèses temporelles donnent
hl - 2 <= hs - 1 <= ho < R_d hs < hs < R_d hl
soit,
R_2T hl <= ho < R_2d hl

La correspondance avec la définition s'établit en posant
B = S
C = 0
A s'obtient à partir de 1 en posant
a = hl - 1
V.a (n) = V.1 (n + 1)

On a bien alors
V.c (n+1) = g*f (V.a (n) , V.b (n))
L'inéquation
R_2T hl <= ho < R_2d hl
devenant alors
R_2T a <= c - 1 < R_2d a
et dmin = 2d
dmax = 2T.

REALISATION bB
Hypothèses temporelles



Equations de fonctionnement
sn+l = f (ln+l , sn)
f (FORCAGE , indeterminé) = sinit = sl
ol = indeterminé
on+l = g (sn)
soit
on+2 = g*f (ln+l , sn)
Les hypothèses temporelles donnent
hl - 2 <= hs -l <= ho -l < R_d hs < hs < R_d hl
ou
R_2T hl <= ho - l < R_2d hl

La correspondance avec les variables de la définition s'établit alors par
B = S
A est obtenu à partir de L en posant
a = hl - l
V.a (n) = V.l (n+l) = ln+l
C est obtenue à partir de O par
c = ho - l
V.c (n) = V.o (n+l) = on+l

On a bien alors V.c (n+l) = g*f (V.a(n) , V.b (n))
et
R_2T a <= c-l < R_2d a
d'où
dmin = 2d
dmax = 2T

3.3 APPLICATION DU MODELE DE DESCRIPTION AUX ARCHITECTURES BASEES SUR UNE LOGIQUE DYNAMIQUE

3.3.1 DESCRIPTION INFORMELLE

Les différentes réalisations étudiées précédemment sont toutes dans le cadre des architectures à base de logique statique, c'est à dire où le point de mémorisation de base est obtenu par rétrocouplage de deux portes NON ET ou NON OU.

Nous avons supposé de plus que ces points de mémorisation étaient sensibles au front d'horloge (Front montant ou descendant ou les deux à la fois comme dans <UNGE>). Toutefois certains des résultats établis précédemment peuvent etre étendus au logiques dites à niveau, quite à considérer le niveau commandant le chargement comme instantané et à le considérer comme une occurence de l'horloge.

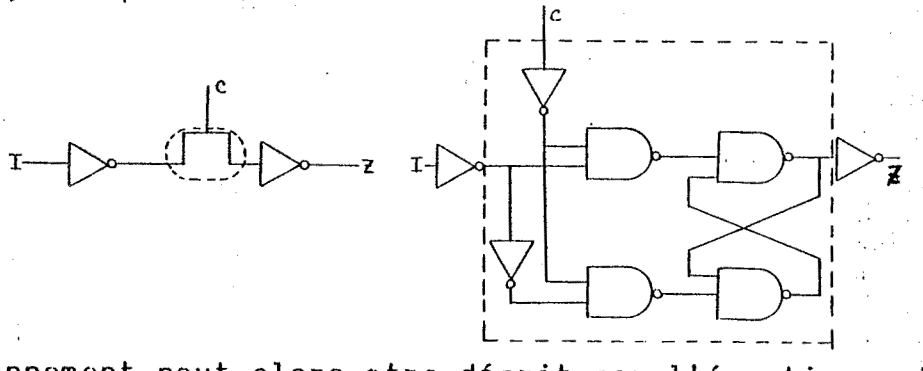
Cette simplification n'est plus possible dans les logiques dynamiques classiquement utilisées dans les technologies M.O.S. C'est dans ce cas la capacité de grille d'un transistor qui sert de mémoire, fugitive, et cette capacité est chargée ou déchargée à travers un transistor utilisé comme interrupteur. La " sortie " du point de mémorisation est donc disponible pendant que le point prend sa valeur.

Le fonctionnement détaillé d'un tel dispositif de mémorisation est décrit dans <CARR>, <MEAD> ou <MURO>.

On étendra donc ici les possibilités de description en passant de l'architecture à un niveau qui tient à la fois de l'architecture et de l'électricité, niveau appelé " Clocked register and logic level " dans <STEF>.

Une telle tentative de description formelle d'architecture dynamique a déjà été faite dans <GORD>, spécialement pour une technologie N-M.O.S. enrichi déplété . L'approche y est très électrique puisqu'on y distingue transistors de charge et transistors de signal.

Une proposition d'une autre façon de modéliser de tels systèmes a été faite dans <PRAS> . L'équivalence des deux structures suivantes y est posée :



Le fonctionnement peut alors etre décrit par l'équation suivante, où t désigne un instant:

Si C(t) = 1 alors Z(t) := I(t) sinon Z(t) := Z(t- Δ t)

Cette équation pose le problème de la signification de l'expression Δ t.

On peut seulement en dire que Δ t est "petit".

De plus elle néglige, sans que cela ne pose de problèmes, la

disparition possible d'information par courant de fuite.

Nous prendrons ici le meme parti de négliger ce phénomène pour deux raisons :

* Il est très complexe à manipuler car les temps de maintien garanti ne sont pas connus avec précision

* Il est possible de s'affranchir de ce problème par l'utilisaation d'un rafraichissement systématique si on le souhaite.

Rappelons aussi les deux modes de fonctionnement d'une porte de transmission évoqués dans <OKAZ> où l'on distingue :

* le mode synchrone où l' on garantit, par construction, que l'entrée I(t) est stable pendant que l'horloge C est haute.

* Le mode asynchrone où cette hypothèse n'est pas faite.

Dans le mode synchrone, la description du fonctionnement du point de mémorisation peut se faire en considérant TOUT le niveau haut comme une occurence d'un événement fictif appelé l'horloge.

On écrirait alors que la valeur mémorisée, et affichée en sortie, est la valeur de l'entrée à la date de cette occurence de l'horloge. Peu importe que cette date ne corresponde pas à une date physique mais à une époque, puisque pendant cette époque rien ne change.

Nous choisirons ici de nous intéresser au mode d'utilisation asynchrone, plus général a priori.

Le point de mémorisation sera assimilé à un dispositif de poursuite-mémoire instantané, ou verrou qui:

* laisse passer l'entrée vers la sortie quand l'horloge est

haute,

* conserve la valeur qu'avait l'entrée la dernière fois que l'horloge est tombée et continue à afficher cette valeur tant que l'horloge est basse.

3.3.2 FORMALISATION DE LA DESCRIPTION

Le modèle par événements, conçu pour décrire des systèmes discrets permet de décrire des variables dont on connait l'événement de changement de valeur. Dans une réalisation fondée sur des points de mémorisation dynamique, l'évolution des variables ne nous intéresse pas totalement.

Habituellement :
Si A est une variable, V.a(n) désigne la n-ième valeur connue de A.
W.a(t) désigne la valeur de A à l'instant t .
D.a est l'événement des prises de valeurs certaines de A.
W.a(t) = V.a * C.a (t) .

Nous utiliserons ici les variables dans une définition plus restreinte qui est la suivante :

W.a(t) désigne la valeur de A à l'instant t .

Pour simplifier l'écriture, nous noterons a au lieu de W.a .

On pourra combiner les variables entre elles, et poser, par exemple

c = b + a

pour

qqst t apra TT c(t) = b(t) + a(t)

Si l'on ne connait que les dates de variation de b, et pas celles de a , on ne peut naturellement déduire celle de c. La notation

c = a + b garde toutefois tout son sens.

Il reste bien sur possible de décrire la suite d'observations que l'on peut faire d'une variable à des dates précisées. Ainsi la variable a, observée aux dates d'occurence d'un événement h donne la suite de valeur

a * D.h .

Mais cette suite de valeurs n'est peut-etre pas la suite de toutes les valeurs prises par a au cours de son évolution.

On utilisera la notation

x = c$y + c`$z

pour décrire la variable définie par

qqst t apra TT x(t) vaut y(t) si la condition c(t) est vraie
et vaut z(t) si la condition c(t) est fausse

3.3.2.1 DESCRIPTION DE CONSTITUANTS

REGISTRE DYNAMIQUE

Si h désigne l'horloge qui commande l'interrupteur d'accès au point de mémorisation, hl les fronts descendants de cette horloge,

x l'entrée du dispositif et y sa sortie :

y = h$x + h`$(x*L.hl)

DELAI

Un delai maximal d sera décrit par une application r de TT dans

TT telle que :

qqst t apra TT t-d <= r(t) <= t

Une variable y, résultant d'une variable x retardée d'au maximum d est alors donnée par :

y = x * r
On pourra traduire ici qu'aucune valeur d'entrée ne peut rattrapper la précédente et sortir avant elle en exigeant que r soit croissante.

CIRCUIT COMBINATOIRE

Un circuit combinatoire réalisant la fonction f sur des entrées x et y, et délivrant une sortie z en un délai maximal d sera décrit par :

ilex r: TT ⟶ TT : I-d <= r <= I
ilex r': TT ⟶ TT : I-d <= r' <= I
tels que
z = f (x*r , y*r')

On simplifiera ici en considérant

z = f (x*r , y*r) = f (x, y) * r

On se propose de démontrer ici que le procédé de construction d'un automate présenté dans <MEAD> fournit bien un automate. En fait il s'agira surtout de montrer sous quelles hypothèses temporelles de fonctionnement ceci est vrai.

3.3.2.2 DESCRIPTION DE LA REALISATION COMPLETE

La réalisation d'un automate d'état fini, présentée dans <MEAD> est constituée

* de deux registres R1 et R2,
* d'un circuit combinatoire entre R1 et R2,
* d'un "chemin de retour " de R2 vers R1

Comme dans la description donnée en référence, on ne précisera pas ici la façon dont se fait l'intialisation du dispositif.

On nommera

t la variable d'entrée avant échantillonnage dans R1

u la variable d'entrée après échantillonnage dans R1

x la variable de sortie avant échantillonnage dans R2

y la variable de sortie après échantillonnage dans R2

m,n,p,q, la variable état en quatre points différents :

m à l'entrée du registre R1,

n entre R1 et le circuit combinatoire

p à l'entrée du registre R2,

q entre R2 et le chemin de retour

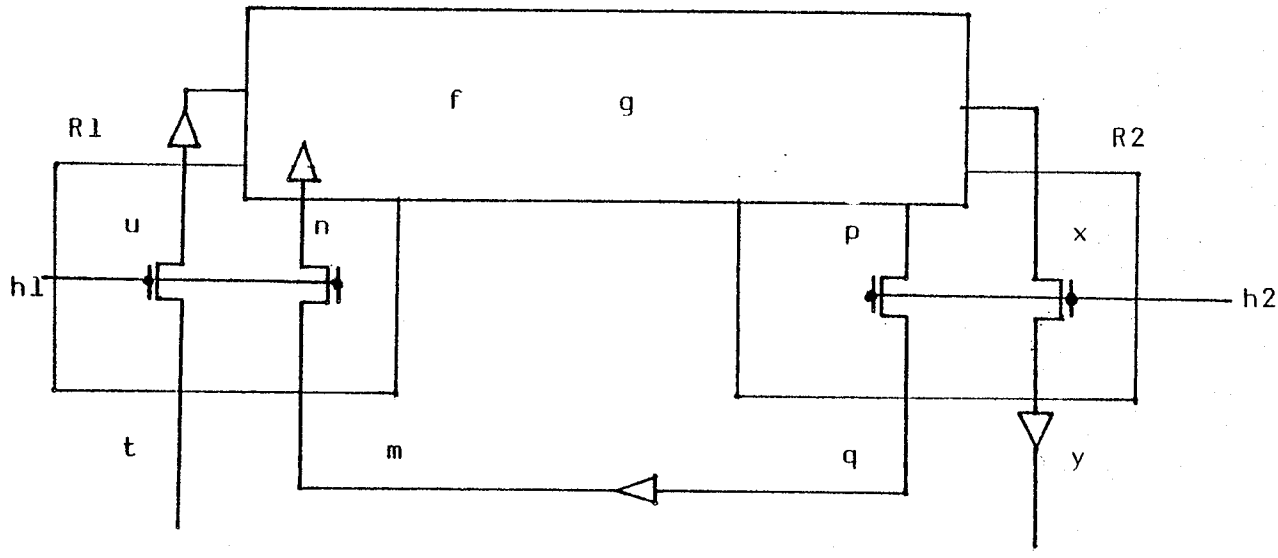
h1 et h2 les horloges respectives de R1 et R2

d le délai maximal du chemin de retour

d' le délai maximal du circuit combinatoire.

Remarquons que l'expression " entre le registre R1 et le circuit combinatoire " résulte d'une certaine abstraction par rapport au crcuit puisque, dans le circuit réel, la grille du transistor de

signal d'une porte de premier niveau dans le ciruit combinatoire fait partie du circuit combinatoire et tient lieu de registre.



Le fonctionnement de l'ensemble de la réalisation est donc décrit par les équations suivantes :

1	ilex r I-d <= r <= I	m = q * r
2		n = h1 $ m + h1` $ (m * L.h1!)
3		u = h1 $ t + h1` $ (t * L.h1!)
4	ilex r' I-d'<=r'<= I	p = f (n * r' , u * r')
5	ilex r'' I-d'<=r''<=I	x = g (n * r'', u * r'')
6		q = h2 $ p + h2` $ (p * L.h2!)
7		y = h2 $ x + h2` $ (x * L.h2!)

La spécification de l'automate à réaliser reste la meme que précédemment. On considèrera ici comme variable d'entrée la variable t observée aux fronts descendants de h1 , comme variable d'état la variable m observée aux fronts descendants de h1, et

comme variable de sortie la variable x observée aux fronts descendants de h2.
La spécification du comportement souhaité pour la réalisation décrite est alors

m * D.h1! * (I+1) = f (m * D.h1! , t * D.h1!)
x * L.h2! * D.h1! = g (m * L.h1! , t * L.h1!) * L.h2! * D.h1!

Cette spécification n'est bien entendue réalisable que sous certaines hypo thèses temporelles . Ces hypothèses sont dans la réalité complexes car étroitement liées à des phénomènes électriques . On n'en considère ici qu'une version simplifiée.
En particulier, le principe meme de fonctionnement du point de mémorisation dynamique suppose que les niveaux hauts de l'horloge sont assez longs pour permettre les charges et décharges des capacités des grilles qui servent de point de mémorisation.
Cette hypothèse est ici supposée vraie, faute de quoi, c'est le fonctionnement meme du registre qui n'est plus garanti.
De plus nous supposerons que les niveaux bas sont suffisamment courts pour qu'il n'y ait pas perte d'information.

Ces deux hypothèses sont pré-requises pour que l'équation caractéristique de fonctionnement du point de mémorisation
y = h $ x + h` $ (x * L.h!)
ait un sens.

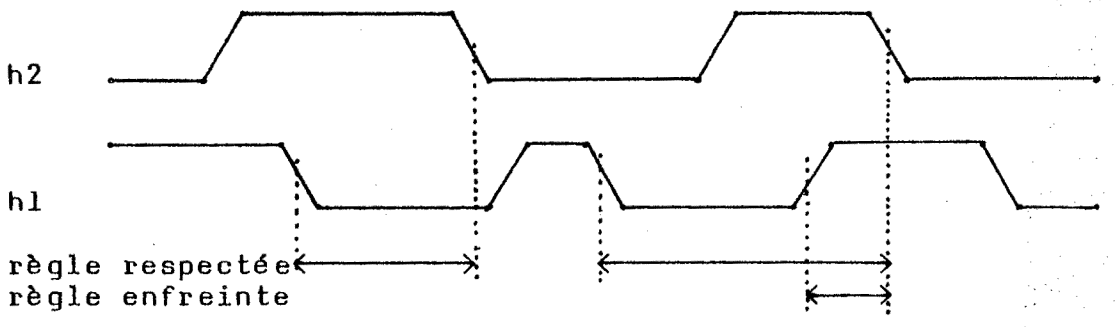
3.3.2.3 HYPOTHESES TEMPORELLES

Les deux autres hypothèses que nous allons discuter sont par contre nécessaires au fonctionnement de la réalisation automate mais pas du point de mémorisation.
Ce sont les deux hypothèses de non recouvrement des deux horloges h1 et h2 et de bon entrelacement de ces deux horloges.

HYPOTHESES DE NON CHEVAUCHEMENT

HY1 et HY1bis

Durant tout intervalle de longueur inférieure ou égale à d', précédant un front descendant de h2, h1 vaut 0.



Cette hypothèse s'écrit

HY1

pour toute fonction a de TT vers TT telle que
I - d' <= a <= I , on a :
h1 * a * D.h2! = 0
ce qui conduit à
L.h1! * a * D.h2! = L.h1! * D.h2!

L'hypothèse "duale" est alors

HY1bis

Pour toute fonction b de TT vers TT telle que
I - d <= b <= I
h2 * b * D.h1! = 0
ou

L.h2! * b * D.h1! = L.h2! * D.h1!

HYPOTHESES DE BON ENTRELACEMENT

Ces hypothèses traduisent la bonne alternance h1 h2 h1 h2 h1...

En particulier, il ne doit pas y avoir deux fronts descendants de l'un entre deux fronts descendants de l'autre.

Cela se traduira par :

HY2

L.h1! * D.h1! * (I+1) = L.h1! * D.h2! * D.h1! * (I+1)
= D.h1!

et l'hypothèse duale

L.h2! * D.h2! * (I+1) = L.h2! * D.h1! * D.h2! * (I+1)
= D.h2!

3.3.3 DEMONSTRATION

La demonstration que la réalisation décrite a bien le comportement escompté se ramène à énoncer :

avec I-d <= r <= I d'après 1

m * D.h1! * (I+1) = q * r * D.h1! * (I+1)

= (h2$p + h2`$(p * L.h2!)) * r * D.h1! * (I+1) d'après 6

= (h2 * r * D.h1! * (I+1))$(p * r * D.h1! * (I+1))

+ (h2` * r * D.h1! * (I+1))$(p * L.h2! * r * D.h1! * (I+1))

= p * L.h2! * r * D.h1! * (I+1) d'après HY1

```
      = p * L.h2! * D.h1! * (I+1)          d'après HY1bis

      = p * L.h2! * D.h1! * (I+1)          d'après HY1bis


      = f(n*r',u*r') * L.h2! * D.h1! * (I+1)  avec I-d'<=r'<=I d'après 4

      = f ( n * r' * L.h2! * D.h1! * (I+1) ,
             u * r' * L.h2! * D.h1! * (I+1) )

      = f ( ( h1$m + h1`$( m * L.h1! )) * r' * L.h2! * D.h1! * (I+1) ,
       ( h1$t + h1`$( t * L.h1! )) * r' * L.h2! * D.h1! * (I+1))
```

soit, d'après HY1

```
m * D.h1! * (I+1) = f ( m * L.h1! * r' * L.h2! * D.h1! * (I+1) ,
                        t * L.h1! * r' * L.h2! * D.h1! * (I+1) )
```

ou

```
m * D.h1! * (I+1) = f ( m * L.h1! * L.h2! * D.h1! * (I+1) ,
                        t * L.h1! * L.h2! * D.h1! * (I+1) )
```

HY2 permet alors de conclure

```
m * D.h1! * (I+1) = f ( m * D.h1! , t * D.h1! )
```

De manière analogue :

```
x * D.h2! = g ( n * r'' , u * r'' ) * D.h2!
          = g ( n * r'' * D.h2! , u * r'' * D.h2! )
          = g ( ( h1$m + h1`$( m * L.h1! ) ) * r'' * D.h2! ,
                ( h1$t + h1`$( t * L.h1! ) ) * r'' * D.h2! )
          = g ( m * L.h1! * D.h2! , t * L.h1! * D.h2! )
```

Il vient alors

x * D.h2! * C.h2! * D.h1! * (I+1) =

g (m * L.h1! * L.h2! * D.h1! * (I+1) ,

t * L.h1! * L.h2! * D.h1! * (I+1))

d'où

x * L.h2! * D.h1! * (I+1) = g (m * D.h1! , t * D.h1!)

Les deux expressions obtenues pour cette réalisation :

m * D.h1! * (I+1) = f (m * D.h1! , t * D.h1!)

x * L.h2! * D.h1! * (I+1) = g (m * D.h1! , t * D.h1!)

sont bien conformes à la spécification générale d'automate donnée précédemment.

W.s * D.s * (I+1) = f (W.1 * D.1' , W.s * D.s)

W.o * D.o' = ge (W.1 * D.1' , W.s * D.s)

On a ici

W.s = m

D.s = D.h1! = D.1'

f = f

W.1 = t

D.o' = L.h2! * D.h1! * (I+1)

ge = g

3.3.4 CALCUL DES DELAIS MINIMAL ET MAXIMAL

On trouve ici Dmin et Dmax pour que

R_Dmax l' <= o' <= R_Dmin l'

Sachant que

D.o' = D.h2! * C.h2! * D.h1! * (I+1)

= D.h2! * C.h2! * l' * (I+1)

de par HY2, on a

L.l' * D.o' = L.h1! * D.o'

= L.h1! * L.h2! * D.h1! * (I+1)

= D.h1!

L.l' * D.o' = D.l'

soit, si l' est périodique, de période T

Dmax = T

Par ailleurs, d'après HY1

L.l' * a * D.o' = L.h1! * a * L.h2! * D.h1! * (I+1)

= L.h1! * L.h2! * D.h1! * (I+1)

= L.l' * D.o'

et ceci pour tout a tel que

I - d' <= a <= I

On a donc Dmin = d'

Voulant, par conformité à la définition que

R_dmax l' <= o' - 1 <= R_dmin l'

il suffit de remarquer que, si l' est périodique de période T,

R_T l' = l' - l

On a alors

R_Dmax l' - l <= o' - l <= R_Dmin l' - l

R_(T+Dmax) l' <= o' - l <= R_(T+Dmin) l'

d'où

dmax = T+Dmax = 2T

dmin = T+Dmin = T+d'

QUATRIEME PARTIE

REALISATION DE MACHINES COMPLEXES SYNCHRONES

4.1 INTRODUCTION

La construction de machines synchrones complexes peut se faire à partir d'une description du fonctionnement de cette machine sous forme d'automate d'état fini. Il se trouve que cette méthode n'est, en général, pas utilisée pour les machines très complexes. Il y a à cela plusieurs raisons :

La principale est sans doute que dans ce cas le nombre d'états est tellement grand que :

* Les méthodes classiques de passage automatique de la description du fonctionnement par automate d'état fini à la description des plans du circuit donnent de "mauvais" résultats. (Les résultats sont justes mais un critère d'évaluation des résultats tel que la surface du circuit est très mal pris en compte).

* Le concepteur risque de se tromper en établissant la description des fonctions f ou g de l'automate. Il y a là un problème de maitrise de la complexité qui n'est pas propre à la

technique de conception de circuits intégrés mais que l'on ne peut pour autant négliger.

Une autre raison importante est la mauvaise adéquation d'une description stricte par automate pour certaines parties de circuit. Nous entendons par stricte description par automate une description faisant apparaitre tous les états successifs.

Ainsi, par exemple, un "automate" constitué de n registres de k éléments binaires chacun et d'un opérateur arithmétique a, à priori, 2 ** n.k états.

L'extension du "chemin de données" à 2k éléments binaires par registre donnerait un automate à 2 ** 2n.k états La modification semble considérable alors qu'elle est très facile a réaliser , il suffit pour s'en convaincre de regarder les chemins de données de circuits de type micro-processeur.

Ces raisons ont conduit les concepteurs à décrire les machines synchrones complexes comme étant constituées de plusieurs automates, reliés entre eux et synchronisés.

Les liaisons entre les automates sont assurées par le fait que les entrées (sorties) des uns sont les sorties (entrées) des autres.

La synchronisation est assurée

* soit par la répartition des références temporelles dans l'ensemble du circuit

* soit par des " synchroniseurs " spécialisés (arbitre , élément de MULLER , ...)

* soit par des échanges de protocoles de communication entre les

différents automates.

Les automates pourront etre, selon les cas,

* très nombreux et tous différents,

* très nombreux et tous semblables,

* peu nombreux .....

Nous étudierons ici quelques façons de relier deux réalisations synchrones d'automate et de les synchroniser.

Nous distinguerons, à priori , trois cas de couplage de deux automates suivant la façon dont se fait l'échantillonnage des entrées de l'un par rapport a l'échantillonnage des sorties de l'autre .

D'abord, on peut faire se succéder ces deux échantillonnages ce sera le premier cas considéré, ou bien ces deux échantillonnages peuvent etre identiques, ce sera le deuxième cas, dans le troisième cas, enfin, on montrera que ces échantillonnages sont parfois supprimés.

Nous prendrons dans l'organisation d'une machine synchrone complexe constituée de deux automates reliés les conventions suivantes :

Le premier automate a comme ensembles

d'entrées L1 X A

de sorties O1 X B

Le deuxième automate a comme ensembles

d'entrées L2 X B

de sorties O2 X A

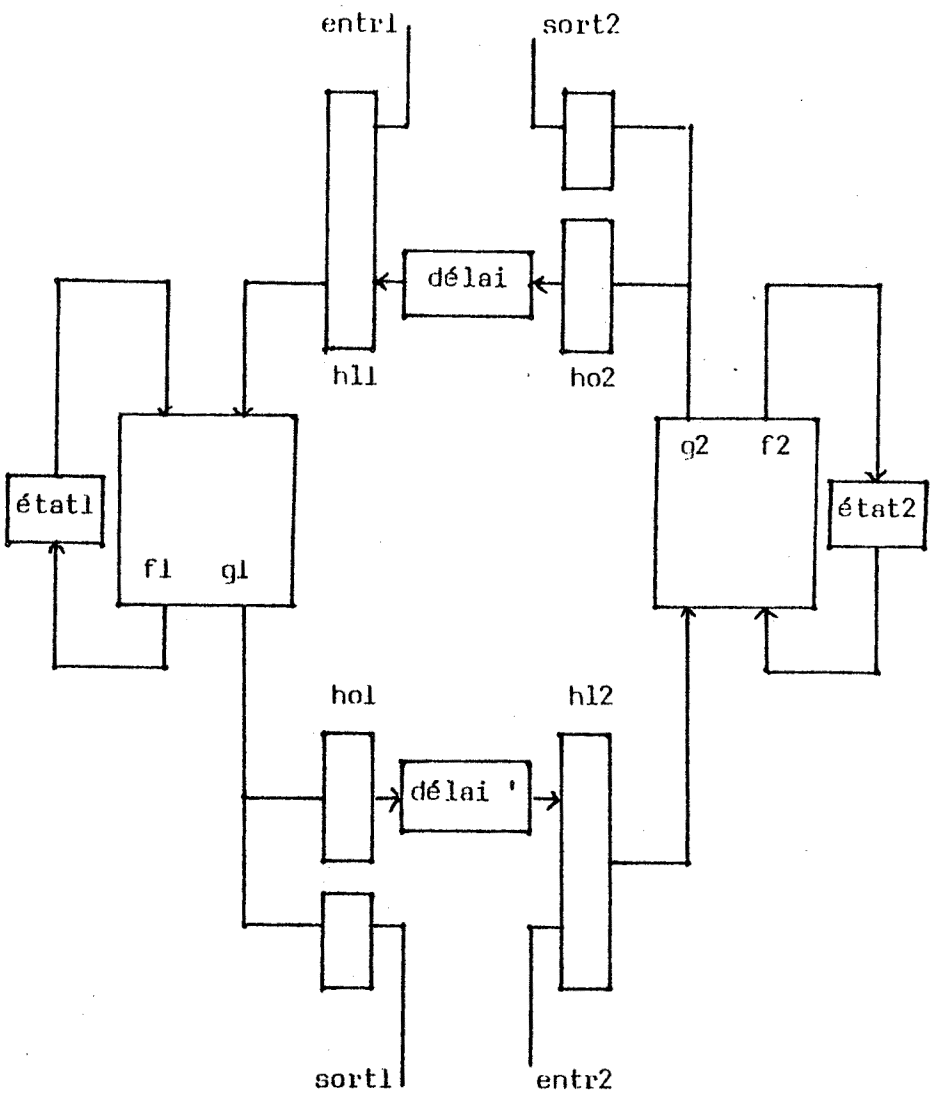
Les composantes dans A et B sont les entrées/sorties de liaisons entre les deux automates, les composantes dans L1, L2, O1, O2 sont les entrées ou sorties "propres" à chacun des automates.
Un de ces ensembles peut ne pas exister. On l'assimilera alors à un singleton ce qui permettra, par exemple, de confondre L2 X B et B si le deuxième automate n'a pas d'entrées propres.
Les fonctions état suivants et sorties seront naturellement notées f1,f2,g1,g2.

4.2 COUPLAGE AVEC ECHANTILLONNAGES SUCCESSIFS

Dans cette organisation, un échantillonnage d'une sortie est suivi d'un échantillonnage de cette même valeur comme entrée par l'autre automate. On distinguera donc deux organisations d'une réalisation de machines complexes avec ce double échantillonnage. Dans l'une, on exige que les séquences entrées de l'un sorties de l'autre soient identiques. Nous donnerons les conditions temporelles pour que cette condition soit ssatisfaite.

Dans l'autre, cette exigence d'identité des séquences n'existe pas . Nous montrerons un exemple de machine où, cette exigence étant absente, les conditions temporelles sont beaucoup moins dures.

Dans les deux cas, le schéma de principe est toutefois le meme :



Si l'on note d (respectivement d') le délai maximal du à la transmission de transfert entre les sorties du deuxième (resp. du premier) et les entrées du premier (resp. du deuxième) les conditions d'égalité des séquences sont alors naturellement

ho1 - 1 <= h12 < R_d' ho1

et

ho2 - 1 <= h11 < R_d ho2

Ces contraintes s'ajoutent à celles caractérisant le fonctionnement de chacun des deux automates.

Remarquons par ailleurs que pour respecter les conditions initiales, il faut peut-etre décaler d'un coup l'une des horloges et poser par exemple :

ho2 - 1 <= h11 - 1 < R_d ho2

Nous reviendrons plus loin en détail sur un tel décalage.
Il existe des cas, très particuliers, où de telles contraintes temporelles n'existent pas. Nous en présenterons un exemple, dont l'intéret est que l'ensemble fonctionne correctement meme si les horloges des deux automates ne sont pas synchronisées.

MACHINE PROTOCOLE OU MACHINE A TRANSMETTRE

(D'après <FUJI>)

Cette machine est constituée de deux automates synchrones & entrées sorties échantillonnées.

Le premier, appelé émetteur, a comme ensembles :

d'entrées { 0,1 } X { 0,1 }
dont les deux composantes sont appelées
demande (entrée propre)
écoutearr (sortie du deuxième)

d'état { CN , CY }

de sorties { 0,1 }
la variable de sortie sera appelée appeldep
C'est une entrée du deuxième automate,
Il n'y a pas de sortie propre.

Il a comme délais dmin1 et dmax1

Ses fonctions sont données, sous formes de Mealy par la table suivante :

(demande, ecoutearr) / état	(0, 0)	(0, 1)	(1, 1)	(1, 0)
CN	CN 0	CN 0	CN 0	CY 1
CY	CY 1	CN 0	CN 0	CY 1

Le deuxième automate, appelé récepteur, a comme ensembles :

d'entrées { 0, 1 } la variable est nommée appelarr
C'est une sortie du premier automate
Il n'y a pas d'entrée propre.

d'état { HN , HY }

de sorties { 0, 1 } X { 0, 1 } dont les composantes sont nommées
ecoutedep, c'est une entrée du premier
laissepasser c'est une sortie propre

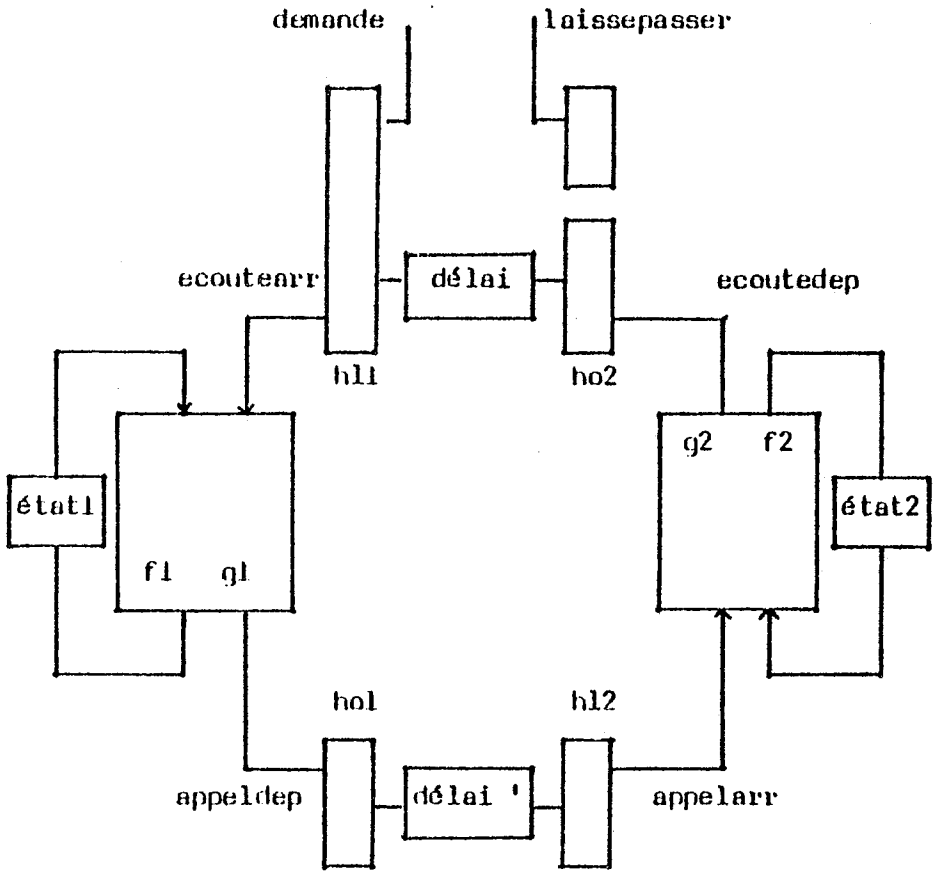
Il a comme délais dmin2 et dmax2

Ses fonctions f2 et g2 sont données sous forme de Mealy par la table suivante :

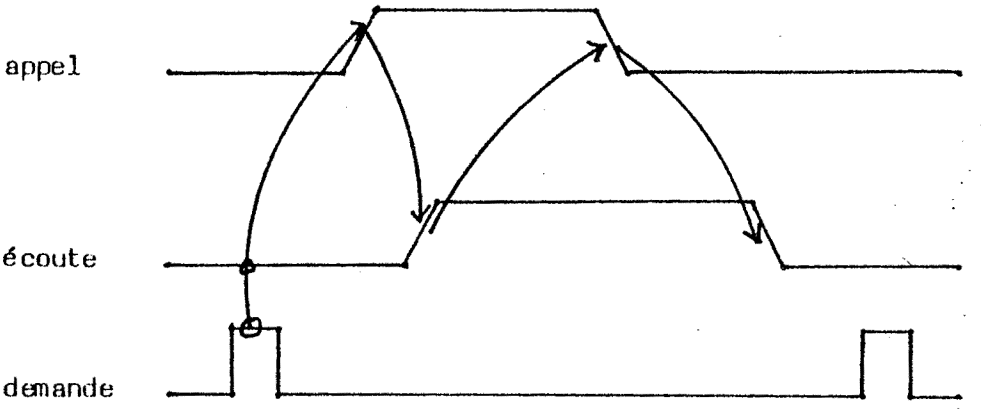
appelarr / état	0		⊥		
HN	HN	0	0	HY	⊥	⊥
HY	HN	0	0	HY	⊥	0

sorties : (ecoutedep,laissepasser)

Les états initiaux des deux circuits sont CN pour le premier et HN pour le second.



Ces deux automates sont les deux automates utilisés en bout de ligne pour permettre une émission de message selon le protocole, classique, de "request-acknowledge" connu par le diagramme suivant :



L'intéret de ce que les horloges de l'émetteur et du récepteur puissent etre indépendantes est que l'émetteur et le récepteur peuvent etre éloignés et n'avoir aucune référence temporelle commune.

4.3 COUPLAGE AVEC ECHANTILLONNAGES SIMULTANES

Dans cette organisation, l'échantillonnage d'une sortie est simultané avec l'échantillonnage d'une entrée de l'autre automate. Cette simultanéité est physiquement assurée par la simple identité de registre de sortie de l'un des automates qui sert aussi de registre d'entrée de l'autre.

On posera donc

$$L2 = O1$$

soit, plus précisement ;

$$h12 = ho1$$

$$V.12 = V.o1$$

Cette convention est arbitraire; on aurait tout aussi bien pu choisir $L1 = O2$. On choisira, pour garder une certaine souplesse dans l'expression des dates d'observation de poser que L1 soit la variable O2 décalée d'un certain nombre, à priori inconnu, de valeurs et donc de dates.

En effet, si l'on peut poser soit l'égalité $L2 = O1$, soit $L1 = O2$, il apparait que de poser $L2 = O1$ ET $L1 = O2$ est certainement une contrainte forte.

On posera donc :

Soit p , entier, le " nombre de coups " de décalage entre L1 et O2.

On a alors

V.11(1), V.11(2), ... V.11(p) à determiner,

V.l1(p+1) = V.o2(1)

et, en général,

V.l1 * (I+p) = V.o2 Les p premières valeurs de L1 ne sont pas nécéssairement les sorties du deuxiéme automate,

de meme, temporellement :

D.h11(1) , D.h11(2), ... D.h11(p) à déterminer,

D.h11(p+1) = D.ho2(1)

et, en général,

D.h11 * (I+p) = D.ho2

On a alors h11 - p = ho2.

On montrera que la détermination des valeurs possibles pour p permet d'en choisir une. Cette valeur de p étant choisie, on en déduit :

* Une période minimale des horloges du système considéré,

* Un " décalage fonctionnel " de q coups entre états d'un automate.

Ce décalage fonctionnel peut etre défini ainsi :

Un état d'un automate dépend

* de ses états prédecesseurs (en fait du dernier)

* de l'influence de ses états prédécesseurs sur l'autre automate.

Ainsi, pour n assez grand,

V.s1(n) = f1(g2 (f2 (g1 (V.s1 (n-q)))))

Dans l'application classique d'un automate de controle et d'une partie opérative, cela exprime le nombre d'états entre une action commandée par la partie controle et la prise en compte possible

du résultat de cette action dans un compte-rendu.

La détermination des valeurs possibles pour p se fait au vu des caractéristiques de synchronisation de chacune des deux réalisations d'automates utilisées.

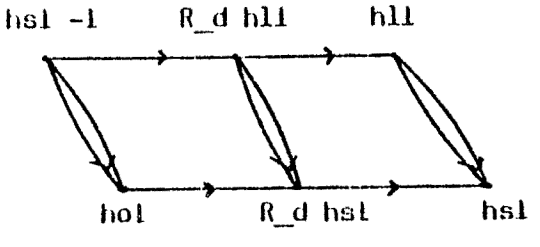
Dans la partie précédente, quatre schémas de synchronisation ont été présentés pour la réalisation 1 et un schéma pour la réalisation 2 . Cela donne à priori vingt cinq cas possibles de connexions de deux automates. Nous n'en étudierons ici deux pour montrer la méthode utilisée dans la détermination de p et dans celle qui en découle de la période minimale d'horloge et du décalage fonctionnel.

Comme précédemment, nous figurerons les relations d'ordre entre les différentes horloges d'une réalisation de façon graphique .

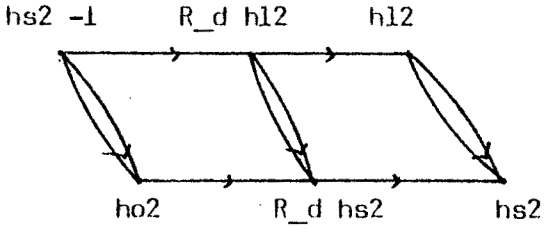
ETUDE DETAILLEE D'UN EXEMPLE

Automate 1 : Réalisation aA :
Automate 2 : Réalisation aA :

Automate 1

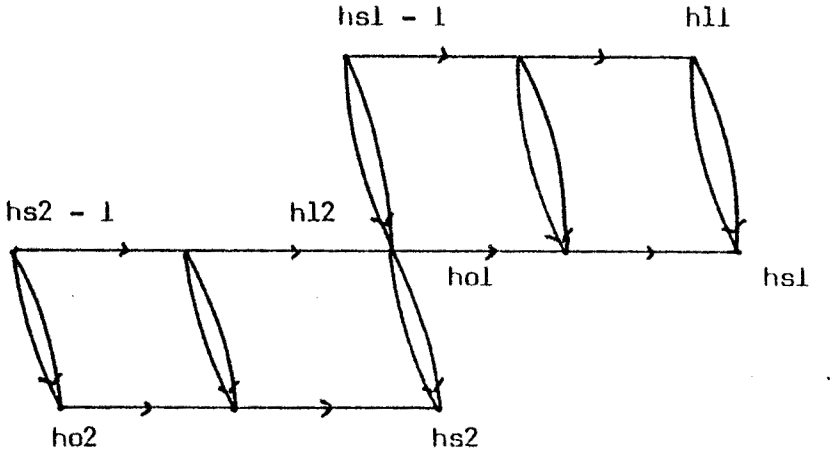


Automate 2



Pour la connexion des deux h12 = ho1

L'information sur les rapports de synchronisation entre les horloges lorsqu'on réunit les deux réalisations est alors décrite par le schéma suivant :



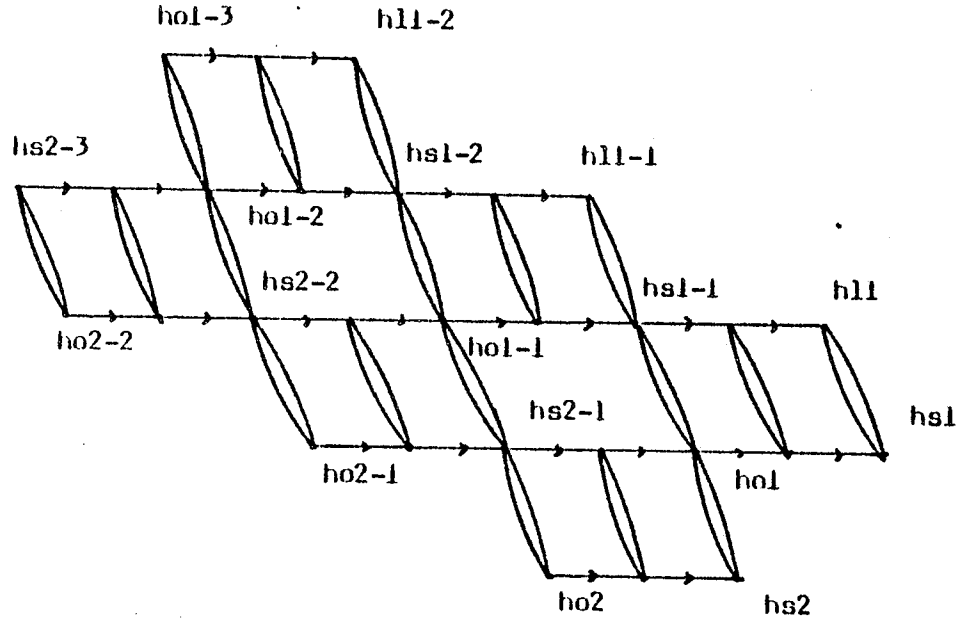
Détermination des valeurs possibles pour p

Le schéma précédent ne donne aucune information sur les valeurs possibles de p telles que h11 - p = ho2. En effet, aucune comparaison n'est faite entre h11 et ho2. Cette information est fournie en remarquant que les relations d'ordre sont conservées

par suppression des premiéres occurences.

Si e < f , e-1 < f-1

Si e <= f, e-1 <= f-1 On a alors, en appliquant plusieurs fois cette propriété, le schéma de synchronisation exprimé par le graphique suivant :



Les deux relations suivantes y apparaissent ;

ho2 - 2 < hl1

hl1 - 2 <= ho2

Comme on cherche p tel que

(I-p) * hl1 = ho2 c'est à dire hl1-p = ho2

L'ensemble des valeurs possibles de p est fourni par

hl1 - p - 2 < hl1 soit p > -2

hl1 - 2 <= hl1 - p soit p <= 2

On a donc p qui est une des quatre valeurs -1, 0, 1, 2.

Avant d'examiner deux de ces quatre solutions, une remarque s'impose à propos de la première solution:

p = -1.

On cherchait implicitement une valeur positive de p en posant

D.hl1(p+1) = D.ho2 (1)

Cela correspondait physiquement à faire démarrer ho2 après hl1. Trouvant p = -1 , c'est à dire hl1 +1 = ho2, on a en ait hl1 qui démarre après ho2 puisque ho2 = hl1-1 (L'écriture hl1 + 1 = ho2 n'ayant pas de véritable signification).

Le choix d'une valeur de p revient alors à choisir un canevas de synchronisation entre les horloges intervenant dans les deux réalisations.

Exemple de choix

p = 2

En posant p = 2, c'est à dire ho2 = hl1 - 2 , on a nécéssairement

ho2 = hl1 - 2 = hs1 - 2 = hl2 - 1 = ho1 - 1 = hs2 - 1 = ho2

(Ceci apparait, sur le schéma de synchronisation d'ensemble des deux automates comme un écrasement sur le dessin représentant le treillis des différentes horloges.)

Détermination des périodes d'horloge

On cherche à calculer les périodes minimales de chacune des horloges intervenant dans la réalisation (si une telle période existe).

Comme dans le cas p = 2 les six horloges sont de meme caractéristiques, on ne calcule que la période de hl1

L'inégalité

hl1 -1 = hs1 -1 < R_d hl1 suffit à déterminer que la période minimale de hl1 est d.

Il est facile de vérififer que cette période convient aussi pour les cinq autres horloges.

Détermination du décalage fonctionnel

Pour deux réalisations aA d'automate, et pour n assez grand, les valeurs de la variable d'état sont données par

V.s1 (n) = f1 (V.l1 (n-1), V.s1 (n-1))
V.o1 (n) = g1 (V.s1 (n))
V.s2 (n) = f2 (V.12 (n-1) , V.s2 (n-1))
V.o2 (n) = g2 (V.s2 (n))

de plus

V.12 = V.o1

et, puisque p = 2,

V.l1 (n) = V.o2 (n-2) Il vient donc, naturellement :

V.s1(n) = f1 (V.l1 (n-1) , V.s1 (n-1))
= f1 (V.o2 (n-3) , ***
= f1 (g2 (V.s2 (n-3) , ***
= f1 (g2 (f2 (V.12 (n-4) , ***
= f1 (g2 (f2 (V.o1 (n-4) , ***
= f1 (g2 (f2 (g1 (V.s1 (n-4), ***

(où l'on remplace ce qui n'est pas l'état par ***)

On obtient donc ici un décalage fonctionnel de 4.

Par contre, le choix de p = 0 conduit à hl1 = ho2 .

On obtient dans ce cas le schéma de synchronisation :

hs1 - 1 = hs2 - 1 , qu'on peut noter hs - 1

h11 = ho1 = h12 = ho2 qu'on peut noter he

Comme hs1 - 1 < R_d h11

et ho2 < R_d hs2 ,

On a ici

hs - 1 < R_d he < R_2d hs

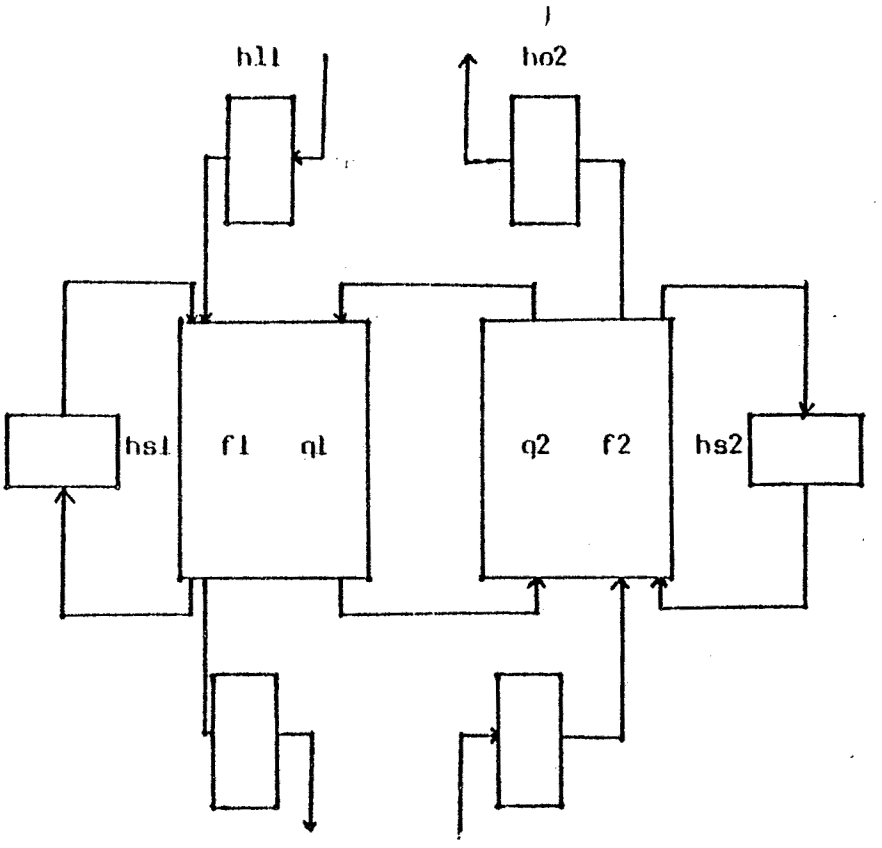
et hs est de période minimale 2d.

Un calcul de décalage fonctionnel exactement analogue au précédent donne ici q = 2.

4.4 COUPLAGE SANS ECHANTILLONNAGE INTERMEDIAIRE

On a justifié l'introduction d'un échantillonnage des entrées d'un automate par des considérations de stabilité. De meme, on a considéré des réalisations à sorties échantillonnées. Il est toutefois généralement admis que si les entrées d'un automate 1 sont les sorties d'un automate 2, la stabilité des entrées de l'automate 1 est garantie par la stabilité de l'état de l'automate 2.

Le schéma de principe devient alors :



L'unanimité semble etre faite concernant les synchronisations dans ce cas : <AUMI>, <MEIN>, <PARK>, <MANG>, <BERG> par exemple notent l'existence de deux horloges : l'une pour hs1 et l'autre pour hs2 ; ce sont les fronts montants et descendants d'une meme horloge qui matérialisent ces deux horloges pour des points de mémorisation sensibles au front .

CINQUIEME PARTIE

EXEMPLES

Dans cette partie, on montre sur quelques exemples simples :

* soit différentes réalisations d'un meme automate decrit sous forme de Moore,

* soit deux descriptions (sous forme de Moore et de Mealy) d'une meme réalisation.

Les réalisations sont décrites au moyen de schémas architecturaux classiques sans conventions de dessin ou de notation particulières.

5.1 DIFFERENTES REALISATIONS D'UN AUTOMATE DECRIT SOUS FORME DE MOORE

5.1.1 SOLUTION A P.L.A.

L'exemple retenu ici comme point de départ est l'automate de controle de feux donné dans ⟨MEAD⟩. Il est décrit sous forme d'automate de Mealy. Pour avoir un automate simple décrit sous forme de Moore, on supprime ici la sortie ST.

La description de l'automate est alors la suivante :

L'ensemble d'entrées est $\{0,1\} \times \{0,1\} \times \{0,1\}$

L'ensemble d'états est $\{Hg, Hy, Fg, Fy\}$

L'ensemble de sorties est $\{(G,R) , (Y,R) , (R,G) , (R,Y)\}$

Dans l'ensemble des entrées les parties " intéressantes " suivantes sont considérées :

CETTL = $\{(x,y,z) \ / \ x = 1 , y = 1\}$

TS = $\{(x,y,z) \ / \ z = 1\}$

CBOUTL= $\{(x,y,z) \ / \ x = 1 , y = 0\}$

Cela correspond, si l'on appelle c, tl, ts , dans cet ordre, les trois composantes du vecteur booleen d'entrée, aux parties qui rendent vraies respectivement

c ET tl

ts

NON c OU tl

La fonction f, de transition est définie par :

si l apra CETTL f (l , Hg) = Hy

sinon f (l , Hg) = Hg

si l apra TS f (l , Hy) = Fg

sinon f (l , Hy) = Hy

si l apra CBOUTL f (l , Fg) = Fy

sinon f (l , Fg) = Fg

si l apra TS f (l , Fy) = Hg

sinon f (l , Fy) = Fy

La fonction g de sortie (en rappellant qu'on fait ici abstraction de la sortie ST) est définie par :

g (Hg) = (G, R)
g (Hy) = (Y, R)
g (Fg) = (R, G)
g (Fy) = (R, Y)

La construction de l'expression régulière correspondant à la description comportementale de ce meme automate est donnée dans °KARL§.

Dans l'ouvrage cité en référence, les fonctions de transition et de sortie sont présentées dans une table de transition . Cette table permet, un codage par un vecteur de booléens étant choisi pour chacun des états, des entrées et des sorties , de passer automatiquement et simplement au schéma des masques de la partie combinatoire de la réalisation si elle est réalisée avec un P.L.A.

La table de transition proposée est la suivante :

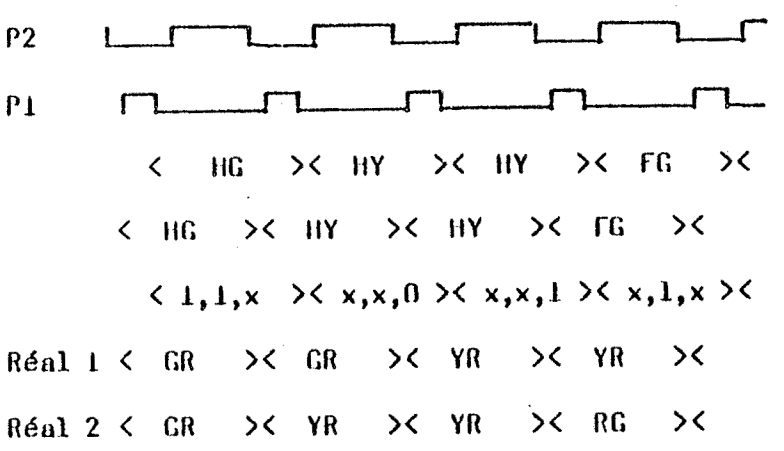
échantillonné en P1				échantillonné en P2		
entrées			état présent	état suivant	sorties	
C	TL	TS			HL	FL
0	x	x	Hg	Hg	G	R
x	0	x	Hg	Hg	G	R
1	1	x	Hg	Hy	G	R
x	x	0	Hy	Hy	Y	R
x	x	1	Hy	Fg	Y	R
1	0	x	Fg	Fg	R	G
0	x	x	Fg	Fy	R	G
x	1	x	Fg	Fy	R	G
x	x	0	Fy	Fy	R	Y
x	x	1	Fy	Hg	R	Y

Le schéma temporel proposé comporte une mémorisation entre le calcul de la fonction f et le calcul de la fonction g. L'absence d'information sur la façon dont l'automate est initialisé ne permet pas de savoir si la réalisation est de type aA, aB, bA ou bB.

Sur la base d'une traduction identique table de transition schéma des masques de la réalisation, une réalisation de type 2, avec calcul de g*f sans mémorisation entre le calcul de f et celui de g, peut etre décrite par la table de transition suivante :

échantillonné en P1				échantillonné en P2		
entrées			état présent	état suivant	sorties	
C	TL	TS			HL	FL
0	x	x	Hg	Hg	G	R
x	0	x	Hg	Hg	G	R
1	1	x	Hg	Hy	Y	R
x	x	0	Hy	Hy	Y	R
x	x	1	Hy	Fg	R	G
1	0	x	Fg	Fg	R	G
0	x	x	Fg	Fy	R	G
x	1	x	Fg	Fy	R	Y
x	x	0	Fy	Fy	R	Y
x	x	1	Fy	Hg	G	R

La différence entre les deux fonctionnements obtenus peut apparaitre sur les chronogrammes ci-dessous :



5.1.2 SOLUTION MICROPROGRAMMEE

Le principe de base de la microprogrammation a été mis au point pour des automates décrit sous forme de Moore ayant la particularité que chaque état a au plus deux successeurs. Ces principes sont naturellement étendus à tous les automates. On s'intéresse ici au cas simple des deux successeurs, c'est à dire qu'à chaque état sp appartenant à l'ensemble S des états correspond une partie Lp de l'ensemble L des entrées telle que la fonction f de transition puisse s'exprimer sous la forme:

si l apra Lp , alors f (l , sp) = sj
sinon f (l , sp) = sk

Un certain nombre de particularités des familles Li de chacun des états amène à des réalisations plus ou moins simples.
Dans le principe initial <WILK>, le cas suivant est considéré :
L'ensemble S des états est partitionné en deux :
S = Sbran U Snon
pour sp apra Snon , Lp est vide ,
pour sp apra Sbran, Lp est une certaine partie B de
l'ensemble L des entrées.
Cela correspond aux états à un seul successeur (dans Snon) et aux états à deux successeurs choisis selon un critère qui est le meme pour tous les états. (Etats sources de branchement dans Sbran).

On a alors :

f (l , sj) = s1		dans le cas où sj appartient à Snon,indépendamment de l
= s2		dans le cas où sj appartient à Sbran et l appartient à ß
= s3		dans le cas où sj appartient à Sbran et l n'appartient pas à ß

Le " calcul " de la fonction f se fait alors par une consultation d'une table écrite en mémoire morte qui délivre :

pour un élément de Snon le code de l'état suivant,

pour un élément de Sbran les codes des deux états suivants potentiels. Le choix entre les deux est réalisé par un multiplexeur comandé par la valeur logique "l appartient ou n'appartient pas à ß".

De façon analogue, le " calcul " de la fonction de sortie g se fait par consultation d'une mémoire morte contenant g(s), pour chaque s.

Un usage fréquent dans les systèmes microprogrammés est de s' "arranger " pour que " le plus souvent possible " le successeur ou l'un des deux successeurs d'un état ait un code facile à établir à partir du code de l'état courant. On peut alors faire un importante économie de place en mémoire morte en supprimant la partie qui délivre l'état suivant et en la remplaçant par un circuit combinatoire extérieur à la mémoire qui calcule alors éffectivement le code de l'état suivant à partir du code de l'état courant.

Cette fonction de transcodage peut etre l'incrémentation, mais ce

n'est pas obligatoire.

Considérant alors l'ensemble des états muni de la relation binaire R définie par :

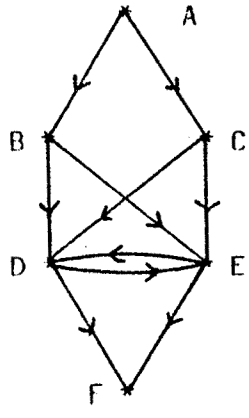
s R s' ssi il existe l tel que f(l, s) = s'

on obtient un graphe

* orienté,
* où tout noeud est de degré 1 ou 2 ,
* ayant une source (l'état initial)

On aimerait alors établir un codage bijectif par des entiers d'un tel graphe tel que l' un au moins des successeurs d'un état à deux successeurs codé n soit codé n+1.

L'exemple suivant montre que cela peut être impossible :



L'arbre des codages possibles donne :

C= N

=> E = N+1

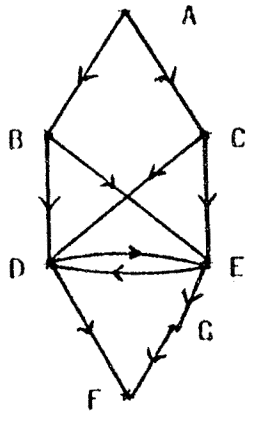
=> D = N+2 ou F = N+2 } impossible

N=/ 0,1,2,3.

L'arrangement évoqué plus haut à propos de codage , doit alors

etre utilisé pour les graphes, très particuliers, où le codage mentionné n'est pas possible. La solution habituellement retenue est de rajouter des états fictifs, rendant le codage possible, et de s'arranger (encore une fois) pour que les sorties de ces états fictifs n'aient pas d'autre influence sur le fonctionnement du circuit que l'introduction d'un retard du à l'introduction d'un état.

Sur l'exemple ci dessus, l'introduction d'un noeud G, entre E et F, permet de proposer le codage suivant :



A=0

B=1

D=2

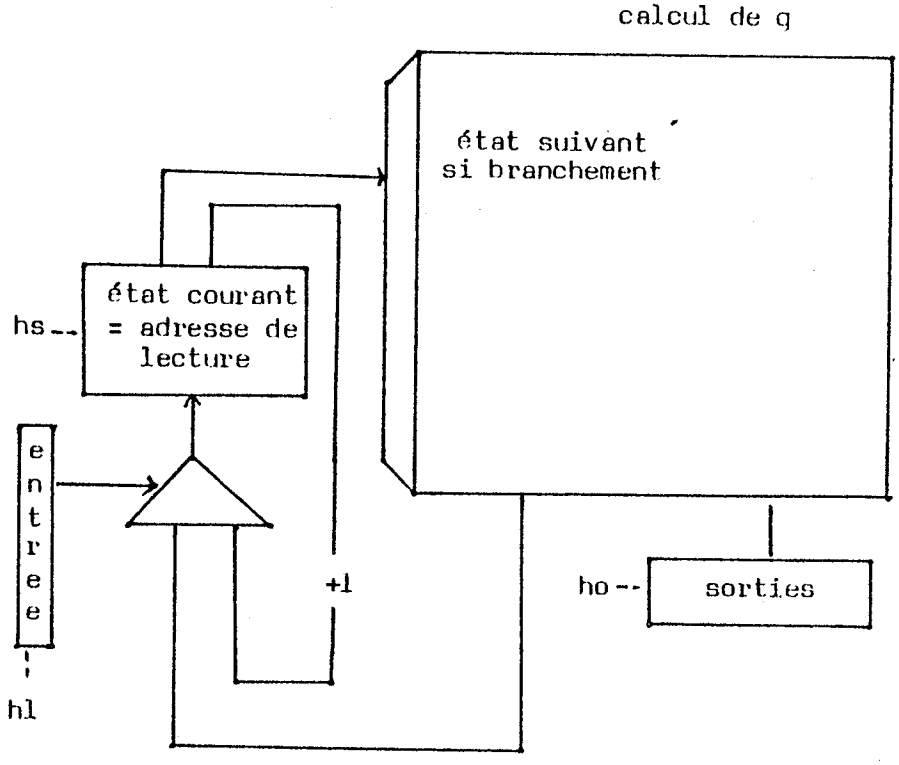
F=3

C=4

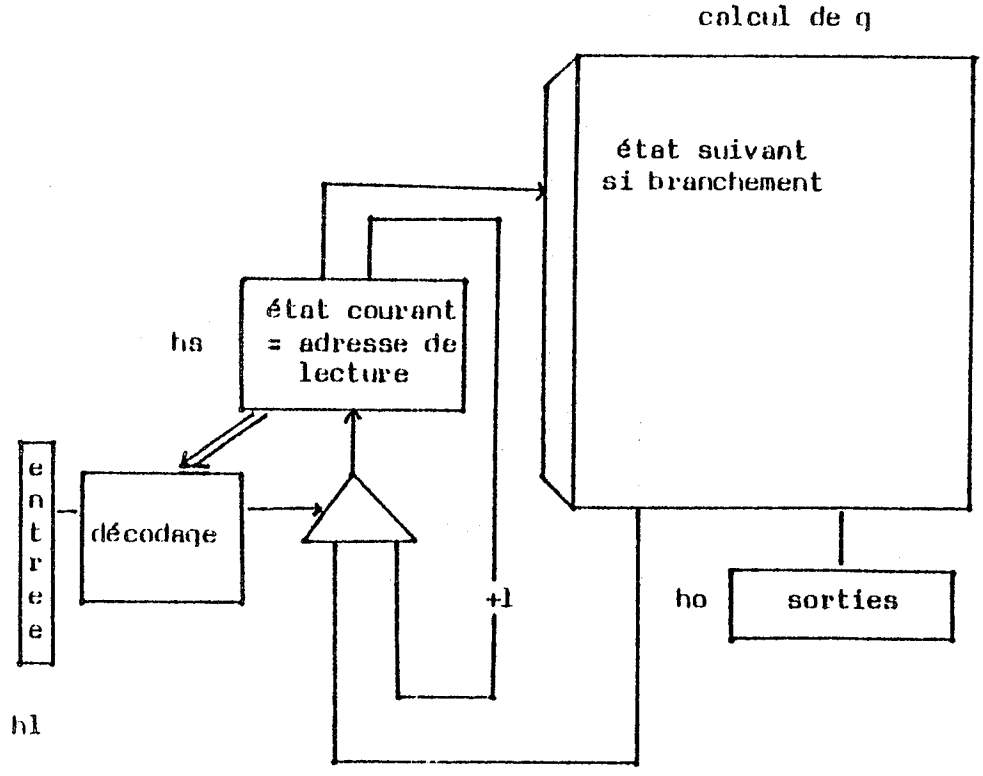
E=5

G=6

Une réalisation microprogrammée d'un automate décrit sous forme de Moore est souvent du type de la réalisation 1, elle est alors organisée selon le schéma :



De nombreuses variantes sont apportées à ce schéma de base pour tenir compte, par exemple, du fait que la partie Lp de l'ensemble des entrées correspondant au choix du successeur de l'état sp n'est , en général, pas la meme pour tous les états. On a alors un schéma de principe comme celui-ci, où la partie décodage calcule, au vu de l'état courant et de l'entrée si il doit y avoir branchement ou incrémentation.



On notera que les réalisations microprogrammées se prètent à priori mal à des réalisations de type 2. On a en effet noté que les réalisations de type 2 ont un intéret lorsque le calcul de la fonction composée g*f n'est pas plus complexe que le calcul de f ou celui de g. Ce n'est évidemment pas le cas pour un calcul par tabulation dans une mémoire morte . Dans le cas simple présenté, il faudrait tabuler dans la mémoire, pour chaque état

* Les sorties correspondant à l'état suivant si il y a un branchement,

* Les sorties correspondant à l'état suivant si il y a incrémentation. Cette solution est peu réaliste.

5.2 DIFFERENTES DESCRIPTIONS D'UNE MEME REALISATION

5.2.1 DESCRIPTION ARCHITECTURALE CONVENTIONNELLE

Le circuit, en fait la partie de circuit , auquel on s'intéressera ici est constitué

* d'un registre dit de commande , chargé sur hl,

* de deux registres, A et B , de données, maitre/esclaves, chargés sur ChA et ChB, ces deux signaux étant synchronisés sur une horloge hs

* d'une unité arithmétique à deux entrées x et y , et trois fonctions :

  . transfert de x (1)

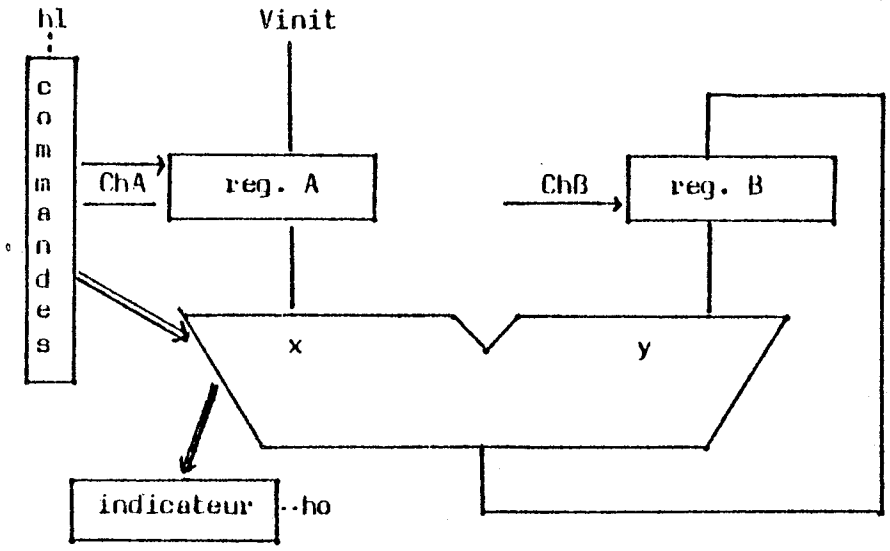
  . calcul de x + y (2)

  . calcul de x - y (3)

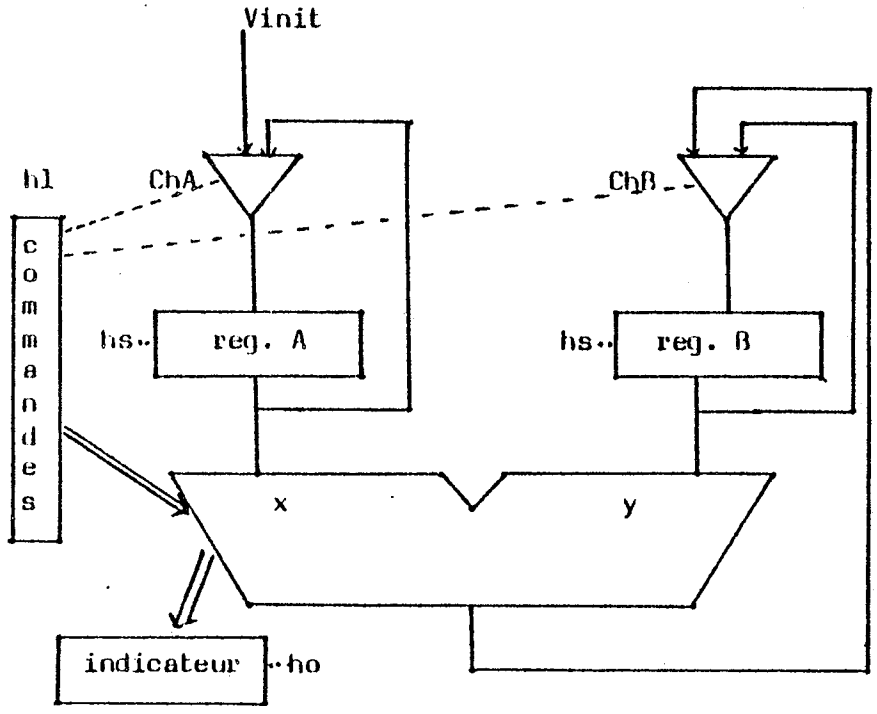
* d'un registre indicateur échantillonnant, sur une horloge ho , VRAI si la sortie de l'unité arithmétique est positive, FAUX si elle est négative ou nulle.

* d'une valeur Vinit, reliée à l'entrée du registre A.

Un schéma architectural conventionnel serait :



On préférera ici le schéma équivalent suivant qui fait apparaitre que, si un des registres de données n'est pas chargé, il garde son ancienne valeur.



5.2.2 DESCRIPTION SOUS FORME D'AUTOMATE DE MEALY

Dans la réalisation ci dessus on appellera état le vecteur
< a , b > des valeurs mémorisées dans les registres A et B.
On décrirait cet automate sous forme de Mealy par sa fonction de transition

où l'on notera

CHA la commande ChA est active et A prend une nouvelle valeur)

CHA' la commande ChA n'est pas active (et A garde son ancienne valeur).

f (CHA, CHB, 1 , <a,b>) = < Vinit , a >

f (CHA, CHB, 2 , <a,b>) = < Vinit , a+b >

f (CHA' , CHB, 1 , <a,b>) = < a , a >

f (CHA' , CHB, 3 , <a,b>) = < a , a-b >

pour i valant 1, 2 ou 3

f (CHA' , CHB' , i, <a,b>) = < a , b >

f (CHA , CHB' , i , <a,b>) = < Vinit , b >

La fonction de sortie g serait de meme décrite par :

g (CHA , CHB , 1 , <a,b>) = (a>0)

g (phi , phi' , 1 , <a,b>) = (a>0)

g (phi , phi' , 2 , <a,b>) = (a+b>0)

g (phi , phi' , 3 , <a,b>) = (a-b>0)

où phi et phi' représentent des valeurs booléennes .

5.2.3 DESCRIPTION SOUS FORME D'AUTOMATE DE MOORE

La description de la meme réalisation sous forme d'automate de Moore peut etre obtenue en prenant le procédé classique qui est de représenter l'état par le triplet

<a,b,prédicat sur la sortie de l'opérateur>

On aurait alors, par exemple ,

f (CHA' , CHB , 3 , <a,b,phi>) = < a, a-b , (a-b>0)>

f (CHA , CHB' , 2 , <a,b,phi>) = < Vinit , b, (a+b>0)>

f (CHA' , CHB' , 1 , <a,b,phi>) = < a, b , (a>0)>

et

g (<a,b,c>) = c

La réalisation qui a été présentée est bien une réalisation de type 2 , à sorties calculées par g*f , de l'automate de Moore décrit ci-dessus.

Remarque

On n'a pas , dans cette réalisation , fait apparaitre de dispositif d'initialisation de l'état. Il faudrait en effet dans cette réalisation :

* forcer Vinit1 dans A lors d'une première "opération"

qui nécessite une entrée de commande

forçage (1) = CHA, phi , phi .

* puis forcer Vinit2 dans A pendant qu'on transfert Vinit1 dans

B à travers l'opérateur arithmétique ce qui nécessite une deuxième opération avec cette fois une entrée de commande

forçage (2) = CHA , CHB , 1 pour se trouver dans un état initial connu.

5.2.4 VARIANTE

Une variante, apparemment mineure dans la réalisation, amène bien sur une modification complète de la description. Il s'agit peut etre là d'un inconvénient d'une description par automate faisant apparaitre explicitement la fonction de transition.

Introduisons dans l'exemple précédent une commande supplémentaire de chargement (indic) ou de non chargement (indic') de l'indicateur de positivité de sortie de l'opérateur arithmétique. En cas de non chargement, l'ancienne valeur est maintenue.
Si l'on désigne l'état par le triplet <a,b,p>, comme précédemment pour un automate de Moore, la description devient maintenant:

f (CHA, CHB, 1, indic', <a,b,p>) = < Vinit, a, p >

f (CHA, CHB, 1, indic , <a,b,p>) = < Vinit, a, (a>0) >

et ainsi de suite. la fonction de sortie est élémentaire puisque g (<a,b,p>) = p Cette description peut aussi etre vue comme une description sous forme de Mealy, il suffit pour cela de poser, selon le procédé classique :

gMealy (entree, <a,b,p>) = gMoore (<a,b,p>) = p

On a là un cas très particulier d'automate puisque il n'y a pas de véritable sortie mais où la sortie est une partie de l'état. Les horloges ho et hs sont dans ce cas naturellement confondues, et la "sortie" délivrée est dans tous les cas celle correspondant au "futur" état comme dans les réalisations de type 2. Une sortie est en effet échantillonnée en meme temps que létat auquel elle correspond.

Une partie de l'étude de Berg <BERG> sur la synchronisation entre deux automates , qu'il appelle controleur et chemin de données , repose sur l'hypothèse que tout chemin de données synchrones est du type de cette variante, c'est à dire a des sorties qui sont une partie de l'état.

Il faut enfin remarquer que, dans cette dernière réalisation, l'entrée

NOP = (CHA' , CHB' , phi, indic')

est telle que, quelque soit l'état s,

f (NOP , s) = s

Cette valeur de l'entrée est donc très utile lorsqu'on veut maintenir le circuit dans un état constant. C'est notamment le cas pour un automate controlé dans le cas où on a rajouté des états fictifs à l'automate "controlant".

B I B L I O G R A P H I E

<AGRE>
P.E. AGRE
A high level silicon compiler
Rapport de " master of science " Massachusetts institute of technology, Janvier 1983

<AISE>
M.A. AISERMAN, L.A. GUSEV, L.I. ROZONOER, I.M. SMIRNOVA, A.A. TAL
Logic, automata and algorithms
Academic press, 1971

<AMB1>
P. AMBLARD, G. SAUCIER
Design methodology of controllers
International conference on circuits and computers, pp 230-233, Port-Chester, 1-3 octobre 1980

<AMB2>
P. AMBLARD, M. CRASTES DE PAULET, J. RARIVOMANANA, G. SAUCIER
CADOC : A functional specification and simulation tool
International conference on computer aided design, pp 65-66
Santa Clara , septembre 1983.

<AUMI>
M. AUMIAUX
Methode de conception de systemes logiques : obtention des signaux de commande.
Minis et micros, pp 41-44, 30 novembre 1981

<BARB>
M. R. BARBACCI, G.E. BARNES, R.G. CATTELL, D.P. SIEWIOREK
The I.S.P.S. Computer Description Language.
Rapport Carnegie Mellon University 1977

<BARC>
N. BAR-CHAIM, I. URY, A. YARIV
Integrated opto-electronics
I.E.E.E. Spectrum, pp 38-45, mai 1982.

<BARR>
J.C. BARROS, B.W. JOHNSON
Equivalence of the arbiter,the synchronizer, the latch and the inertial delay.
I.E.E.E. Transactions on computers, pp 603-614, juillet 1983

<BENZ>
C. BENZAKEN
Cours d'algebre appliquee : Cours polycopie
U.S.M. Grenoble, 1975

<BER2>
H.K.BERG
A model of timing characteristics in computer control
Microprocessing and microprogramming, pp 206-218, juillet 1979

<BER1>
M.H. BERGER, S.K.WEIDMANN
Merged transistor logic : A low cost bipolar logic concept.
I.E.E.E. Journal of Solid state circuits, pp 340-351, octobre 1972

<BEYL>
A.M. BEYLS, B. HENNION, J. LECOURVOISIER, G. MAZARE, A. PUISSOCHET
A design methodology based upon symbolic layout and integrated C.A.D. tools.
19 eme Design Automation Conference, pp 872-875, Las Vegas, 14-16 juin 1982.

<BOCH>
G.V. BOCHMANN
Hardware specification with temporal logic : an example.
I.E.E.E. Transactions on computers, pp 223-231 , mars 1982

<BONO>
A. BONOMO
Simulazione di sistemi digitali descritti al livello funzionale in linguagio S.D.L.
Tesi di Laurea Facolta di scienze MFN Torino 1981

<BREN>
R.P. BRENT, H.T. KUNG
A regular layout for parallel adders.
I.E.E.E. Transactions on computers, pp 260-265, mars 1982.

<BROF>
S.BROFFERIO
Two cases studies in teaching controller design in electronic systems.
Microprogramming and microprocessing, pp 160-168, mars 1981.

<BRO1>
H. BROWN, C. TONG, G. FOYSTER
PALLADIO : An exploratory environment for circuit design dans <COM2>

<BRO2>
D. W. BROWN
A state machine synthesizer
18 eme Design automation conference, pp 301-305, Nashville, 29 ju-1 juillet 1981

<BRYA>
R.E. BRYANT
A switch-level model of M.O.S. logic circuits.
1 ere International conference on V.L.S.I., pp 329-340, Edinburgh, 18-21 aout 1981.

<CALD>
S.H. CALDWELL
Switching circuits and logical design.
J. Wiley and sons, New-York, 1958

<CARD>
L. CARDELLI, G. PLOTKIN
An algebraic approach to V.L.S.I. design
1ere International conference on V.L.S.I. (VLSI 81), pp173-182, Edinburgh,
18-21 aout 1981

<CARR>
W.R. CARR, J.P. MIZE
MOS/LSI design and application
Mc Graw hill, 1972

<CART>
W.C. CARTER, W.H. JOYNER, D. BRAND
Symbolic simulation for correct machine design.
16 eme Design Automation Conference, pp 280-286, San Diego, 25-27 juin 1979.

<CAS1>
P. CASPI, N. HALBWACHS
An approach to real time systems modelling.
International conference on Distributed Computing Systems, Miami,
octobre 1982

<CAS2>
P. CASPI, N. HALBWACHS
Conception certifiee de systemes distribues : un exemple.
Rapport final de l' Action Thematique Programmee C.N.R.S. Parallelisme, Cooperation, Synchronisation, juillet 1983

<CLOU>
A.W. NAGLE, R. CLOUTIER, A.C. PARKER
Synthesis of hardware for the control of digital systems
I.E.E.E. Transactions on computer aided design of integrated circuits and systems, pp 201-212, octobre 1982.

<COFF>
E.G. COFFMANN
A formal microprogram model of parallelism and register sharing.
Symposium on computers and automata, New-York, 13-15 avril 1971.

<COM1>
Hardware description language applications : Voices from the tower of BABEL
Computer, numero de juin 1977

<COM2>
New V.L.S.I. tools
Computer, numero de decembre 1983

<CONW>
L. CONWAY
The M.P.C. adventures : experiences with the generation of V.L.S.I. design and implementation methodologies.
Microprocessing and microprogramming, novembre 1982.

<CULI>
K. CULIK, H. JURGENSEN
Programmable finite automata for V.L.S.I.
International journal of computer mathematics, pp 259-275, 1983

<DAVI>
R. DAVID
Modular design of asynchronous circuits defined by graphs.
I.E.E.E. Transactions on computers, pp 727-737, aout 1977.

<DIRE>
S.W. DIRECTOR, A.C. PARKER, D.P. SIEWIOREK, D.E. THOMAS
A design methodology and computer aids for digital V.L.S.I. systems.
I.E.E.E. Transactions on circuits and systems, juillet 1981.

<McFA>
M.C. Mc FARLAND
On proving the correctness of optimizing transformations in a digital design automation system.
18 eme Design Automation Conference, pp 90-97, Nashville, 29-1 juin-juillet 1981

<FLOY>
R.W. FLOYD, J.D. ULLMAN
The compilation of regular expressions into integrated circuits.
Journal of the A.C.M. ,pp 603-622,juillet 1982

<FOST>
M.J. FOSTER, H.T. KUNG
Recognize regular languages with programmable building blocks
Journal of digital systems, pp 323-332, Hiver 1982

<FUJI>
M. FUJITA, H. TANAKA, T. MOTO-OKA
Verification with Prolog and temporal logic
Sixieme international conference Computer hardware description languages , pp 103-114 , Pittsburgh, 23-25 mai 1984

<GILO>
W.K. GILOI, H. LIEBIG
A formalism for description and synthesis of logical algorithms and their hardware implementation.
I.E.E.E. Transactions on computers, septembre 1974.

<GOR1>
M.J.C. GORDON
Register transfer systems and their behaviour.
5 eme International conference on computer hardware description languages and their applications, pp 23-36, Kaiserslautern, 7-9 septembre 1981.

<GOR2>
M.J.C. GORDON
A very simple model of sequential behaviour of N-M.O.S..
1 ere internatonal conference on V.L.S.I. , pp85-94, Edinburgh, 18-21 aout 1981.

<GRAF>
Collectif
Normalisation de la representation du cahier des charges d'un automatisme logique
Automatique et informatique industrielle, nov-decembre 1977

<GRAS>
W. GRASS
A synthesis system for P.L.A. based programmable hardware
Euromicro journal, pp 15-31, aout 1983

<GUST>
R.N. GUSTAFSON, F.J. SPARACIO
I.B.M. 3081 Processor unit : design considerations and design process.
I.B.M. Journal on research and development, pp 12-21, janvier 1982

<MAFE>
L.J. MAFER, A.C. PARKER
A formal method for the specification, analysis and design of register-transfer level digital logic.
I.E.E.E. Transactions on computer-aided design, pp 4-17, janvier 1983

<MALB>
N. MALBWACHS
Modelisation et analyse des systemes temporises
These d'etat, Grenoble, 1984

<MART>
R.W. MARTENSTEIN
Fundamentals of structured hardware design.
North-holland 1977

<MAVI>
G.L. MAVILAND, A. TUSZYNSKI
Time-dependent logic equations.
International Conference on Circuits and Computers, pp 746-750, Port Chester, 1-3 octobre 1980

<MENR>
M. MENRY, G. CRINON, J.LACOUR
Conception de circuits integres logiques M.O.S.T. par la methode de l'espalier.
Colloque international sur la microelectronique avancee, pp 440-448, Paris, 6-10 avril 1970

<MILL>
F.J. MILL, G.R. PETERSON
Digital systems : hardware organization and design.
J. Wiley and sons 1978

<JOHN>
D.W. JOHNSON
Go from flowchart to hardware.
Electronic design, pp90-95, 1 septembre 1976.

<KANG>
S. KANG W. M. VAN CLEEMPUT
Automatic P.L.A. synthesis for D.D.L.p description
18 eme Design automation conference, pp 391-397, Nashville, 29 ju-1 juillet 1981

<KARL>
A.R. KARLIN, H.W. TRICKEY, J.D. ULLMANN
Experience with a regular expression compiler.
International Conference on Computer Design, pp 656-665, Port Chester,31-3 Octobre-novembre 1983.

<KATZ>
R.H. KATZ, S. WEISS
A standard design frame for V.L.S.I. circuit prototyping.
Journal of V.L.S.I. and computer systems, pp 101-114, 1 er trimestre 1983.

<KOOM>
C.J. KOOMEN
C.C.S. as a possible semantic background for S.D.L.
S.D.L. Newsletter, pp 63-83, novembre 1982.

<KUNT>
J. KUNTZMANN
Algebre de Boole
Dunod, 1968

<LANZ>
LANZINI
S.D.L. and Petri nets.
S.D.L. Newsletter

<LEIN>
S. LEINWAND, T. LAMDAN
Models of control at register transfer level.
I.F.I.P. Conference, pp 163-168 , 1980

<LEWK>
K.D. LEWKE, F.J. RAMMIG
Description and simulation of M.O.S. devices in register transfer languages.
2 nd international V.L.S.I. conference, Trondheim, 16-19 aout 1983

<LIU >
T. LIU, K.R. HOHULIN, L. SHIAU, S. MUROGA
Optimal one bit full adders with different types of gates
I.E.E.E. Transactions on computers, pp 63-70, Janvier 1974

<LOOM>
H.H. LOOMIS Jr, M.R. Mc COY
A theory of high speed clocked logic.
I.E.E.E. Conference on switching circuit theory and logical design, University of Michigan, 6-8 octobre 1965.

<MAJO>
J. MAJOS, J.L. LARDY
The multidrain M.O.S. transistor.
4 ème E.S.S.C.I.R.C. , pp 133-135, Amsterdam, 18-21 septembre 1978

<MALA>
Y. MALACHI, S.S. OWICKI
Temporal specifications of self-timed systems.
Conference on V.L.S.I. systems and computations, pp 203-212 , Pittsburgh, 19-21 octobre 1981

<MALL>
R.MALLADI, G. SERRERO
An automatic design tool for symbolic layout of circuits.
7 ème E.S.S.C.I.R.C., pp 31-33, Fribourg, 22-24 septembre 1981.

<MANG>
D. MANGE, E. SANCHEZ, A. STAUFFER
Systemes logiques programmes.
Presses polytechniques romandes, 1982

<MANN>
Z. MANNA, A. PNUELI
Verification of concurrent programs : the temporal framework.
Stanford university, computer science departement, juin 1981.

<MEAD>
C. MEAD, L. CONWAY
Introduction to V.L.S.I. systems.
Addison-Wesley, 1980.

<MILI>
R. MILNER
A calculus of communicating systems.
Springer-verlag, 1980

<MIL2>
G.J. MILNE
The correctness of a simple silicon compiler.
6 eme symposium Computer hardware descripion languages, pp 1-12, Pittsburgh, 23-25 mai 1983.

<MOAL>
M. MOALLA
Specification et description sure d'automatismes discrets complexes basees sur l'utilisation du GRAFCET et des reseaux de Petri
These d'etat, Grenoble, 1981

<MURO>
S. MUROGA
V.L.S.I. systems design
J. Wiley & sons , 1979

<NAGL>
A.W. NAGLE, R. CLOUTIER, A.C. PARKER
Synthesis of hardware for the control of digital systems. I.E.E.E. transactions on computer-aided design of integrated circuits and systems, pp 201-212, octobre 1982.

<OKAZ>
K. OKAZAKI, T. MORIYA, T. YAHARA
A multiple delay simulator for M.O.S. L.S.I. circuits
20 eme Design automation conference, pp 279-285, Miami, 27-29 juin 1983

<OLIV>
V. OLIVE, D. ROUQUIER
Une methode automatique de synthese de partie controle definie par un GRAFCET.
Conference I.F.I.P, Paris, 1983.

<PAWL>
A. PAWLAK
Microprocessor systems modeling with MODLAN
20 eme Design Automation Conference, pp804-811, Miami, 27-29 juin 1983

<PECH>
M. PECHOUCEK
Anomalous response times of input synchronizers.
I.E.E.E. Transactions on computers, pp 133-139, fevrier 1976.

<PETE>
J.L. PETERSON
Petri net theory and the modelling of systems.
Prentice-Hall.

<PIG1>
C. PIGUET
Synthese de systemes logiques asynchrones a l'aide des proprietes des fonctions logiques decroissantes.
These de l'ecole polytechnique federale de Lausanne, 1981.

<PIG2>
C. PIGUET, A. STAUFFER, J.F. PEROTTO, J.J. MONBARON
Le sequenceur d'un microprocesseur.
Bulletin de l'association suisse des electriciens, pp 126-132, 1979.

<PRAS>
B. A. PRASAD
Modelling techniques for dynamic logic and test pattern generation of a microprocessor.
Symposium on design automation and microprocessors, pp 100-107, Palo Alto, 24-25 fevrier 1977

<PURU>
S. PURUSHOTHAMAN, P.A. SUBRAHMANYAM
An algebraic basis for specifying and reasonning about protocols for designing self-timed circuits.
2 eme international conference on V.L.S.I. , pp 133-144, Trondheim, 16-19 aout 1983

<RAMM>
F.J. RAMMIG
Hierarchical modular description of V.L.S.I. systems
V.L.S.I. and software engineering workshop, pp 112-116, Port-chester, 4-6 octobre 1982.

<dROS>
J. de ROSNAY
Les bio-transistors : la microelectronique du XXI eme siecle.
La recherche, pp 870-872, juillet-aout 1981.

<SAUC>
G. SAUCIER
Codage des automates asynchrones
These d'etat , Grenoble, 1970

<SEIT>
C. SEITZ
System timing.
Chapitre 7 de <MEAD>, pp218-262.

<SHOS>
R.E. SHOSTAK
Formal verification of circuit designs
6 eme conference Computer Hardware Description Languages, pp 13-30, Pittsburgh, 23-25 mai 1983

<SIFA>
J. SIFAKIS
Modeles temporels des systemes logiques.
These universite scientifique et medicale de Grenoble, 1974

<SISK>
J.F. SISKIND, J.R. SOUTHARD, K.W. CROUCH
Generating custom high performance V.L.S.I. designs from succint algorithmic descriptions.
Conference on advanced research in V.L.S.I., pp 28-40, Cambridge (Massachusets), 25-27 janvier 1982

<SOU1>
J.R. SOUTHARD, A. DOMIC, K.W. CROUCH
Lincoln boolean synthesizer.
Lincoln laboratory, M.I.T. , septembre 1982.

<SOU2>
J.R. SOUTHARD
Mac Pitts : an approach to silicon compilation.
dans <COM2>

<STE1>
M. STEFIK, L. CONWAY
Towards the principled engineering of knowledge.
The Artificial Intelligence magazine, ete 1982.

<STE2>
M. STEFIK, D.G. BOBROW, A. BELL, H. BROWN, L. CONWAY, C. TONG
The partitionning of concerns in digital system design.
Conference on advanced research in V.L.S.I., pp 43-52, Cambridge (Massachussets) 25-27 janvier 1982

<SUBR>
P.A. SUBRAHMANYAM
Synthesizing V.L.S.I. circuits from behavioural specifications : a very high level silicon compiler and its theoretical basis.
2 eme International conference on V.L.S.I. , pp 195-210, Trondheim, 16-19 aout 1983

<SUZU>
N. SUZUKI, R. BURSTALL
Sakura : A V.L.S.I. modelling language.
Conference on advanced research in V.L.S.I., pp 201-209, Cambridge (Massachussets) , 25-27 janvier 1982.

<TEEL>
B. TEEL D. WILDE
A logic minimizer for V.L.S.I. P.L.A. design
19 eme Design automation conference, pp 156-162, Las vegas, 14-16 Juin 1982

<TRED>
N. TREDENNICK
How to flowchart for hardware.
I.E.E.E. Computer, pp 87-102, decembre 1981.

<TRIC>
H.W. TRICKEY
Good layouts for pattern recognizers.
I.E.E.E. transactions on computers, pp 514-520, juin 1982.

<UMRI>
Z.D. UMRIGAR, V. PITCHUMANI
Formal verification of real time hardware design.
20 eme Design automation conference, pp 221-227, Miami, 27-29 juin 1983.

<UNGE>
H. UNGER
Double edge triggered flip-flops
I.E.E.E. Transactions on computers, pp 447-451, juin 1981

<VITT>
E. VITTOZ, C. PIGUET, W. HAMMER
Model of the logic gate.
Journees d'electronique E.P.F.L., pp D7.1-D7.13, Lausanne, 18-20 octobre 1977

<VUIL>
J. VUILLEMIN
A very fast multiplication algorithm for V.L.S.I. implementation.
Integration The V.L.S.I. journal, pp 39-52, avril 1983.

<WAGN>
T.J. WAGNER
Verification of hardware designs thru symbolic manipulation.
Symposium on design automation and microprocessors, pp 50-53, Palo alto, 24-25 fevrier 1977.

<WALD>
K. WALDSCHMIDT, R. FAHNRICH, D. TAVANGARIAN
Computer aided design of microprogrammed control units with Sandra
Euromicro 1980

<WANN>
D.F. WANN, M.A. FRANKLIN
Asynchronous and clocked control structures for V.L.S.I. based interconnexion networks.
I.E.E.E. transactions on computers, pp 284-293, mars 1983.

<WILK>
M.V. WILKES
The best way to design an automatic calculating machine (1951),reproduit dans
Microprocessing and microprogramming, automne 1981

<WINK>
D. WINKEL, F. PROSSER
The art of digital design : An introduction to top-down design.
Prentice-Hall, 1980.

<WOJC>
W.S. WOJCIECHOWSKI
Structured digital systems design in multiple-valued logic.
12 eme international symposium on multiple valued logic, pp 206-222, Paris, 25-27 mai 1982.

<ZIMM>
G. ZIMMERMANN
The MIMOLA design system.
16 eme Design automation conference, pp 53-58, San Diego, 25-27 juin 1979.

AUTORISATION de SOUTENANCE

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

VU le rapport de présentation de Madame le Professeur G. SAUCIER

Monsieur Paul AMBLARD

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

Fait à Grenoble, le 21 mai 1984

Le Président de l'I.N.P.-G

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

P.O. le Vice-Président,