Projet CASCADE : une approche de la simulation hiérarchisée multi-modes

Marc Humbert

THESE

présentée à

l'Institut National Polytechnique de Grenoble

pour obtenir le grade de
DOCTEUR-INGENIEUR
«Informatique»

par

Marc HUMBERT

PROJET CASCADE: UNE APPROCHE DE LA SIMULATION HIERARCHISEE MULTI-MODES.

Thèse soutenue le 29 octobre 1984 devant la commission d'examen.

G. VEILLON	Président
B. BAYLAC	Examinateurs
J. FONLUPT	
J. MERMET	
C. LE FAOU	

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

ECOLE NATIONALE SUPERIEURE DES MINES DE SAINT-ETIENNE

Directeur	: Monsieur M. MERMET
Directeur des Etudes et de la formation	: Monsieur J. LEVASSEUR
Directeur des recherches	: Monsieur J. LEVY
Secrétaire Général	: Mademoiselle M. CLERGUE

Professeurs de 1ère Catégorie

COINDE	Alexandre	Gestion
GOUX	Claude	Métallurgie
LEVY	Jacques	Métallurgie
LOWYS	Jean-Pierre	Physique
MATHON	Albert	Gestion
RIEU	Jean	Mécanique - Résistance des matériaux
SOUSTELLE	Michel	Chimie
FORMERY	Philippe	Mathématiques Appliquées

Professeurs de 2ème catégorie

HABIB	Michel	Informatique
PERRIN	Michel	Géologie
VERCHERY	Georges	Matériaux
TOUCHARD	Bernard	Physique Industrielle

Directeur de recherche

LESBATS	Pierre	Métallurgie

Maîtres de recherche

BISCONDI	Michel	Métallurgie
DAVOINE	Philippe	Géologie
FOURDEUX	Angeline	Métallurgie
KOBYLANSKI	André	Métallurgie
LALAUZE	René	Chimie
LANCELOT	Francis	Chimie
LE COZE	Jean	Métallurgie
THEVENOT	François	Chimie
TRAN MINH	Canh	Chimie

Personnalités habilitées à diriger des travaux de recherche

DRIVER	Julian	Métallurgie
GUILHOT	Bernard	Chimie
THOMAS	Gérard	Chimie

Professeur à l'UER de Sciences de Saint-Etienne

VERGNAUD	Jean-Maurice	Chimie des Matériaux & chimie industrielle

****

Je tiens à remercier tous les membres de ce jury :

Monsieur le Professeur G. VEILLON, Directeur de l'E.N.S.I.M.A.G., de m'avoir fait l'honneur d'accepter de le présider.

Monsieur J. MERMET, Maître de Recherche au C.N.R.S. et Directeur du Laboratoire ARTEMIS, pour m'avoir accueilli dans son équipe et avoir dirigé ce travail.

Monsieur B. BAYLAC, Responsable du Service Caractérisation, Modélisation et Simulation de la Société THOMSON-EFCIS, pour avoir accepté d'être le rapporteur de cette thèse.

Monsieur C. LE FAOU, Ingénieur C.N.R.S. et Chef du projet CASCADE, avec qui j'ai toujours eu le plus grand plaisir à travailler et dont j'ai pû apprécier les qualités humaines au cours des années.

Monsieur J. FONLUPT, Professeur à l'E.N.S.I.M.A.G., qui a guidé mes premiers pas dans la Recherche et dont je garde un souvenir reconnaissant.

Je n'oublie pas tous les Membres du Laboratoire ARTEMIS que j'ai appris à connaître et à apprécier au cours de mes années passées en son sein ; en particulier l'un des derniers arrivés, Mohamed Nazir HAKIM qui fut le premier à me proposer son aide dans le sprint final et Claudine MEYRIEUX pour m'avoir aidé à la réalisation matérielle ; ainsi que D. Iglésias et tout le Service de Reprographie de l'Institut IMAG.

Enfin, je remercie tous les Membres de l'Equipe CAD du Centre de Technologies Informatiques de PHILIPS à Fontenay-aux-Roses dont j'ai toujours apprécié la gentillesse et la bonne humeur au cours de mes "stages parisiens".

... En fin de compte, je remercie tout le monde !

Sans oublier... moi-même d'avoir eu le courage de rédiger une thèse... et le traitement de texte RUNOFF du VAX qui m'a permis de la frapper avec conviction : les côtés un peu obsolètes de cet outil ont finalement un certain charme à l'heure où n'importe quel micro en offre un beaucoup plus performant.

De toutes façons, la nouvelle génération de traitement de texte "scientifique" est en gestation et il me reste simplement à déplorer d'avoir rédigé avec un peu d'avance (mais oui...) sur la sortie du prototype SPARADRAP [Hull.83] .

Que d'heures ingrates passées devant mon terminal, ce merveilleux outil ne m'eût-il pas épargnées !

PLAN

CHAPITRE I : L'ÉVOLUTION DE LA SIMULATION DE SYSTÈMES LOGIQUES

CHAPITRE II : LA CONSTRUCTION D'UN MODÈLE : ORDONNANCEMENT.

CHAPITRE III : MÉCANISMES DE SIMULATION.

INTRODUCTION

Ce travail a trait à la simulation des systèmes logiques. et se rattache de ce fait à la CAO de VLSI très en vogue actuellement.

C'est en effet devenu un lieu commun que d'insister sur la complexité croissante des petites "puces" qui rend nécessaires l'étude et le développement de nouveaux outils de CAO susceptibles d'aider les concepteurs dans leur lourde tâche.

La simulation n'est peut-être pas la panacée: certains pensent, sans doute avec raison, que l'on pourra un jour s'en passer lorsque des outils performants de synthèse et de compilation logique seront disponibles. Un simulateur est un outil somme toute assez limité car il permet uniquement de vérifier le bon fonctionnement d'un circuit par rapport à un certain modèle.

Pourtant dans le contexte actuel , on utilise encore énormément les simulateurs: il est préférable, en effet, de passer un peu de temps (humain) à décrire un circuit et un peu (ou beaucoup) de temps (machine) à le simuler plutôt que de passer beaucoup de temps à construire un prototype "hard".

Et comme cette situation risque de durer encore de longues années, beaucoup de gens pensent encore à améliorer les simulateurs existants de manière à augmenter la productivité des concepteurs. C'est l'un des objectifs du projet CASCADE* auquel nous avons participé.

Dans le premier chapitre de cette thèse, nous nous intéressons à l'évolution de la simulation en essayant de mettre en évidence certaines idées nouvelles qui sont apparues au cours des dernières années telles les méthodes de décomposition en simulation "électrique", la simulation au niveau "transistor", la simulation concurrente, les architectures spécialisées et la simulation multi-modes... Ces nouveaux axes de développement ne sont d'ailleurs pas exclusifs et dans le projet CASCADE, ils sont tous étudiés sauf (pour l'instant) les architectures spécialisées.

(*) CASCADE est une marque déposée de l'ENSIMAG (Conception Assistée de Systèmes et Circuits Analogiques et Digitaux Electroniques).

Dans le cadre de ce projet CASCADE, nous avons participé à l'étude et à la réalisation du simulateur multi-modes. Il s'agit bien sûr d'un travail d'équipe vu l'ampleur de la tâche: les chapitres 2 et 3 de cette thèse détaillent plus particulièrement notre participation personnelle à cette tâche et sont loin de couvrir tout le travail réalisé par notre équipe.

Nous avons d'abord mis au point des mécanismes d'ordonnancement du modèle: c'est l'objet du deuxième chapitre. L'ordonnancement est une partie de la compilation du modèle servant à préparer statiquement la simulation. Cette préparation joue à deux niveaux: on ordonne statiquement la hiérarchie correspondant au modèle et on ordonne aussi les parties fonctionnelles de ce modèle. Ces mécanismes sont actuellement en fonctionnement dans le prototype CASCADE.

Le troisième chapitre correspond à une partie dont nous avons participé à la définition mais dont la réalisation a été assurée par un autre membre de l'équipe: Il s'agit des mécanismes de simulation permettant de simuler un modèle dont certaines parties sont décrites à des niveaux différents à l'aide de modes de simulation différents. Un premier prototype de ce simulateur fonctionne actuellement dans notre équipe; on pourra trouver en annexe quelques exemples de simulation qu'il a permis de réaliser.

CHAPITRE 1

L'ÉVOLUTION DE LA SIMULATION DE SYSTÈMES LOGIQUES

CHAPITRE I

L'EVOLUTION DE LA SIMULATION DE SYSTEMES LOGIQUES

Le but de ce chapitre n'est pas de dresser un état de l'art exhaustif sur ce vaste sujet qu'est la simulation des systèmes logiques. Plus modestement, nous nous contenterons d'essayer de mettre en évidence l'évolution des idées à propos de la gestion du temps et des mécanismes de séquencement en simulation.

Après quelques rappels concernant la description et la modélisation du matériel, nous définissons les notions communes aux différents types de simulateurs. Suit un rapide historique des méthodes utilisées en simulation au niveau électrique jusqu'à l'actuelle "troisième génération de simulateurs" basée sur l'idée de DECOMPOSITION.

Avec l'évolution de la simulation logique, on aura une idée des efforts constants d'amélioration des méthodes classiques. La vitesse d'exécution est déterminante à ce niveau et bien souvent, les travaux en simulation logique portent sur des petites astuces susceptibles d'augmenter le fatidique "nombre de portes évaluées par seconde".

Quelques idées nouvelles sont néammoins apparues, telles la simulation concurrente imaginée par E. Ulrich pour la simulation de pannes (Ulr.73) et appliquée maintenant à la simulation "normale"; ou encore les architectures spécialisées. Les "machines de simulation" se développent très vite actuellement. On notera aussi l'émergence de la simulation au niveau "transistor" bien appréciable pour les concepteurs travaillant en technologie MOS.

On passe enfin aux tentatives d'intégration: de nombreuses équipes travaillent sur des projets de simulateurs multi-niveaux ou multi-modes qui représentent peut-être la seule solution pour la simulation des circuits de demain: c'est en tous cas le point de départ du projet CASCADE avec l'étude d'un simulateur hiérarchisé multi-modes basé sur un langage de description multi-niveaux.

I.1 IDEES GENERALES SUR LA SIMULATION

I.1.1 DESCRIPTION ET MODELISATION

La presque totalité des outils CAO en conception de circuits utilisent des modèles plus ou moins sommaires des circuits étudiés.

En général, ces modèles sont construits à partir de descriptions du circuit et peuvent être définis à différents niveaux d'abstraction.

Ainsi, en "montant" à l'échelle d'abstraction décrite plus loin, on s'éloigne des phénomènes physiques en faisant des approximations sur le comportement temporel et logique du système. Si en prenant de la hauteur on perd de la précision, on y gagne en efficacité (du point de vue de la rapidité) : le processus de modélisation-simulation est très marqué par la recherche de ce compromis EFFICACITE / PRECISION.

Pour modéliser un circuit ou un système logique, l'emploi des langages de description de matériel s'est généralisé: on pourra trouver une étude complète sur ces langages dans (Bor.81).

Nous évoquerons cependant les niveaux de description habituellement reconnus:

ECHELLE D'ABSTRACTION

- NIVEAU ELECTRIQUE ou ANALOGIQUE ("circuit" en Anglais)

- NIVEAU TRANSISTOR ou INTERRUPTEUR ("switch" en Anglais)

  Approximation du niveau précédent (en discrétisant les conductances) (En fait, historiquement c'est le niveau suivant que l'on a détaillé un peu pour s'adapter aux technologies MOS).

- NIVEAU LOGIQUE ou "PORTE LOGIQUE"

  Ce niveau est purement structurel. Le système est vu comme un réseau de portes interconnectées.

- NIVEAU TRANSFERT DE REGISTRES (RTL)

  En général, ce niveau est le niveau charnière entre le Structurel et le Comportemental. A ce niveau, on a déjà spécifié certaines informations structurelles comme la connexion de certaines boîtes.

- Le HAUT DE L'ECHELLE ne sera pas abordé dans ce travail. On peut y trouver le niveau ARCHITECTURAL si l'on s'interesse déjà à une représentation physique et le niveau ALGORITHMIQUE si l'on s'intéresse à la fonction remplie par le système

On peut imaginer toutes les variations et combinaisons possibles. La raison en est simple: Les gens ayant développé un simulateur pour un niveau (par exemple le niveau porte) cherchent à étendre la portée de ce simulateur en rajoutant des élements au langage de description (Par exemple, on ajoute couramment des éléments fonctionnels aux simulateurs au niveau Porte Logique)

En fait comme l'affirme D.Borrione (Bor.81), il existe un nombre "à priori aussi grand que l'on veut" de niveaux d'abstraction.

Du point de vue de la simulation qui est celui qui nous intéresse ici, on peut déjà distinguer deux grandes classes de modèles:

- Les modèles continus où l'on décrit l'évolution continue du temps. Les variables du système prennent aussi des valeurs réelles.

- Les modèles discrets où l'on fait des approximations en considérant que les variables du système prennent un nombre fini de valeurs et que le temps progresse de manière discrète (On ne peut observer le modèle qu'à certaines dates).

IMPORTANCE DU MODELE

Bien que l'étude des modèles ne soit pas notre objet ici , on ne saurait trop insister sur l'importance des outils de modélisation mis à la disposition du concepteur.

I.1.2 ALGORITHMES

La GESTION DES EVENEMENTS représente la partie la plus importante de tout simulateur.

Les descripteurs d'évènements doivent être connectés dans une certaine structure qui définisse l'ordre suivant lequel ces évènements seront exécutés.

On peut envisager différentes manières de la concevoir (Ler.80).

I.1.2.1 SEQUENCEMENT STATIQUE

On détermine statiquement (de manière automatique ou non) une liste d'évènements à évaluer à une date donnée.

Cette liste étant "préparée" statiquement, il est souhaitable de tenir compte des contraintes de précédence d'évènements. En clair, il est bon d'ordonner les évènements de manière à respecter leurs conditions d'évaluation.

L'avantage de ce type de séquencement est la facilité avec laquelle on peut, à partir d'un évènement, obtenir "l'évènement suivant". (suivant dans la liste)

Il y a par contre un inconvénient qui peut être important dans certains cas: chaque évènement de la liste est traité même si ce n'est pas nécessaire.

Dans ce type de séquencement la gestion du temps est particulièrement simple. Il existe généralement une horloge qui pilote la simulation. Le temps progresse ainsi de pas d'horloge en pas d'horloge.

I.1.2.2 SEQUENCEMENT DYNAMIQUE

La philosophie du séquencement dynamique est tout autre. Ce n'est plus un système d'horloge à fréquence fixée qui commande la gestion du temps mais l'évolution du modèle lui-même. Ce type de simulation colle de plus près à la propagation de l'activité dans le modèle. La temps progresse d'une date d'évenement à une autre. La gestion du temps est assurée par un ECHEANCIER.

L'échéancier peut être considéré comme un ensemble d'évènements structuré sous forme d'une table ou d'une liste. (on pourra trouver une étude assez complète sur les différentes structures d'écheancier dans (Ler.80)).

I.2 EVOLUTION DE LA SIMULATION ELECTRIQUE

Les simulateurs électriques sont les simulateurs les plus utilisés dans les équipes de conception.

On utilise ce type de simulateurs lors de l'étude et la conception des éléments de base (portes, bascules etc). Ils fournissent les niveaux des signaux au cours du temps, et toutes informations nécéssaires, en fait, pour modéliser le composant à un niveau plus haut. Ils sont, bien sûr, étroitement liés à la technologie.

Ils ont pour inconvénient majeur le fait de ne pas pouvoir simuler des circuits trop importants.

I.2.1 POSITION DU PROBLEME

Le modèle est un système algébro-différentiel non linéaire de la forme:

$$F(z(t),z'(t),t) = 0$$

I.2.2 SIMULATION STANDARD

Les premiers simulateurs électriques datent d'une vingtaine d'années: ces simulateurs de "première génération" étaient caractérisés par un schéma d'intégration explicite.

L'amélioration des méthodes d'intégration conduit alors à une deuxième génération de simulateurs qui arrivent à maturité dans les années 1970. De nombreux logiciels ont été développés: le plus connu est SPICE (Nag.75), mais on peut aussi citer IMAG3 développé dans notre équipe (Rey.78).

I.2.3 UNE TROISIEME GENERATION DE SIMULATEURS

Une remarque s'impose à propos de la simulation électrique: en général, à un instant donné, il y a seulement une faible partie du circuit qui est active; et en général, plus la taille du circuit est importante, plus le "pourcentage d'activité" dans ce circuit diminue. Il est assez naturel d'essayer d'exploiter cette grande inactivité dans le circuit: on en vient alors à des méthodes de partitionnement, de décomposition des systèmes étudiés, ces décompositions pouvant être effectuées au niveau du système algébro-différentiel de départ ou au niveau du sytème non linéaire ou encore au niveau du système linéarisé (Has.81). Les premiers travaux sur les méthodes de décomposition ont été

concrétisés par le fameux MOTIS (CGK.75) qui causa une petite révolution dans le domaine de la simulation électrique. Pendant un certain temps les simulateurs au niveau "timing" ont connu un certain succés d'autant plus que le fait d'avoir décomposé le système permet de les intégrer facilement dans un simulateur mixte Logique/Electrique avec un mode de séquencement évènementiel exploitant l'inactivité de certains sous-systèmes. Les inconvénients de ces simulateurs au niveau "timing" viennent de leur mauvaise aptitude à traiter les circuits avec des sous-circuits fortement couplés (par exemple lorsqu'on a des transistors interrupteurs: modèles bidirectionnels qui créent aussi des problèmes pour les simulateurs au niveau Porte Logique, ainsi que nous le verrons par la suite).

Tous les simulateurs dits "de la troisième génération" utilisent des méthodes de décomposition; et en schématisant on pourra dire que les uns seront basés sur une décomposition SPATIALE tandis que les autres seront basés sur une décomposition TEMPORELLE (Has.81), (RuD.83).

a/ APPROCHE SPATIALE

On garde la convergence et la stabilité de l'approche standard dont c'est en fait l'évolution naturelle.

Le circuit est décomposé en sous-circuits si possible faiblement couplés mais le contrôle du temps, lui, reste centralisé. Globalement, le temps avance de la même manière dans les sous-circuits. La période est divisée en pas (ces pas étant habituellement de longueur inférieure aux temps de montée des signaux): on parlera de Méthode INCREMENTALE. Outre une revue de ces méthodes dans (HaS.81), on pourra trouver un étude intéressante menée dans notre laboratoire (Bon.83).

b/ LES METHODES DE RELAXATION

On parlera aussi de méthodes indirectes ou d'approche TEMPORELLE: le principe est d'accorder un certain contrôle du temps aux sous-systèmes, ce qui permet d'exploiter la dispersion de l'activité du circuit au cours du temps: il s'agit alors d'obtenir les "ondes temporelles" pour toute la période au niveau sous-circuit.

Au niveau linéaire, on peut remplacer la factorisation de type LU utilisée dans la première approche par une décomposition de type GAUSS-SEIDEL. A chaque itération, on se contente alors de résoudre un système triangulaire. Par contre, les propriétés de convergence de l'algorithme standard sont perdues.

On peut aussi utiliser une méthode de relaxation au niveau du système non linéaire, mais les simulateurs les plus récents attaquent le problème au niveau du système différentiel.

Pour ce qui est des réalisations existantes, on peut citer le plus connu de ces simulateurs de troisième génération: RELAX, développé à partir des idées de Sangiovanni-Vincentelli (LRS.82), (LeS.83).

I.3 LA SIMULATION DISCRETE

Par simulation discrète, on entend la simulation de deux états (0 ou 1) ou davantage.

De très nombreux simulateurs logiques ont été réalisés durant les vingt dernières années. Pendant un certain temps on est resté au niveau "PORTE LOGIQUE" qui est d'ailleurs toujours très utilisé.

On a cherché ensuite à monter dans l'échelle d'abstraction, ce qui a donné des simulateurs aux niveaux Transfert de Registres, puis aux niveaux algorithmique et système.

A l'opposé, si les simulateurs au niveau Porte Logique ont donné toute satisfaction pour les technologies bipolaires ou alors pour les conceptions bâties autour de la notion de porte logique (réseaux prédiffusés par exemple); il n'en est pas de même dès que l'on veut construire un circuit utilisant toutes les possibilités de la technologie MOS et en particulier les "transistors interrupteurs", que l'on ne peut plus modéliser efficacement avec des portes.)

I.3.1 SIMULATION AU NIVEAU PORTE:

UNE RECHERCHE CONSTANTE DE L'EFFICACITE MAXIMALE ...

La simulation au niveau porte sert à vérifier le comportement logique d'un système, mais aussi son comportement temporel avec la détection des "aléas" et des "courses critiques".

(Les simulateurs à ce niveau sont aussi des outils très employés dans le domaine du test mais ce n'est pas notre propos ici).

Les premiers simulateurs de ce type utilisaient un séquencement statique (SeF.62). Le premier simulateur "dirigé par évènements" est imaginé par E. Ulrich (Ulr.65).

Depuis les méthodes ont été sans cesse améliorées.

On pourra trouver un très bonne description des simulateurs logiques classiques dans (BrF.76).

I.3.1.1 CARACTERISTIQUES

PARTICULARITES DE LA SIMULATION AU NIVEAU PORTE LOGIQUE

- Les éléments de base sont les portes: elles correspondent aux opérateurs logiques ET,OU ,etc ..., toutes opérations pouvant facilement être exécutées sur la plupart des calculateurs. On a vu que l'évolution actuelle conduisait à une inflation du nombre d'éléments primitifs, ainsi d'ailleurs que du nombre de valeurs logiques: cela reflète en fait l'inadéquation d'un simulateur au niveau porte pour certains modèles, nous le verrons plus loin.

- Modélisation du comportement temporel:

  Les comportements temporels sont, eux-aussi, plus ou moins évolués. Ainsi, certains simulateurs travaillent avec des retards nuls; d'autres avec des retards "unité", des retards variables ou même des retards aléatoires. Il en est aussi qui distinguent le temps de montée et le temps de descente d'un signal. Ces informations sur les retards peuvent provenir d'une simulation électrique ou de l'analyse sur une partie déjà connue et réalisée.

- Unidirectionnalité

  On obtient le comportement des sorties directement à partir du comportement des entrées. c'est cela qui a conduit au séquencement évènementiel.

- les connections sont passives et sans mémoire

- Les circuits sont faiblement interconnectés car on limite le nombre de connections par sous-circuits pour des raisons de performance.

- L'activité est très localisée à la fois dans l'espace et dans le temps. De plus le pourcentage de portes actives diminue avec l'augmentation de la taille du circuit.

- Le niveau Porte Logique est intéressant pour le choix du séquencement.

  Si l'évolution dans presque tous les simulateurs récents a été d'abandonner le mode compilé pour un mode évènementiel, on peut noter que l'on retrouve un séquencement compilé dans la " machine d'IBM " dont nous parlerons plus loin.

- Enfin on peut conclure que la caractéristique essentielle de ce niveau est la recherche de l'efficacité maximale. En effet, si la puissance de calcul des ordinateurs hôtes augmente sans cesse, la taille des circuits à simuler augmente aussi et les problèmes restent identiques.

I.3.1.2 AMELIORATIONS DES METHODES CLASSIQUES

- Amélioration des structures d'échéancier (roue temporelle) (Ulr.78), (PhT.78).

- Simulation parallèle

  En effectuant des opérations sur des mots, on peut multiplier la vitesse par un facteur égal à la longueur du mot de la machine hôte.

- Techniques d'accès par table (Ulr.80)

  L'idée est de grouper certaines informations (valeurs des entrées et sortie de la porte par exemple) pour former une adresse d'entrée dans une table donnant en résultat une ou des actions à exécuter.

- Evolution des modèles:

  Comme nous allons le voir plus loin, on doit parfois augmenter le nombre d'éléments primitifs et affiner le comportement temporel des modèles.

I.3.1.3 LE PROBLEME DE LA SIMULATION DES CIRCUITS MOS

Pour la plupart des simulateurs logiques existants, les modèles de circuits sont en fait des réseaux de portes logiques dans lesquels chaque porte produit des valeurs de sortie dépendant uniquement des valeurs appliquées sur les entrées et éventuellement d'un certain état interne (informations uniquement structurelles).

Ces simulateurs logiques traditionnels s'avèrent inefficaces pour simuler correctement des circuits MOS.

Certains ont tenté de résoudre le problème de la manière suivante:

- Définition de nouveaux éléments primitifs correspondant à certaines structures MOS: PORTES TRISTATE, OPERATEURS CABLES, ELEMENTS UNIDIRECTIONNELS ou même BUS PRECHARGES ou TRANSISTORS BIDIRECTIONNELS.

- Ajout de nouvelles valeurs logiques : l'état Haute Impédance est apparu depuis longtemps mais maintenant on en vient à distinguer des valeurs basses et fortes... L'escalade a même atteint des sommets: on a vu des simulateurs à 32 valeurs logiques !

Ces "astuces" ne sont pas très intéressantes car elles nécessitent un travail fastidieux de traduction qui, fait plus grave, est source d'erreurs.

LES INCONVENIENTS SONT NOMBREUX:

- Les interconnections sont passives et ne peuvent rien mémoriser. Il faut imaginer des boucles de rétro-action pour mémoriser une information.

- Ces simulateurs ne sont pas adaptés à l'usage de la logique câblée.

- Il y a un risque d'erreur lors de la prise en compte de la bidirectionnalité. Les transformations manuelles qui sont nécessaires sont pénalisantes. Le coût devient excessif.

- D'une manière générale, on ne sait plus bien où l'on en est: le modèle ne représente plus rien pour le concepteur après toutes ces transformations artificielles.

LA SOLUTION:

Il apparait dés lors préférable d'interrompre cette escalade et de rajouter un nouveau barreau sur l'échelle d'abstraction. C'est ce qu'a fait R. Bryant avec son simulateur au niveau transistor (Bry.81).

I.3.2 LA SIMULATION AU NIVEAU TRANSISTOR

... UN CARREFOUR DE LA SIMULATION MOS

De nos jours beaucoup de concepteurs travaillant en technologie MOS préfèrent optimiser leur circuit en le concevant directement au niveau transistor sans même passer par l'intermédiaire du niveau porte.

Le modèle est adapté à la réalité physique à la fois structurellement et fonctionnellement.

VUE DU BAS DE L'ECHELLE

Il est à noter que si cet exposé introduit ces simulateurs comme une spécialisation des simulateurs logiques, on peut aussi partir du bas de l'échelle et regarder un simulateur à ce niveau comme une approximation d'un simulateur électrique. On peut d'ailleurs remarquer que l'étude mathématique d'un modèle au niveau transistor est beaucoup plus fouillée qu'au niveau porte.

Le premier de ces simulateurs a été MOSSIM1 (Bry.81)

Depuis , de très nombreux simulateurs à ce niveau ont été décrits. En ce moment ils sont dans ce qu'on pourrait qualifier de phase de jeunesse et les méthodes sont loin d'être aussi clairement définies que pour le niveau porte.

LES MODELES CLASSIQUES

EN PRATIQUE, un modèle au niveau transistor se présente comme un ensemble de NOEUDS actifs connectés par des INTERRUPTEURS actifs.

Les objets du modèle sont donc:

a) Les NOEUDS qui peuvent être:

- EXTERNES: dans ce cas ce sont des entrées du modèle correspondant à des Sources de courant donc à un "signal fort".

- OU INTERNES: ils assurent alors une fonction de mémorisation (due aux capacités), et sont caractérisés par une "taille", valeur discrète indiquant la charge en fonction des autres noeuds avec lesquels ils la partagent. Eux aussi peuvent fournir des signaux forts. Au cours de la simulation, ils pourront changer d'état logique s'ils sont connectés à un signal fort par un ensemble d'interrupteurs fermés.

b) Les INTERRUPTEURS qui correspondent au "chemin" entre le drain et la source (ou entre la source et le drain car il n'y a pas de sens prédéfini). Ce chemin est caractérisé par une conductance: les interrupteurs auront une certaine "force" pouvant prendre un nombre fini de valeurs.

SIGNAUX LOGIQUES

Un SIGNAL LOGIQUE est caractérisé par le doublet (ETAT logique plus FORCE).

L'ETAT logique correspondant à un VOLTAGE et la FORCE à la CONDUCTANCE de l'equivalent Thevenin.

LES METHODES

Venons en enfin à l'aspect qui nous intéresse dans cette étude: le séquencement de la simulation.

Les algorithmes travaillent donc sur des modèles que l'on peut voir comme des réseaux de noeuds et de transistors.

Là encore, il y a plusieurs approches:

- MOSSIM1 (Bry.81) utilise des méthodes issues de la Théorie des Graphes
- ESIM (Ter.82) utilise une technique de relaxation très empirique.
- Dans MOSSIM2 (Bry.82), R.Bryant définit une "algèbre de forces "; l'algorithme de simulation consiste alors à résoudre un système d'équations récurrentes dans cette algèbre.
- PARCHEMIN (HHL.82), calcule directement les signaux logiques en s'inspirant des méthodes numériques .

LES MODELES DU TEMPS employés sont encore beaucoup trop sommaires:

On ne se pose pas de problème d'évolution dans le temps puisqu'on cherche uniquement l'état stable du circuit. un grand effort est à porter là dessus: si les premiers simulateurs à ce niveau fonctionnaient avec des "retard unité", on commence maintenant à voir apparaitre des simulateurs à retard variable (DDS.83), (MSR.84).

En fait, leur idée consiste à isoler les parties bidirectionnelles et à simuler le reste, comme un simulateur logique classique avec un séquencement évènementiel: c'est déjà de la simulation mixte ...

I.3.3 LES SIMULATEURS AUX NIVEAUX PLUS HAUTS

Nous ne nous étendrons pas sur ce sujet qui a été déjà longuement étudié dans notre équipe avec par exemple CASSANDRE (Mer.73); LASCAR (Bor.76); LASSO (Gra.81). On pourra trouver une synthèse de ces travaux dans (Mer.80).

I.4 LES AXES DE DEVELOPPEMENT ACTUELS

Nous laisserons de côté les développements récents spécifiques au niveau électrique qui ont déjà été évoqués précédemment.

Nous parlerons d'abord de deux axes marquants dans le domaine de la simulation logique: la simulation concurrente et le "hardware spécialisé".

Ces deux approches nous semblent intéressantes car elles correspondent l'une, à une approche logicielle du parallélisme; l'autre, à une approche matérielle.

Enfin il est à noter que pour ces deux approches, si les premières réalisations touchent le niveau porte logique que l'on peut considérer comme le niveau de base; des études sont en cours pour les autres niveaux.

I.4.1 SIMULATION CONCURRENTE

La paternité de la simulation concurrente revient à E. Ulrich ((Ulr.73). C'était au départ une méthode employée pour la simulation de pannes, mais depuis le même E. Ulrich a proposé de l'étendre à la simulation "dite normale" (Ulr.83).

Décrivons en rapidement le principe: Le principe consiste à simuler plusieurs "hypothèses" en parallèle, "les unes contre les autres". En simulation de pannes, on introduit plusieurs pannes en même temps dans le modèle, alors qu'en simulation normale, on introduit plusieurs séquences d'entrée simultanément. On exploite alors le fait que les différentes séquences d'entrée auront des effets "à peu près identiques dans le circuit": les parties actives du circuit seront sensiblement les mêmes, il y aura "peu de différences" entre les divers états du modèle. D'où l'idée de ne propager que ces différences dans le modèle; ce qui permet de réaliser un gain important en temps de simulation.

I.4.2 MACHINES SPECIALISEES

Pour améliorer les performances des simulateurs, l'idée d'utiliser des architectures spécialisées est assez naturelle car lea algorithmes de simulation sont en général "hautement parallélisables".

Depuis la première réalisation d'IBM (Den.82), de nombreuses autres "machines de simulation" ont été développées, essentiellement aux niveaux Porte logique et Transistor qui sont les niveaux pour lesquels la vitesse d'exécution est la plus cruciale.

On peut ainsi citer

- La Machine de Simulation d'IBM : YSE (Den.82). Cette machine utilise un modèle du temps très rudimentaire (retard unité) et un séquencement compilé.

- La Machine de ZYCAD qui est déjà sur le marché: "Logic evaluator"

- La Machine de DAISY CORPORATION qui est également commercialisée: MEGALOGICIAN

- La Machine "Multi-niveaux": HAL de NEC (SKO.85)

- La Machine de BELL LABORATORIES: "Logic Simulation Machine" (ALM.82)

- et bien d'autres encore sont en développement ...

Dans une machine de simulation logique, seul le coeur du simulateur est cablé: il faut un calculateur hôte pour assurer l'interface avec l'utilisateur (et compiler son modèle).

Donnons un petit aperçu de l'architecture de ces machines.

Le principe de base est de traiter en parallèle les évènements ayant la même date d'occurrence. On distingue deux approches:

- L'approche: "architecture distribuée", utilisée par exemple dans les machines de ZYCAD, NEC et la deuxième machine des BELL LABS, consiste à partitionner le modèle en sous-réseaux que l'on simule en parallèle. L'affectation d'un sous-réseau à un processeur est statique (sauf dans le cas de la machine de Bell).

- l'approche: "architecture pipe-line" dans laquelle on affecte chaque tâche de la simulation à un processeur spécifique. Les évènements ayant la même date d'occurence peuvent donc etre traités en parallèle. Cette approche est adoptée par DAISY et les BELL LABS pour leur première machine.

Remarque: la machine YSE d'IBM combine, elle, les deux approches avec un réseau de processeurs parallèles dont chacun a une architecture pipe-line.

I.5 LA SIMULATION MULTI-MODES

Un simulateur multi-niveaux est basé sur une structure hiérarchique où les différents niveaux communiquent entre eux par l'intermédiaire d'une base de données commune. Chaque niveau ajoute un peu plus de précision au précédent. Au niveau de la simulation , c'est en fait la notion de macro-modèle qui est utilisée.

Dans la simulation MULTI-MODES, les différents programmes opèrent au même niveau mais la fonction est différente: les simulateurs multi-modes permettent la simulation SIMULTANEEE d'un circuit à différents niveaux d'abstraction. On mesure aisément l'intérêt, pour un concepteur, de pouvoir simuler les parties critiques d'un circuit à un "bas niveau d'abstraction" tandis que l'on simule le reste du circuit à un niveau plus élevé.

Ils représentent donc une solution au problème de la recherche d'un compromis entre le temps de simulation et la précision souhaitée.

On peut citer quelques réalisations existantes, mais cette liste est loin d'être exhaustive: SPLICE (New.80), DIANA (Dem.80), .SABLE (HiV.79), VISTA (GGP.82), MIXS (SYA.82), CMU-DA (ThN.83), DIALOG (Dem.82), THEMIS (DDS.84) ...

Aucun de ces simulateurs ne couvre l'échelle complète des niveaux: c'est l'ambition du projet CASCADE en cours de développement dans notre équipe:

I.6 LE PROJET CASCADE

Le projet CASCADE a son origine dans les travaux menés par l'équipe CAO de l'IMAG depuis une quinzaine d'années dans le domaine des langages de description de matériel et dans le domaine de la simulation, travaux sont évoqués au chapitre suivant.

L'occasion de concrétiser ces recherches s'est manifestée au moment de l'obtention d'un contrat avec la CEE regroupant différents partenaires Européens Universitaires ou Industriels. Citons ainsi le groupe PHILIPS avec en France le Centre de Technologies Informatiques (C.T.I) et R.T.C, aux Pays-Bas P.T.I, en Angleterre T.M.C; l'Ecole Polytechnique de Turin en Italie; la société SGS à Milan. Le maître d'oeuvre de ce projet est le laboratoire ARTEMIS de l'IMAG.

L'objectif du projet CASCADE est de réaliser un système intégré de CAO pour les circuits VLSI offrant une aide au concepteur à toutes les étapes du processus de conception depuis les spécifications initiales jusqu'à la réalisation physique. On pourra trouver une présentation complète de ce système dans (Mer 83).

Nous nous contenterons ici d'effectuer une énumération de quelques modules du système qui sont développés dans notre laboratoire:

* un simulateur multi-niveau hiérarchisé qui est l'objet de l'étude qui va suivre.

* un éditeur graphique (Mar.84)

  Cet éditeur permet de générer automatiquement la description textuelle en langage CASCADE d'un circuit décrit graphiquement au départ.

* un compilateur logique (Dur.84)

  Ce compilateur est en cours de réalisation. Ses principales caractéristiques sont d'être un outil modulaire et interactif. Il se compose:

  - De modules spécifiques (pour la synthèse de PLA par exemple).

  - D'un module d'implémentation "naive", qui donne un premier résultat.

  - D'un module d'optimisation.

CHAPITRE II

LA CONSTRUCTION D'UN MODÈLE : ORDONNANCEMENT

CHAPITRE II

LA CONSTRUCTION D'UN MODELE : ORDONNANCEMENT

Ce chapitre commence par une présentation rapide du langage CASCADE ainsi que l'organisation générale de la compilation.

Une partie du travail réalisé se situe dans ce cadre: il s'agit de l'ordonnancement du modèle que l'on peut définir comme la préparation statique du modèle en vue de la simulation.

Aprés avoir détaillé les mécanismes d'ordonnancement que nous avons défini et implémenté dans la version prototype du système CASCADE, nous proposons une réorganisation de tout le processus de compilation .

Note:

Dans ce chapitre et le suivant, nous utiliserons les mots "boucle" ou "bouclage" à la place du mot "circuit" habituellement utilisé en Théorie des Graphes; ceci afin d'éviter toute confusion avec le mot circuit employé dans le sens "circuit logique". De même, le lecteur habitué au vocabulaire de la Théorie des Graphes aura pû remarquer que pour désigner une arborescence, nous employons parfois à tort le mot "hiérarchie": c'est en fait pour se plier à l'usage en cours dans le domaine de la conception de circuits.

II.1 LE LANGAGE CASCADE

Le langage CASCADE est la concrétisation des travaux effectués à Grenoble menés d'abord par C. Le Faou (Lef.74) et J. Mermet (Mer.73) puis par D. Borrione (Bor.81) à l'aide de l'approche CONLAN.

II.1.1 PRESENTATION GENERALE DU LANGAGE

Le langage CASCADE permet donc de couvrir tous les niveaux d'abstraction de l'échelle définie dans le chapitre précédent. Il est de plus compatible avec les langages existants développés précédemment par l'équipe CAO de l'IMAG (Mer.83).

Nous détaillerons ici les grandes lignes du langage: pour avoir plus de détails, on pourra se rapporter à (Bor.84).

IDEE DE BASE:

UN NOYAU COMMUN ...

CASCADE doit être considéré comme une famille de langages, définis autour d'une grammaire, d'une syntaxe et d'une sémantique unique. Toutes ces notions communes définissent un noyau commun à tous les niveaux.

.. COMPLETE PAR DES SOUS-LANGAGES

Il existe un certain nombre de sous-langages pré-definis, qui regroupent des données et des porteuses, des fonctions, des procedures et des descriptions considérées commes primitives pour chaque niveau de description.

D'une manière générale, on pourra distinguer trois parties dans une description CASCADE:

- La partie "description de réseau", qui est purement structurelle.
- La partie "description de conditions"
- La partie "relations" correspondant à des équations soumises aux conditions précédentes.

CASCADE EST UN LANGAGE CARACTERISE PAR SA MODULARITE

Un module présent dans le modèle correspond à un exemplaire d'une DESCRIPTION; il peut être partitionné en plusieurs modules interconnectés, chaque "sous-module" étant lui-même décomposable ... On obtient alors une structure arborescente pouvant avoir une profondeur arbitraire.

Tous les modules peuvent contenir une partie structurelle et une partie comportementale (relations entre objets d'interface et objets locaux); ils sont interconnectés par leurs porteuses d'interface.

Les instructions de la partie comportementale sont concurrentes et non procédurales (leur ordre d'écriture n'a pas - ou du moins ne doit pas avoir- d'influence sur le résultat).

Le "contrôle", s'il y en a un, est exprimé par les primitives de contrôle (conditions, automates) choisies pour chaque sous-langage.

II.1.1.1 NIVEAUX COUVERTS

Le nombre de sous-langages n'a à priori pas de limite car le système Cascade permet à l'utilisateur de définir son propre langage a partir d'un langage de référence prédéfini.

On peut représenter ces sous-langages prédéfinis sur l'échelle d'abstraction de la manière suivante:

NIVEAU DE MODELISATION	LANGAGE
SPECIFICATION	CASPEL
SYSTEME	LASSO
COMPORTEMENTAL	LASCAR Synchrone
	LASCAR Asynchrone
RTL TRANSFERT DE REGISTRES	CASSANDRE Synchrone
	CASSANDRE Asynchrone
PORTE LOGIQUE	POLO
TRANSISTOR	CASTOR
ELECTRIQUE	IMAG

Par la suite, nous nous intéresserons uniquement à ces niveaux prédéfinis et essentiellement aux niveaux allant de IMAG à LASCAR qui sont les niveaux rattachés directement au hardware.

II.1.1.2 NOTION D'"OBJET"

Le langage CASCADE contient deux sortes d'objets : les valeurs et les porteuses.

a/ un "type de valeur" est un domaine mathématique sur lequel une ou plusieurs fonctions sont définies. Par exemple: entier, logique, réel, caractère...

b/ le "type de porteuse" est une notion plus générale. Il est défini par:

- le type de valeur qu'il peut prendre.
- sa valeur par défaut.
- les opérations susceptibles de modifier sa valeur.

Un registre, par exemple, est un type de porteuse.

II.1.1.3 PARTIE STRUCTURELLE D'UNE DESCRIPTION

Chaque description possède une interface qui est le seul moyen de communication entre un module associé à cette description et son environnement. La description d'un circuit précise donc les connexions des objets d'interface des différents modules.

II.1.1.4 PARTIE COMPORTEMENTALE D'UNE DESCRIPTION

La partie comportementale d'une description décrit les relations entre les objets internes et les objets d'interface.

Les instructions de la partie comportementale dépendent beaucoup du niveau de description. D'ailleurs, à certains niveaux comme POLO, la partie comportementale est totalement absente. Au niveau LASSO, le modèle est décrit comme un graphe de contrôle. Au niveau CASSANDRE, on pourra trouver des instructions de chargement de registre, des automates d'état finis, des instructions de connexion...

II.1.1.5 FORME GENERALE D'UNE DESCRIPTION

Une description est composée des parties suivantes, dont seule la première n'est pas optionnelle:

- L'en-tête indique dans quel sous-langage la description est écrite, nomme la description et déclare ses attributs et son interface.

- Les assertions permettent à l'utilisateur de préciser des contraintes sur les attributs et les objets d'interface, ou bien des relations temporelles entre les entrées et les sorties. Elles sont écrites sous forme de prédicat.

- Dans le corps de la description proprement dit, on trouve:

- L'indication des descriptions, fonctions et procédures externes qui seront apellées.

- La déclaration des objets internes et des boites imbriquées.

- Un ensemble d'instructions décrivant les relations entre objets et modules internes, ainsi que le fonctionnement de la description.

II.1.1.6 UN EXEMPLE DE DESCRIPTION CASCADE

L'exemple suivant décrit un multiplieur 8 bits au niveau LASCAR

```
lanref LASCAR_S

<< on indique le niveau de langage  >>

description MULTV1
<< nom de la description englobante >>
( in HORLOGE H ;  in SIGBO EM(0:7) , DEPART , RE ;
  out SIGBO SM(0:15) , OCCUPE )
<< description de l'INTERFACE >>

corps

declaration
<< declaration des variables internes >>
SIGBO S(0:7) , RS ;  REGBO A(0:7) , B(0:7) ;

externe
FULLADDV4 , AUTOMATV1 ;
unite
FULLADDV4 ADD8 (A,B,RE,S,RS) ;
AUTOMATV1 AUTO (H,EM,DEPART,S,RS,A,B,SM,OCCUPE) ;
<<Le modele est construit a l'aide de deux sous-modules qui
sont des instances des descriptions  FULLADDV4 et AUTOMATV1>>
findescription
```

Donnons aussi la description de l'additionneur 8 bits référencée par la précédente:

```
lanref LASCAR_S

description FULLADDV4
 (in REGBO A(0:7) , B(0:7) ; in SIGBO RE ; out SIGBO S(0:7) , RS)

 corps
 declare   SIGBO R(1:7) ;
 externe ADDV1 ;
 use ADDV1 ADD7 (A(7),B(7),RE,S(7),R(7)),
           ADD6 (A(6),B(6),R(7),S(6),R(6)),
           ADD5 (A(5),B(5),R(6),S(5),R(5)),
           ADD4 (A(4),B(4),R(5),S(4),R(4)),
           ADD3 (A(3),B(3),R(4),S(3),R(3)),
           ADD2 (A(2),B(2),R(3),S(2),R(2)),
           ADD1 (A(1),B(1),R(2),S(1),R(1)),
           ADD0 (A(0),B(0),R(1),S(0),RS);
<< Ici, on utilise 8 exemplaires de la meme description ADDV1 >>
findescription
```

Enfin la description de l'additionneur élémentaire (1 bit) est la suivante:

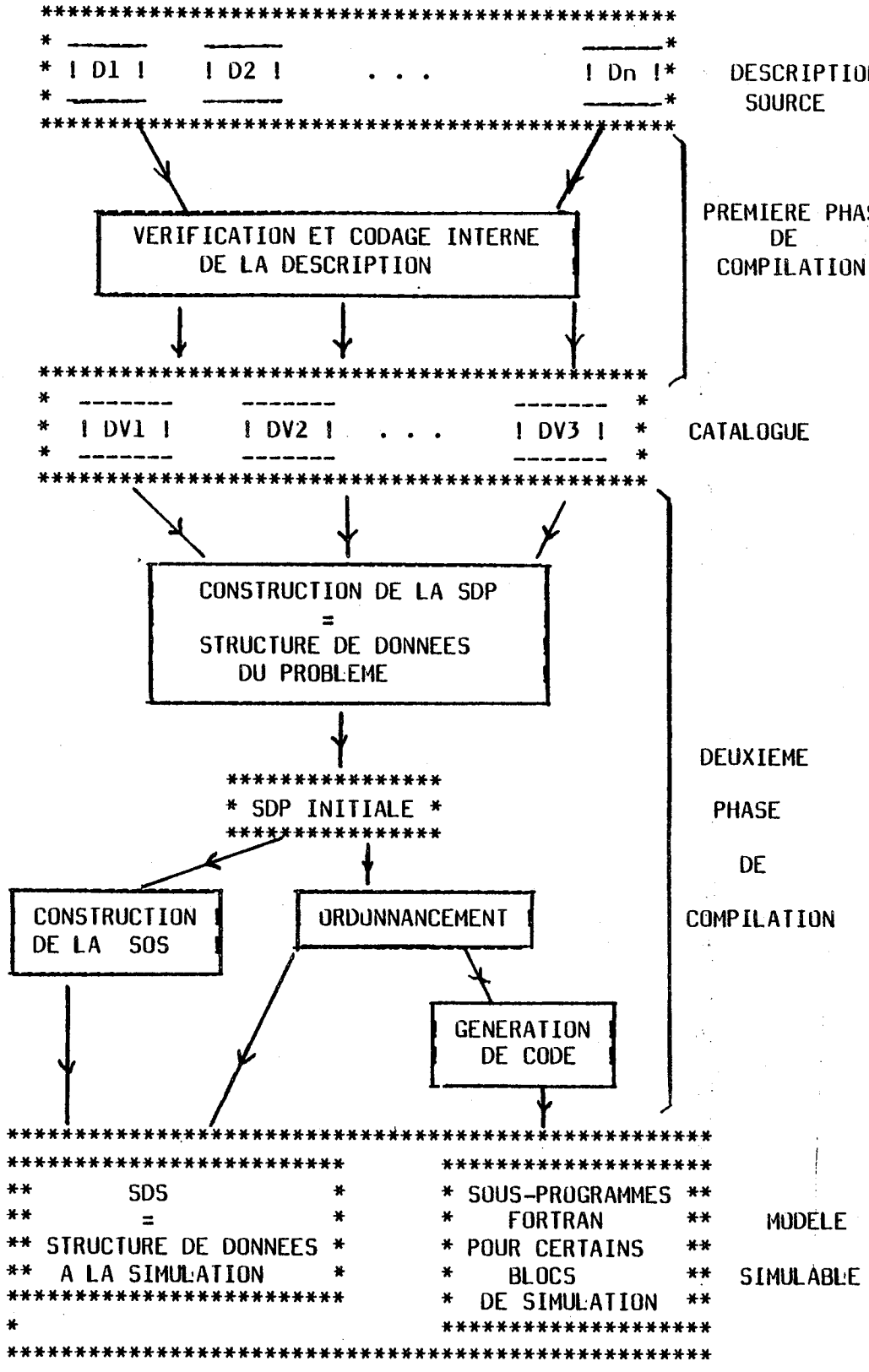
```
lanref LASCAR_S
description ADDV1
   (in  REGBO A,B ;
    in  SIGBO RE ;
    out SIGBO ST,RS)

    corps
    declare
    SIGBO S1,R1,R2 ;
    relations
    R1 .= A & B ,
    S1 .= A xor B ,
    R2 .= RE & S1 ,
    ST .= RE xor S1 ,
    RS .= R1 | R2
 findescription
```

Le langage CASCADE est la base de nombreux outils CAO, outils qui nécessitent tous une certaine compilation du langage.

Dans ce travail, nous nous intéressons uniquement à la compilation orientée simulation.

II.1.2 COMPILATION DU LANGUAGE

L'idée directrice de toute la compilation est de conserver une structure hiérarchisée pendant tout le traitement jusqu'à la simulation.


D1
D2
Dn
DESCRIPTIONS SOURCE
VERIFICATION ET CODAGE INTERNE DE LA DESCRIPTION
PREMIERE PHASE DE COMPILATION
DV1
DV2
DV3
CATALOGUE
CONSTRUCTION DE LA SDP = STRUCTURE DE DONNEES DU PROBLEME
SDP INITIALE
CONSTRUCTION DE LA SOS
ORDONNANCEMENT
GENERATION DE CODE
DEUXIEME PHASE DE COMPILATION
SDS = STRUCTURE DE DONNEES A LA SIMULATION
SOUS-PROGRAMMES FORTRAN POUR CERTAINS BLOCS DE SIMULATION
MODELE SIMULABLE

II.1.2.1 PREMIERE PHASE DE COMPILATION

Dans la première phase, on ne considère pas encore de modèle général: les descriptions sont traitées les unes aprés les autres.

Cette premiere phase a pour buts:

- De vérifier les descriptions source:

  Le VERIFICATEUR effectue une vérification syntaxique et sémantique du source CASCADE. Il génère une première structure intermédiaire.

- De structurer les segments des modèles CASCADE avant de les ranger dans un CATALOGUE (Cau.83).

  Ce catalogue permet d'ailleurs de construire l'interface entre le système CASCADE et d'autres outils de CAO. Pour chaque application, on construira, à partir de ce catalogue, une Structure de Données du Problème (SDP) spécialisée (Mer.83).

II.1.2.2 DEUXIEME PHASE DE COMPILATION

La deuxième phase de compilation prend les descriptions vérifiées dans le catalogue et produit un MODELE prêt à être simulé.

Elle comporte trois étapes:

- Première étape: Construction de la Structure de Données du Problème (SDP) initiale (Lef.84).

- Deuxième étape: Construction de la Structure Orientée Simulation (SOS). On pourra trouver dans (Rou.84) tous les détails au sujet de l'organisation de la zone valeur du modèle.

- Troisième étape: Ordonnancement avec Construction de la Structure de Données à la Simulation (SDS) et Génération de code.

II.2 MECANISMES D'ORDONNANCEMENT

La première phase pouvait être vue comme la compilation "classique" du langage. la deuxième phase est davantage orientée vers la simulation : c'est en quelque sorte le travail "statique" destiné à rendre la simulation plus efficace.

II.2.1 BUT ET LIMITES DE L'ORDONNANCEMENT

Le modèle contenu dans la SDP n'est pas directement simulable. Aucun algorithme de séquencement ne pourrait directement travailler sur cette structure dans laquelle cohabitent des modules de niveaux différents et connectés de manière différente.

Revenons un peu sur les problèmes liés au séquencement de la simulation qui ont déjà été abordés dans l'état de l'art du chapitre 1. On a vu, en particulier, la différence entre un mode de simulation continu et un mode discret ainsi que la différence entre un séquencement statique et un séquencement dynamique. Il est clair que pour avoir une simulation efficace, tous ces modes doivent pouvoir cohabiter. Il s'agit donc de trouver une structure pour la simulation ainsi qu'un mécanisme de contrôle général qui le permettent. Le choix de ce "noyau de synchronisation" sera discuté au chapitre suivant.

Faisons ici une remarque: l'ordonnancement est une opération STATIQUE (séquencement statique), et ne sert qu'à ordonner des opérations instantanées ou à détecter des bouclages instantanés.

Aussi dès que l'on fait apparaître, dans la modélisation, un retard entre l'activation de deux opérations, l'ordonnancement ne sert à rien et il faut adopter un mode de séquencement dynamique. Cet ordonnancement statique n'aura donc pas la même utilité pour tous les niveaux de modélisation.

ORDONNANCEMENT ET NIVEAUX DE MODELISATION

A un instant donné de la simulation, la détermination du comportement du modèle se fait en évaluant toutes les expressions et en exécutant toutes les fonctions décrites, valides a cet instant. Ceci est vrai quelque soit le type de modèle simulable et en particulier quelque soit le niveau de modélisation. Cependant, suivant le niveau de langage de la description, la manière d'exprimer le séquencement des actions va varier et sera plus ou moins explicite. Dans certains cas, il sera plus intéressant de "préparer" ce séquencement à la compilation; dans d'autres cas, il sera préférable de le contrôler à la simulation.

- Aux niveaux "fonctionnels", on l'a déjà vu, le choix d'un séquencement compilé est intéressant: à un niveau "Transfert de Registre" l'essentiel de l'activité provient des Horloges; il est alors assez naturel d'opter pour un séquencement statique "dirigé par horloge" avec "Liste fixée d'évènements" (Ces listes fixées correspondent au futurs blocs de simulation).

  S'il est vrai que pour une telle liste d'évènements (instructions CASSANDRE par exemple), on peut très bien prendre un séquencement tout à fait élémentaire et ne rien ordonner (A la simulation on applique un algorithme de stabilisation tel qu'il a été défini pour les parties bouclées (voir plus loin ou (BDG.84)), il sera beaucoup plus efficace de les ordonner statiquement.

- Un modèle au niveau logique "colle de plus près" à la représentation physique. La propagation de l'activité suit la structure du circuit. Il est important d'avoir à ce niveau un mécanisme de Trace sélective d'activité. En général, à ce niveau, on modélise des retards sur les portes: la gestion du temps est alors plus efficace avec un mode de séquencement dynamique. Remarquons qu'il est quand même intéressant de détecter les bouclages instantanés qui peuvent correspondre à des erreurs de conception.

ORDONNANCEMENT ET HIERARCHIE

La discussion précédente était une discussion classique pour un modèle non hiérarchisé où "tout est à plat à la simulation".

Dans CASCADE l'un des objectifs consiste à conserver une hiérarchie jusqu'à la simulation. Nous avons choisi d'ordonner aussi les modules de composition, cet ordonnancement structurel servant à construire une hiérarchie de simulation qui ne soit pas bouclée et qui permette éventuellement un séquencement statique.

NOS OBJECTIFS:

Le modèle hiérarchisé est multi-niveaux: Suivant le niveau de modélisation l'ordonnancement sera plus ou moins utile mais nous avons décidé d'appliquer un traitement de base commun à tous les modules: son but sera de rendre le modèle simulable en conservant un maximum de latitude pour le choix futur des modes de simulation, et de faire certains traitements à la compilation susceptibles de préparer le séquencement de la simulation.

II.2.2 DEFINITIONS ET STRUCTURES DE DONNEES

II.2.2.1 LA SDP: STRUCTURE DE DONNEES ARBORESCENTE DU MODELE .

La base de départ de l'ordonnancement est la SDP. Cette SDP est issue du catalogue et contient toutes les informations nécessaires à une simulation du modèle. Elle se présente comme une arborescence de modules.

Un MODULE correspond à un exemplaire de description utilisé pour la modélisation.

Un module A, a une description de référence . Cette description peut utiliser des exemplaires d'autres descriptions c-a-d d'autres modules . Ces modules sont les DESCENDANTS de A dans l'arborescence .

On dira dans ce cas que A est un MODULE DE COMPOSITION.

Dans le cas contraire, A sera un MODULE FEUILLE.

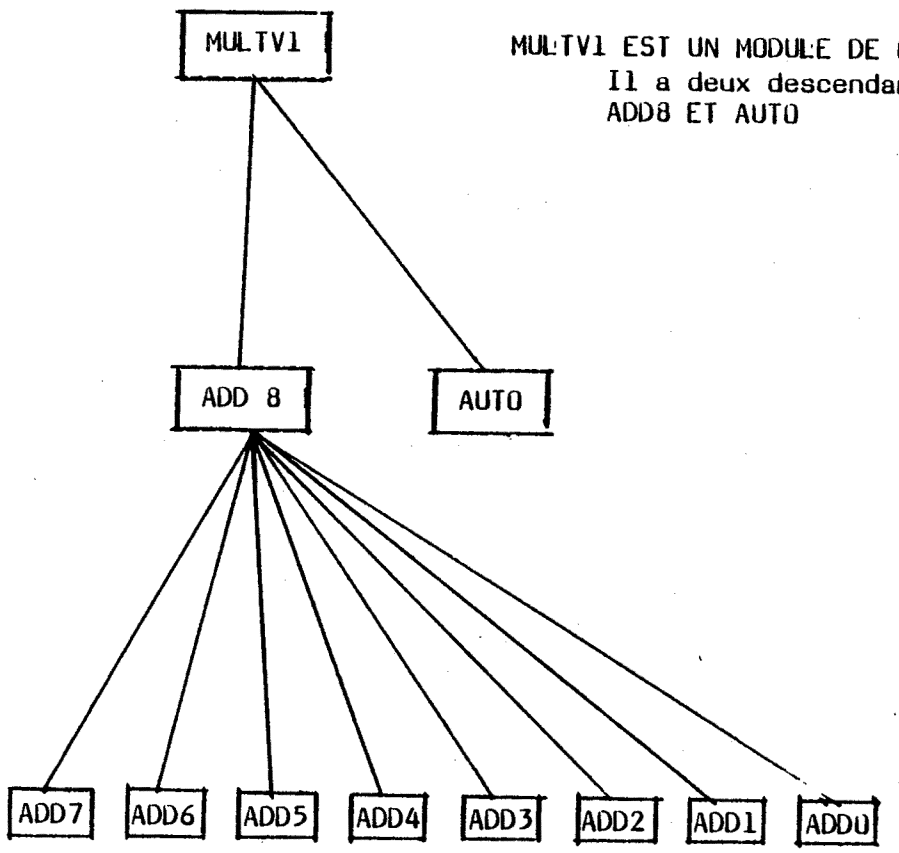
Il existe de nombreux types de modules feuille, suivant le niveau de modélisation en particulier.

La structure de cette arborescence n'est pas définitive et ne correspond pas encore au modèle simulable. En particulier elle est factorisée, en ce sens que la "descendance" d'un module décomposable n'est représentée qu'une fois si ce module est multi-référencé. De plus, pour pouvoir appliquer notre algorithme de séquencement nous allons ordonner et restructurer le modèle.

UN MODELE CASCADE

Reprenons l'exemple du multiplieur dont nous avons donné la description au niveau LASCAR

ARBORESCENCE INITIALE



II.2.2.2 POINT DE VUE INTERNE A UN MODULE:

Après ce point de vue global sur l'arborescence du modèle, nous allons maintenant avoir un point de vue interne à un module:

II.2.2.2.1 OBJETS INTERNES A UN MODULE

On a vu qu'un module pouvait avoir des "modules descendants" et qu'ainsi d'un point de vue "extérieur", il pouvait exister une sous-arborescence de racine A. Ces modules descendants forment une première classe d'objets internes: On les appelle "éléments de réseau".

Ce module peut aussi "contenir" d'autres objets "internes" qui n'apparaissent pas dans la structure arborescente. Ces objets peuvent être des variables ou des "pattes" (objets d'interface).

II.2.2.2.2 LIENS ENTRE OBJETS INTERNES A UN MODULE

Les liens entre tous ces objets internes (modules descendants y compris) peuvent être de différentes sortes: Instructions ou Connexions par synonymie.

A partir de ces liens, nous construisons une structure de travail pour l'ordonnancement: le GRAPHE D'INCIDENCE.

Le graphe d'incidence est une construction artificielle représentant les liens de précédence, entre certains objets internes à un module, à un instant donné.

La construction du graphe d'incidence à l'intérieur d'un module (d'une description) est actuellement réalisée par le catalogueur. Les sommets du graphe d'incidence peuvent être associés à tous les objets internes à valeur instantanée (Les registres n'en font pas partie).

On a un ARC d'un sommet A vers un sommet B si et seulement si l'objet correspondant au sommet A doit être évalué avant l'objet correspondant au sommet B. Ces contraintes de précédence sont déduites de certaines instructions et connexions décrites dans le module de la manière exposée dans les deux paragraphes suivants:

II.2.2.2.3 INSTRUCTIONS

Il existe de nombreux types d'instructions selon le niveau de modélisation en particulier.

a/ CAS D'UNE INSTRUCTION D'AFFECTATION D'UNE VARIABLE A VALEUR INSTANTANEE: G

(une variable à valeur instantanée est en partie gauche de l'instruction).

Si on trouve en partie droite de l'instruction:

- Un objet S de type signal non affecté d'un retard:

  On a alors un lien d'incidence (arc du graphe d'incidence) allant du sommet associé à S vers le sommet associé à G.

- Une patte d'un élément de réseau: E, non affectée d'un retard:

  On a alors un lien d'incidence allant du sommet associé à E vers le sommet associé à G.

- Tout autre objet en partie droite ne produit pas de lien d'incidence.

b/ CAS D'UNE INSTRUCTION D'AFFECTATION D'UNE PATTE D'UN ELEMENT DE RESEAU E

Si on trouve en partie droite de l'instruction:

- Un objet S de type signal non affacté d'un retard:

  On a alors un lien d'incidence (arc du graphe d'incidence) allant du sommet associé à S vers le sommet associé à E.

- Une patte d'un élément de réseau E' non affectée d'un retard:

  On a alors un lien d'incidence allant du sommet associé à E' vers le sommet associé à E.

- Tout autre objet en partie droite ne produit pas de lien d'incidence.

c/ CAS D'UNE INSTRUCTION DE CHARGEMENT DE REGISTRE

Les instructions de chargement de registre sont propres aux niveaux fonctionnels. Elles n'influent pas sur l'ordonnancement car les valeurs de registre ainsi calculées ne seront utilisées qu'au cycle de calcul suivant.

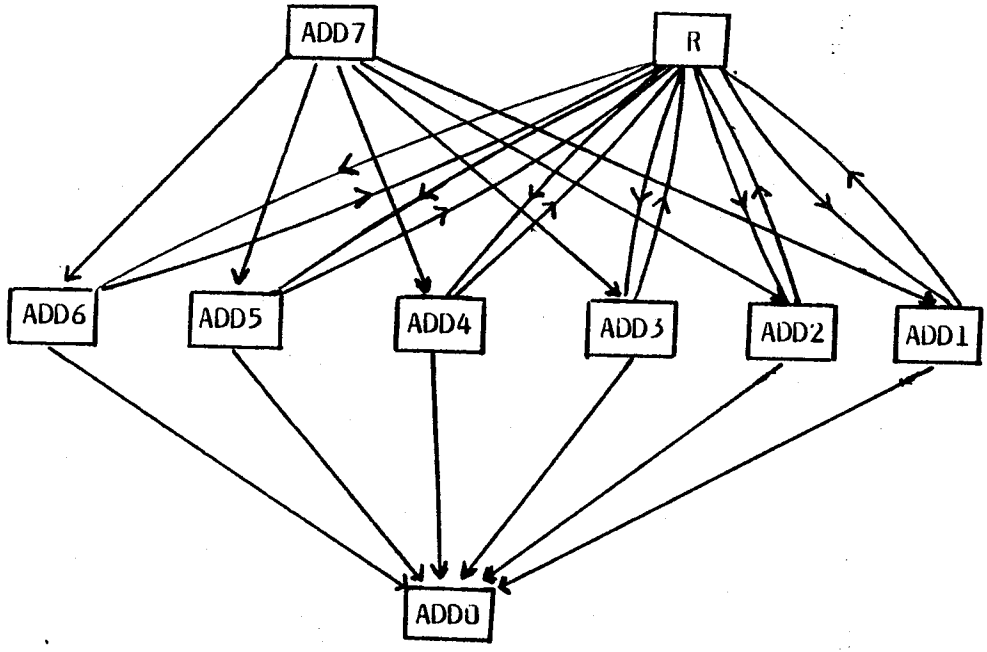
Donc si l'on ne considère que le séquencement à un instant donné, il suffit de placer ces instructions en fin de calcul (lorsque tous les signaux ont ete calculés) pour être sûr de satisfaire aux contraintes de précédence.

II.2.2.2.4 SYNONYMIES

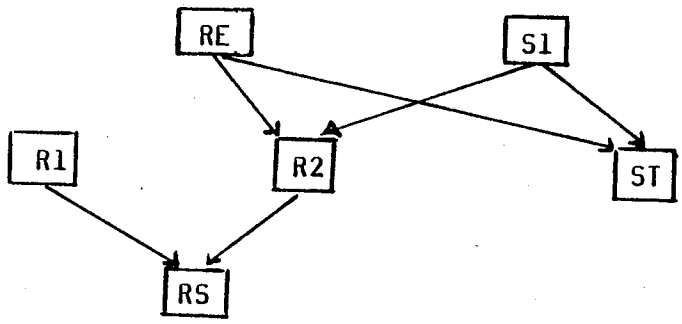
Les connexions par synonymie entre pattes d'éléments de réseau produisent des liens d'incidence entre les éléments de réseau touchés.

Par exemple, dans le cas de l'additionneur 8 bits : ADD8, on aura des liens d'incidence entre les 8 additionneurs élémentaires suivant le schéma ci-après:

GRAPHE D'INCIDENCE DE L'ADDITIONNEUR 8 BITS : ADD8



GRAPHE D'INCIDENCE DE L'ADDITIONNEUR 1 BIT CORRESPONDANT A LA DESCRIPTION ADDV1 DONNEE PRECEDEMMENT



II.2.3 PRINCIPE GENERAL DE L'ORDONNANCEMENT

Nous avons défini un algorithme d'ordonnancement travaillant sur l'arborescence de modules contenue dans la SDP du modèle.

PARCOURS DE L'ARBORESCENCE:

Le parcours de l'arborescence est réalisé "en largeur", niveau par niveau. Pour chaque noeud de l'arbre, on considère le module associé et on ordonne tous les sommets du graphe d'incidence. En général, ce graphe d'incidence comporte deux types de sommets: Les sommets associés à des modules et les sommets associés à des variables:

II.2.4 L'ORDONNANCEMENT D'UN MODULE: TRAITEMENT GENERAL

Ce traitement est effectué sur le GRAPHE D'INCIDENCE défini précédemment L'algorithme proposé combine deux modes d'exploration du graphe d'incidence:

- L'exploration "en largeur d'abord" pour la numérotation des sommets
- L'exploration "en profondeur d'abord" pour la detection et le marquage des Composantes Fortement Connexes.

L'algorithme travaille ainsi en deux passes sur le graphe d'incidence.

a/ PREMIERE PASSE

* Détection des Composantes Fortement Connexes
* Premier Traitement de ces CFC (Restructuration éventuelle du modèle)

b/ DEUXIEME PASSE

- Numérotation des sommets
- Construction d'un ORDRE sur les objets
- Preparation finale du module
  - Restructuration pour un module de composition
  - Ordonnancement des instructions pour un module feuille

II.2.4.1 DETECTION DES COMPOSANTES FORTEMENT CONNEXES

La recherche des composantes fortement connexes dans un graphe est un problème classique en Théorie des Graphes et de surcroît "bien résolu" en ce sens qu'il existe de "bons" algorithmes pour le traiter.

Un algorithme classique (en O(n2)) consiste à rechercher l'ensemble de tous les prédécesseurs et l'ensemble de tous les successeurs d'un sommet donné . L'intersection de ces deux ensembles constitue la composante fortement connexe à laquelle appartient ce sommet.

Nous utiliserons l'algorithme de TARJAN (Tar.72) qui offre l'avantage d'etre en O(n) + O(p) (n etant le nombre de sommets et p le nombre d'arcs du graphe)

Expliquons le principe de cet algorithme sans en donner toutefois de justification ce qui serait trop long. (On pourra trouver ces justifications dans (Tar.72))

PRINCIPE DE L'ALGORITHME DE TARJAN

L'algorithme de Tarjan peut être considéré comme un ensemble d'explorations récursives du graphe.

En effet, le graphe est parcouru "en profondeur d'abord" en suivant la stratégie élémentaire suivante:

ON CHOISIT UN SOMMET DE DEPART: D TANT QUE L'ON PEUT "DESCENDRE" DANS LE GRAPHE , ON DESCEND.

Nous allons maintenant détailler ce principe de descente en montrant comment il permet de repérer les composantes fortement connexes du graphe.

On considère deux piles PATH et VISIT. La première: PATH sert à mémoriser le chemin qui va du sommet D placé en son début vers le sommet S placé à sa tête . La pile VISIT, quant à elle, contient tous les sommets visités au cours de la descente récursive à partir de D.

Deux renseignements supplementaires sont associés à un sommet: S

- Son indice dans VISIT. c'est le numéro d'ordre dans l'exploretion de l'arborescence où se trouve le sommet. Nous le noterons INDICE.

- Son LIEN mémorisé dans le tableau LIEN. c'est le minimum entre l'indice cité précédemment et l'indice de certains de ses sommets successeurs. (voir algorithme de descente). C'est à partir d'un test sur ce LIEN que l'on met en évidence une CFC.

A partir du sommet placé en tête de PATH, que nous appelerons S, on cherche à "descendre" dans le graphe .

S est supposé avoir pour indice TOPA dans PATH et TOVI dans VISIT.

Remarque: TOVI est toujours supérieur ou égal à TOPA ; à un certain stade de l'algorithme, on trouvera les sommets de la CFC au sommet de VISIT, entre les indices TOPA et TOVI.

ALGORITHME DE TARJAN

```
tant que (il existe un sommet de depart) faire
    empiler sommet dans VISIT et PATH
    TOVI <--1 ; TOPA <--1 ; VISIT(1) <-- D ; PATH(1) <--D ;
    tant que (VISIT et PATH non vides) faire
       S = TETE DE PATH
       DESCEN(S)
    fin tantque
fintantque
```

DESCENTE DANS LE GRAPHE A PARTIR DU SOMMET S

```
PROCEDURE DESCEN(S)
 si  (Il existe un sommet successeur T)
         - Qui est actuellement dans VISIT
           (= on a deja visite  ce sommet a partir d'un
            autre de ses predecesseurs)
           alors   LIEN(S)   <--   MIN ( LIEN(S) , LIEN(T))
         - Qui n'a JAMAIS ete dans VISIT (c-a-d qui n'est pas
           actuellement dans VISIT et qui n'a pas deja ete
           range dans une CFC:
           alors
           on empile ce sommet dans VISIT et dans PATH
           TOVI <-- TOVI +1 ;  TOPA <-- TOPA+1 ;
           VISIT (TOVI) <-- T ;  PATH (TOPA) <-- T ;
           LIEN (T) <-----  TOVI (= indice du sommet dans VISIT)
 sinon
           (il n'existe pas de sommet successeur)
            si  LIEN(S) = INDICE(S) alors
               S et les sommets places au-dessus de S dans VISIT
               forment une CFC.
               Le cardinal de cette CFC est egal a TOVI - TOPA + 1
               - Sous certaines conditions, on applique une
                 operation de restructuration de reseau.
                 (voir plus loin).
               - On depile VISIT en otant tous les sommets de
                 la CFC.
               - On depile aussi PATH en otant S du sommet de
                 cette pile.
            sinon si LIEN(S) < INDICE(S) alors
                     - on depile PATH en otant de la tete de
                       cette pile.
                       soit T le nouveu sommet situe en tete.
                     - LIEN(T) = Min ( LIEN(S) , LIEN(T) ) .
                  finsi
            finsi
```

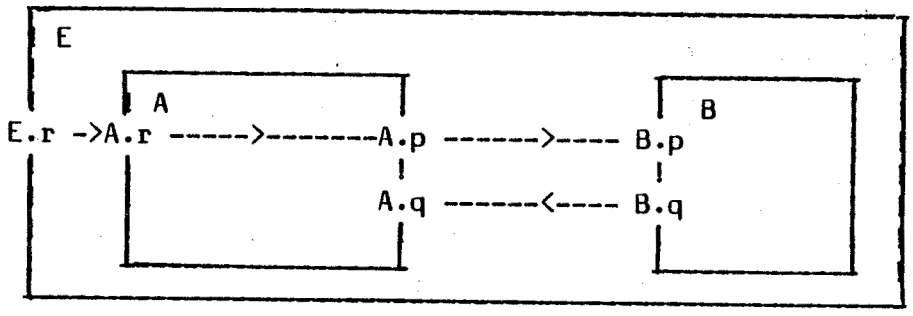
II.2.4.2 TRAITEMENT DES COMPOSANTES FORTEMENT CONNEXES:

On a vu comment les CFC étaient détectées dans le graphe d'incidence. Ces composantes fortement connexes peuvent correspondre à un ou plusieurs bouclages instantanés. Un tel bouclage peut être faux s'il peut disparaitre à la simulation; ou vrai, s'il ne peut pas disparaitre à la simulation qu'il soit du à la volonté du concepteur du modèle ou à une erreur de conception.

Détaillons maintanant cette notion de fausse boucle.

a/ FAUSSES BOUCLES DUES A LA MODULARITE

EXEMPLE



Sur ce schéma, on a représenté le point de vue de l'intérieur de E (= extérieur pour A et B) en même temps que les points de vue internes à A et B. On voit alors trés bien que les liens d'incidence entre A et B (provoqués par les instructions entre p.B et p.A d'une part et entre q.A et q.B d'autre part) créent un bouclage si l'on reste au niveau de E mais que ce bouclage disparait si l'on supprime les contours de A et B parce qu'il n'existe pas de chemin de q.A vers q.B dans le graphe d'incidence interne à A.

Lorsque l'on détecte un bouclage dans un module de composition, la stratégie suivie actuellement consiste à restructurer le réseau par les opérations ENGLOBER-ECLATER. Cela a pour effet de reporter le bouclage à un niveau plus bas de l'arborescence et de faire disparaitre ainsi certaines "fausses boucles dues à la modularité".

Remarque:

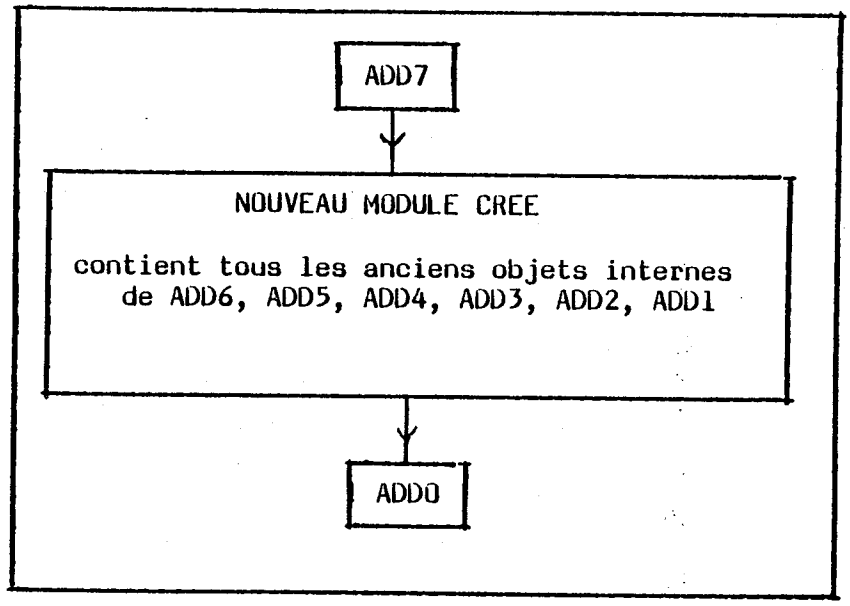
Les opérations de restructuration de réseau: ENGLOBER/ECLATER sont expliquées dans (Gen.83).

Décrivons les rapidement

- L'opération ENGLOBER crée un nouveau module en contractant 2 ou plusieurs modules, alors que l'opération ECLATER supprime le contour d'un module. Ces deux opérations sont généralement combinées.

  EXEMPLE: Reprenons le cas de l'additionneur 8 bits: ADD8 . On a détecté un bouclage dans le graphe d'incidence et on applique les opérations ENGLOBER/ECLATER sur les 6 additionneurs.

  On obtient alors le schéma suivant:



b/ BOUCLAGE DANS UN MODULE TERMINAL

Lorsque l'on détecte un bouclage dans un module terminal, il peut s'agir d'une vraie boucle ou d'une fausse boucle.

Les fausses boucles peuvent être dues aux tableaux ou aux conditions

- Les fausses boucles dues aux tableaux ressemblent beaucoup aux précédentes. Elles sont aussi provoquées par une modélisation trop globale. Par exemple, si l'on considère la description de l'additionneur 8 bits : ADD8, on remarque qu'il y a un bouclage dans le graphe d'incidence qui est provoqué par la modélisation de la retenue par un objet de type tableau.

  Nous avons étudié un "cassage" systématique de ces fausses boucles pendant la compilation, mais ce traitement peut être très coûteux vu la complexité des mécanismes d'indexation dans le langage.

- Les fausses boucles dues aux conditions sont plus complexes à traiter:

  En effet; statiquement le bouclage existe (il y a un circuit dans le graphe d'incidence) mais il disparait dynamiquement à un instant donné de la simulation.

  Exemple de fausse boucle due aux conditions

  Soit la description:

```
...
relation
 Si (condition) alors A .= B sinon B.=A
...
```

  Le graphe d'incidence associé est bouclé mais ce bouclage n'existera pas à la simulation car on n'exécutera jamais les 2 instructions au même cycle. Nous avons aussi étudié des méthodes pour "casser" ces fausses boucles pendant la compilation: ainsi on peut remarquer que dans certains cas, il existe un ordre statique sur des instructions d'une fausse boucle. (Dans un module feuille, le but final n'est pas de trouver un ordre sur les objets mais un ordre sur les instructions)

C'est le cas, dans l'exemple très simple déjà évoqué:

si (cond) alors A.= B sinon B.= A

on a bien un bouclage dans le graphe d'incidence mais on peut donner un ordre statique sur les deux instructions (les 2 ordres possibles sont corrects).

Là encore cette analyse statique apparaît trop coûteuse par rapport aux résultats obtenus et nous avons préféré aborder différement le problème ainsi que nous le verrons par la suite.

II.2.4.3 NUMEROTATION DES SOMMETS

Après application de l'algorithme de Tarjan et restructuration d'un module de l'arborescence, le graphe réduit obtenu est sans circuits. La numérotation des sommets est alors un problème très facile. Elle fournit un ordre partiel dans le graphe.

A partir de cet ordre partiel, nous choisissons un ORDRE total symbolisé par le tableau ORDRE:

ALGORITHME

DESCENTE "EN LARGEUR D'ABORD" DANS LE GRAPHE

NUMEROTATION DES SOMMETS ---> EMPILEMENT DANS ORDRE
QUI CONTIENT UN DES
ORDRES TOTAUX ASSOCIES
A CETTE MODELISATION

ORDRE est une pile contenant tous les sommets déjà ordonnés NOR est le sommet de cette pile alors que IO en est l'indice courant.

```
tantque IO < NOR faire
    N <-- ORDRE (IO)
    Pour tout S, sommet successeur du sommet N faire
       NP = nombre de predecesseurs non ranges dans ORDRE

       si (NP =0) alors
          ORDRE(NOR) <-- S
          NIVEAU(S)  <-- 1 + MAX (NIVEAU(P))
                  (avec P = predecesseur de S)
       finsi

    fin pour

    IO <-- IO+1

fintantque
```

II.2.5 TRAITEMENT D'UN MODULE DE COMPOSITION

Nous intéressons ici au cas d'un module de composition qui est un futur BLOC DE COMPOSITION DE LA SDS (non terminal dans la hiérarchie du modèle simulable = voir chapitre suivant) , et la stratégie de construction choisie nous impose de garder un modèle uniquement structurel à ce niveau, c-a-d ne contenant pas d'instructions de "comportement". Pour atteindre cet objectif,il s'agit donc de restructurer ce module de la meilleure manière possible suivant des critères qui, pour l'instant sont assez empiriques mais qui pourront être améliorés par des mesures de performance sur les premières versions du simulateur. En particulier, il sera très important de faire des mesures sur l'influence du nombre de modules pour un niveau de l'arborescence, ainsi que de leur taille, sur l'efficacité de la simulation.

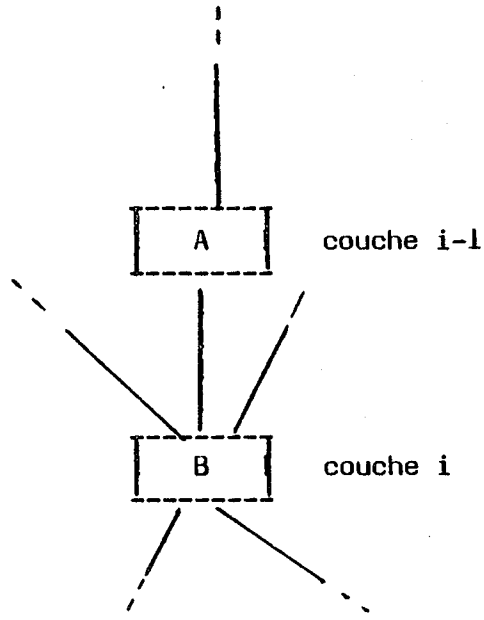
L'essentiel de cette restructuration consiste à "englober" tous les sommets du graphe d'incidence correspondant à des variables isolées de manière à ne conserver que des éléments de réseau interconnectés par "synonymie".

Actuellement la procédure adoptée est la suivante: on englobe tous les "paquets" de variables situés entre deux éléments de réseau dans la pile ORDRE. Ainsi on est sûr de ne pas créer de circuits dans le graphe réduit. Il serait possible d'optimiser ce processus suivant un certain critère: par exemple, on peut vouloir minimiser le nombre de blocs crées par ces opérations ENGLOBER tout en ayant ,bien sûr, un graphe réduit sans circuits.

Remarque: le nombre minimal de modules à créer est au plus égal au plus long chemin dans le graphe.

Le cas le plus défavorable est celui où on a des sommets avant chaque élément de réseau (ou des instructions d'affectation sur les pattes d'entrée).

Montrons que dans ce cas, il faut construire un module pour chaque "couche" si l'on ne veut pas créér de circuits dans le graphe.



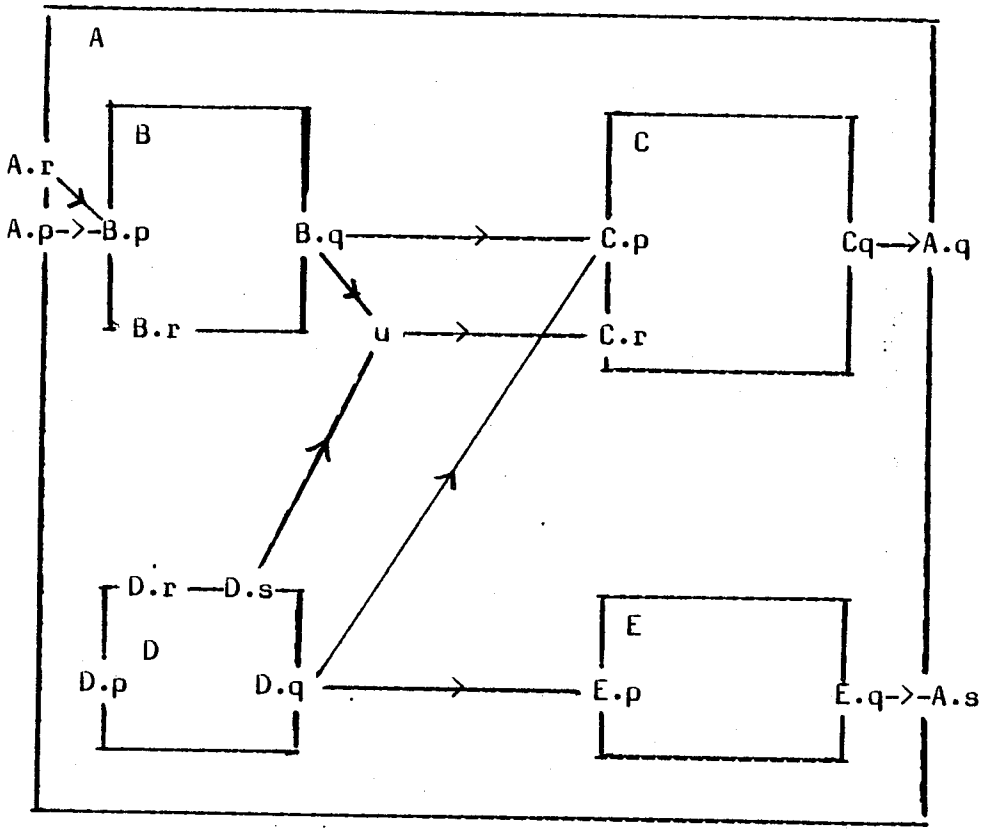
Le graphe (sans circuits) a été numéroté. B appartient à la couche i. Il existe au moins un sommet de la couche i-1 telle que l'on ait un arc de A vers B.

Supposons que l'on veuille ranger les instructions d'entrée sur B dans le module des entrées de la couche i-1 : E(i-1)

Alors dans le graphe d'incidence, on aura un arc de A vers E(i-1) car l'arc précédent de A vers B se dédouble en un arc de A vers E(i-1) plus un arc de E(i-1) vers B. On a donc un circuit entre A et E(i-1).

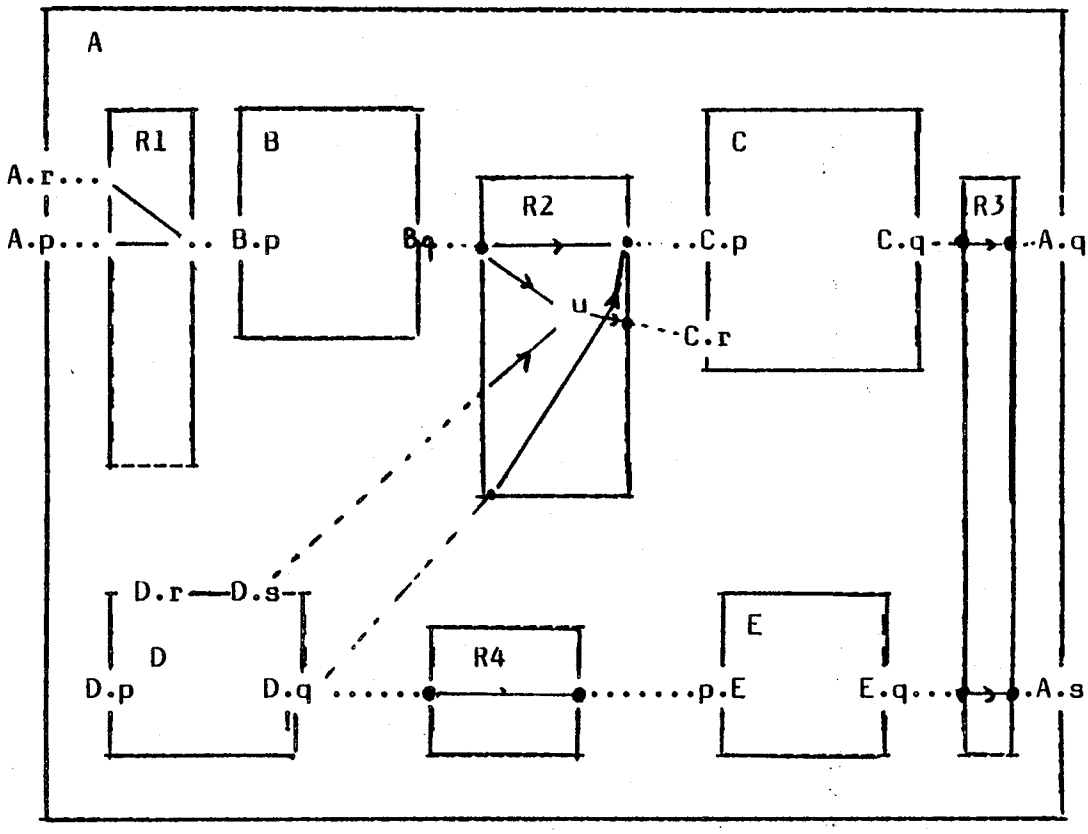
Il s'ensuit qu'il faut créér au moins un module par "couche" du graphe.

EXEMPLE DE RESTRUCTURATION D'UN MODULE DE COMPOSITION




```
...
description DA
...

   externe DB,DC,DD,DE;
...

relations
   C.p  .= B.q & D.q ,
   u .= B.q & D.s ,
   C.r  .= u ,
   A.s  .= E.q ,
   E.p  .= D.q ,
   B.p  .= A.p & A.r,
   A.q  .= C.q ,
   A.s  .= E.q
```

Aprés restructuration on obtient réseau contenant 4 nouveaux modules: R1, R2, R3 et R4.



II.2.6 TRAITEMENT D'UN MODULE FEUILLE

II.2.6.1 PRESENTATION

Une fois l'ordonnancement structurel terminé (éventuellement, il s'est traduit par une "mise à plat" de tout le modèle); Il s'agit d'ordonner " l'intérieur" des boites fonctionnelles si leur niveau de modélisation le nécessite.

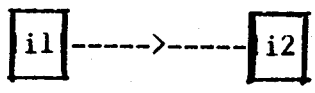
En effet, pour une boite correspondant aux niveaux Cassandre et similaires, le choix d'un mode de séquencement statique implique un travail préalable d'ordonnancement des instructions de la boite. Cet ordre sur les instructions pourra, en général, être tout naturellement déduit de l'ordre déterminé précédemment sur les objets du modèle. La dernière étape consistera à générer le code (Fortran) qui sera exécuté à la simulation (BDG.84).

Regardons d'abord ce qui se passe dans le cas où les instructions ne sont pas soumises à condition.

II.2.6.2 ORDONNANCEMENT NON CONDITIONNEL DES INSTRUCTIONS

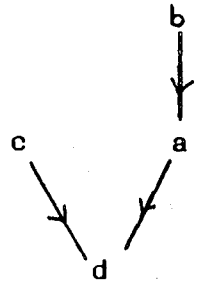
Il s'agit d'obtenir un ordre d'évaluation de ces instructions compte tenu de certaines contraintes de précedence. Ces contraintes de précedence peuvent être traduites par un graphe dans lequel on associe un sommet à chaque instruction, et dans lequel un arc relie deux instructions il et i2 si et seulement si il doit être évaluée avant i2

```
instruction il  a .= F1(b,c)
instruction i2  d .= F2(a,c)
```



On ramène alors le problème à la détermination d'un ordre (ou d'une numérotation) sur les sommets de ce graphe.

Ce graphe sur les instructions est en quelque sorte dual du graphe d'incidence sur les sommets. D'ailleurs plutôt que de représenter ce graphe pour les modules fonctionnels, on préfère actuellement travailler systématiquement pour tous les modules sur le graphe d'incidence entre sommets et déduire l'ordre sur les instructions de l'ordre sur les sommets. Pour l'exemple précédent le graphe d'incidence était:



L'ordre trouvé sur les sommets est

c, b , a , d

A partir de cet ordre il est facile de trouver un ordre d'évaluation des instructions par la procédure suivante:

Pour tout sommet S de ORDRE faire

- Si S apparaît en partie gauche d'une instruction alors
  - Ranger l'instruction dans la pile des instructions ordonnée
- finsi

finpour

On obtient ici l'ordre trivial : i1, i2 .

REMARQUE:

Les instructions de chargement de registre, ou plus généralement les instructions "retardées" seront systématiquement placées à la fin du bloc. Leur ordre n'a aucune importance puisque l'on est sûr d'avoir à disposition les bonnes valeurs des objets situés en partie droite de l'instuction.

II.2.6.3 GENERATION DE CODE

Après avoir ordonné les instructions de ce module terminal, il reste à générer du code Fortran qui, une fois compilé, sera appelé par le simulateur lorsque l'on voudra connaitre le fonctionnement de ce bloc à un instant donné de la simulation. Tous les détails techniques concernant cette génération de code pourront être trouvés dans (BDG.84). Notons quand même que les modules correspondant à des parties bouclées (rappelons que les parties bouclées ont été isolées dans des nouveaux modules) seront traités par un algorithme de stabilisation. C'est ce même

algorithme qui était appliqué systématiquement dans CASSANDRE (Bre.77) et que l'ordonnancement permet d'éviter dans les cas où il n'y a pas de bouclages.

II.3 REORGANISATION FUTURE DE LA COMPILATION

A la lumière des premières expériences de modélisation et de simulation, nous avons envisagé une refonte totale du processus de compilation.

Ce processus comprendra les étapes suivantes:

II.3.1 SEPARATION ENTRE PARTIE STRUCTURELLE ET PARTIE FONCTIONNELLE

Une des caractéristiques du langage CASCADE pour les "hauts niveaux d'abstraction" (Cassandre et au-dessus) est de pouvoir mélanger des informations structurelles et fonctionnelles à l'intérieur d'une même description.

Or pour certaines applications, dont fait partie la simulation, il est intéressant de séparer la partie fonctionnelle et la partie structurelle dans chaque module.

Cette séparation interviendra en fin de phase 1 (après la verification). Dans la LIBRAIRIE DES DESCRIPTIONS VERIFIEES ainsi obtenue (et qui correspond à une extension du "catalogue" actuel, les descriptions non terminales ne comporteront que des informations STRUCTURELLES de connexion de modules. Ces connexions ne pourront être que des SYNONYMIES.

On peut noter que ce travail est déjà réalisé dans la version actuelle mais il se passe plus tard, au cours de l'ordonnancement, et l'inconvénient est de ne pas pouvoir le décompiler en langage source.

II.3.1.1 STRATEGIES DE PARTITIONNEMENT

On a vu qu'une description générale contenait des éléments de reseau interconnectés (partie structurelle déjà décrite par l'utilisateur) ainsi qu'une partie "relations" décrivant des relations de "fonctionnement" entre objets internes. Le but du partionnement est de "mettre ces instructions de fonctionnement en boite".

Cette mise en boite des objets internes et des instructions de fonctionnement peut être faite d'une manière "radicale" en créant une seule boîte fonctionnelle.

Le risque est d'avoir ainsi un module "fortement connecté" au reste du réseau , et de créer ainsi de nouveaux bouclages dans le graphe d'incidence.

En fait le problème posé est le même que celui de la restructuration d'un module après ordonnancement, problème que nous avons abordé précédemment.

On peut alors proposer la même stratégie:

Pour tout module faire

- Créer un "module d'entrée" dans lequel on range les instructions où les pattes d'entrée du module apparaissent en partie gauche.

- Dans ce module, ranger les objets et les instructions correspondantes en remontant dans le graphe d'incidence.

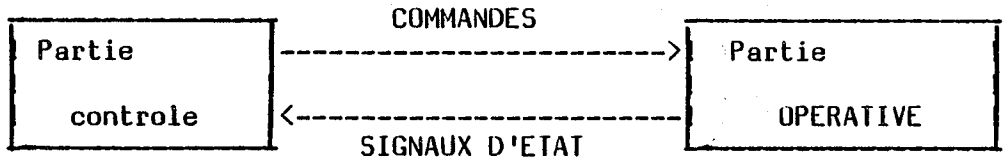
Créer finalement un module englobant les pattes de sortie du module englobant et leurs prédécesseurs.

Cette méthode ne crée pas de nouveau circuit dans le graphe d'incidence.

II.3.2 SEPARATION ENTRE PARTIE CONTROLE ET PARTIE OPERATIVE D'UN MODELE

Tout circuit (ou modèle de circuit) peut être découpé en deux parties: une partie CONTROLE et une partie OPERATIVE :

- La partie contrôle du modèle a pour fonction de fournir des commandes à la partie opérative.
- La partie opérative reoit ces commandes ainsi que des données venant de l'extérieur. Elle a pour fonction d'effectuer des traitements sur ces données suivant les valeurs des commandes.



II.3.2.1 CONSTRUCTION DE LA PARTIE CONTROLE

Le module contrôle aura pour sorties toutes les CONDITIONS régissant le comportement du reste de la description.

On le construit en "remontant" dans le graphe à partir des conditions jusqu'à obtenir des registres ou des pattes d'entrée du module englobant.

II.3.3 ORDONNANCEMENT DES MODULES (ORDONNANCEMENT STRUCTUREL)

Cet ordonnancement correspond à une simplification de l'ordonnancement actuel pour un module de composition. Le principe en sera identique:

- Détection des bouclages et restructuration éventuelle.
- Dans le graphe sans bouclages qui en résulte, on numérote les modules.

II.3.4 ORDONNANCEMENT FONCTIONNEL DES MODULES FEUILLES

Ici l'ordonnancement des instructions sera un "Ordonnancement conditionnel": on se propose d'étudier l'interaction entre l'ordre d'évaluation des instructions et la hiérarchie des conditions.

Cette hiérarchie au départ, reflète l'imbrication des conditions telle que l'a décrite l'utilisateur; comme dans un langage de programmation, elle indique aussi un certain ordre d'évaluation, qui est associé à la profondeur d'imbrication.

Cet ordonnancement conditionnel est en quelque sorte un ordonnancement temporel, il n'est pas forcément "compatible" avec l'ordonnancement "spatial" déterminé par les contraintes de précédence.

Le problème posé peut se formuler de la manière suivante:

Est-il intéressant de faire une étude statique plus poussée du modèle pour gagner du temps à la simulation au détriment de la place mémoire et d'un surcoût à la compilation ?

II.3.4.1 OPTIMISATION DU CODE GENERE

On se place donc dans le contexte d'un Module Terminal Fonctionnel. On cherche à générer le code le plus efficace possible. Cette efficacité peut être mesurée par deux critères:

- Taille du code généré (occupation mémoire)
- Efficacité à l'exécution

UNE EBAUCHE DE "MESURE"

On peut donner quelques éléments de mesure: à l'exécution l'efficacité du code sera fonction:

- du nombre de tests (de conditions) réalisés: NT .
- du nombre d'instructions exécutées: NI .

Si l'on considère que les valeurs vraies ou fausses des conditions sont equiprobables, on peut facilement déterminer, pour une module, l' espérance mathématique de NT et de NI.

Mais la meilleure mesure sera peut-être donnée par l'expérience, et après avoir expérimenté différentes stratégies d'optimisation sur de nombreux types de modules, nous pourrons faire certains choix.

Remarque: Les modules peuvent être distingués par les données suivantes:

a/ La densité du graphe d'incidence entre instructions

- Nombre de sommets de ce graphe (ie nombre d'instructions)
- Nombre d'arcs du graphe: Ce nombre caractérise l'interdépendance des instructions.

b/ Hiérarchie des conditions:

- Nombre de racines de la hiérarchie
- Profondeur des imbrications
- Nombre moyen d'instructions attachées à un sommet de cette hiérarchie

II.3.4.2 DEFINITIONS DE STRATEGIES

Une première stratégie est l'approche suivie jusqu'à présent: On ordonne toutes les instructions du module (si c'est possible) sans tenir compte de l'imbrication des conditions portant sur ces instructions On n'a alors aucune répétition d'instructions donc un gain potentiel au niveau de la taille du code généré. Par contre on a une perte potentielle au niveau de l'exécution du code avec davantage de tests de validation d'instructions à réaliser. De plus, comme nous l'avons vu précédemment on va devoir "stabiliser toutes les fausses boucles".

Une deuxième stratégie consiste à conserver autant que possible une structure de conditions imbriquées dans le code généré. Cette solution apparaît plus coûteuse à la compilation car on a des traitement supplémentaires à réaliser, mais on doit avoir un gain à l'exécution car on fait moins de tests. C'est cette deuxième approche qui sera réalisée dans le nouveau processus de compilation: nous donnons d'abord une méthode exhaustive qui développe l'arborescence de toutes les configurations possibles, puis nous donnons un nouvel algorithme imaginé par M. Pressman (PrF.84).

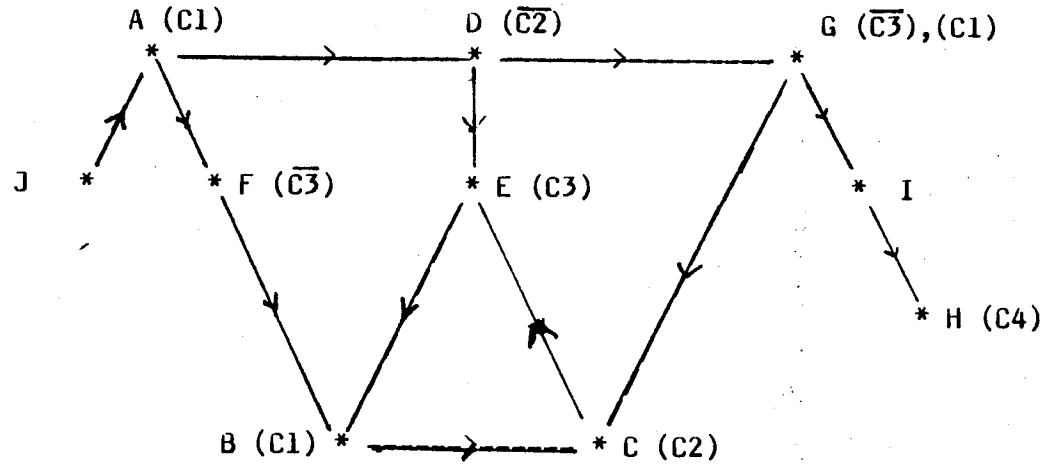
II.3.4.3 METHODE EXHAUSTIVE

Exemple: On veut ordonner la suite d'instructions suivantes:

On note par (C1), l'expression de la condition No 1

```
si (C1) alors  b <-- a  (instruction A )
               e <-- d    (B)
finsi
si (C2) alors  f <-- e   (C)
        sinon  f <-- b   (D)
finsi
si (C3) alors  d <-- f   (E)
        sinon  d <-- b   (F)
               si C1 alors g <-- f  (G)
               finsi
finsi
si (C4) alors  i <-- h   (H)
finsi
h <-- g  (I)
a <-- c  (J)
```

Le graphe d'incidence sur les instructions est le suivant:



Pour chaque sommet (associée à une instruction) on a indiqué les conditions portant sur l'instruction.

DEVELOPPEMENT DE LA HIERARCHIE DES CONDITIONS

Note: stab(B,C,E) signifie que l'on boucle sur ces 3 instructions jusqu'a stabilisation

```
si (C1) alors si (C2) alors si (C3) alors si (C4) alors
                                                J,A,stab(B,C,E),I,H
                                                    sinon
                                                J,A,stab(B,C,E),I
                                            finsi
                                      sinon si (C4) alors
                                                J,A,F,B,C,G,I,H
                                            sinon J,A,F,B,C,G,I
                                            finsi
                                finsi
                          sinon si (C3) alors si (C4) alors
                                                J,A,D,E,B,I,H
                                            sinon J,A,D,E,B,I
                                            finsi
                                      sinon si (C4) alors
                                                J,A,F,B,D,G,I,H
                                            sinon J,A,F,B,D,G,I
                                            finsi
                                finsi
                finsi
      sinon si (C2) alors si (C3) alors si (C4) alors
                                                J,C,E,I,H
                                            sinon J,C,E,I
                                            finsi
                                      sinon si (C4) alors
                                                J,F,C,I,H
                                            sinon J,F,C,I
                                      finsi
                                finsi
                          sinon si (C3) alors si (C4) alors
                                                J,D,E,I,H
                                            sinon J,D,E,I
                                            finsi
                                      sinon si (C4) alors
                                                J,D,F,I,H
                                            sinon J,D,F,I
                                            finsi
                                finsi
                finsi
finsi
```

En remarquant l'équivalence suivante (du point de vue de l'ordre):

```
si C alors B1                    B1
          B2                     si (C) alors B2
          B3        <===>                sinon B'2
     sinon B1                    finsi
          B'2                    B3
          B3
finsi
```

On peut faire "remonter" certaines instructions dans la hiérarchie des conditions. L'expression précédente se ramène alors à l'expression simplifiée:

```
J
si (C1) alors A
            si (C2) alors si (C3) alors stab(B,C,E)
                                  sinon F,B,C,G
                       finsi
                 sinon D
                       si (C3) alors E
                                  sinon F,G
                 finsi
                 B
            finsi
       sinon si (C2) alors C
                          si (C3) alors E
                                  sinon F
                          finsi
                    sinon D
                          si (C3) alors E
                                  sinon F
                          finsi
              finsi
finsi
I
si (C4) alors H finsi
```

EVALUATION: Si on note S(3) le nombre d'instructions exécutées lorsque l'on stabilise les 3 instructions bouclées, on a:
E(NT) = 4 E(NI) = 43/8 + 1/8 * (S(3))

Cette méthode peut être intéressante dans certains cas (Dans cet exemple, le résultat est tout à fait satisfaisant), mais si le nombre de conditions est très important, la mise en oeuvre devient complexe et coûteuse.

II.3.4.4 UNE AUTRE METHODE

On donne ici le principe de la heuristique mise au point par M. Pressman (Pre.84). On considère dans le graphe d'incidence des instructions, les instructions non soumises à condition et sans prédécesseur soumis à condition. On les rajoute à la description et on les supprime définitivement du graphe. On choisit ensuite une condition C et l'on rajoute à la description déjà construite toutes les instructions soumises à C. Ainsi on n'aura plus à tester C. Par contre, on peut être amené à considèrer bien d'autres conditions qui ne sont pas encore traitées.

La description désirée s'obtient de la manière suivante:

- On rajoute à la description "si C alors" et l'on supprime provisoirement du graphe toutes les instructions soumises à (non C).

- On considère l'ensemble des instructions I, sans prédécesseur soumis à une autre condition que C, soumises à C ou non soumises à condition mais précédant au sens large une instruction soumise à C et on les rajoute à la description puis on les supprime du graphe.

- Si il reste des instructions soumises à C, on risque d'être obligé de considèrer une nouvelle condition C':

  - soit parce qu'il existe une instruction soumise à C' précédant au sens large une instruction soumise à C

  - soit parce qu'il existe une instruction soumise à la fois à C et à C'

- On considère alors les cas "si C' alors" et "si non C' alors" séparément de manière à continuer d'éliminer les instructions soumises a C. Quand il n'existe plus de conditions du type précédent, on construit le graphe réduit de l'ensemble des instructions restantes qui sont soumises à C ou qui précédent une instruction soumise à C et on les ordonne. Toutes les instructions soumises à la condition C sont supprimées définitivement du graphe et on continue jusqu'à ce qu'il ne reste plus aucun sommet dans le graphe.

Pour l'exemple précédent, le résultat de l'algorithme est constitué par la liste d'instructions suivante:

```
J
si (C2 barre) alors si (C1) alors A, D
                    sinon D
                finsi
finsi
si (C1) alors A
                si (C3 barre) alors F, B
                      si (C2) alors C, G
                            sinon G
                      finsi
                sinon si (C2) alors stab ( B, C, E )
                            sinon E, B
                      finsi
                finsi
finsi
I
si (C2) alors C finsi
si (C3) alors E finsi
si (C3 barre) alors F finsi
si (C4) alors H finsi
```

EVALUATION:

$\overline{E}(NT) = 7.5$ $\quad E(NI) = 51/8 + 1/8 * S(3)$

Ce résultat est moins bon que le précédent, mais on ne peut pas vraiment en tirer de conclusions car il s'agit d'un exemple petit.

Il reste encore bien des tests à faire avant de pouvoir mettre au point une stratégie convenable. En outre, il serait intéressant de mener une étude théorique poussée sur la notion d'optimisation du code.

CHAPITRE III

MÉCANISMES DE SIMULATION

CHAPITRE III

MECANISMES DE SIMULATION

Dans le chapitre précédent, il était question de la compilation du modèle simulable. Le temps était figé et l'on s'intéressait principalement aux aspects statiques.

Ce chapitre traite de la mécanique de simulation proprement dite. On étudie l'évolution du modèle au cours du temps.

Nous avons défini des mécanismes qui utiliseront les résultats de l'ordonnancement , ils travailleront sur la structure de données qui en est issue: la SDS (Structure de Données à la Simulation) et s'appuieront dans certains cas sur les listes fixées d'évènements qui ont été générées. Ainsi, notre première approche de la simulation multi-modes est basée sur cette idée de hiérarchie préparée statiquement. Le premier protoype de ce simulateur fonctionne actuellement en couvrant les niveaux CASSANDRE, POLO et IMAG.

L'objectif d'avoir un simulateur mieux adapté au niveau POLO, nous a conduit à définir un autre mode de séquencement, purement dynamique. Nos partenaires Italiens du projet ont développé un prototype de simulateur au niveau POLO qui sera prochainement intégré au système CASCADE. Ce mode de séquencement dynamique est aussi la base d'une deuxième stratégie de simulation multi-modes.

Mais avant de nous intéresser à ces mécanismes de séquencement, nous allons détailler un peu la SDS.

III.1 LA SDS: STRUCTURE DE DONNEES A LA SIMULATION

III.1.1 UNE HIERARCHIE DE BLOCS

L'un des objectifs de la compilation était de conserver une structure arborescente pour le modèle, jusqu'à la simulation. Aussi, la Structure de Données à la Simulation se présente comme une hiérarchie de blocs.

On distingue deux types de blocs suivant leur position dans l'arborescence:

- Les blocs de composition qui correspondent aux noeuds non terminaux de l'arborescence.
- Les blocs de simulation qui correspondent aux noeuds terminaux.

Chaque bloc est identifié par un DESCRIPTEUR qui lui permet de communiquer avec son mécanisme de contrôle.

III.1.2 STRUCTURE D'UN DESCRIPTEUR DE BLOC

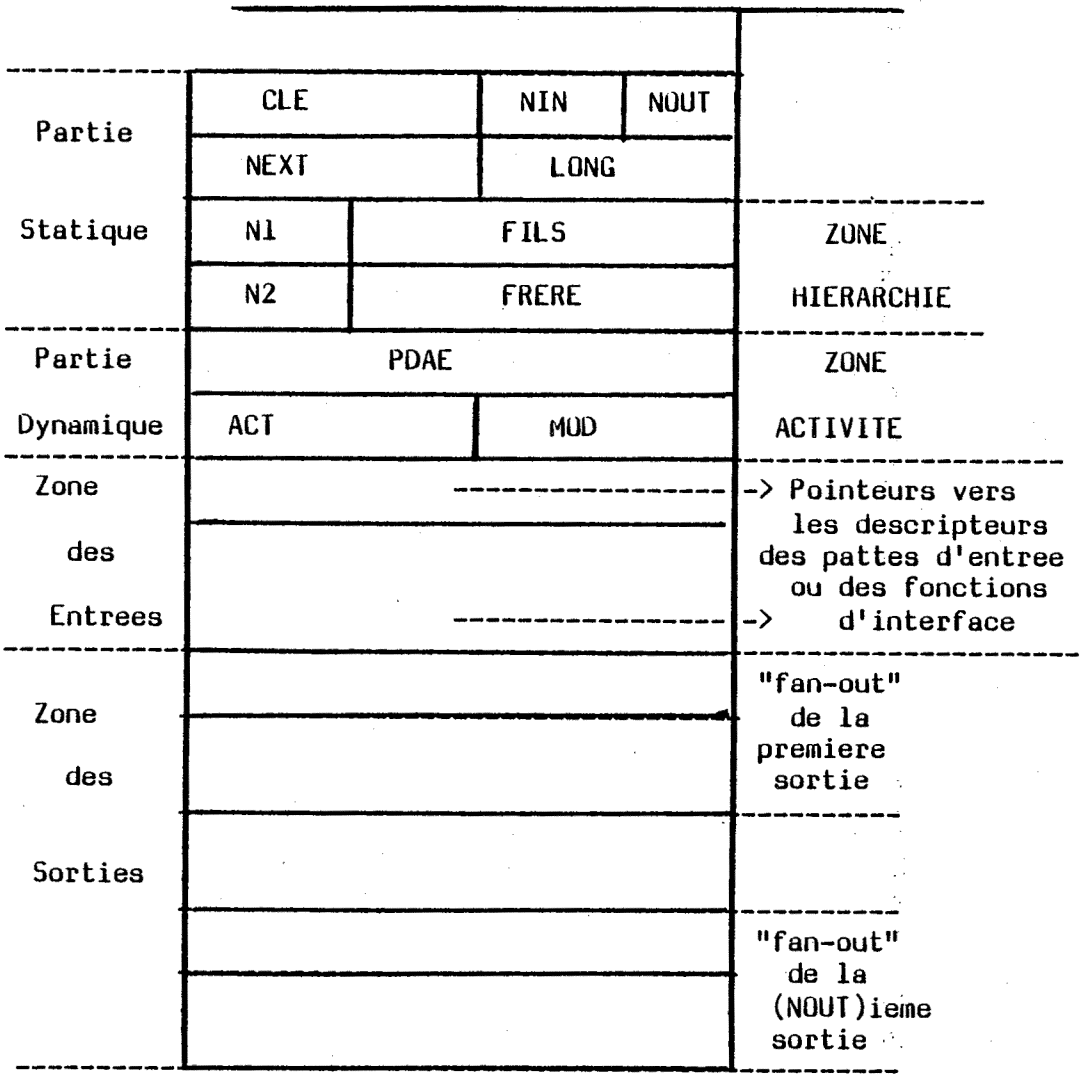
Deux objectifs nous ont guidé pour la définition de ces descripteurs de blocs .

1. Harmonisation maximale entre les différents niveaux de modélisation
2. Efficacité maximale pour chaque niveau

Ces deux objectifs étant hélas souvent contradictoires.

Détaillons d'abord la structure de base d'un descripteur de bloc. Cette structure de base peut être utilisée pour tous les types de blocs pourvu qu'ils n'aient pas de rapport avec le mode de séquencement appelé CTAB qui sera étudié plus loin.

STRUCTURE DE BASE D'UN DESCRIPTEUR DE BLOC



SIGNIFICATION DES DIFFERENTS CHAMPS

CLE Identificateur du type de bloc
peut aussi servir au sequencement en mode CTAB
(voir plus loin)

NIN : Nombre de pattes in

NOUT : Nombre de pattes out

NEXT : Chainage descripteur suivant

LONG : Longueur du descripteur

N1 : Nombre de blocs fils

FILS : Pointeur vers premier bloc fils

N2 : Nombre de blocs freres

FRERE : Pointeur vers frere suivant ou
vers Bloc pere, si c'est le dernier frere

PDAE : Prochaine Date d'Activation Enregistree
ou pointeur vers echeancier dans certains cas

ACT : Indicateur d'activite

MOD : Indicateur de modification d'une sortie

CAS PARTICULIER D'UN DESCRIPTEUR DE BLOC AU NIVEAU POLO

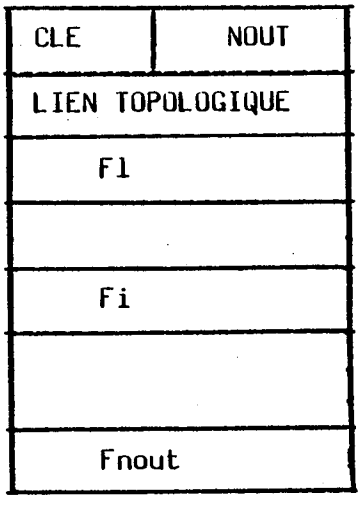
Le niveau POLO (= niveau porte) est particulier pour différentes raisons: la principale est, comme nous le verrons plus loin, que nous aurons deux modes de séquencement pour ce niveau..

Avec le premier mode, nous ne faisons pas de différence avec un bloc CASSANDRE. les descripteurs de blocs POLO peuvent donc être identiques aux précédents du moins si l'on envisage de s'en tenir à ce mode de séquencement.

Comme la structure est générée à la compilation, il est préférable de générer systématiquement une structure maximale permettant de simuler un module POLO avec l'un ou l'autre mode.

Avec le deuxième mode, que nous appelerons mode CTAB, un module POLO est systématiquement considéré comme un bloc de composition, et les objets primitifs: les portes, sont traités comme des blocs feuilles et ont de ce fait un descripteur de bloc particulier. Ce descripteur est d'ailleurs la base de travail du mode de séquencement CTAB; il contient en fait toutes les informations statiques nécessaires au séquencement. Nous verrons par la suite qu'il existe aussi des descripteurs dynamiques associé au niveau POLO.

DESCRIPTEUR DE BLOC "PORTE ELEMENTAIRE POLO"



NOUT est le nombre d'elements connectes a la sortie de cette porte ("fan-out")

--> pointeur vers descripteur dynamique

On a un mot par "fan-out"
Chaque mot est divise en deux champs
Ainsi le premier champ de Fi contient le numero d'ordre de l'entree du ieme fan-out de la porte. Le deuxieme champ contient un pointeur vers le descripteur statique de ce fan-out.

III.2 SEQUENCEMENT DES BLOCS ET GESTION DU TEMPS

On peut formuler le problème de la manière suivante:

Comment gérer le temps dans une arborescence de blocs dans laquelle les blocs feuilles travaillent suivant des mécanismes différents ?

En schématisant un peu, on peut dire que l'essentiel du travail de simulation se passe dans les Blocs de Simulation, les Blocs de Composition ne servant qu'à propager le contrôle du temps dans le modèle. D'ailleurs, dans certains cas, la hiérarchie pourra être réduite à un seul Bloc.

Il est alors assez naturel de "partir du bas" en expliquant les mécanismes de base dans les blocs de simulation; puis de "remonter" les contraintes sur la gestion du temps dans les blocs de composition jusqu'au bloc racine. Nous étudierons ainsi les différents scénarii possibles concernant la cohabitation des modes de simulation.

A PROPOS DU NIVEAU DE MODELISATION

Il s'agit donc de choisir un mode de séquencement pour chacun des blocs de la hiérarchie. Ce choix dépendra avant tout du niveau de modélisation.

Ainsi si l'on considère, par exemple, un module décrit au niveau POLO, il est possible d'utiliser différents modes de simulation comme nous le verrons plus loin.

III.2.1 LA GESTION DU TEMPS DANS LES BLOCS DE SIMULATION

Les blocs de simulation correspondent aux noeuds terminaux de l'arborescence . Ils représentent la partie fonctionnelle du modèle.

Pour ce qui est du MODE de simulation, on distingue actuellement deux types de blocs de simulation dans notre système:

III.2.1.1 LES BLOCS DE SIMULATION A LISTE FIXEE

Ces blocs sont des BLOCS DE SIMULATION COMPILES: Un sous-programme FORTRAN a été généré pour chaque bloc; il correspond à une séquence (si possible ordonnée) d'instructions.

Ces instructions sont éventuellement conditionnées et sont suivies par les instructions de contrôle de la simulation (vérification de conflit, passage ancienne valeur --> nouvelle valeur, etc) qui permettent la simulation d'opérations concurrentes. Rappelons aussi que si ce bloc de simulation a été isolé comme bloc bouclé, alors on produit en plus, des instructions de stabilisation. (BDG.84)

A l'intérieur d'un tel bloc, il n'y a plus aucun parallélisme dans l'exécution des opérations puisque tout le fonctionnement a été codé dans un langage séquentiel. C'est le type de bloc qui utilise le plus les résultats de l'ordonnancement. Dans ce cas, on peut dire que presque tout le travail de la simulation a déjà été fait à la compilation.

NIVEAUX DE MODELISATION COUVERTS

Ils correspondent principalement aux niveaux CASSANDRE et LASCAR dans lesquels on n'a conservé que la partie fonctionnelle.

Ils peuvent correspondre aussi à un bloc POLO si l'on compile ce dernier. (ce qui n'est pas intéressant sauf si l'on a fait une modélisation "sans retards").

ACTIVITE D'UN TEL BLOC:

Un Bloc de Simulation a été défini comme un "atome" dans les tests d'activité; c'est à dire que l'activation d'un tel bloc est de type tout ou rien. Le seul type d'évènement que l'on puisse définir à propos d'un tel bloc est son activation. Notons ici que si la gestion de l'activité du Bloc est alors particulièrement simple, l'efficacité de la simulation pourra s'avérer insuffisante si la taille du bloc est trop grande (ce qui entraine l'exécution d'actions inutiles). Lors du choix du mode de simulation, il s'agira de prendre en compte ce paramètre.

Un Bloc de simulation à liste fixée est actif à l'instant Tl si l'une des conditions suivantes au moins est remplie:

(a) Une entrée change à l'instant Tl

(b) L'activation a déjà été enregistrée à cette date pour une raison interne.

La première condition peut correspondre à différents cas:

- L'entrée en question peut être une "entrée primaire" (ceci dans le cas où le bloc est simulé tout seul).

- Cette variation d'entrée peut correspondre à une "propagation immédiate d'activité"

- elle peut enfin correspondre à un "top d'horloge"

La deuxième vient du fait que des éléments temporels ont été introduits dans la description: Il s'agit de RETARDS. Un retard de n (unités de temps) portant sur un objet situé en partie droite d'une instruction implique que la valeur de cet objet ne sera disponible que n unités de temps plus tard, ou plus exactement dans notre modèle, qu'il faut aller rechercher dans l'historique des valeurs la valeur de l'objet n unités de temps plus tôt.

ECHEANCE , GENERATION D'ECHEANCE

On a vu que le seul évènement pouvant se produire pour un bloc est son activation. Cependant cette activation peut être "enregistrée" pour différentes dates du futur: on parlera de différentes échéances pour le bloc.

Ces échéances peuvent avoir deux causes:

- Les horloges

  En Cassandre, un registre est chargé sous portée d'une horloge et a donc un retard égal au pas de l'horloge dans notre modélisation.

- Les retards modélisés dans la description. Le traitement associé est décrit dans l'exemple ci-dessous:

EXEMPLE

Supposons qu'un module contienne l'expression

A .= B!5! + C

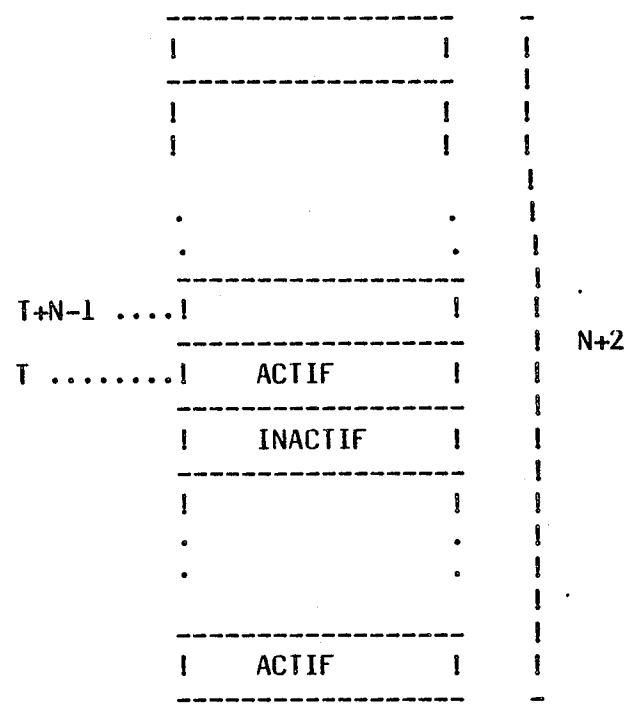
A,B et C étant des objets internes à ce module.

Si à la date t, B change de valeur, alors il faut générer une echéance pour le bloc de simulation à la date t + 5.

STRUCTURE DE L'ECHEANCIER

La structure de l'échéancier associé à un tel bloc est particulièrement simple: Il s'agit d'une sorte de calendrier d'activation avec pour chaque date, un indicateur d'activité du bloc. La structure choisie est une table circulaire ayant pour longueur le maximum des retards portant sur les objets du bloc.

T = date courante N = retard maximum dans le bloc



Dans le cas où on a un échéancier pour un bloc de simulation, c'est le champ PDAE du descripteur du bloc qui pointe dessus. (voir structure du descripteur définie auparavant).

Vis à vis du contrôle général du temps dans l'arborescence, la seule information à transmettre est la tête de cet échéancier, c'est à dire la Prochaine Date d'Activation Enregistrée pour ce bloc (PDAE).

III.2.1.2 LES BLOCS DE SIMULATION EN MODE CONTINU (BSMC)

Ils correspondent à un modèle décrit au niveau électrique (IMAG)
.

Ce modèle est écrit sous la forme d'un système d'équations algébro-différentielles non-linéaires de la forme

$$F(z(t),z'(t),t) = 0 \qquad (1)$$

Les blocs de simulation au niveau IMAG contiennent un ensemble de procédures pour

- calculer F
- calculer le Jacobien de F
- faire une transformation LU de cette matrice Jacobienne
- résoudre les systèmes triangulaires résultants

Une étude est menée dans le cadre du projet CASCADE, sur une amélioration des méthodes de résolution. (voir chap 1 ou (Bon.83)).

En peu de mots, il s'agit de partitionner F en blocs faiblement couplés susceptibles d'évoluer indépendemment au cours du temps (avec des pas différents en particulier).

Si l'on se réfère à l'état de l'art du premier chapitre, il s'agit d'une décomposition SPATIALE.

ACTIVITE D'UN TEL BLOC

Un bloc de simulation continue est actif à la date Tl, si et seulement si l'une au moins des deux conditions suivantes est remplie:

(a) Une entrée a changé à la date Tl

(b) Aprés une activation précédente, Tl a été enregistrée comme prochaine date d'activation.

Là encore, la deuxième condition nécessite quelques explications avec en particulier l'introduction de la notion de TEMPS LOCAL.

Considèrons la formulation générale du modèle mathématique de la formule (1). Le système d'équations doit être intégré avec les conditions initiales:

(t = t1
(z' = z'(t1)
(z = z(t1)

Ces conditions initiales correspondent à l'état du système à la fin de la précédente activation du bloc.

Le bloc appelant fournit la date T2 de la prochaine variation sur une entrée enregistrée pour ce bloc comme borne supérieure de l'intervalle de temps sur lequel on intègre.

A la différence d'une simulation discrète, dans laquelle on exécute des actions à une date fixée, on a besoin ici d'un "intervalle de temps" pour travailler. Dans un contexte de simulation hiérarchisée, le mécanisme de séquencement doit fournir, non plus une seule date d'activation comme dans le cas discret mais un intervalle de temps, c'est à dire en fait une deuxième date T2, borne supérieure de l'intervalle d'intégration. T2 sera dterminée de telle manière qu'aucun changement extérieur ne vienne modifier à priori les processus d'activation internes au bloc, sur l'intervalle (T1,T2). (Le calcul de T2 sera abordé en étudiant la gestion du temps dans un bloc de composition).

On peut alors considérer que l'on gère un TEMPS LOCAL à ce bloc; ce temps local: t, permettant détudier l'évolution du bloc de manière autonome entre T1 et T2.

RETOUR DE CONTROLE

On rend le contrôle au bloc appelant lorsque l'une des trois conditions suivantes est vérifiée:

(a) A LA DATE t2 (<T2) RELATIVE AU TEMPS LOCAL, on détecte l'inactivité du bloc. Ce dernier est alors marqué inactif et une activation future ne pourra être provoquée que par un nouveau changement d'entrée.

(b) A LA DATE LOCALE t2 , on détecte une variation sur une sortie du bloc (ce qui peut avoir des effets sur les autres blocs) t2 est alors enregistrée comme prochaine date d'activation.

(c) LE TEMPS LOCAL A ATTEINT SA BORNE t2 = T2

Alors T2 est enregistrée comme prochaine date d'activation.

Remarque: Sous les conditions (b) et (c) ; le bloc n'est pas encore stabilisé.

III.2.2 SEQUENCEMENT DANS LES BLOCS DE COMPOSITION

... OU COMMENT UNE MODELISATION MULTI-NIVEAUX CONDUIT

A UNE SIMULATION MULTI-MODES...

On a donc vu les mécanismes de base pour les niveaux terminaux de la hiérarchie. Les problèmes étaient bien localisés: il s'agissait de déterminer pour chaque bloc de base le meilleur mécanisme de simulation ... A la limite, si l'on s'arrêtait là, on aurait des simulateurs travaillant de manière indépendante et si possible optimale (sauf pour le cas d'un modèle POLO comme nous le verrons plus loin).

Notre objectif est plus ambitieux puisqu'il s'agit d'intégrer tous ces modes dans un simulateur hiérarchisé et d'imaginer un mécanisme général de contrôle.

La tâche la plus ardue va consister à faire communiquer entre eux tous les simulateurs de base qui travaillent avec des modes différents.

Le noyau de synchronisation doit permettre la cohabitation de différents MODES de simulation. Ces modes correspondant aux modes de base définis plus haut pour les blocs de simulation mais aussi à d'autres modes à définir pour les blocs de composition.

ETUDIONS CE QUI SE PASSE DANS UN BLOC DE COMPOSITION

Les blocs de composition sont associés aux noeuds non terminaux de l'arborescence. Ils ne contiennent pas de partie fonctionnelle. Ils sont destinés à conserver une gestion du temps hiérarchisée (les tests d'activité sont réalisés aussi "haut" que possible).

L'un des objectifs de CASCADE étant de faire de la simulation multi-modes, les blocs de composition peuvent "contenir" d'autres blocs modélisés à différents niveaux.

Essayons de clarifier un peu cette notion de niveau de modélisation pour un bloc de composition:

A priori, on peut se demander pourquoi distinguer des niveaux de modélisation dans des blocs qui se présentent tous de la même manière c'est à dire comme un RESEAU STRUCTUREL DE BLOCS INTERCONNECTES ?

En fait, au moment de la simulation, il faut s'affranchir de l'idée de niveau de language et voir davantage le "niveau de modélisation" sous l'aspect mode de séquencement.

Pour un bloc de composition les questions qui se posent sont les suivantes:

- Le contrôle est-il donné pour un instant figé ou pour un intervalle de temps ?
- Dans le second cas, il y a une gestion du temps à assurer ... comment gérer ce temps local ? comment l'intégrer dans la gestion du temps global ?
- Dans chaque cas, comment choisir le meilleur mode de séquencement des blocs fils ?

En suivant toujours notre stratégie de souplesse maximale, nous devons permettre un large éventail de choix:

Prenons le premier problème, celui de la présence d'un temps local:

Si l'on part du bas de l'arborescence pour choisir, il parait intéressant de donner la gestion d'un temps local à tous les blocs ascendants d'un bloc à temps local.

Bien évidemment, pour le bloc racine le temps local se confond avec le temps général.

III.2.2.1 BLOC DE COMPOSITION SANS TEMPS LOCAL.

Dans cette catégorie, nous rangeront les blocs de composition n'ayant pas dans leurs descendants (au sens large) de blocs avec temps local.

Remarque: La propriété: "avoir un temps local" sera définie en remontant dans l'arborescence. Elle sera déterminée à la compilation.

Un Bloc de composition sans temps local pourra avoir, comme ses fils, un mode de séquencement statique: Ce mode de séquencement s'appuiera tout naturellement sur l'ordre statique des blocs tel qu'il a été déterminé au cours de la phase d'ordonnancement.

C'est un séquencement "spatial" en ce sens qu'il est basé sur les liens topologiques entre éléments.

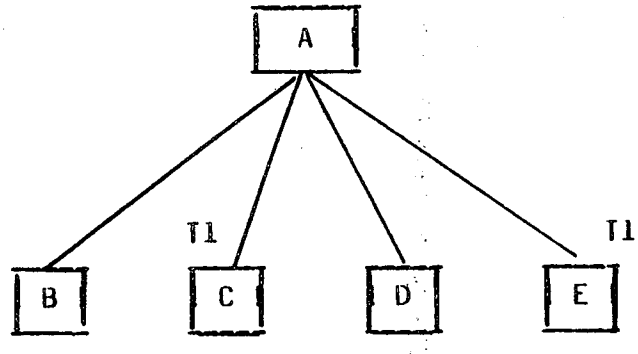
a/ MECANISME D'ACTIVATION AVEC LISTE FIXEE DE BLOCS

Lorsqu'un tel bloc est activé à la date Tl, il transmet à ses fils la condition d'activité instantanée due à un changement de l'une au moins de ses entrées.

l'algorithme d'activation est alors le suivant:

- Sélectionner dans la liste fixée des fils, le premier qui soit activable à la date Tl
- Tant qu'il existe un fils dont la PDAE est Tl faire
  - . Donner le contrôle à ce bloc
  - . Quand le contrôle revient de l'aval, mettre à jour les PDAE des fils.
  - . Si une ou plusieurs sorties ont varié alors transmettre les conditions d'activité instantanée .
  - . fin tantque
- Rendre le contrôle au bloc père

ILLUSTRATION



A la date Tl, A recoit le contrôle: Il choisit C qui est son premier fils dont la PDAE est Tl.

La propagation instantanée d'activité (trace sélective) entraine l'activation de C à la date Tl

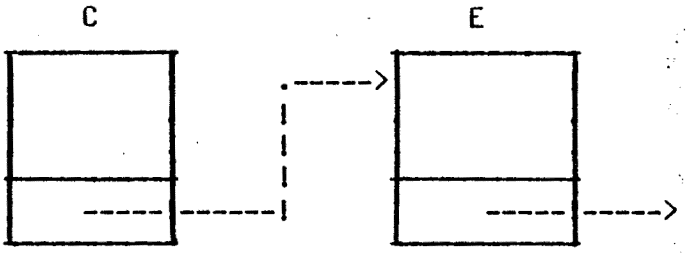
Enfin, on exécute E qui avait déjà Tl comme PDAE.

b/ MECANISME D'ACTIVATION AVEC ECHEANCIER

On peut introduire un échéancier élémentaire dans un tel bloc de composition sous forme d'un simple CHAINAGE DYNAMIQUE ENTRE BLOCS ACTIFS.

Le descripteur de bloc contient alors un pointeur vers un autre bloc dont la PDAE est identique.

Ainsi, dans l'exemple précédent, on aura un chainage direct de C vers E.



Quand le contrôle est donné à un noeud de l'arbre à une date t, on accède alors à l'écheancier qui nous donne directement TOUS les blocs fils actifs à cette date; c'est à dire C et E dans notre exemple. Il faut quand même assurer le propagation immédiate de l'activité. On a vu que l'activation de C rendait le bloc D actif: Il faudra alors chainer dynamiquement le bloc D au bloc E.

L'efficacité d'un tel échéancier par rapport à la liste fixée doit dépendre du nombre de blocs fils. Des tests seront réalisés.

III.2.2.2 BLOC DE COMPOSITION A TEMPS LOCAL

POURQUOI UN TEMPS LOCAL ?

Certains modes de simulation ont besoin d'un intervalle de temps pour travailler.

Si l'on veut conserver la hiérarchie à la simulation, il faut donc admettre l'idée d'une décentralisation du contrôle du temps. (DECOUPLAGE TEMPOREL).

Remarque: Dans tout ce qui suit, on utilisera la lettre T (majuscule) pour désigner le temps général dans l'arborescence et la lettre t (minuscule) pour désigner le temps local dans le bloc courant.

GESTION DU TEMPS LOCAL

Lorsqu'un tel bloc est activé à la date T1, on lui fournit en plus une date T2 correspondant à la prochaine date d'évènement possible (variation possible d'une entrée). La transmission de l'activité instantanée se fait de la même manière que pour un Bloc "à temps figé". Ensuite à partir de T1, on simule un évolution du temps indépendante des autres branches de l'arborescence.

On définit donc un temps local : t sur l'intervalle (T1,T2). t progresse alors pas par pas jusqu'à la date T2 au plus tard.

(T2 a été définie par le mécanisme de contrôle aux niveaux supérieurs).

Il devra cependant rendre le contrôle à t=t2 avec t2 < T2 dans les cas suivants:

(a) Tous les blocs fils ont été marqués inactifs.

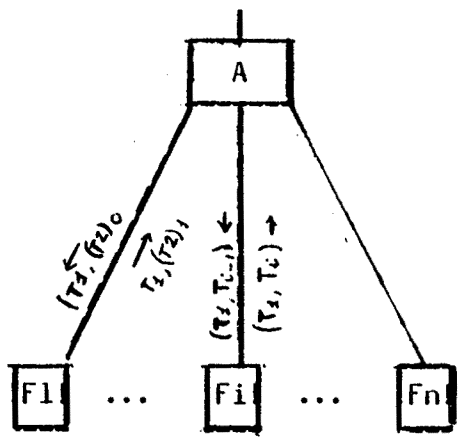
(b) Une de ses sorties a varié.

PROGRESSION DU TEMPS LOCAL

Le temps t progresse donc par pas jusqu'à la date T2 au plus tard. Soit A un bloc de composition à temps local appelé à l'instant T1 avec contrôle sur l'intervalle (T1,T2).

T2 est initialement fournie par le bloc appelant de A (bloc père) mais va évoluer avec l'activation des blocs fils de A.

On définit alors une suite de dates (T2)i avec i=1, .. n n étant le nombre de blocs fils actifs (ou qui le deviendront), de A



La suite des (T2)i est décroissante.

Le bloc A appelle d'abord le bloc F1 avec l'intervalle $(T1,(T2)_0)$. F1 rend le contrôle en indiquant l'intervalle $(T1,(T2)_1)$ avec

$(T2)_1 \leqslant (T2)_0$

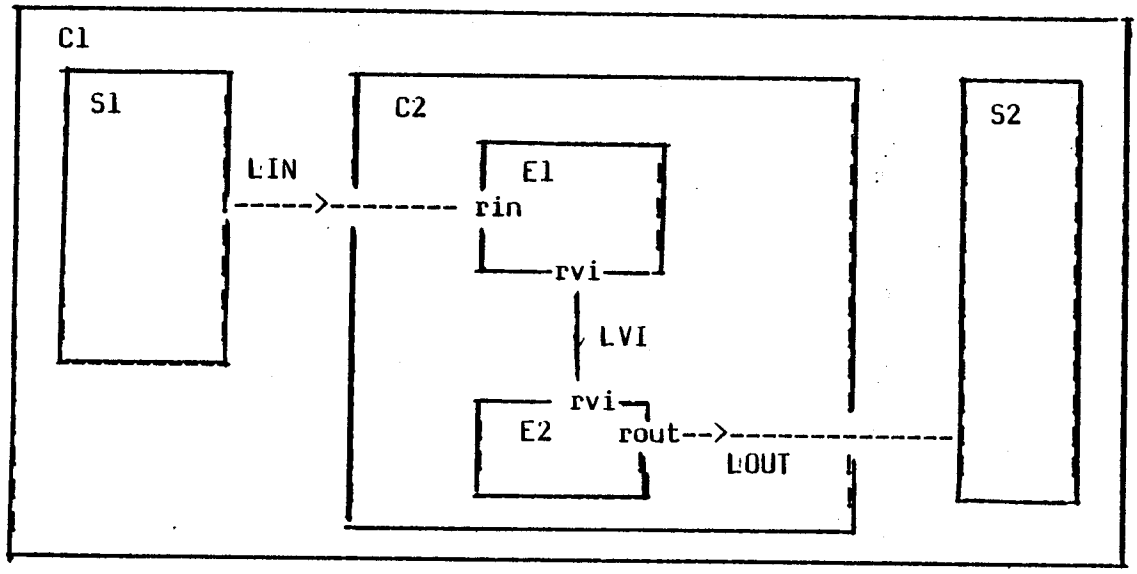
et ainsi de suite: Lorsque A appelle le bloc Fi, il lui fournit une date (T2)i-1 et récupère en retour une date $(T2)i \leqslant (T2)i-1$...

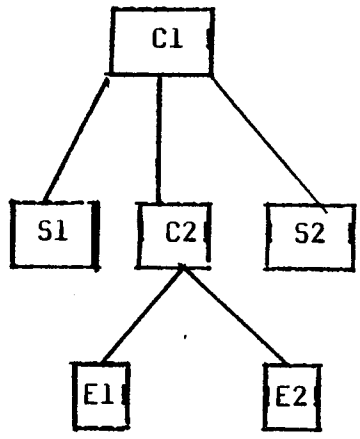
EXEMPLE

UN SCENARIO DE SIMULATION MULTI-MODE CASSANDRE/IMAG AVEC GESTION D'UN TEMPS LOCAL DANS UN BLOC DE COMPOSITION

On considère un bloc C1 modélisé au niveau CASSANDRE

C1 contient trois blocs S1, C2 et S2 , C2 étant aussi un bloc de composition CASSANDRE et S1 et S2 étant des blocs de simulation modélisés au même niveau.





Le bloc C2 contient deux blocs feuilles modélisés au niveau IMAG (et travaillant donc en mode continu) E1 et E2.

Le bloc C1 donne le contrôle du temps à la date T1, au bloc C2. On indique aussi à C2 que T2 est la PDAE générale c'est à dire que c'est la borne supérieure de l'intervalle de TEMPS LOCAL accordé.

C2 en prenant le contrôle, va démarrer son temps local t .

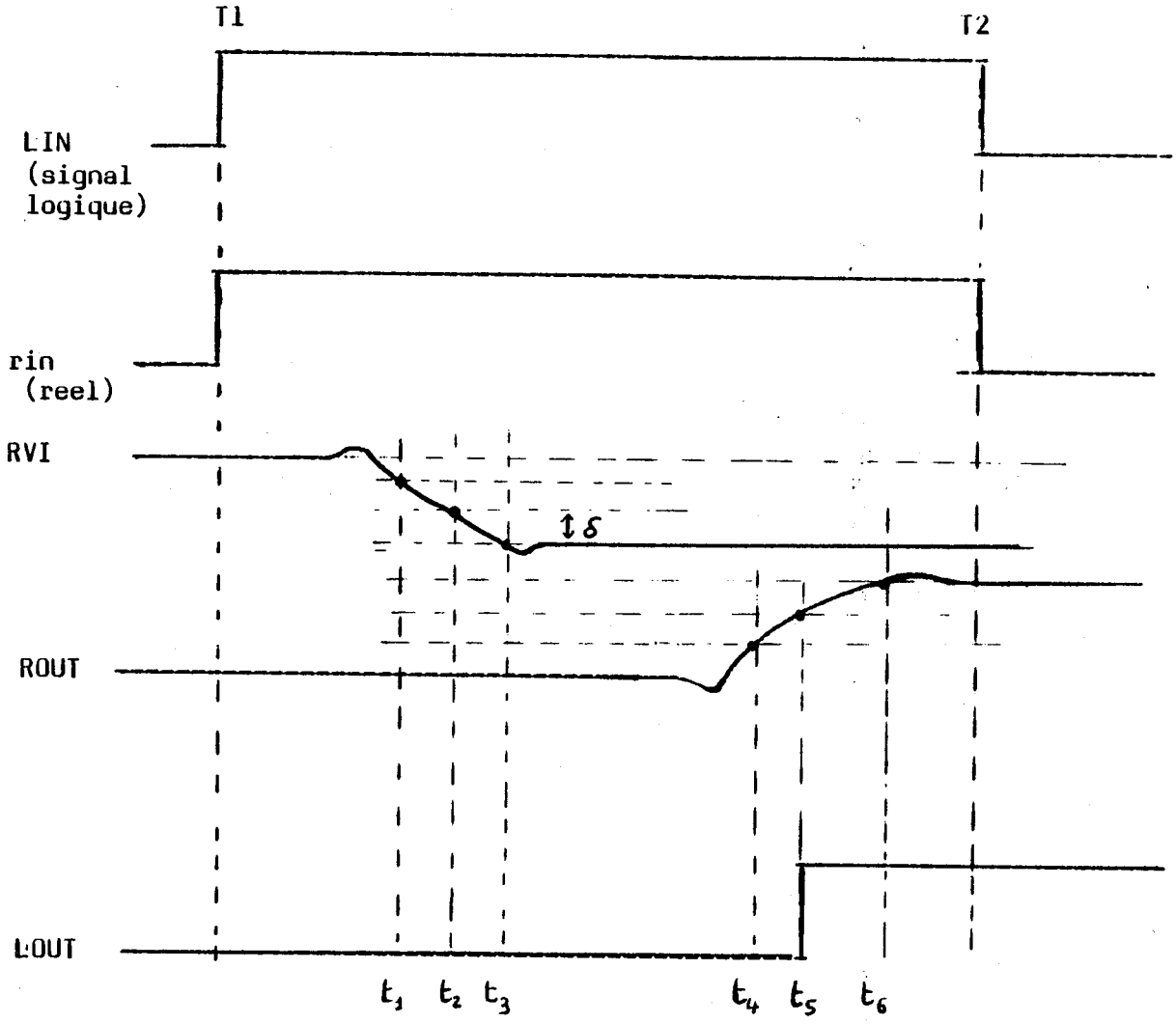
Suivons alors ce qui se passe sur le tableau: C2 va conserver le contrôle du temps(local) jusqu'à la date t5, date à laquelle on note une variation de LOUT.

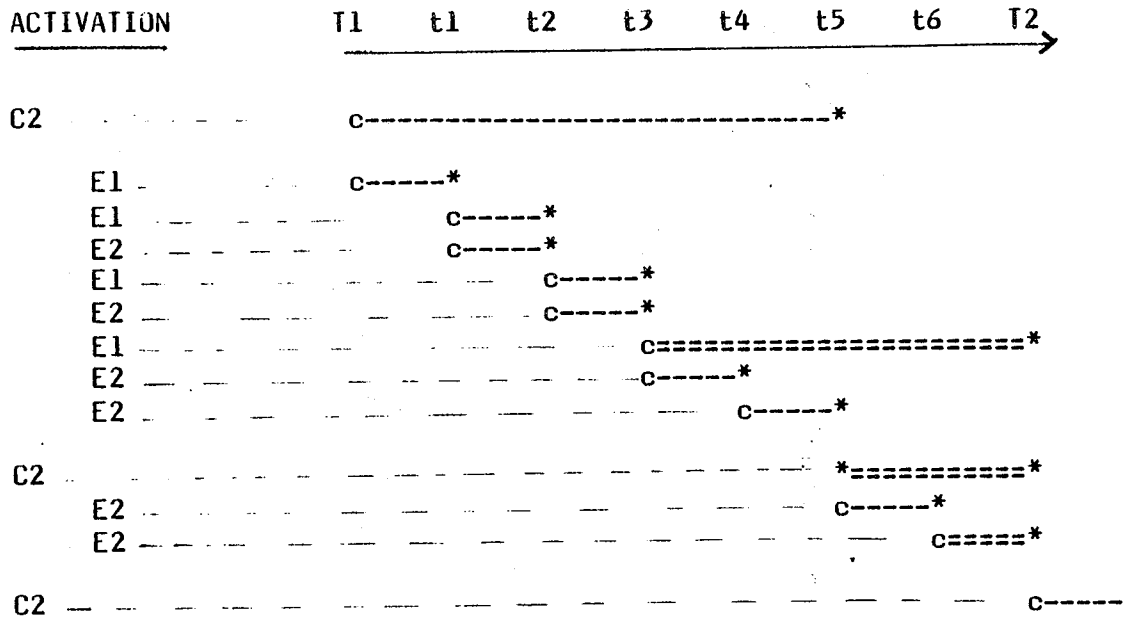
C2 rend alors le contrôle à C1 en demandant à être réactivé à la date t5 pour terminer son travail. On revient ainsi au temps globalà la date T1 à laquelle on avait laissé le bloc C1.

A un instant ultérieur de la simulation (quand le temps général a progressé jusquà t5, le bloc C1 rendra donc le contrôle à C2 sur l'intervalle (T1,T2) avec T1=t5.

C2 reprend donc l'évolution de son temps local. Il détecte une variation de LOUT à la date T1 et fait progresser son temps local t jusquà T2. Il rend enfin le contrôle à C1, l'horloge générale du système étant toujours à T1.

C1 doit alors transmettre la variation instantanée de LOUT enregistrée à la date T1.


T1
T2
LIN
(signal
logique)
rin
(reel)
RVI
↕δ
ROUT
LOUT
t1 t2 t3
t4 t5 t6



LEGENDE:

c Contrôle donné par le bloc appelant (bloc père) au bloc courant à la date courante

----* Bloc en activité jusqu'à l'évènement suivant

====* Bloc au repos jusqu'à l'évènement courant

III.2.3 LE PROTOTYPE DU SIMULATEUR CASCADE

Les idées développées plus haut ont servi de base au prototype du simulateur CASCADE actuellement en fonctionnement à l'IMAG. Ce prototype représente une première approche de la simulation multi-modes.

III.2.3.1 LES NIVEAUX COUVERTS

Ce simulateur prototype permet actuellement de simuler un modèle hiérarchisé dans lequel cohabitent des blocs modélisés aux niveaux LASCAR, CASSANDRE, POLO, et IMAG. L'extension aux niveaux CASTOR et LASSO sera réalisée prochainement.

III.2.3.2 MODES DE SIMULATION DISPONIBLES

Dans ce prototype, les blocs de simulation IMAG travaillent en mode continu classique (à noter cependant que les méthodes de simulation développées par M. Bona (Bon.83) seront très prochainement intégrées).

Les autres Blocs de simulation sont compilés et travaillent en mode Liste Fixée (Le code généré correspond à une liste fixée d'instructions). On notera ce mode TCLF (Terminal, Compilé, Liste Fixée).

Pour les blocs de composition, différents cas peuvent se produire:

a) S'il n'y a pas de bloc terminal IMAG dans les descendants on peut avoir:

- Le Mode CLF (Composition, Liste Fixée de blocs)
- Le Mode CLFECH (Composition, Liste Fixée et ECHéancier)

b) S'il y a un bloc terminal IMAG dans les descendants il faut un temps local on peut alors avoir:

- Le Mode CLFTL (Composition, Liste Fixée de blocs, gestion d'un temps local)
- Le Mode CLFECHTL (Composition, Liste Fixée, ECHéancier et gestion d'un Temps Local).

III.2.3.3 CONVERSION ENTRE LES VALEURS

Un des problèmes qui se pose en simulation multi-modes est la conversion entre les valeurs des variables qui peuvent être différentes suivant les niveaux. Par exemple lorsqu'un bloc modélisé au niveau IMAG cohabite avec un bloc modélisé au niveau POLO, il faut faire la conversion entre des valeurs réeles et des valeurs discrètes.

Dans CASCADE ces conversions sont effectués par l'appel de Fonctions de conversion (Rappelons que le langage CASCADE comporte les notions de fonctions et procédures).

III.2.3.4 EXEMPLES DE FONCTIONNEMENT A LA SIMULATION

On pourra trouver en annexe des exemples de simulation avec ce prototype.

III.2.4 RETOUR SUR LE NIVEAU POLO

Nous avons vu qu'une description POLO pouvait être simulée en utilisant un mode de séquencement compilé. Cela peut être intéressant dans le cas d'une modélisation élémentaire du temps (peu ou pas de retards sur les éléments ou alors retard unité) si l'on veut se contenter de vérifier le comportement logique du modèle. L'intégration avec le mécanisme général de contrôle hiérarchisé est alors très aisée puisque la mécanique de simulation est la même. Avec cette approche, on peut imaginer un modèle où tout est à plat (pas de hiérarchie): on a alors un seul BLOC DE SIMULATION qui peut être de taille très importante. Mais on peut aussi hiérarchiser le modèle : on a alors des blocs de composition POLO, ce qui peut permettre de gagner en efficacité en exploitant l'inactivité potentielle d'une partie du modèle.

Cependant, dès que l'on désire introduire des retards dans la description du modèle, le fait d'utiliser un tel mode de simulation peut s'avérer très pénalisant: on risque de gaspiller beaucoup de temps à calculer des élements inactifs si l'on n'a pas hiérarchisé le modèle ou à parcourir la hiérarchie dans le cas contraire.

Aussi, dans le cadre du projet CASCADE nous avons envisagé d'introduire un nouveau mode spécialement adapté à POLO, lorsque c'est ce niveau qui est le centre d'interêt principal d'une simulation. Nous appelerons ce mode mode CTAB (Composition, séquencement évènementiel dirigé par TABles).

Ce simulateur spécialisé POLO a été défini en s'inspirant des idées d'E. ULRICH avec les techniques d'accès par table pour une évaluation rapide des éléments ((BrF.76), (Ulr.80), et un mécanisme très perfectionné pour la gestion des évènements basé sur une structure de listes convergentes ((Ulr.78), (PhT.78)).

Il est actuellement en développement chez nos partenaires Italiens ((CGM.84),(BGM.84)).

III.2.4.1 DESCRIPTION RAPIDE DU SIMULATEUR

Les constituants de base de ce simulateur sont les suivants:

1. La structure statique issue de la compilation du modèle
2. L'échéancier
3. La structure dynamique reprèsentant les évènements
4. la table de contrôle
5. la table d'aiguillage.

a) STRUCTURE STATIQUE

Comme son nom l'indique, elle reprèsente les liens statiques entre les objets du modèle c'est à dire les interconnexions entre les portes.

Chaque objet du modèle possède un DESCRIPTEUR STATIQUE.

Cette structure correspond à notre SDS, ce qui permet d'envisager de faire cohabiter ce mode avec les précédents. Ce descripteur SDS a déjà été décrit précédemment.

b) L'ECHEANCIER

Le simulateur est un simulateur dirigé par évènements. La structure de cet échéancier est largement inspirée de celle d'Ulrich (PhT.78). Le principe est de faire cohabiter une structure de table pour la "roue temporelle" et des structures de listes d'évènements très sophistiquées les listes convergentes.

c) STRUCTURE DYNAMIQUE

A chaque EVENEMENT est associé un DESCRIPTEUR DYNAMIQUE. Les chaînages entre ces descripteurs représentent les flots d'information circulant entre les objets (réels ou virtuels) du modèle. Tous les descripteurs correspondant à des évènements simultanés sont chainés ensemble, le premier étant pointé par l'échéancier (Notons qu'en fait, il n'y a pas une seule liste dévènements par élément de la "roue temporelle" mais une structure à deux dimensions: liste horizontale + liste verticale (PhT.78)).

STRUCTURE D'UN DESCRIPTEUR DYNAMIQUE

CLE	POINTEUR VERS ZONE RETARDS
LIEN TOPOLOGIQUE	
	No MACHINE CONCURRENTE
	MOT D'AIGUILLAGE
DATE D'EVENEMENT ENREGISTREE	
CHAINAGE ENTRE EVENEMENTS	
CHAINAGE DE PILE	

d) LA TABLE DE CONTROLE

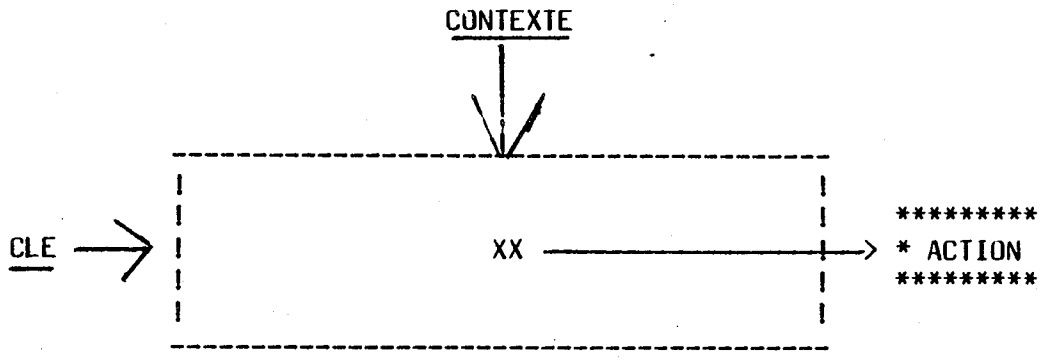
On a vu qu'il existait dans chaque descripteur, statique ou dynamique, une information appelée CLE.

Une autre information joue un rôle décisif dans le contrôle de la simulation: Il s'agit du CONTEXTE. Précisons un peu cette notion:

Le CONTEXTE indique quel type d'action on est en train de réaliser. En fait on contrôle le parcours des différentes listes.

L'idée de la table de contrôle consiste à concaténer la CLE et le contexte afin de former une adresse d'entrée dans une table qui en sortie aiguille vers une liste d'actions à entreprendre.

Cette table peut être reprèsentée schématiquement par un tableau à deux dimensions:



AVANTAGES DE LA TABLE DE CONTROLE

1. SOUPLESSE Les modifications de l'algorithme de simulation sont faciles: il suffit en effet de modifier supprimer ou ajouter des "contextes".

2. EFFICACITE A l'aide de cette table, on va de 5 à 10 fois plus vite qu'avec un arbre de décision ordinaire.

e) LA TABLE D'AIGUILLAGE

Cette table est utilisée pour un évaluation rapide des éléments: Le descripteur dynamique d'un élément contient un mot spécial appelé MOT D'AIGUILLAGE. Ce mot d'aiguillage est formé par la concaténation de certaines informations sur lélément à savoir:

- Le type d'élément (porte NAND, NOR ...)
- Un indicateur d'activité (évènement enregistré ou non)
- L'état logique des entrées
- L'état logique de la (ou des sorties)

et il est utilisé comme adresse d'entrée dans une table: La Table d'aiguillage. En sortie de cette table on a un pointeur vers une action à exécuter.

MECANIQUE GENERALE DE SIMULATION

La simulation en mode CTAB se déroule en deux passes:

Au cours de la première passe, on prend le premier évènement pointé par l'échéancier. La table de contrôle qui supervise le processus indique l'action à entreprendre: Si on a une variation de sortie d'une porte, on met à jour la valeur de cette sortie dans le descripteur de la porte puis on s'occupe de toutes les portes connectées à cette sortie (les "fan-out"): on met à jour leurs entrées et on les range dans une pile provisoire.

Au cours de la seconde passe, on regarde toutes les portes rangées dans cette pile provisoire en utilisant la table d'aiguillage. Si on peut prévoir un changement de sortie pour certaines d'entre elles, alors on les introduit dans l'echéancier. L'intérêt d'avoir ces deux passes est d'éviter tout les calculs inutiles dans le cas de plusieurs évènements sur la même porte et dont les effets s'annulent.

Dans le cas où il n'y a pas de bouclages sans retards (on notera donc l'intéret de les détecter à la compilation quelque soit le niveau de modélisation) le circuit est dans un état stable après ces deux passes.

III.2.4.2 UNE DEUXIEME STRATEGIE POUR LE "MULTI-MODES"

Nous considérerons maintenant un bloc POLO comme un bloc de composition, tous les éléments primitifs POLO étant des blocs de simulation à fonctionnement élementaire donné par une table.

Nous pouvons alors imaginer de mixer ce mode de simulation : CTAB avec le mode Liste Fixée, en considérant un modèle où le bloc racine est un bloc de composition POLO ayant pour fils des éléments primitifs POLO ou des Blocs RTL (Cassandre ou LASCAR).

Le mode maître dans la hiérarchie ainsi créé est alors le mode CTAB qui traite des descripteurs de blocs correspondant soit aux éléments primitifs POLO, soit aux Blocs Fonctionnels. Ces blocs fonctionnels peuvent être des blocs de simulation (a l'appel du bloc, on execute alors le code généré) ou des blocs de composition (a l'appel du bloc a un instant donné , on reprend alors le mode Liste Fixée dans la sous-arborescence issue du bloc en question.

Ce séquencement multi-mode "CTAB-LISTE FIXEE" est assez voisin de celui décrit dans (She.78).

III.2.5 INTEGRATION DU NIVEAU TRANSISTOR DANS CASCADE

Nos partenaires Italiens du projet CASCADE développent un simulateur au niveau transistor CASTOR. Les mécanismes de simulation sont basés sur une technique de relaxation (MSR.84). Par la suite, ce mode de simulation sera appelé TRELAX. La structure de données à la simulation est presque identique à celle de POLO de manière à permettre la simulation mixte POLO/CASTOR. Les évènements sont ainsi chaînés suivant des listes convergentes.

Actuellement, dans ce simulateur prototype, il n'y a pas de hiérarchie:

Là encore se pose le choix de savoir s'il faut considérer un bloc CASTOR comme un bloc de simulation (terminal dans la hiérarchie) ou alors comme un bloc de composition voyant les éléments primitifs de CASTOR (noeuds et transistors) comme des blocs feuille élémentaires.

Notre idée à propos du niveau Transistor, est de faire autant que possible du mode mixte CASTOR/POLO ; c'est à dire que l'on simule le plus de choses possibles au niveau POLO et l'on passe au niveau TRANSISTOR uniquement dans les cas où l'on ne peut pas faire autrement (éléments bidirectionnels). Cette idée est à rapprocher de celle des auteurs de LOGMOS (DDS.83).

Cela nous conduit au paragraphe suivant où il est justement question de clarifier notre approche de la simulation multi-modes.

III.2.6 UNE APPROCHE DE LA SIMULATION MULTI-MODES

A ce niveau de l'exposé, il serait bon de résumer nos idées pour la simulation multi-modes dans CASCADE. Le modèle est donc une hiérarchie de blocs avec la possibilité d'avoir certains niveaux de modélisation et certains modes de simulation pour chaque bloc. Les modes de base sont définis de manière à pouvoir travailler de manière efficace dans chaque bloc. Enfin on essaye de permettre un maximum de cohabitations entre les les différents modes dans la hiérarchie.

III.2.6.1 RECAPITULATION DES MODES POSSIBLES

Les tableaux suivants rappellent les modes de simulation possibles suivant les niveaux:

BLOCS DE SIMULATION

- NIVEAUX TERMINAUX DE LA HIERARCHIE -

Mode / Niveau	COMPILE LISTE FIXEE	RELAXATION	CONTINU
BLOC FEUILLE LASCAR & CASSANDRE	oui = MODE TCLF		
BLOC FEUILLE POLO	oui = MODE TCLF		
BLOC FEUILLE CASTOR		oui = Mode TRELAX	
BLOC FEUILLE IMAG			oui = mode TC

- BLOCS DE COMPOSITION -

Mode / Niveau	LISTE FIXEE SEULE		LISTE FIXEE AVEC CHAINAGE DYNAMIQUE		EVENEMENTIEL "PUR" AVEC ACCES PAR TABLE ET LISTES CONVERGENTES
	SANS TEMPS LOCAL	AVEC TEMPS LOCAL	SANS TEMPS LOCAL	AVEC TEMPS LOCAL	
BLOC DE COMPOSITION LASCAR & CASSANDRE	oui = mode CLF	oui = mode CLFTL	oui = mode CLFECH	oui = mode CLFECHTL	
BLOC DE COMPOSITION POLO	oui = mode CLF	oui = mode CLFTL	oui = mode CLFECH	oui = mode CLFECHTL	oui = mode CTAB

III.2.6.2 COHABITATION DES MODES

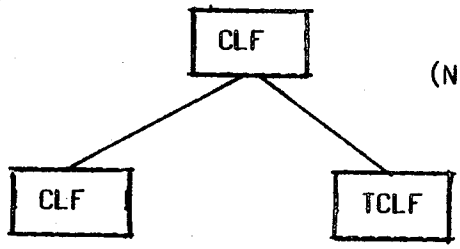
Laissons de côté les cas particuliers de modèles non hiérarchisés pour lesquels il suffit d'utiliser le mode de simulation adéquat tel qu'il a été défini dans le premier tableau.

Pour un modèle hiérarchisé, nous allons résumer les cohabitations permises entre les différents modes telles qu'elles ont été définies précédemment. On distinguera deux stratégies de base suivant le mode de séquencement choisi pour le bloc racine (mode de séquencement maitre).

Pour mettre en évidence, ces cohabitations entre modes, il suffit d'étudier un seul niveau de hiérarchie. On appelera mode maître de la simulation le mode du bloc racine.

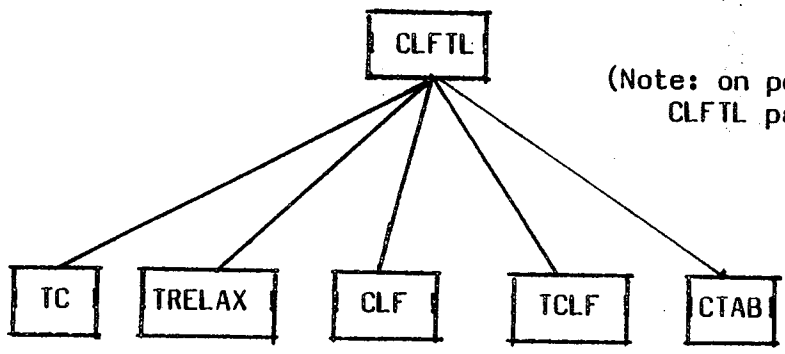
PREMIER CAS: le mode maitre est le mode Liste fixée (ou ses variantes).

1/ SANS TEMPS LOCAL



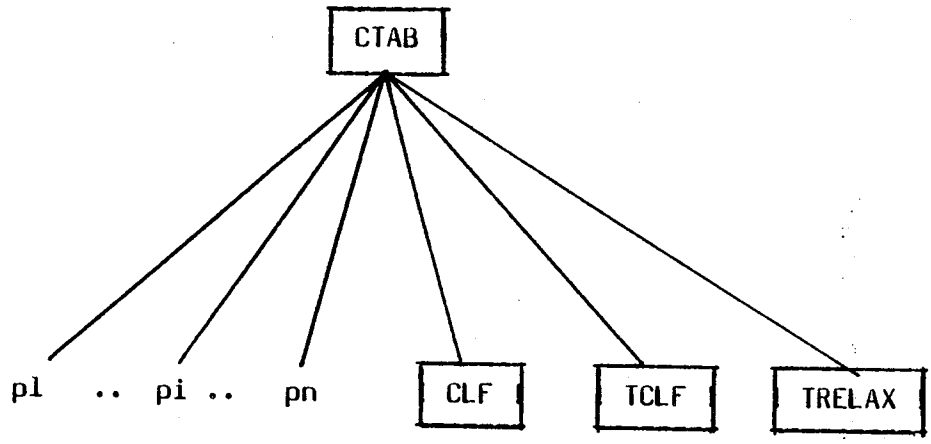
(Note: on peut remplacer CLF par CLFECH)

2/ AVEC TEMPS LOCAL



(Note: on peut remplacer CLFTL par CLFECHTL)

DEUXIEME CAS: Le mode maitre est le mode CTAB



Remarque: p1,p2, ... pn sont des éléments primitifs POLO.

III.3 LE MODE CONCURRENT DANS CASCADE

Nous avons décidé d'introduire les principes de la simulation concurrente dans CASCADE. Nous avons déjà parlé au chapitre 1 de l'idée de E. Ulrich, d'appliquer ces principes à la simulation dite "normale" (par opposition à la simulation de fautes) (Ulr.83). Le simulateur PULO avec mode CTAB, que nous venons de voir a d'ailleurs été conçu avec l'optique d'intégrer la simulation concurrente. La Structure de Données à la Simulation a été définie en ce sens.

Dans un premier temps c'est donc au niveau PULO que fonctionnera le mode concurrent CONCUR.

La Structure de Données du simulateur PULO a été prévue pour permettre la simulation concurrente (MSR.84).

La différence essentielle est qu'au lieu d'avoir un seul descripteur dynamique associé à un élément à un instant donné, on a une liste de descripteurs dynamiques. Le premier de ces descripteurs est particulier: il représente la "machine de référence" c'est `a dire l'image de l'activité correspodant à la séquence d'entrée que l'on a particularisée comme référence de la simulation. Les autre descripteurs sont associés aux "machines concurrentes" c'est à dire qu'elles représentent l'image des différences observées (pour l' élément courant) entre les différentes séquences d'entrée appliquées "en concurrence" avec la séquence d'entrée "de référence.

DESCRIPTION SOMMAIRE DE L'ALGORITHME

Le principe de base de l'algorithme est le même que celui que nous avons vu précédemment pour la simulation non concurrente (mode CTAB), tout au moins si l'on considère la simulation de la "machine de référence".

La situation est tout de même un peu plus complexe à cause des parcours des listes de machines concurrentes. On a aussi des problèmes liés à la convergence et à la divergence: On dit que l'on a DIVERGENCE pour un élément lorsque à un moment de la simulation une nouvelle "différence" apparait à ce niveau (correspondant à une séquence d'entrée qui ne s'était pas "manifestée" jusque là pour cet élément). Il faut alors créer en nouveau descripteur dynamique. On a CONVERGENCE dans le cas contraire, c'est à dire lorsque l'on est certain de ne plus avoir de différence par rapport à la machine de référence.

EXTENSION AUX AUTRES NIVEAUX

Une étude a démarré dans notre équipe pour étendre le mode concurrent aux autres niveaux.

CONCLUSION

Nous avons présenté notre travail dans le cadre du projet CASCADE. Ce projet doit se terminer dans un an et nous entrons maintenant dans une phase d'optimisation de la version prototype:

En ce qui concerne la compilation et les mécanismes d'ordonnancement, il s'agit maintenant de réaliser la nouvelle version telle qu'elle a été définie au chapitre 2. Après avoir éffectué un grand nombbre d'expériences sur des modèles variés, il sera possible de définir des règles pour la restructuration des modèles, en particulier pour la taille des blocs de simulation qui est un élément déterminant pour l'efficacité de la simulation. Enfin, il convient de pousser plus loin l'étude sur l'ordonnancement conditionnel des instructions dans un module fonctionnel.

Au niveau de la simulation, le travail à réaliser est encore important: Par exemple, il faut noter que certains modes de simulation (pour le niveau CASTOR en particulier) sont encore à implémenter.

Mais d'une manière générale, il s'agit maintenant de réaliser de nombreux tests afin de valider notre approche de la simulation multi-modes. Dans cet objectif, nous pensons qu'une collaboration étroite avec des concepteurs (parmi nos partenaires Européens par exemple) permettra de mettre en évidence les cohabitations intéressantes de modes de simulation.

Dans le domaine des réalisations à plus long terme, (après la fin du projet) on peut imaginer plusieurs directions. L'une d'entre elles concerne l'augmentation de la parallélisation dans CASCADE. Ainsi que nous l'avons signalé au premier chapitre, on peut envisager une approche logicielle ou une approche matérielle.

- La première étant l'extension de la simulation concurrente aux autres niveaux. Un projet a débuté dans ce sens dans notre équipe.

- La deuxième pourrait consister en l'étude d'une "machine de simulation multi-modes" Dans une première étape, il est envisagé de "câbler" uniquement certaines parties de CASCADE.

ANNEXES

ANNEXE

Nous donnons ici des exemples de simulation de modèles décrits à l'aide du langage CASCADE.

Ces exemples n'ont pas la prétention de montrer les "performances" du simulateur car ce dernier n'existe qu'à l'état de prototype et bien des optimisations sont encore à réaliser avant de parvenir à une version utilisable avec profit par des concepteurs. Par contre, ils permettent de valider en partie nos mécanismes.

Le premier exemple est un modèle de circuit que nous avons conçu, il y a quelque temps, en collaboration avec l'équipe d'architecture des ordinateurs de l'IMAG dans le cadre du projet MPC (BCC.82). A l'époque, comme nous ne disposions pas encore d'un prototype de CASCADE, nous avions simulé ce circuit avec le simulateur CASSANDRE. Nous avons depuis réalisé une description CASCADE que nous avons simulé avec le prototype.

Le deuxième exemple est un exemple de simulation multi-modes. Une partie du circuit est décrite au niveau CASSANDRE, une autre au niveau IMAG. Là encore il convient de préciser que cet exemple ne prétend pas être un exemple de simulation multi-modes réaliste (l'intérêt de faire directement cohabiter un module CASSANDRE modélisé sans retards avec un module IMAG est très discutable pour un concepteur!) mais ici encore un exemple de test pour nos mécanismes.

1.0 PREMIER EXEMPLE:

UN CIRCUIT INTEGRE DE GENERATION ET ANALYSE DE VECTEURS POUR LE TEST DE CARTES A MICROPROCESSEUR

1.1 SPECIFICATIONS DU CIRCUIT

L'un des axes de recherche actuels dans le domaine du test concerne les méthodes pour éliminer la génération de vecteurs de test, la commandabilité et l'observabilite dans un circuit.
La méthode proposée ici consiste à générer des vecteurs et analyser "sur place" les réponses en utilisant des registres normaux

Le circuit étudié est construit autour d'un registre de base devant permettre les fonctions suivantes:

- Génération de Vecteurs PARALLELE
- Génération de Vecteurs SERIE
- Analyse de signature PARALLELE
- Analyse de signature SERIE

Le circuit pourra ainsi fonctionner suivant deux modes:

a) Dans le premier mode, le circuit est connecté à un microprocesseur qui le contrôle pour tester la carte.

b) Dans le second mode, le circuit travaille tout seul et doit se suffire à lui-même.

On peut bien sûr utiliser plusieurs de ces circuits sur une même carte avec certains d'entre eux travaillant en générateurs, d'autres en analyseurs.

Le mode de fonctionnement du registre est indiquée par un registre de contrôle, lui-même chargé à partir de la valeur des broches de contrôle. Le choix du polynôme générateur du registre à décalage doit être adapté à tous les modes de fonctionnement. Finalement, le choix s'est porté sur le polynôme suivant: (BCC.82)

$$P = x^{16} + x^9 + x^7 + x^4 + 1$$

1.2 CONSTRUCTION D'UN MODELE CASCADE

Les spécifications fonctionnelles sont traduites par une description au niveau LASCAR-S:

```
lanref LASCAR_S
<< CIRCUIT DE GENERATION DE VECTEURS ET D'ANALYSE DE SIGNATURE >>
description GEN1
  ( in HORLOGE H ;
    in SIGBO ARET , PARAL , GENER , RESB , CSCLEB , CSCB ,RESHB ,
    SETHB, VMA , LECT , DONVAL , XEPARA(0:15) , XESER ,XECRIT ;
    out SIGBO SM(0:15) , OCCUPE , XLEC , XSSER ,XSPARA(0:15) )
corps
  declare
   HORLOGE HREDEC , HRECON ;
<<          charge le registre a decalage                          >>
<<          permet de charger le registre fonctionnel              >>
   REGBO RRESB , RPARAL , RGENER , RARET , REGDEC(0:15) ;
<<   4 Registres de controle  +   Registre a decalage              >>
   SIGBO  S(0:15) ,C1 ,C2 ,C3 , C4 ,C5 , C6 , C7 ,C8 ;
  externe
  REDEC , RECON , CONTRO , ENRD ,SORED ,HORL ;

  use  REDEC RD ( HREDEC , S , REGDEC );
<<     REGISTRE A DECALAGE            >>

  use ENRD ERD ( C1 , C2 ,C3  ,CSCLEB ,RESHB ,SETHB ,VMA , LECT ,
                 XEPARA ,XESER, XECRIT,
                 RRESB,RPARAL ,RGENER ,    REGDEC ,      S ) ;
<<     ENTREES DU REGISTRE A DECALAGE >>

  use SORED SRD (C4 , C5 , C6 ,REGDEC , XLEC ,XSSER , XSPARA  ) ;
<<     SORTIES DU REGISTRE A DECALAGE    >>

  use CONTRO CT ( CSCLEB , CSCB ,VMA , LECT ,SETHB, RESHB ,DONVAL ,
                  RRESB , RPARAL ,RGENER ,RARET ,
                  C1 , C2 , C3 ,C4 ,C5 ,C6 ,C7 ,C8) ;
<<      PARTIE CONTROLE :  LES SORTIES SONT DES CONDITIONS   >>

  use HORL HORED(H , C7 , HREDEC) ,
<< generation de l'Horloge du registre fonctionnel >>
          HOREC(H , C8 , HRECON );
<< generation de l'horloge du registre de controle >>

  use RECON  RCRES(HRECON , RESB , RRESB) ,
             RCPAR(HRECON , PARAL , RPARAL) ,
             RCGEN(HRECON , GENER , RGENER ) ,
             RCARET(HRECON , ARET  , RARET ) ;
findescription
```

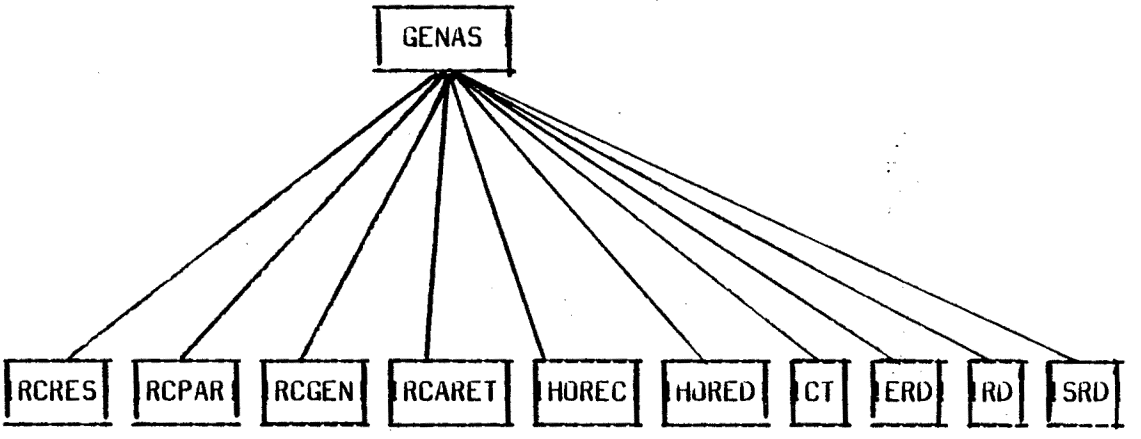
```
lanref LASCAR_S

description CONTRO

(in SIGBO CSCLEB , CSCB , VMA , LECT ,SETHB, RESHB , DONVAL  ;
                in REGBO RRESB ,RPARAL ,RGENER ,RARET;
                out SIGBO C1 , C2 , C3 ,C4 ,C5 , C6 ,C7 , C8 )

<<      PARTIE CONTROLE :  LES SORTIES SONT DES CONDITIONS   >>

  corps
  declare
   SIGBO SRES , SPREC , INIT , SANPAR ;

<<  SRES = 1  SI UN DES DEUX RESET ACTIF (=1)              >>
<<  INIT = 1  SI LE RESET HARD OU LE SET HARD  SONT ACTIFS >>
<<  SPREC = 1 SI DECALAGE DES BITS DE 1 A 15               >>
<<  SANPAR = 1 SI ANALYSE PARALLELE                         >>

  relations
    INIT .= ¨(RESHB ù SETHB) ,
    SANPAR .= (¨RGENER) & RPARAL ,
    SRES .= ¨(RESHB & RRESB) ,
    SPREC .= ¨(SANPAR & CSCLEB) ,

    C1 .= SRES ,
    C2 .=  (¨SRES) & SPREC ,
    C3 .=  (¨SRES) & (¨SPREC) ,
<< CONDITIONS DE CALCUL DES ENTREES DU REGISTRE A DECALAGE >>

    C4 .=  ¨(CSCLEB) & LECT & VMA ,
    C5 .=  CSCB & RGENER & (¨RPARAL) ,
    C6 .=  CSCB & RGENER & RPARAL ,
 << CONDITIONS DE TRANSFERT SUR LES BUS DE SORTIES  >>

    C7 .=  (DONVAL & CSCB & (¨RARET)) ù (VMA &(¨CSCLEB)) ,
  << CONDITION D'HORLOGE POUR LE REGISTRE DE CONTROLE >>

    C8 .=  INIT ù (¨(CSCB) & VMA)
<< CONDITION D'HORLOGE POUR LE REGISTRE A DECALAGE >>

findescription
```

```
lanref LASCAR_S
description ENRD( in SIGBO C1 , C2 ,C3 , CSCLEB , RESHB,
        SETHB, VMA , LECT, XEPARA(0:15) ,XESER, XECRIT ;
in REGBO RRESB ,RPARAL, RGENER, REGDEC(0:15); out SIGBO S(0:15))

<<                  ENTREES DU REGISTRE A DECALAGE              >>

 corps
 declare  SIGBO F1 , F2 , F3 , ZA ,X ,X1,
                X2 ,X3 ,X4 ,X5 , Y ,SRES , SANPAR ;
<<   F1 ,F2 ,F3          RETOURS DU REGISTRE A DECALAGE          >>
<<  X,X1,X2,X3,X4,X5,Y  FILS INTERMEDIAIRES                      >>
<<  SRES                 = 1 SI RESET (HARD OU SOFT) DU REGISTRE >>
<<                       A DECALAGE                              >>
<<  SANPAR               = 1 SI ANALYSE PARALLELE                >>

   relations
   F1 .= REGDEC(15) xor REGDEC(11) ,
   F2 .= F1 xor REGDEC(8) ,
   F3 .= F2 xor REGDEC(6) ,
   SANPAR .= (¨RGENER) & RPARAL ,
   SRES .= ¨(RRESB & RESHB) ,
   ZA .= XESER xor F3 ,
   Y  .= XEPARA(0) xor F3 ,
   X  .= (¨SETHB) ù X1 ù X2 ù X3 ù X4 ù X5 ,
   X1 .= CSCLEB & RGENER & F3 ,
   X2 .= CSCLEB & SANPAR & Y  ,
   X3 .= CSCLEB & (¨RPARAL) & (¨RGENER) & ZA  ,
   X4 .= (¨CSCLEB) & LECT & REGDEC(15) ,
   X5 .= (¨CSCLEB) & (¨LECT) & VMA & XECRIT ,
   si C1 alors
        pour I depuis 0 jusqua 15 repeter S(I) .= `0 finpour
   sinon
        S(0) .= X
   finsi ,
    si C2 alors
        pour I depuis 1 jusqua 15 repeter
             S(I) .= REGDEC(I-1)
        finpour
    finsi ,
   si C3 alors
        pour I depuis 1 jusqua 15 repeter
             S(I) .= REGDEC(I-1) xor XEPARA(I)
        finpour
    finsi
findescription
```

```
lanref LASCAR_S
description SORED
  ( in SIGBO  C4 ,C5 ,C6  ; in REGBO REGDEC(0:15) ;
    out SIGBO XLEC , XSSER ,XSPARA(0:15) )

<<                    SORTIES DU REGISTRE A DECALAGE                    >>

  corps
   relations
   si C4 alors XLEC .= REGDEC(15)     finsi,
   si C5 alors XSSER .= REGDEC(15)    finsi,
   si C6 alors XSPARA .= REGDEC       finsi
findescription



lanref LASCAR_S
description REDEC (in HORLOGE HREDEC ; in SIGBO A(0:15) ;
                   out REGBO RB(0:15) )

<<                     REGISTRE A DECALAGE                              >>

  corps
  relation
  !HREDEC! RB <= A ;

  findescription

lanref LASCAR_S
description RECON (in HORLOGE HRECON; in SIGBO A; out REGBO RB )

<<                    REGISTRE DE CONTROLE                              >>

  corps

  relation ,
  !HRECON! RB <= A ;

  findescription


lanref LASCAR_S
description HORL (in HORLOGE HE ;  in SIGBO C ;  out HORLOGE HS )

<<                    GENERATION D'HORLOGE                              >>

 corps
  relation
   si C alors HS .= HE finsi
  findescription
```

La description initiale correspond à un découpage en modules tel que devrait en produire la "nouvelle compilation CASCADE" avec séparation de la partie structurelle et de la partie fonctionnelle.

Ainsi structurellement, au niveau englobant, "GENAS" se présente comme un réseau de 10 modules interconnectés par synonymie.



LA SEPARATION ENTRE PARTIE CONTROLE ET PARTIE OPERATIVE qui sera faite aussi automatiquement au cours de la nouvelle compilation a été faite ici manuellement.

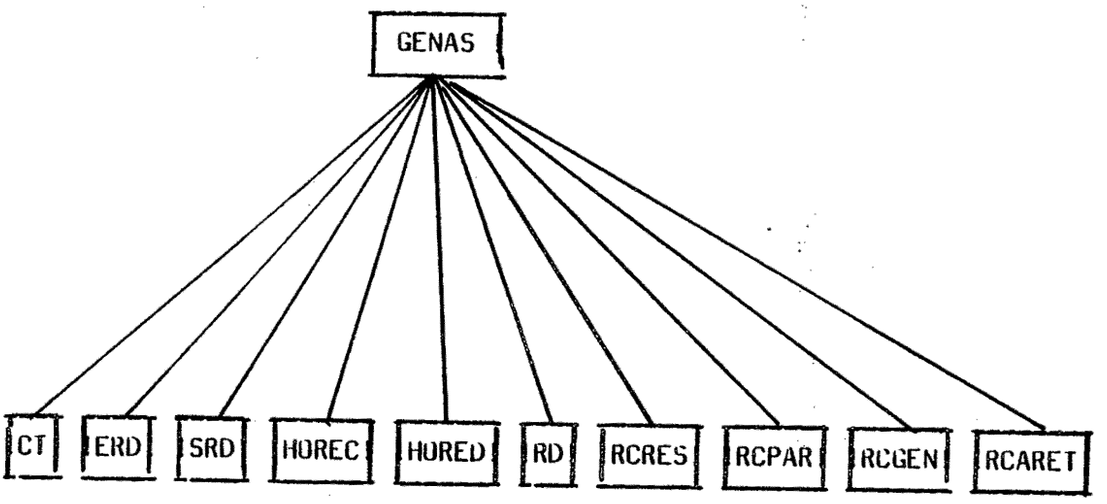
Ainsi, la partie OPERATIVE du circuit se compose du module RD correspondant à la description du registre à décalage (REDEC), ainsi que des modules ERD (description ENRD) qui calcule les entrées du registre à décalage à l'aide de la boucle de rétro-action correspondant au polynôme P et SRD qui correspond à l'aiguillage des sorties du registre vers les pattes de sortie du circuit (description SORED).

La partie CONTROLE du circuit se compose des quatre registres de contrôle: RCRES, RCPAR, RCGEN et RCARET; des deux "blocs horloge" HOREC et HORED; et enfin du bloc CT (description CONTRO).

Les sorties de CT sont les conditions qui contrôlent la partie opérative.

1.2.1 MODELE APRES ORDONNANCEMENT

Nous représentons ci-dessous la structure arborescente obtenue après l'ordonnancement structurel:



CE DESSIN REPRESENTE L'ARBORESCENCE EN PRE-ORDRE.

CORRESPONDANCE ENTRE BROCHAGE DU CIRCUIT ET ENTREES/SORTIES DE LA DESCRIPTION

BROCHES	ENTREES/SORTIES	COMMENTAIRES
E	H	HORLOGE
ARRET/MARCHE	ARET	ARRET SI 1 LE REGISTRE FONCTIONNEL CONSERVE SES DONNEES SANS LES MODIFIER
PARALLELE/SERIE	PARAL	PARALLELE SI 1 - SERIE SI 0
GENERATION/ANALYSE	GENER	GENERATION SI 1 - ANALYSE SI 0 MODES DE FONCTIONNEMENT DU REGISTRE
RESET	RESB	RESET SOFT ACTIF SI 0
CS LECT/ECR	CSCLEB	CHIP SELECT CONTROLE LECTURE/ECRITURE ACTIF SI 0. PERMET L'INITIALISATION DU REGISTRE FONCTIONNEL.

CS CONTR	CSCB	CHIP SELECT CONTROLE ACTIF SI 0 : DANS CE CAS LE REGISTRE DE CONTROLE EST CHARGE A PARTIR DES BROCHES DE CONTROLE, ELLE-MEMES CONNECTEES AU BUS DONNEES. LE REGISTRE FONCTIONNEL CONSERVE SA VALEUR.
RESET HARD	RESHB	RESET HARDWARE ACTIF SI 0
SET HARDWARE	SETHB	SET HARDWARE ACTIF SI 0 LORSQUE LES SET ET RESET HARD SONT ACTIFS, ILS PROVOQUENT UN SET OU UN RESET DU REGISTRE FONCTIONNEL ET LE CHARGEMENT DU REGISTRE DE CONTROLE A PARTIR DES BROCHES.
VMA	VMA	DONNEES VALIDES SI 1
VD	DONVAL	INDIQUE QUE LES DONNEES A ANALYSER SONT VALIDES.
ANOMALIES	ANO	SORTIE INDIQUANT UNE ANOMALIE
R/W	LECT	LECTURE SI 1 -- ECRITURE SI 0
E/S PARALLELES	é XEPARA é XSPARA	BUS E/S PARALLELE = ENTREES " = SORTIES
E/S SERIE	é XESER é XSSER	BUS E/S SERIE = ENTREES " = SORTIES
LECTURE/ECRITURE	é XLEC é XECRIT	BUS LECTURE/ECRITURE = LECTURE = ECRITURE

1.3 SIMULATION DU MODELE

Nous donnons un exemple de session de simulation CASCADE dans laquelle on pourra distinguer 3 étapes:

- Initialisation des registres de contrôle
- Initialisation du registre à décalage
- Fonctionnement du registre en mode "GENERATION PARALLELE" avec résultats sur le bus d'E/S parallèles.

```
******** Session de Simulation CASCADE ********
```

1/

```
 STO H=1
.
 STO GENER=`1
.STO PARAL=`1
.STO RESB=`1
.STO CSCLEB=`1
.STO RESHB=`1
.STO SETHB=`1
.STO VMA=`1

 RUN 1
 Date Logique demandee
            1
.PRI HRECON
  1
.PRIRGENER
  `1
.
 PRI RPARAL
  `1
```

2/

```
 STO CSCB=`1
.STO CSCLEB=`0
.STO LECT=`0
.STO XECRIT=`1
.

 RUN 2
 Date Logique demandee
            2
.PRI REGDEC
  `1000000000000000
.
 STO XECRIT=`0
.RUN 6
 Date Logique demandee
            6
.PRI REGDEC
  `0000100000000000
```

```
.RUN +2
 Date Logique demandee
          15
.PRI REGDEC
  `1100111110000100
```

3/

```
 STO CSCLEB=`1
.STO DONVAL=`1
.
 RUN +3
 Date Logique demandee
          18
.PRI REGDEC
  `1001100111110000
.
 PRI XSPARA
  `0011001111100001
.

 RUN +5
 Date Logique demandee
          23
.PRI XSPARA
  `0000100110011111
.

 RUN +10
 Date Logique demandee
          33
.PRI XSPARA
  `1110010001000010
.
 RUN +20
 Date Logique demandee
          53
.PRI XSPARA
  `1010100011011011
.

 RUN +30
 Date Logique demandee
          83
.PRI XSPARA
  `0111110000100101
.


 END

       ******  FIN de Session de Simulation CASCADE*:
```

2.0 DEUXIEME EXEMPLE :

LE CIRCUIT TESTLE

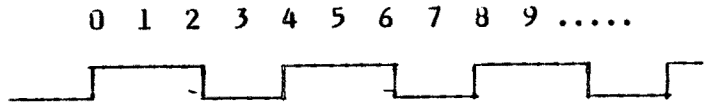
Ce modèle est tout à fait artificiel et a pour seul but de tester la cohabitation entre le mode de simulation continu et le mode de simulation Liste fixée.

```
lanref CASSANDRE_S

description TESTLE   (in HORLOGE H ; out SIGBO S)

<< CIRCUIT DE TEST  - SIMULATION MIXTE CASSANDRE/IMAG >>

 corps
 declaration  REGBO   R ;
 externe   RC,DEC ;
 unite
  RC  RC1  (R , S) ;                MODULE IMAG
  DEC DEC1 (H , R) ;                MODULE CASSANDRE
findescription


lanref CASSANDRE_S
description DEC  (in HORLOGE H ; out REGBO R)
 corps
 declaration  REGBO   R1 ;  HORLOGE CK ;
 relation
  si R1 = `0  alors CK .= H  finsi ,
  !H! R1 <= ¨R1 ;
  !CK! R <= ¨R ;
findescription

reflan IMAG_F
description RC  (in VAR IN ; out VAR OUT)
corps
declare  VAR  VR,VC,A,B,C,D; VEL  V1; IEL  I1 ; NDEL N1,N2,N3 ;
 use
  E E1 (N1,N2,IN) ;
  R R1 (N2,N3,VR) ;
  C C1 (N3,N1,VC) ;
 relations_elec
  OUT := V1 ,
  VR  := 10.0 ,
  VC  := 15.0P ,
  V1  := v (N3,N1) ,
  I1  := i (R1.P1) ,
  A   := B + exp (40.0 * V1) ,
  B   := sqrt (I1 + 5.0) + vabs(C) ,
  C   := V1 / max (I1,2.0) ,
  D   := VR
 findescription
```

2.1 SIMULATION DU MODELE

******** Session de Simulation CASCADE ********

EN SORTIE DE DEC1 ON OBTIENT LA CONFIGURATION SUIVANTE POUR R



ON VA OBSERVER, TOUT AU LONG DE CETTE SIMULATION, LES VALEURS SUCCESSIVES PRISES PAR LES ENTREES/SORTIES DE RC,
DU POINT DE VUE LOGIQUE ET DU POINT DE VUE ELECTRIQUE

```
 TRACE DE LA SIMULATION                          COMMENTAIRES
 ----------------------                          ------------
                                               (LES COMMANDES DE SIMULATION
                                               (SONT PRECEDEES D'UN POINT

.STO H=1                                      (POSITIONNEMENT DE L'HORLOGE
.RUN 0
  Date Logique demandee
            0
 INITIALISATION DU BSE D'ADRESSE :    22
  BTU                                            ( BTU = BASIC TIME UNIT
      0.100E-06                                  (  Valeur de BTU exprimee
  Date Electrique courante                       (       en format REEL :
      0.000E+00
  Date Electrique demandee
      0.100E-06
  ON ACTIVE LE BLOC D'ADRESSE   40               ( = BLOC LOGIQUE
.PRI R                                           ( COMMANDE D'IMPRESSION
 L'objet specifie est :
      R
   `1
.PRI S
 L'objet specifie est :
      S
   `0
.BOX /TESTLE/RC1                                 ( COMMANDE DE POSITIONNEMENT
 Le module specifie est :                        ( DANS LA HIERARCHIE
      /TESTLE/RC1
.PRI IN
 L'objet specifie est :
      IN
```

```
      0.000E+00
.PRI OUT
 L'objet specifie est :
      OUT
      0.000E+00
.BOX /TESTLE
 Le module specifie est :
      /TESTLE
**************************************************************************

                          ( DEUXIEME CYCLE DE SIMULATION )
                            ---------------------------

.RUN +1                          (ON INDIQUE JUSQU'A QUELLE DATE
  Date Logique demandee          (ON SIMULE
            1
  BTU
      0.100E-06
  Date Electrique courante
      0.100E-06
  Date Electrique demandee
      0.200E-06
  ON ACTIVE LE BLOC D'ADRESSE    22      (= BLOC ELECTRIQUE DEC1
  ON ACTIVE LE BLOC D'ADRESSE    40      (= BLOC LOGIQUE RC1

.PRI R
 L'objet specifie est :
      R
   `1

.PRI S                                   (S EST LA SORTIE DU BLOC
 L'objet specifie est :                  (ELECTRIQUE TRANSFORMEE EN
      S                                  (LOGIQUE.
   `0

.BOX /TESTLE/RC1                         (COMMANDE DE POSITIONNEMENT
 Le module specifie est :                (DANS LA HIERARCHIE
      /TESTLE/RC1

.PRI IN
 L'objet specifie est :                  (VALEUR DE R TRANSFORMEE
      IN                                 (EN REEL
      0.100E+01

.PRI OUT                                 (VALEUR DE LA SORTIE REELLE
 L'objet specifie est :                  (DU BLOC ELECTRIQUE
      OUT
      0.100E+01

.BOX /TESTLE
 Le module specifie est :
      /TESTLE
```

```
**************************************************************************

                                        ( TROISIEME CYCLE )
                                          --------------
.RUN +1
  Date Logique demandee
            2
  BTU
      0.100E-06
  Date Electrique courante
      0.200E-06                         ( INTERVALLE
  Date Electrique demandee              ( D'INTEGRATION
      0.300E-06
  ON ACTIVE LE BLOC D'ADRESSE   22      ( BLOC ELECTRIQUE
  ON ACTIVE LE BLOC D'ADRESSE   40      ( BLOC LOGIQUE
.PRI R
 L'objet specifie est :
      R
   `U
.PRI S
 L'objet specifie est :
      S
   `1
.BOX /TESTLE/RC1
 Le module specifie est :
      /TESTLE/RC1

.PRI IN
 L'objet specifie est :
      IN
      0.100E+01

.PRI OUT
 L'objet specifie est :
      OUT
      0.100E+01

.BOX /TESTLE
 Le module specifie est :
      /TESTLE

   ...

   ...


.RUN +1
  Date Logique demandee
           13
  BTU
      0.100E-06
```

```
  Date Electrique courante
      0.130E-05
  Date Electrique demandee
      0.140E-05
  ON ACTIVE LE BLOC D'ADRESSE    22
  ON ACTIVE LE BLOC D'ADRESSE    40

.PRI R
 L'objet specifie est :
      R
   `1
.PRI S
 L'objet specifie est :
      S
   `0
.BOX /TESTLE/RC1
 Le module specifie est :
      /TESTLE/RC1
.PRI IN
 L'objet specifie est :
      IN
      0.100E+01
.PRI OUT
 L'objet specifie est :
      OUT
      0.100E+01
.BOX /TESTLE
 Le module specifie est :
      /TESTLE

.END

        ******  FIN de Session de Simulation CASCADE******
```

BIBLIOGRAPHIE

(ABK.77) Abramovici M., Breuer M.A., Kumar K.
"Concurrent Fault Simulation and Functionnal Level Modeling"
Proc. 14th DAC, June 1977, pp 128-135 .

(ALM.82) Abramovici M., Levendel Y.H., Menon P.R.
"A Logic Simulation Machine"
Proc. 19th DAC, June 1982, pp 65-73 .

(BCC.82) Borrione D., Caubet B., Courtois B., Guyot A., Humbert M., Ronald A.
"Un circuit integre de generation et analyse de vecteurs pour le test de cartes a microprocesseur."
RR IMAG NO 293 , Mars 1982 .

(BDG.84) Borrione D., Genevey J., Decouchant V.
"Generation de code"
Rapport CEE CASCADE - Jan 1984

(Ben.79) Bening L.
"Developments in computer simulation of gate level physical logic".
Proc. 16th DAC, June 1979, pp 561-567.

(BGM.84) Berardo G., Gai S., Mezzalama M., Somenzi F., Spalla., Zanello R.
"CERES Project Concurrent Gate Level Simulator"
Rapport CEE - CERES - Juin 1984-

(BHL.83) Borrione D., Humbert M., Le Faou C.
"Hierarchical Mixed-mode Simulation in the Cascade Project"
Proc. of the VLSI conference. TRONDHEIM - Aout 1983.

(BMB.83) Borrione D., Mermet J., Battistoni G., Prinetto P.
"A Time Profile Description Language For System Specification and Simulation"
RR IMAG NO 406 - Nov 1983 -

(BMP.84) Battistoni G., Mermet J., Prinetto P.
"CERES Project: The simulation Environment"
Rapport de contrat CEE - Juin 1984 -

(BPV.84) Battistoni G., Prinetto P., Vercellone V.
"The Input Waveform Management System"
Rapport de contrat CEE Juin 1984

(Bon.83) Bona M.
"Quelques Outils Numeriques Pour la Resolution De Systemes Algebrodifferentiels de Grande Dimension."
These DI 1983 - USMG -

(Bor.76) Borrione D.
"LASCAR Un langage pour la Simulation et l'Evaluation des Architectures d'Ordinateur"
These de troisieme cycle . Grenoble 1976.

(Bor.81) Borrione D.
"Langages de description de systemes logiques - Proposition pour une methode formelle de definition"
These d'etat INPG . Juillet 1981.

(Bor.84) Borrione D.
"Manuel de reference du langage CASCADE"
Rapport de contrat CEE - Aout 1984

(Bre.76) Bressy Y.
"Version asynchrone du langage CASSANDRE"
Seminaire de Programmation du LA 7 - Nov 1975 -

(Bre.83) Bressy Y.
"Superviseur elementaire de simulation"
Rapport de contrat CEE - Sept 1983

(BrF.76) Breuer M.A., Friedman A.D.
"Diagnosis and Reliable Design of Digital Systems"
Potomac, MD: Comp.Sc.press, 1976.

(Bry.82) Bryant R.E.
"Switch Level Modeling of MOS Digital Circuits"
Proc. IEEE Int. Symp. on circuits and systems.
Rome, Italy, May 1982 , pp 68-71.

(CGk.75) Chawla B.R., GUMMEL H.K., KOZAK P.
"MOTIS a MOS Timing Simulator"
IEEE trans. on Circ. and Sys. Vol CAS 22 Dec 1975.

(CLN.84) Chen C.F., Lo C.Y., Nham H.N., Subramaniam P.
"The Second Generation MOTIS Mied-Mode Simulator"
Proc. of the 21st DAC - Albuquerque USA - June 1984-
paper 2-2.

(DAR.80) De Man H., Arnout G., Reynaert P.
"Mixed-Mode Circuit Simulation Techniques and Their Implementation in DIANA"
Computer Design Aids for VLSI Circuits,
NATO Advanced Study Institutes Series - July 1980 .

(DaT.80) D'Abreu M.A., Thompson E.W.
"An accurate functional level concurrent fault simulator"
Proc. 17th Design Automation Conference, June 1980,
pp 210-213.

(DCF.84) D'Abreu M.A., Cheong K.L., Flanagan C.T.
"ORACLE: A Simulator for bipolar and MOS IC Design."
Proc. of the 21st DAC - Albuquerque USA - June 1984-
paper 21-2.

(DDS.82) De Man H., Dumlugol D., Stevens P., Schrooten G., Bolsens I.
" LOGMOS: A Transistor Oriented Logic Simulator With Assignable Delays"
Proc. of ICCC'82 - New-York Oct 1982.

(Den.82) Denneau M.M.
"The Yorktown Simulation Engine"
Proc. 19th DAC, June 1982 .

(Dma.82) De Man H.J.,
"Mixed-mode Simulation for MOS VLSI: Why, where and How?
Proc. IEEE Int. Symp. on circuits and systems.
Rome, Italy, May 1982 , pp 699-701.

(DGW.82) Daseking H.W., Gardner R.I., Weil P.B.
"VISTA: A VLSI CAD System"
IEEE trans. on CAD, Vol. CAD-1, pp 2-12, Jan 1982.

(DMS.83) De Micheli A., Sangiovanni-Vincentelli A.
"Symmetric Displacement Algorithms"
IEEE trans. on CAD, Vol. CAD-2, No 3, pp 166-175,
Jul 1983.

(DSS.84) Doshi M.H., Sullivan R.B., Schuller D.M.
"Themis Logic Simulator, a mixed-mode, multi-level,
hierarchical, interactive digital circuit simulator."
Proc. of the 21st DAC - Albuquerque USA - June 1984-
paper 2-4.

(Dur.84) Durand Y.
"Le compilateur Logique du Systeme CASCADE
Etude et specifications."
Rapport de DEA Informatique , Septembre 1984.

(ELD.82) Engel W.L., Laur R., Dirks H.
"MEDUSA A simulator for modular circuits"
IEEE trans. on CAD Int. Circ. Syst ,
vol CAD-1 pp 85-93 - Apr 82.

(Gen.83) Genevey J.
"Fonctions de restructuration de reseau"
Rapport de contrat CEE - 1983 -

(Ger.81) Gerritsen R.B.W.
"A Novel Hybrid Simulation Procedure"
Delft University of TecHnology Report. Jan 1981.

(Gra.81) Grabowiecki J.F.
"LASSO: Un langage pour la description et la simulation
des systemes informatiques - Mise en oeuvre -"
Memoire CNAM - 31 Mars 1981 -

(HaS.81) Hachtel G.D., Sangiovanni-Vincentelli A.L.
"A survey of third generation simulation techniques"
Proc. IEEE, vol 69, pp 1264-1280, Oct 81 .

(Hey.83) Heydemann M.H.
"A survey of MOS Logic simulation tools"
Proc of ESSCIRC 83 - Lausanne - Septembre 1983.

(HHL.82) Heydemann M.H., Hachtel G.D., Lightner M.R.
"Implementations Issues in Multiple Delay
Switch Level Simulation"
Proc. of the ICC'82 conference - New-york, Oct 82 .

(Hum.83) Humbert M.
"Ordering"
Rapport de contrat CEE - 15 Jun 1983 -

(Hum.84) Humbert M.
"Construction of SDS"
Rapport de contrat CEE - 16 Jan 1984 -

(HuV.83) Humbert M., Uvietta P.
"SPARADRAP: un Syteme expert de Production Automatique et Rapide d'Articles, de Documents, de Rapports et Autres Publications"
Note interne ARTEMIS.

(JeL.68) Jensen R.W., Lieberman M.D.
"IBM Electronic Circuit Analysis Program"
Englewood cliffs, N.J.: Prenctice-Hall, 1968.

(HiV.79) Hill D., VanCleemput W.
"A Tool For Generating Structured, Multi-Level Simulators"
Proc. 16th Design Automation Conference,
June 1979, pp 272-279.

(Lef.74) Le Faou C.
"Conception Assistee par Ordinateur: IMAG2 "
Jerusalem Conference on Information Technology
Jerusalem Aout 1970.

(Lef.82) Le Faou C.
CASCADE: Systeme de CAO integre pour Ensembles Logiques et Electroniques.
Introduction du niveau electrique.
RR IMAG No 308 Juillet 1982.

(Ler.80) Leroudier J.
"La Simulation a evenements discrets"
Monographies d'Informatique AFCET
Ed. Paris 1980.

(LeS.82) Lelasramee E., Ruehli A.E., Sangiovanni-Vincentelli A.L.
"RELAX: A new circuit simulator for large scale MOS integrated circuits"
Computer Aided Design - Vol 15 No 5 - Sept 1983.

(LeS.82) Lelasramee E., Ruehli A.E., Sangiovanni-Vincentelli A.L.
"The Waveform Relaxation Method For Time Domain Analysis of Large Scale Integrated Circuits"
IEE trans. on CAD of Integrated Circuits ans Systems
Vol CAD.1 No 3 Jul 1982.

(Mar.84) Marty J.C.
"Un editeur graphique pour le systeme CASCADE: EDICAS"
These de troisieme cycle INPG - Oct 1984.

(Mer.73) Mermet J.
"Etude Methodologique de la Conception Assistee par Ordinateur des systemes logiques: CASSANDRE "
These d'etat USMG. Avril 1973.

(Mer.80) Mermet J.
"Functionnal Simulation"
Computer Design Aids for VLSI Circuits
NATO Advanced Institutes Series, July 1980.

(Mer.83) Mermet J.
"Circuits and Systems Computers Aided Design & Engineering: CASCADE"
CAPE Conference Avril 1983.

(MSR.84) Mani M., Somenzi F., Rossi U.
"CERES project Switch Level Simulation"
Rapport de projet CEE - CERES - Jul 1984.

(Nag.75) Nagel L.
"SPICE 2 : A computer program to simulate Semi-conductor Circuits"
ERL.Memo No UCB/ERL M 75/520 May 1975.

(PhT.78) Phillips N.D., Tellier J.G.
"Efficient Event Manipulation: The Key To large Scale Simulation"
Proc. Test Conference - 1978.

(PrF.84) Pressman M., Fonlupt J.
"Ordering with Conditions"
Rapport de contrat CEE - Aout 1984 -

(Rou.84) Routin J.
"SOS - Structure Orientee Simulation"
Rapport de contrat CEE - Juillet 1984-

(Rue.81) Ruehli A.E.
"Survey of Analysis, Simulation and Modeling for Large Scale Logic Circuits"
Proc. 18th Design Automation Conference,
June 1981, pp 124-132.

(RuD.83) Ruehli A.E., Ditlow G.S.
"Circuit Analysis, Logic verification, and Design Verification for VLSI"
Proc. of the IEEE, Vol 71, NO 1, pp 34-48 Jan 83.

(SaD.82) Sakallah K.A., Director S.W.
"An Event Driven Approach for Mixed Gate and Circuit Level Simulation"
Proc. of Int. Symp. on Circ. and Sys.
Rome , Italie Mai 1982.

(Sar.84) Sarrette C.
"Introduction du niveau electrique dans la phase 2 de compilation"
Rapport de contrat CEE - Jan 1984 -

(SeF.62) Seshu S., Freeman D.N
"The diagnosis of asynchronous sequential switching systems"
IEE. Trans. Elec. Comp pp 459-465 - Aug 1962

(Ser.84) Serlet B.
"Description Structurelle et Simulation de Circuits Integres"
These de troisieme cycle. Univ. de PARIS-SUD.
- Janvier 1984 -

(SKO.83) Sasaki T., Koike N., Ohmori K., Tomita K.
"HAL: A Block Level Hardware Logic Simulator"
Proc. 20th Design Automation Conference,
June 1983, pp 150-156.

(SYA.81) Sasaki T., Yamada A., Aoyama T., Hasegawa K.,
Kato S., Sato S.
"Hierarchical Design Verificatin For Large Digital Systems"
Proc. 18th Design Automation Conference,
June 1981, pp 105-112.

(SYK.81) Sasaki T., Yamada A., Kato S., Nakazawa T., Tomita K., Nomizu N..
" MIXS: A Mixed Level Simulator for Large Digital System Logic Verification"
Proc. 17th Design Automation Conference, June 1980, pp 626-634.

(She.79) Sherwood W.
"A hybrid scheduling technique for hierarchical logic simulators"
Proc. 16th DAC, 1979 , pp 249-254.

(Tar.72) Tarjan R.
"Depth-first search and linear graph algorithms"
Siam J. on Comput. Vol 1, No 2, June 1979.

(Ter.82) Terman C.
"ESIM - Event Driven Switch Level Simulator, User's Manual,"
U-C Berkekey, CS division, Oct. 4 1982.

(ThN.83) Thomas D.E., Nestor J.A.
"Defining and Implementing a Multi-level Design Representation With Simulation Applications"
IEEE Trans. on Computer-Aided Design, Vol.CAD-2 NO. 3, July 1983.

(TWW.84) Tham K., Willoner R., Whimp D.
"Functional Design Verification by Multi-level Simulation."
Proc. of the 21st DAC - Albuquerque USA - June 1984- paper 21-2.

(Ulr.65) Ulrich E.G.
"Exclusive simulation of activity in digital networks"
Com. of ACM ,vol 12, nO 2 , pp 102-110, Feb 1965;

(UlH.82) Ulrich E.G., Herbert D.
"Speed and Accuracy in Digital Network Simulation based in structural modeling"
Proc. 19th DAC, Las Vegas USA - 1979 , pp 587-593 .

(Ulr.80) Ulrich E.G.
"Table lookup techniques for fast and flexible digital logic simulation"
Proc. 17th Design Automation Conference, June 1980, pp 560-563.

(Ulr.83) Ulrich E., Kearney M., Tellier J., Demba S.
"Design Verification for very large digital networks based on concurrent simulation and clock suppression."

AUTORISATION de SOUTENANCE

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

VU les rapports de présentation de Messieurs

. J. MERMET, Maître de recherche
. B. BAYLAC, Responsable du service CMS, Efcis

Monsieur HUMBERT Marc

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

Fait à Grenoble, le 8 octobre 1984

Le Président de l'I.N.P.-G

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

P.O. le Vice-Président,