IMHOTEP : un générateur automatique d'architectures pour circuits intégrés de filtrage numérique

Jean-Frédéric Reyss-Brion

THESE

présentée par

Monsieur REYSS-BRION Jean-frédéric

Ingénieur IEG

pour obtenir le titre de

DOCTEUR

DE L'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

(arreté ministériel du 5 juillet 1984)

Spécialité : Informatique

==========

IMHOTEP

Un générateur automatique d'architectures pour circuits intégrés de filtrage numérique.

==========

Date de soutenance : 24 Mai 1985

Composition du Jury :

Monsieur	BOLLIET	Président
Messieurs	CAMUS	
	GASTINEL	
	MAZARE	
	MOSSIERE	Examinateurs

Laboratoire de Génie Informatique
Institut de Mathématiques Appliquées de Grenoble

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Président: Daniel BLOCH

Vice-Présidents: B. BAUDELET
H. CHERADAME
R. CARRE
J.M PIERRARD

Année universitaire 1984-1985

Professeur des Universités

ANCEAU	François	E.N.S.I.M.A.G
BARIBAUD	Michel	E.N.S.E.R.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
FONLUPT	Jean	E.N.S.I.M.A.G
FOULARD	Claude	E.N.S.I.E.G
GANDINI	Alessandro	U.E.R.M.C.P.P
GAUBERT	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
JALINIER	Jean-Michel	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
LESIEUR	Marcel	E.N.S.H.G
LESPINARD	Georges	E.N.S.H.G
LONGEQUEUE	Jean-Pierre	E.N.S.I.E.G
LOUCHET	François	E.N.S.E.E.G
MASSELOT	Christian	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 Associés

BLACKWELDER	Ronald	E.N.S.H.G
HAYASHI	Hirashi	E.N.S.I.E.G
PURDY	Gary	E.N.S.E.E.G

Professeurs Université des Sciences Sociales (Grenoble II)

BOLLIET	Louis
CHATELIN	Françoise

Chercheurs du C.N.R.S

CARRE	René	Directeur de recherche
FRUCHART	Robert	Directeur de recherche
JORRAND	Philippe	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	Maître de recherche
BORNARD	Guy	Maître de recherche
DAVID	René	Maître de recherche
DEPORTES	Jacques	Maître de recherche
DRIOLE	Jean	Maître de recherche
GIGNOUX	Damien	Maître de recherche
GIVORD	Dominique	Maître de recherche
GUELIN	Pierre	Maître de recherche
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	Jean-Claude	Maître de recherche
MERMET	Jean	Maître de recherche
MUNIER	Jacques	Maître de recherche
PIAU	Monique	Maître de recherche
PORTESEIL	Jean-Louis	Maître de recherche
THOLENCE	Jean-Louis	Maître de recherche
VERDILLON	André	Maître de recherche
SUERY	Michel	Maître de recherche

Personnalités habilitées à diriger des travaux de recherche
(Décision du Conseil Scientifique)

E.N.S.E.E.G.

ALLIBERT	Colette	DIARD	Jean Paul	NGUYEN TRUONG	Bernadette
BERNARD	Claude	EUSTATHOPOULOS	Nicolas	RAVAINE	Denis
BONNET	Roland	FOSTER	Panayotis	SAINFORT	(CENG)
CAILLET	Marcel	GALERIE	Alain	SARRAZIN	Pierre
CHATILLON	Catherine	HAMMOU	Abdelkader	SIMON	Jean Paul
CHATILLON	Christian	MALMEJAC	Yves (CENG)	TOUZAIN	Philippe
COULON	Michel	MARTIN GARIN	Régina	URBAIN	Georges (Laboratc des ultra-réfracta ODEILLO).

E.N.S.E.R.G.

BARIBAUD	Michel	CHEHIKIAN	Alain	HERAULT	Jeanny
BOREL	Joseph	DOLMAZON	Jean Marc	MONLLOR	Christian
CHOVET	Alain				

E.N.S.I.E.G.

BORNARD	Guy	KOFMAN	Walter	MAZUER	Jean
DESCHIZEAUX	Pierre	LEJEUNE	Gérard	PERARD	Jacques
GLANGEAUD	François			REINISCH	Raymond

E.N.S.H.G.

ALEMANY	Antoine	MICHEL	Jean Marie	ROWE	Alain
BOIS	Daniel	OBLED	Charles	VAUCLIN	Michel
DARVE	Félix			WACK	Bernard

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

BERT	Didier	COURTOIS	Bernard	FONLUPT	Jean
CALMET	Jacques	DELLA DORA	Jean	SIFAKIS	Joseph
COURTIN	Jacques				

U.E.R.M.C.P.P.

CHARUEL	Robert

C.E.N.G.

CADET	Jean	JOUVE	Hubert (LETI)	PERROUD	Paul
COEURE	Philippe (LETI)	NICOLAU	Yvan (LETI)	PEUZIN	Jean Claude (LE
DELHAYE	Jean Marc (STT)	NIFENECKER	Hervé	TAIEB	Maurice
DUPUY	Michel (LETI)			VINCENDON	Marc

Laboratoires extérieurs :

C.N.E.T.

DEMOULIN	Eric	GERBER	Roland	MERCKEL	Gérard
DEVINE	R.A.B.			PAULEAU	Yves

I.N.S.A. Lyon

GAUBERT	C.

****

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

****

A mes parents,

A ma femme,

Qui m'ont toujours encouragé.

AVANT-PROPOS

Le travail présenté dans cette thèse a été réalisé au laboratoire de Génie Informatique de l'IMAG, dans le cadre d'une collaboration du groupe Parallélisme, Communication et Circuits Intégrés avec le Centre Norbert-Segard de Grenoble.

Il a été effectué en liaison avec le département Architectures des Micros-Systèmes, division Conception des Circuits Intégrés du CNET-CNS.

Cette étude a été soutenue financièrement par le contrat DAII-IMAG numéro 82 PE 0215 tout d'abord, puis par la CEE, au sein du projet CVT (CAO VLSI FOR TELECOMMUNICATIONS) numéro 3744/81 (sous-tâche 1.8).

REMERCIEMENTS

Je tiens à remercier ici Monsieur le Professeur BOLLIET, de l'Université Scientifique et Médicale de Grenoble, pour l'honneur qu'il me fait en acceptant de présider mon jury.

J'exprime ma reconnaissance à Monsieur CAMUS, Directeur du CNET Grenoble pour m'avoir permis de développer ce travail au CNS dont la réputation scientifique est internationale, me fournissant ainsi un environnement dont rêveraient tous les thésards. Je le remercie également de l'honneur qu'il me fait en participant à mon jury.

Mes plus vifs remerciements vont ici à Monsieur GASTINEL, Chargé de Recherches au Laboratoire d'informatique Expérimentale de l'Ecole Normale Supérieure pour avoir accepté à la fois d'être rapporteur de ce travail et membre de mon jury. Ses critiques et son jugement seront pour moi un enseignement enrichissant.

Que ma vive gratitude soit ici exprimée à Monsieur le Professeur MAZARE qui m'a non seulement accueilli dans son équipe "Parallélisme, Communication et Circuits Intégrés" et fourni les ressources financières nécessaires, mais qui ne m'a également jamais mesuré conseils et suggestions.

Je suis aussi très sensible ,à l'intérêt que Monsieur le Professeur MOSSIERE, directeur du Laboratoire de Génie Informatique de l'IMAG, a porté à mes travaux en acceptant de participer à ce jury.

Mes chaleureux remerciements et le témoignage de mon amitié vont à toute les personnes de la division "Conception de Circuits Intégrés" du CNET Grenoble pour leur collaboration amicale et leur disponibilité quotidienne.

Ma reconnaissance respectueuse va tout d'abord à Monsieur GERBER, chef de la division CCI, pour les marques régulières d'intéret qu'il a porté à mon travail, et pour avoir accepté d'en être le rapporteur.

Que Monsieur Gérard MICHEL, chef de département au CNET-Grenoble, soit assuré de ma profonde sympathie et de ma reconnaissance marquée, pour avoir été non seulement l'instigateur de ce travail, mais également le conseiller attentif et compétent qui en fit la qualité. Trois ans de relations de travail quotidiennes et chaleureuses seront pour moi un excellent souvenir.

Je tiens aussi à citer tout spécialement le secrétariat de CCI, en les personnes de Mme VEILLEROT et Mlle QUEYRON qui ont donné à ce travail sa forme finale. Je les remercie sincèrement pour leur compétence et leur disponibilité qui n'eurent d'égal que leur gentillesse.

J'adresse enfin mes remerciements au personnel administratif de l'IMAG qui me soutint toujours malgré mon éloignement, et au personnel administratif du CNET qui eut l'amabilité de me faire bénéficier de ses services.

RESUME

A l'heure où la phase de dessin des circuits intégrés devient le goulôt d'étranglement entre la demande et la production, les outils de conception automatique prennent un intérêt grandissant.

Les Compilateurs de Silicium, forme la plus achevée de tels outils, automatiseront complètement le dessin des masques d'un circuit à partir de sa description fonctionnelle.

Nous présentons dans ce rapport un générateur automatique d'architectures pour des circuits intégrés de filtrage numérique. La description d'un algorithme de filtrage assorti d'une contrainte "temps réel" est fournie au générateur, ainsi qu'une liste d'opérateurs à structure parallèle pris dans une bibliothèque de cellules.

L'outil se charge alors de générer une structure de machine utilisant un sous-ensemble de la liste d'opérateurs, et réalisant l'algorithme désiré dans le temps requis. Cette structure est optimisée en ce qui concerne la surface (surface des opérateurs + surface des connexions) tout en respectant la rapidité requise, par l'utilisation de la technique du multiplexage.

Enfin, l'architecture trouvée est fournie sous la forme d'une partie opérative et d'un graphe d'états donnant le séquencement à lui appliquer.

Ce générateur d'architecture est par ailleurs relié à un générateur de plans de masse et à une bibliothèque de cellules paramétrables; l'ensemble constituant un compilateur de silicium pour filtres numériques.

MOTS-CLES

- CIRCUITS INTEGRES - CONCEPTION AUTOMATIQUE - COMPILATEUR DE SILICIUM
- GENERATION D'ARCHITECTURE - FILTRAGE NUMERIQUE - TEMPS-REEL - MULTIPLEXAGE
- BIBLIOTHEQUE DE CELLULES - VLSI - DESCRIPTION COMPORTEMENTALE -

ABSTRACT

Nowdays, as the design stage of integrated circuits is the most important part of the time to market, tools for an automated design take a growing interest.

Silicon Compilers will perform a fully automated design of circuits from their functionnal description.

We introduce here an automated architecture generation tool dedicated to digital filtering circuits. This tool needs the description of a filtering algorithm, a real time constrain and a set of parallel operators taken from a cell library as an input. It provides then the user with a machine description using operators of the inner set and completing the wanted algorithm at the required frequency.

This machine structure is optimized as far as global area is concerned by using the multiplex technic. This machine is described as an operative part and a state graph that gives the scheduling to be applied to the operative part.

At last, this tool is part of a silicon compiler that also includes a floor-plan generator and a flexible cell library.

KEY WORDS

- INTEGRATED CIRCUITS - AUTOMATED DESIGN - SILICON COMPILER - REAL-TIME -
- ARCHITECTURE GENERATION - DIGITAL FILTERING - MULTIPLEX - VLSI -
- CELL LIBRARY - BEHAVIORAL DESCRIPTION -

TABLE DES MATIERES

1 - INTRODUCTION

L'avènement des technologies VLSI (très haute densité d'intégration) a rendu possible l'apparition de circuits contenant plus de 100.000 transistors.

Ces circuits permettront d'intégrer des plaques entières qui supportent actuellement beaucoup de logique périphérique à côté de quelques composants plus complexes. Ainsi, la demande pour des produits de plus en plus performants et personnalisés est en très forte hausse : les circuits VLSI représentent 1 % du marché des circuits intégrés, et cette part croît de 22 % par an.

Cette forte croissance est sous-tendue par la recherche de la performance, le souci de réduire l'encombrement et d'assurer l'inviolabilité d'applications industrielles de plus en plus nombreuses, le tout sur fond de rentabilité économique.

Face à cette demande, les technologies ont rapidement progressé, en doublant la densité d'intégration tous les deux ans au maximum pour atteindre environ un million de transistors par circuit aujourd'hui. Le plus sévère goulot d'étranglement entre la demande et la production de ces nouveaux composants se situe à présent au niveau du dessin des masques. Pour prendre conscience de cette situation, il suffit de rappeler quelques faits :

- Avec des méthodes classiques, le temps de conception d'un circuit VLSI est souvent supérieur à la durée de vie économique du produit, qui diminue d'ailleurs de plus en plus.

- Il faut environ 40 hommes x années pour réaliser un circuit de 40.000 transistors depuis sa définition jusqu'à la mise sur le marché.

- Pour un circuit de 250.000 transistors, le dessin coûte 10 millions de dollars, soit 40 dollars par transistors.

- Enfin, pour concevoir un circuit destiné à une technique particulière, il faut souvent être spécialiste dans cette discipline, et cet apprentissage est coûteux en hommes et en temps.

Cet état de fait nous fait irrésistiblement penser à une situation analogue, et cette comparaison mettra en valeur les immenses potentialités de la micro-électronique :

L'industrie du livre, c'est elle dont il s'agit, a connu les mêmes étapes tout au long de son histoire. Avant Guttemberg, la production des livres était l'oeuvre d'écrivains ou de copistes, essentiellement des moines. Face à une demande très vive, les copistes, dont la capacité de production était humainement limitée, devaient choisir les livres qu'ils allaient recopier selon certains critères. Ainsi, ils sélectionnaient les oeuvres de grande audience comme la Bible, ou effectuaient des travaux sur commande, mais toujours à des tarifs élevés.

Puis l'imprimerie de Guttemberg modifia profondément le marché des livres. Le manuscrit fut converti en une série de planches de lettres par le typographe, puis l'imprimerie à presse autorisait une large diffusion. De cette façon, la demande fut satisfaite, mais les livres produits restèrent des oeuvres de large diffusion à cause du coût et du temps de réalisation des planches de lettres. Le typographe devint alors le passage obligé et coûteux pour tous les candidats à l'impression.

Enfin, l'avènement de la composition informatisée dans le monde de la presse permit à chaque journaliste de supprimer l'étape de typographie et de se retrouver en prise directe avec l'imprimerie et les rotatives. Le temps écoulé entre l'évènement et sa large diffusion est ainsi réduit à quelques heures. Notons également l'apparition de procédés tels que l'offset qui permet à l'écrivain de produire un texte particulier en petites séries.

Le parallélisme entre le marché de l'écrit et celui de la micro-électroniqe est édifiant : si l'équivalent de l'avant Guttemberg se perd dans la mémoire des pionniers de la micro-électronique, le tandem typographe - imprimeur correspond à la réalité d'aujourd'hui. Si les technologies de production sont parfaitement opérationnelles et permettent une large diffusion des produits à des coûts sans cesse décroissants, le concepteur de circuits a bien du mal à satisfaire la demande des candidats à l'intégration.

Nous assistons donc au même phénomène de concentration des efforts sur les produits de large diffusion que dans l'industrie du livre. Les produits de faible diffusion ayant du mal à passer dans ce marché de la micro-électronique.

Conscients de cette situation, les industriels du silicium adoptent en ce moment la même démarche que ceux du livre. Ils essayent d'amener ces équivalents des journalistes et écrivains que sont les concepteurs de systèmes, à concevoir eux-mêmes leur circuit intégré.

Le problème est, bien entendu, de leur fournir l'équivalent de la composition assistée par ordinateur des imprimeurs ou de la reprographie offset afin de soulager les équipes de "typographes - concepteurs VLSI" maison.

Si ce défit est relevé victorieusement, l'industrie du silicium prendra probablement un essor difficilement appréhendable aujourd'hui.

Nous allons à présent analyser la démarche des industriels visant à amener les concepteurs systèmes à la conception VLSI, et nous dirons qu'elle a pris successivement deux formes.

La première fut de développer des gammes de circuits programmables par l'usager afin d'évacuer vers celui-ci la spécificité de l'application. C'est ainsi que les microprocesseurs possédant des particularités diverses innondèrent le marché avec une bonne rentabilité, dans la mesure où les coûts de conception étaient divisés par le grand nombre d'exemplaires vendus.

Néanmoins, ces circuits programmables ne résolvèrent pas tous les problèmes, en particulier celui posé par les petites fonctions logiques périphériques qui sont toujours réalisées par des boîtiers standard. De plus, les concepteurs de systèmes voulurent des circuits capables d'intégrer des fonctions de plus en plus rapides et complexes, pour lesquelles les solutions microprogrammées sont loin d'être optimales.

Conscients de ce fait, les industriels du silicium eurent une seconde attitude : ils constatèrent que, si l'on estime à 400.000 environ le nombre de concepteurs systèmes dans le monde, on ne dénombre que 2.000 concepteurs de circuits intégrés. De plus, comme la compétence leur faisait souvent défaut pour concevoir des circuits spécifiques, ils décidèrent d'inciter les concepteurs systèmes à concevoir eux-mêmes leurs circuits.

Telle est donc à présent une tendance importante dans la stratégie des industriels proposant des circuits à la demande. Dans cette stratégie, quatre orientations sont apparues chronologiquement et coexistent à présent en se partageant le marché.

La première manière d'introduire les ingénieurs systèmes à la conception fut de leur proposer des microprocesseurs dont la mémoire interne (ROM) était importante et programmable par masque durant la dernière étape de fabrication du circuit. Ceci évitait les boîtiers de ROM externe, accélérait donc les lectures en mémoire et offrait une inviolabilité relative. Cette manière de procéder n'était pas révolutionnaire, et présentait une continuité par rapport aux pratiques antécédentes.

La deuxième initiative consista à offrir le moyen d'intégrer, en quelques boîtiers, toute la logique aléatoire périphérique de certaines applications. Ce fut l'apparition des réseaux prédiffusés de portes logiques dont les concepteurs d'applications dessinèrent le masque des connexions. Des outils informatisés apparurent rapidement pour aider à passer du schéma logique désiré au circuit prédiffusé correctement réalisé, en minimisant les interventions humaines.

L'apparition de réseaux de 5.000 portes logiques donne un grand intérêt à ces circuits dont le marché croît de 60 % par an. Néanmoins, des inconvénients comme le nombre limité de broches d'entrées-sorties, ou le fait que les portes inutilisées grèvent le coût final, limitent l'utilisation des réseaux prédiffusés à certaines classes d'application.

La troisième manière d'impliquer les usagers dans la conception des circuits est apparue pour combler les lacunes des réseaux prédiffusés, et est connue sous le nom de méthode des cellules standard.

Les cellules disposées sur le circuit sont ici choisies par l'utilisateur dans une bibliothèque fournie par l'industriel. Ces cellules ont des caractéristiques physiques et électriques figées, mais offrent des fonctionnalités beaucoup plus complexes que les portes logiques. L'usager dispose donc les variétés choisies selon un plan de masse imposé, généralement constitué par des bandes séparées par des canaux d'interconnexions.

Cette méthode évite la perte de place dûe aux cellules inutilisées comme pour les réseaux prédiffusés, mais la standardisation des cellules et du plan de masse occasionne encore des pertes de surface. Cette manière de concevoir les circuits existe chez les industriels et commence à être accessible aux clients au fur et à mesure qu'apparaissent des logiciels d'aide à la conception adaptés.

La dernière manière, et, pour certains circuits la meilleure de toutes, serait de proposer aux concepteurs de systèmes d'utiliser des assembleurs ou des compilateurs de silicium. Ces outils informatiques constitueront, pour les concepteurs qui dessinent leur circuit au micron, transistor par transistor, le même progrès que l'usage d'un assembleur, ou encore mieux d'un compilateur pour des gens qui écriraient leur programme en binaire. Les assembleurs, et à fortiori, les compilateurs de silicium sont encore des outils de laboratoire, bien qu'une société californienne, (Silicon Compilers Inc.) utilise un assembleur de silicium avec succès pour réaliser des circuits sur commande. Cette société a en effet réalisé, en deux hommes x années, la partie opérative du MICROVAX de DEC qui comprend 37.000 transistors.

L'assembleur de silicium peut être vu comme une extension de la méthode des cellules standard. C'est en effet une collection d'outils comprenant une bibliothèque de cellules précaractérisées paramétrables notamment en nombre de bits, d'un outils d'aide au placement des blocs, et d'un traceur de connexions. L'usager décrit donc successivement chaque bloc du circuit dont il doit avoir défini la structure matérielle au préalable, et obtient le masque de son circuit en fin de traitement.

Le compilateur de silicium accepte une description du circuit au niveau logico-fonctionnel, et établit lui-même la structure matérielle de la meilleure machine possible, en termes de surface et de consommation, réalisant la fonction désirée. Il est donc capable de faire des optimisations.

Ces deux sortes d'outils permettront au concepteur de systèmes d'intégrer lui-même son application rapidement en les utilisant soit chez lui en accord avec un fondeur de silicium, soit chez le fondeur lui-même.

L'objet de ce travail est l'étude d'une partie d'un compilateur de silicium spécialisé dans les circuits de filtrage numérique. Ce compilateur, nommé EDIFICE, est en cours d'achèvement au département AMS du CNET-Grenoble. Nous exposerons ici la réalisation du générateur automatique d'architectures qui, partant d'une description de l'algorithme de filtrage, fourni la structure optimisée d'une machine capable de réaliser le filtrage à la cadence désirée.

Le plan de cet exposé comprendra neuf chapitres articulés ainsi :

Après un premier chapitre d'introduction, les chapitres deux et trois feront respectivement une étude bibliographique sur les compilateurs de silicium d'une part, et sur les algorithmes de filtrage numérique d'autre part. Nous définirons alors un langage de description pour ces algorithmes de filtrage dans le chapitre quatre, et une bibliothèque d'opérateurs de filtrage dans le chapitre cinq.

Le chapitre six sera consacré à l'exposé de la méthode que nous avons utilisée pour traiter le problème de la génération automatique d'architectures, et le chapitre sept à celui de l'organisation finale du générateur IMHOTEP.

Les exemples traités pour valider et tester IMHOTEP sont regroupés au chapitre huit qui précède les conclusions et remarques du dernier chapitre.

Le lecteur est invité, tout au long du texte, à se reporter aux annexes techniques et bibliographiques situées en fin d'ouvrage.

2 – ETUDE BIBLIOGRAPHIQUE SUR LES COMPILATEURS DE SILICIUM

2-1- Introduction

2-2- Problèmes posés par la compilation de silicium

2-2-1 Architecture des compilateurs de silicium

2-2-1-1 Structure d'une chaîne de conception
2-2-1-2 Analyse du cheminement humain dans la chaîne de conception
2-2-1-3 Intégration de la chaîne de conception dans un compilateur de silicium.

2-2-1 Etude de différents compilateurs de silicium

2-2-2-1 Introduction
2-2-2-2 MacPitts
2-2-2-3 FIRST
2-2-2-4 ARSENIC
2-2-2-5 LOUVAIN
2-2-2-6 Conclusion

2 - ETUDE BIBLIOGRAPHIQUE SUR LES COMPILATEURS DE SILICIUM

2-1- Introduction

La première apparition du terme "compilateur de silicium" dans la presse scientifique date de 1980, et est due à Dave Johannsen, alors étudiant en PHD dans l'équipe de Carver Mead.

Son étude a fait quelques émules dans le monde et nous avons recensé cinq travaux connus sur ce thème, en ce qui concerne les publications étrangères. Il est par ailleurs intéressant de remarquer que D. Johannsen est co-fondateur de la première société commerciale utilisant un assembleur de silicium (Silicon Compilers Inc).

Le terme même de "compilateur de silicium" recouvre des réalités différentes selon les auteurs, et, le sujet étant complexe, les équipes qui travaillent sur ce sujet s'intéressent souvent à une restriction du problème.

Nous allons donc essayer d'exposer correctement le problème dans sa généralité, avant d'examiner la manière dont les travaux les plus avancés ont tenté de le résoudre.

2-2- Problèmes posés par la compilation de silicium

2-2-1 Architecture des compilateurs de silicium

Pour décrire brièvement un compilateur de silicium (CS), on peut dire que c'est un outil qui accepte en entrée les spécifications fonctionnelles du circuit à réaliser, et qui fournit en sortie les masques de ce circuit.

D'après cette définition idéale, le CS devrait être l'interlocuteur de l'ingénieur système qui désire intégrer une partie de son système. Ceci implique donc que le CS remplace deux ou trois catégories d'individus selon le circuit à réaliser : en remontant dans la chaîne de conception, il s'agit donc successivement du spécialiste en conception, du spécialiste en architectures de micro-systèmes et parfois du spécialiste en algorithmique (dans le cas du filtrage numérique entre autres).

2-2-1-1 Structure d'une chaîne de conception

Examinons les rôles des intervenants dans une chaîne de conception en dégageant les particularités de chacun :

- l'ingénieur système :

Ce futur client du CS exprime des besoins en circuits réalisant des fonctions de plus ou moins haut niveau :

Ce peut être un circuit gérant un protocole de communication sur réseau local, un automate réalisant un réseau de PETRI ou un filtre réalisant telle fonction de transfert dans telles conditions.

L'ingénieur système considère souvent le circuit comme une boite noire dont il n'a que faire de connaître les détails internes. Il ne délivre que des SPECIFICATIONS FONCTIONNELLES.

C'est le niveau numéro 1 d'une chaîne de conception.

- le spécialiste en algorithmique

Selon la nature des spécifications fonctionnelles, le rôle de ce deuxième intervenant peut être nul. Ce sont tous les cas où les spécifications fonctionnelles sont assez précises pour être aussi une description comportementale du circuit.

Ceci n'est pas le cas pour les circuits de filtrage numérique : nous verrons au chapitre 3 qu'il existe plusieurs algorithmes pour réaliser une fonction de transfert donnée.

La première tâche du spécialiste en algorithmique est de transformer les spécifications fonctionnelles en DESCRIPTION COMPORTEMENTALE par le choix d'un ALGORITHME optimisé.

La deuxième, plus difficile, est de faire un choix entre plusieurs algorithmes en prenant en compte les contraintes de l'intégration.

C'est le niveau numéro 2 d'une chaîne de conception.

- le spécialiste en architectures de micro-systèmes :

A partir de la description comportementale du circuit à réaliser, cet intervenant doit imaginer la structure d'une machine qui puisse réaliser à partir des entrées, le comportement désiré des sorties.

La contrainte temps réel éventuelle, occultée au niveau algorithmique, réapparait ici ; et les contraintes de l'intégration prennent une acuité croissante.

Les techniques utilisées sont variées, et vont du plaquage direct de la description comportementale sur des opérateurs pour la plus simple, jusqu'à l'utilisation du multiplexage, du pipe-line avec extraction de parallèlisme pour les plus sophistiquées. Les vecteurs de test sont issus du travail de l'architecture. Le résultat est une DESCRIPTION STRUCTURELLE du circuit à réaliser, et est le fait du niveau numéro 3 d'une chaîne de conception.

- le spécialiste en conception

Partant de la description structurelle du circuit, il faut réaliser tous les blocs qu'elle comporte, puis les placer et les interconnecter.

L'usage d'une bibliothèque de cellules est très courant à ce niveau, mais il faut ajuster les délais de propagation qui dépendent du plan de masse obtenu.

C'est à ce niveau qu'apparaissent beaucoup d'outils qui aident le concepteur, ce qui fait que la frontière entre l'architecture et la conception a tendance à s'estomper.

La conception est dans le 4ème et dernier niveau de la chaîne. Le problème du test ne se situe pas, à notre avis, à un niveau précis, mais il est présent à chaque étape de la conception. Cet aspect de la conception sera profondément influencé par l'arrivée des CS qui permettront probablement des approches plus systèmatiques.

Nous allons à présent étudier l'impact du CS sur l'organisation de cette chaîne de conception.

2-2-1-2 Analyse du cheminement humain dans la chaîne de conception

Après avoir étudié les étapes de la conception d'un circuit , nous allons voir quelle est la démarche des concepteurs pour franchir ces stades. De nombreuses discussions avec les concepteurs du CNS m'ont montré que leur démarche n'est pas vraiment structurée, même si elle comporte les étapes dont nous avons parlé plus haut.

En effet, les concepteurs suivent des règles et des principes issus de leur expérience : ils ont tendance à optimiser des critères très globaux sur l'ensemble du processus de conception. Ces critères sont :

- le produit surface x temps de conception
- répétitivité du dessin
- possibilités de la technologie, choix entre différentes technologies
- testabilité globale
- etc...

De plus, les concepteurs ne comparent entre elles qu'un petit nombre d'architectures, étant donné le travail que nécessite l'élaboration de chacunes d'elles.

Nous constatons donc que les concepteurs sont guidés vers une "bonne" solution par leur expérience et leur vue globale de tous les stades du procédé. Leur solution est bonne selon des critères multivariables qui sont difficiles à transposer au cas d'un CS (le produit surface x temps de conception n'a plus vraiment de sens).

Nous devons à présent examiner ce que peut être le fonctionnement d'un CS, nous inspirer des qualités humaines des concepteurs humains pour enrichir ses algorithmes et dégager des méthodes nouvelles pour pallier aux défauts humains.

2-2-1-3 Intégration de la chaîne de conception dans un CS

Notre hypothèse de départ est que le compilateur doit être spécialisé à chaque étape de la conception pour être efficace. Ceci est une approche différente par rapport à celle du concepteur qui peut explorer plusieurs styles d'architectures.

En effet, si l'on considère les spécifications fonctionnelles comme une entrée du compilateur, celui-ci comporte quatre étapes présentant de fortes interactions de proximité : spécifications - algorithmique - structure et conception.

Selon le type d'applications que doit traiter le CS, le niveau algorithmique sera présent ou non. De même, les mécanismes régissant le niveau architectural seront différents selon le type d'architecture que l'on utilise (série, parallèle, systolique...).

Enfin, la partie implantation utilisera des règles différentes suivant le style d'implantation visé (bit-slice, ou non).

Ces considèrations montrent que, pour longtemps encore, les compilateurs de silicium seront dédiés à des applications précises et seront construits autour de choix et d'idées directrices bien arrêtés.

Ceci sera vérifié pour chaque exemple de CS étudié plus loin, et sera garant du sérieux de l'étude.

La structure la plus complète d'un CS s'étendra donc sur les quatre niveaux que nous avons présentés, et fera une hypothèse simplificatrice sur certains niveaux.

A titre d'exemple, on peut envisager un CS qui :

- ne traite que les filtres numériques (niveau "spécifications")
- utilise les algorithmes WDF et BIQUAD (niveau "comportemental")
- qu'il réalise en architecture série (niveau "structurel")
- et qu'il implémente avec des cellules aboutées, et routage par dessus les cellules (niveau "plan de masse")

Nous pouvons introduire à ce stade une précision sur le terme "compilateur de silicium" : un logiciel qui ne prendrait en compte que les niveaux 1, 3 et 4, avec un niveau 3 atrophié serait un assembleur de silicium.

Par niveau 3 atrophié, nous entendons qu'il se contenterait de prendre un opérateur par opération, et donc de "plaquer" la description comportementale dans le silicium sans aucune optimisation.

Les critères qu'utilisent les concepteurs pour apprécier les architectures n'ont pas grand sens pour un compilateur. Il se peut que l'utilisation d'un calculateur annule leur représentativité (répétitivité, surface x temps de conception) ; il se peut aussi que le critère soit fortement lié au style d'architecture utilisé (testabilité) et soit donc une constante pour un compilateur donné.

Une qualité essentielle des concepteurs est la vue globale qu'ils ont sur l'ensemble de la chaîne ; et qui leur permet de faire des choix à un niveau, en évaluant les conséquences au niveau suivant.

Ceci peut être réalisé de deux manières par un CS : la première est d'adopter une démarche figée à chaque étape, ce qui favorisera les évaluations depuis un niveau supérieur. (Ceci est particulièrement vrai pour les étapes 3 et 4). La deuxième est de procéder par itérations pour minimiser un critère qui peut être la surface totale du circuit.

Il est difficile d'aller plus loin dans la description générale d'un CS. Disons pour résumer que la démarche globale est descendante suivant 4 niveaux plus ou moins développés. A chaque niveau, une stratégie fixée favorise l'obtention de bonnes performances et facilite les évaluations depuis le niveau supérieur. Enfin, des itérations entre niveaux peuvent être utilisées pour minimiser la surface du circuit.

La figure 2.1 reprend cette organisation.

Ingénieur système : client du compilateur	1 - Spécifications fonctionnelles : Contrainte "temps-réel" présente ou non.
Spécialiste en algorithmique : - analyse - simulation fonctionnelle - vérifications des spécifications	2 - Description comportementale : En termes d'opérations interconnectées contrainte "temps-réel" occultée
Spécialiste en architectures - simulation logico-fonctionnelle - simulation électrique simplifiée	3 - Description structurelle : Machine réalisant l'algorithme Multiplexage - Pipeline - Parallèlisme Contraine "temps réel" présente
Spécialiste en conception - simulation électrique fine - vérification de règles de dessin	4 - Description des masques : Placement et routage des blocs

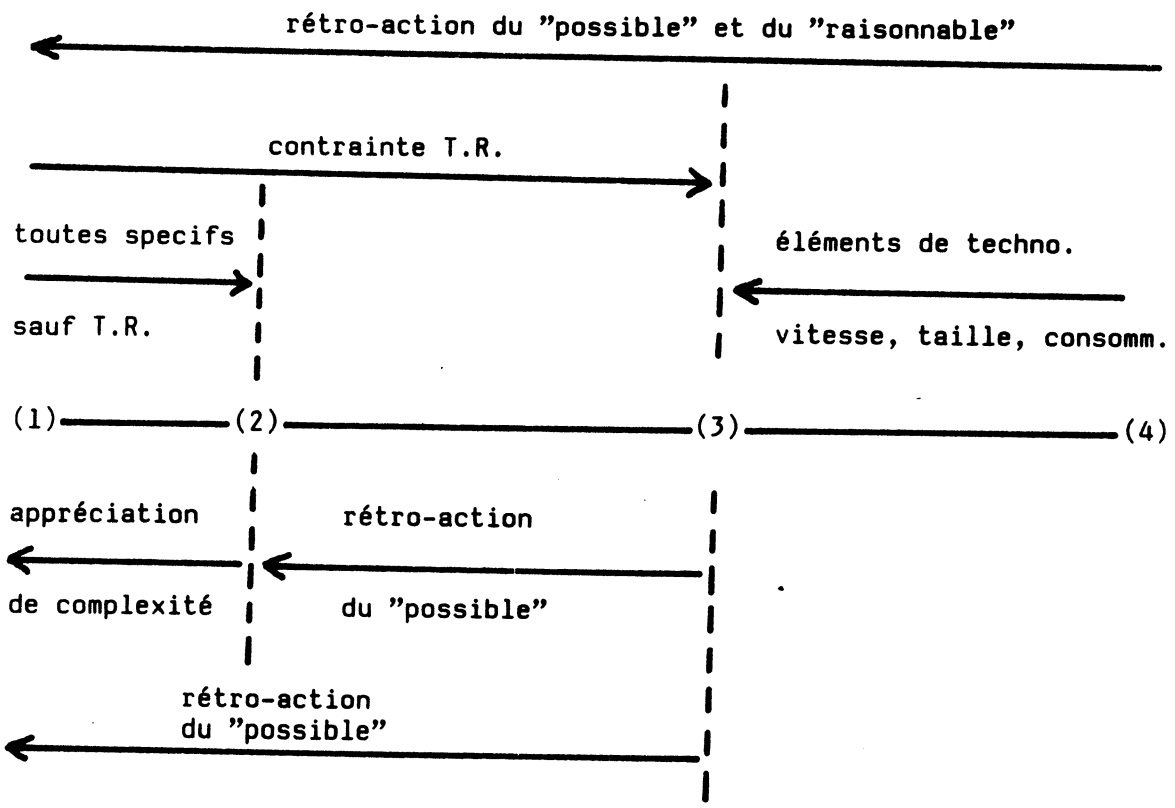


Fig : 2.1

2-2-2 Etudes des différents compilateurs de silicium

2-2-2-1 Introduction

Les compilateurs de silicium sont encore des objets de laboratoire, et les seules informations disponibles le sont par voie de publication dans les congrès. Ceci nous oblige à une certaine prudence, et à analyser soigneusement les exemples fournis pour séparer les idées restées en l'état des réalisations effectives.

Pour chacun des exemples que nous avons sélectionnés, nous essayons de les situer par rapport aux étapes de la conception dont nous avons parlé plus haut : cela favorise les comparaisons et permet entre autres de séparer les compilateurs des assembleurs de silicium.

2-2-2-2 Le compilateur de silicium MACPITTS

Ce travail est en cours au MIT depuis 1980, (B2.3) est certainement un des plus connus.

La définition fonctionnelle du circuit se fait par un langage de type "transfert de registre" comportant des déclarations, des branchements conditionnels et des sous-programmes pouvant être imbriqués. Un programme MacPitts consiste en un ensemble de processus éxécutés en parallèle, chaque processus étant divisé en étapes éxécutées séquentiellement. Les opérations éxécutées par un processus dans une étape donnée durent une période d'horloge.

Nous constatons donc que MacPitts ne comporte pas de niveau 2 et que la spécification fonctionnelle est aussi une description comportementale.

Le niveau description structurelle n'est pas réduit à une simple correspondance bijective entre opérations et opérateurs : cela fait de MacPitts un véritable compilateur de silicium.

MacPitts fait une extraction du parallèlisme, détecte les chemins exclusifs et les opérations en séquence pour déterminer le matériel minimum en évitant les conflits d'accès. Le parallèlisme est favorisé par l'attribution de bus locaux dotés de multiplexeurs.

Nous remarquons que MacPitts ne peut prendre en compte les contraintes temps réel : l'architecture produite est unique et, si elle est trop lente, il faudra redessiner les opérateurs qu'elle utilise.

Cette remarque montre que le niveau structurel de MacPitts n'est pas très perfectionné puisqu'il se contente de faire les économies les plus immédiates par rapport à la correspondance bijective évoquée plus haut.

Le contrôle de la rapidité du circuit repose donc sur le choix des opérateurs, et non sur leur nombre, leur multiplexage ou leur degré de pipeline.

Ces opérateurs sont générés automatiquement grâce à une bibliothèque de cellules paramétrables.

La partie contrôle utilise une horloge unique et une variante de machine d'état fini. Ce séquenceur comporte un compteur pour éviter de répéter une étape identique sur de nombreuses périodes d'horloge. Le remplacement de ce compteur par une pile permet de gérer les appels aux sous-programmes.

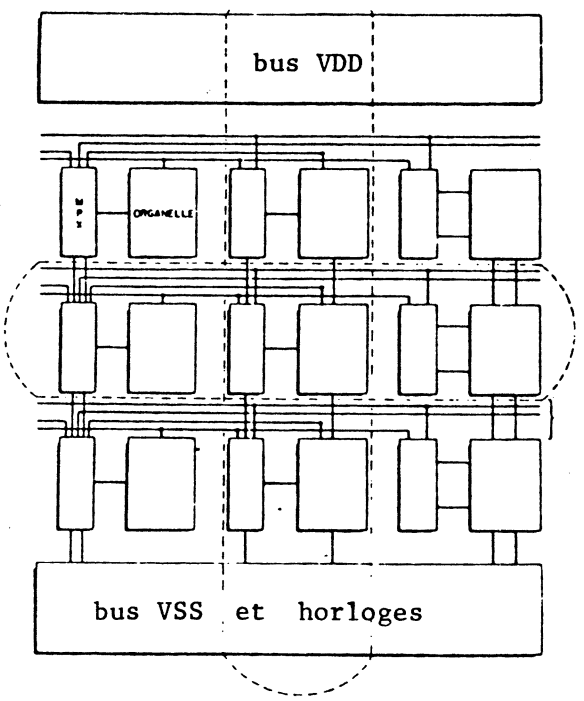


Fig : 2.2

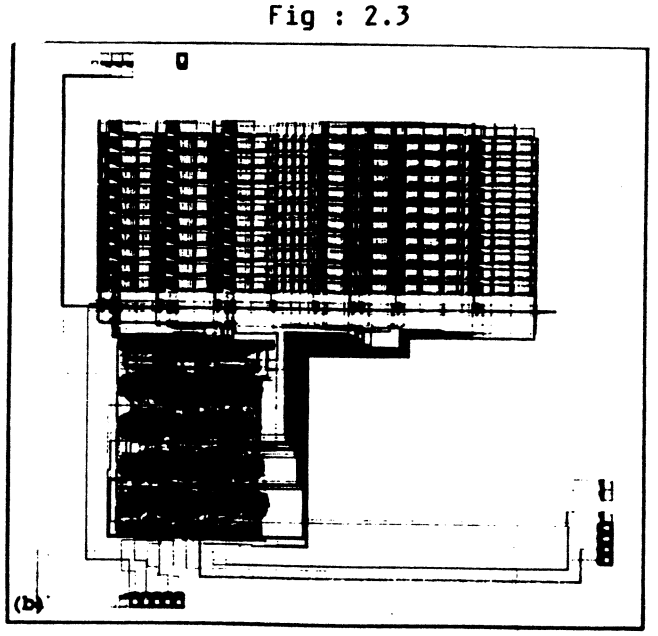
(d'après MacPitts B2.3)

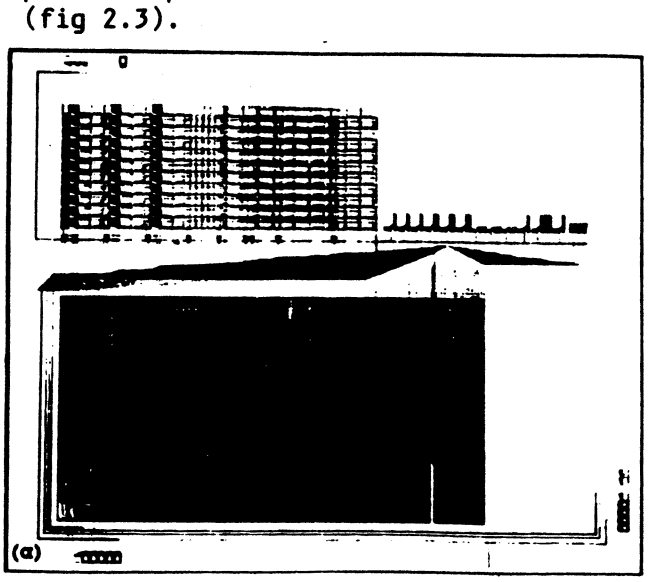
Enfin, le niveau "conception" de MacPitts utilise une implantation en tranches de bits avec plan de masse standard pour améliorer l'efficacité (fig 2.2). Un routage de canal est utilisé entre les tranches de bits des différents opérateurs, et un routage plus général connecte les plots aux différents blocs.

En conclusion, MacPitts nous semble être un produit en cours d'évolution, dont les étapes 1 et 4 sont particulièrement soignées, et dont l'étape 3 a été réalisée proprement, mais reste ouverte pour de nombreuses optimisations.

Les résultats disponibles en (B 2.3) montrent une bonne compacité pour chaque bloc généré, mais une perte de surface importante lors de la juxtaposition des parties opératives et contrôle. (fig 2.3).

Fig : 2.3





2-2-2-3 L'assembleur de silicium FIRST

Ce logiciel a été développé à EDINBOURG, (B 2.7) et est dédié aux circuits de traitement du signal.

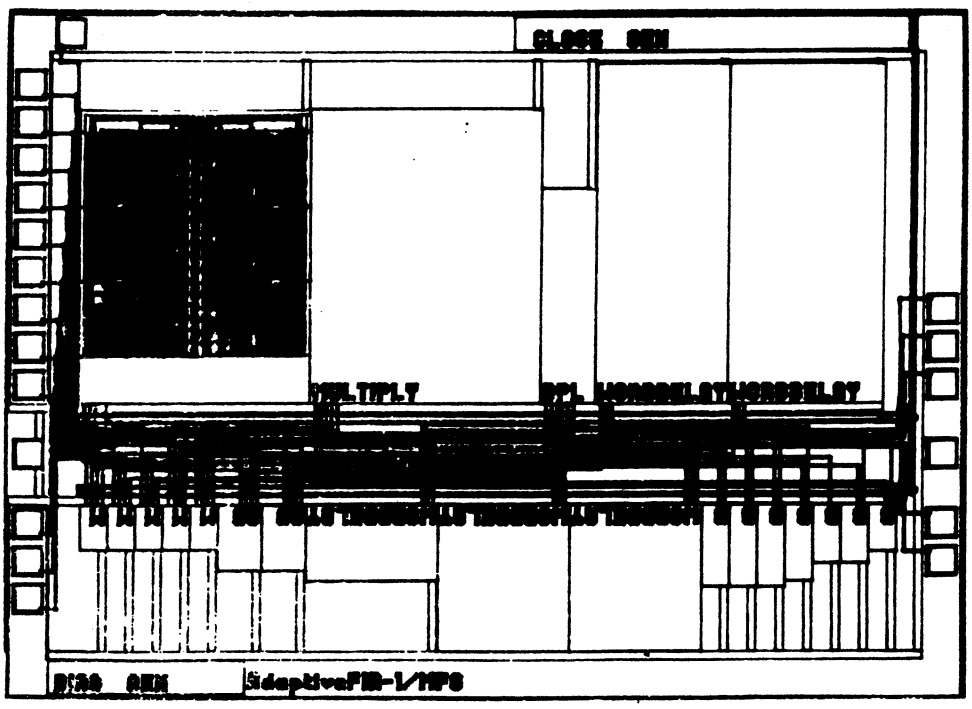
La description du circuit se fait directement au niveau structurel sous la forme d'un réseau d'opérateurs interconnectés. Ce langage autorise une définition procédurale des blocs importants.

Ceci justifie l'appellation d'assembleur de silicium puisque la tâche de FIRST est alors de composer un plan de masse pour les opérateurs fournis. Ces opérateurs sont légèrement paramètrables et ont une interface rigide avec le canal de communication. Cela en autorise la conception par des concepteurs chevronnés, et l'utilisation automatique par FIRST.

Le plan de masse se fait selon une architecture série : un bus de communication central et deux rangées d'opérateurs de part et d'autres. Le séquencement est fait par une horloge à deux phases non recouvrantes.

FIRST est orienté traitement du signal par la nature des opérateurs qu'il manipule, mais l'aspect temps réel doit être assuré dans la description structurelle d'entrée puisque FIRST ne la modifie pas.

Les performances en surface des circuits obtenus sont intéressantes (25 % de perte de Si) mais l'utilisation d'une architecture série réduit l'importance de la surface perdue par les bus. L'essentiel des pertes se trouve probablement dû aux hauteurs différentes de chaque opérateur (fig 2.4).



(d'après FIRST B2.7)

Fig : 2.4

2-2-2-4 Le compilateur de silicium ARSENIC

L'article dont nous disposons (B 2.6) décrit un projet de compilateur de silicium mené à l'université d'Illinois (USA). Le degré d'avancement du travail semblait faible à l'époque où a été écrit cet article, mais l'exposé a eu le mérite de définir exactement les termes "compilateur" et "assembleur" de silicium.

L'analyse par niveaux faite ici est similaire à celle que nous avons développée dans ce chapitre.

Le langage de description fonctionnelle de ARSENIC est une liste d'algorithmes. Chacun d'eux est écrit dans un langage sans GOTO comprenant des déclarations et des instructions de branchement conditionnel.

Ces algorithmes sont destinés à s'éxécuter en parallèle, et il n'y a à ce moment aucune relation entre les variables et les futurs registres.

Le niveau 2 dit "algorithmique" est donc incorporé au niveau 1 selon notre décomposition par niveaux de la chaîne de conception.

L'approche adoptée pour le niveau 3 (description structurelle) est intéressante car c'est la plus élaborée de toutes celles rencontrées jusqu'à présent. L'ensemble des algorithmes est transformée en graphe "flot de données" où chaque noeud représente une micro-opération.

ARSENIC ferait alors un inventaire des fonctions mises en oeuvre dans ce graphe, et allouerait au circuit un certain nombre de modules. Ces modules contiennent soit de la mémoire, soit de petites UAL (Unités Arithmétiques et Logiques) réalisant certaines fonctions et différentes les une des autres.

Il suffirait alors de dire qu'une micro-instruction éxécute un certain nombre de noeuds dans le graphe, et de faire une partition du graphe pour décider quelle micro-opération sera effectuée par telle micro-instruction pendant la période d'horloge considérée.

Nous faisons remarquer que l'idée qui consiste à utiliser des graphes et des techniques de partitionnement inspirée des méthodes de Project Management se retrouve chez d'autres auteurs (B2.5).

Le mode d'élaboration de la description structurelle est ici strictement synchrone dès le départ. Par ailleurs, l'utilisateur peut limiter le nombre de modules qu'ARSENIC utilisera pour faire son architecture. Cela laisse supposer qu'ARSENIC utilise le multiplexage, mais cela n'est probablement qu'une idée à la date de parution de l'article.

L'aspect temps réel de certains algorithmes n'est pas intégré au mode de travail d'ARSENIC, et après une évaluation à postériori, une optimisation des opérateurs du chemin critique serait éventuellement à faire.

La description structurelle de l'architecture étant terminée, ARSENIC passe à l'implantation. Chaque module possède un plan de masse fixé selon qu'il comporte de la mémoire ou non. Le contrôle local de chaque module y est incorporé, et les modules opérateurs sont organisés en tranches de bits.

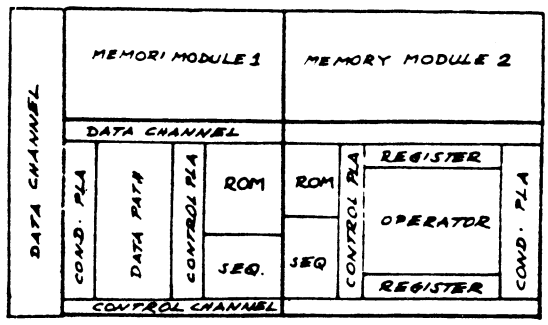


Fig : 2.5 : sytème à :

- deux modules mémoire
- deux modules processeur

(d'après ARSENIC B2.6)

Les modules sont alors juxtaposés, et des canaux de communication pour données et contrôle global sont intercalés si nécessaire (fig 2.5).

Nous conclurons en disant qu'ARSENIC contient beaucoup d'idées intéressantes qu'il conviendra de valider. C'est le premier compilateur de silicium qui traite de manière approfondie le niveau structurel, même si l'article laisse dans le vague la frontière entre ce qui est réalisé et ce qui reste à faire.

2-2-2-5 Boite d'outils CAO réalisée à LOUVAIN (Belgique)

L'université catholique de LOUVAIN a réalisé un ensemble d'outils dont l'activation et l'enchainement sont faits par l'utilisateur, et qui sont dédiés à la réalisation de circuits de Traitement du Signal.

Enchainer ces outils permet de couvrir chacuns des stades de la conception, moyennant une stratégie fixée pour chaque stade.

Les spécifications fonctionnelles peuvent être décrites de deux manières :

- si il s'agit d'un circuit différent d'un filtre, la description se fait sous forme d'un réseau d'opérations interconnectées. Les opérations sont celles que l'on rencontre dans le domaine du traitement du signal.

- si il s'agit d'un filtre numérique, la description se fait sous forme d'une fonction de transfert, rapport signal à bruit et fréquence d'échantillonnage.

Le niveau algorithmique ou de "description comportementale" est activé uniquement dans le cas des filtres. Dans le cas contraire, la description d'entrée est déjà au niveau du comportemental. Ce niveau utilise deux algorithmes différents pour calculer l'architecture du filtre : l'algorithme de REMEZ et l'algorithme WAVE DIGITAL FILTER. Ces algorithmes s'efforcent de minimiser la taille des coefficients et leur nombre de bits à 1 afin d'utiliser le plus possible des multiplieurs spécialisés.

L'approximation des cycles limites est calculée et une description comportementale de l'architecture est donnée dans la même forme que pour les algorithmes différents des filtres.

L'hypothèse qui dirige les algorithmes de la partie "description structurelle" de ce compilateur est l'utilisation d'une architecture série. Dans une version précédente (B 2.8) le graphe des opérations était simplement "plaqué" sur le silicium en prenant un opérateur par opération.

Dans la version actuelle un système expert est en cours de développement pour utiliser les techniques de multiplexage et de pipeline afin d'améliorer le "plaquage" initial.

La dernière étape d'implantation se fait en utilisant les cellules d'une bibliothèque, dans deux technologies CMOS et NMOS suivant deux stratégies différentes.

En NMOS, les cellules ont des contacts par aboutement et les connexions globales en métal. Les horloges et alimentations sont transmises par aboutement à travers les cellules.

En CMOS, la méthode est différente mais non exposée.

Des résultats sont présentés (fig 2.6 à 2.7) qui montrent de bonnes densités d'intégration pour un compilateur de silicium. Il faut toutefois se souvenir que chaque opération est controlée par l'opérateur humain, ce qui augmente certainement les performances en surface de l'ensemble.

	Ref.	Function(s)	Transistors Technology	Active area (mm^2)	Pole-zero Main characteristics
CHIP 1	Fig. 4	9th order dedicated LDI Chebyshev (and test of NORA-CMOS library)	4500 NORA-CMOS 5 µm	10.5	16 bit I/O Max. clock rate : 11 MHz/1.1 mm^2 per pole-zero / 650 kHz max sample rate. Scan path testable design.
CHIP 2	Fig. 5	Weakly programmable digital audio. Can be programmed into offset / low-pass / and programmable parametric equalizer.	5850 NORA-CMOS 5 µm	18.6	16 bit in/out. 32 bit internal leads to internal S/N ratio better than 100 dB. 44,1 kHz sample rate - 2 stereo channels. Notice programmability leads to lower density but is possible.
CHIP DESIGN 3	Fig. 6	Complete digital audio signal processor with 10 graphic equalizers (only one in Fig. 6).	33000 nMOS 3 µm	25	In processing. Shows power of LEGO style and bit-serial compact chip for very high quality custom consumer DSP. Guarantees internal 100 dB S/N per 12 functions.
CHIP DESIGN 4	-	TD - FDM 16 channel wave digital transmultiplexer [22] (Floorplan design using cell library).	66000 NORA-CMOS 5 µm 2 µm	97 30	This is a multirate filter realizing an equivalent of 446 poles. This extreme density of 0,07 mm^2/pole-zero (2 µm) is due to high degree of multiplexing is not possible with switched-capacitors. Can be made scan path testable !

Table I : CHIP 1, CHIP 2 have been designed and processed. CHIP 3 is in processing. CHIP 4 is a design excercise for a very large DSP system.

(d'après LOUVAIN B2.8)

Fig : 2.6

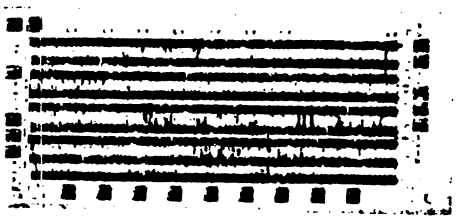


Fig. 4. NORA-CMOS 9th order dedicated LDI filter on 10,5 mm**2 in 5µm technology.

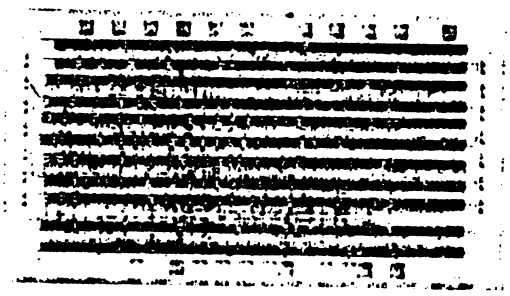


Fig. 5. NORA-CMOS a digital-audio filter programmable as offset, scratch and parametric equalizer.

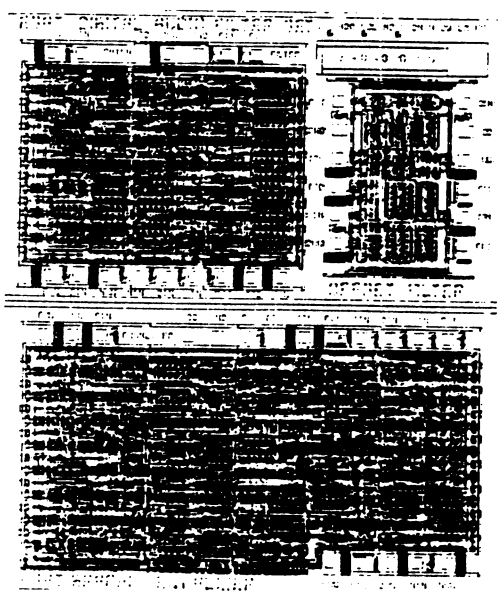


Fig. 6. Three of the 12 filters for a digital audio signal processor chip designed in LEGO style. Realizes 44 programmable poles on 25 mm**2 in 3µm n-MOS.

(d'après LOUVAIN B2.8)

Fig : 2.7

2-2-2-6 Conclusion

L'étude de ces compilateurs de silicium nous a permis de dégager plusieurs idées :

- un processus de conception se décompose en 4 niveaux bien précis : spécifications, description comportementale, structurelle et implantation.

- un véritable compilateur de silicium part au moins de la description comportementale et possède un module de définition structurelle non dégénéré.

- pour être efficace, un compilateur de silicium doit avoir une idée directrice pour chaque niveau qu'il traite.

Nous allons donc nous servir de ces idées pour définir notre travail :

IMHOTEP est un générateur automatique d'architectures. Il accepte en entrée une description comportementale du filtre et une valeur de contrainte "temps réel", et il produit en sortie la description structurelle de la machine réalisant l'algorithme sur une surface de silicium minimale. IMHOTEP automatise donc le passage du niveau 2 au niveau 3 de la chaine de conception et il utilise des opérateurs le plus souvent parallèles. Il accepte les définitions comportementales comprenant un nombre fini de types d'opérations, et à structures invariantes dans le temps (pas de multiplexeur).

Par ailleurs, IMHOTEP est une partie d'un compilateur de silicium pour filtres numériques. La partie "algorithmique" de ce compilateur délivre la description comportementale du filtre en utilisant essentiellement les algorithmes BIQUAD, FIR et TREILLIS. La partie "élaboration de l'architecture" est réalisée par IMHOTEP et la partie "implantation" utilisera une architecture dédiée en tranches de bits, s'appuyant sur une bibliothèque de cellules NMOS.

Nous remarquons que ce projet est très proche du travail réalisé à LOUVAIN, avec les différences suivantes :

- les algorithmes de la partie 2 sont différents (BIQUAD au lieu de WDF).

- l'architecture de la partie 3 est parallèle et multiplexée (au lieu de série).

- l'enchainement des opérations entre 3 et 4 sera automatique dès que le niveau 2 aura produit son résultat (sans intervention humaine).

- mais que les types de schémas d'entrée sont limités aux seuls filtres numériques.

3- ETUDE DES DIFFERENTS ALGORITHMES DE FILTRAGE NUMERIQUE

3-1- Introduction au filtrage d'un signal

3-1-1 Notions de filtrage
3-1-2 Conversion analogique-numérique
3-1-3 Notions préliminaires de filtrage numérique

3-2- Le filtrage numérique d'un signal

3-2-1 Introduction

3-2-1-1 Les différentes familles de filtres
- les filtres à fonction de transfert invariable
- les filtres à fonction de transfert variable

3-2-1-2 La définition d'un filtre

- les paramètres de filtrage
- les paramètres liés à l'arithmétique
- les paramètres liés à l'environnement

3-2-2 Réalisation de la fonction de filtrage

3-2-2-1 Les filtres à réponse impulsionnelle finie (RIF)

3-2-2-2 Les filtres à réponse impulsionnelle infinie (RII)

3-2-2-2-1 Les méthodes de synthèse analogique
3-2-2-2-2 Utilisation de méthodes itératives : synthèse dans le plan des Z

3-2-2-3 Modifications de H(f) apportées par l'arithmétique

3-2-2-3-1 Influence du nombre de bits des coefficients
3-2-2-3-2 Influence du nombre de bits des mémoires de données
- bruit de calcul dû à l'arrondi après multiplication
- bruit de calcul dû aux dispositifs de saturation

3-2-2-3-3 Auto oscillations ou cycles limites

3-2-2-4 Conclusion

3-3- Les différentes structures de filtres numériques

3-3-1 Les filtres en chaînes de quadripôles

3-3-1-1 Les filtres en treillis
3-3-1-2 Les filtres en échelles simulées
3-3-1-3 Les filtres d'onde numériques
3-3-1-4 Conclusion

3 - ETUDES DES DIFFERENTS ALGORITHMES DE FILTRAGE NUMERIQUE

3-1- Introduction au filtrage d'un signal.

3-1-1 Notion de filtrage.

Tout signal analogique s(t) peut être représenté par une fonction complexe S(f) appelée spectre du signal s(t).

Le filtrage d'un signal consiste en une altération controlée de son spectre. Cette altération est totalement décrite par une fonction complexe H(f), appelée fonction de transfert du filtre, et ayant le même domaine de définition que S(f).

Le résultat du filtrage de S par H, ou signal filtré U(f), est défini par U(f) = S(f) x H(f) ; ou par u(t) = (s*h)(t) dans le domaine temporel avec "*" représentant l'opérateur de convolution.

On obtient les grandeurs en majuscules à partir des grandeurs en minuscules par une transformée de Fourier.

La fonction de transfert H(f) définit une opération linéaire dans l'espace des fréquences. Cette opération est physiquement réalisable moyennant une condition de stabilité (la réponse temporelle n'augmente pas indéfiniment si l'entrée cesse d'être non nulle) et une condition de causalité (la réponse ne peut apparaître qu'après l'action).

Les méthodes de synthèse classiques de H(f) impliquant uniquement des contraintes sur l'amplitude conduisent à des fonctions de transfert dites à déphasage minimal. Des contraintes supplémentaires sur la phase, déphasage linéaire par exemple, ne peuvent conduire qu'à une augmentation du degré de H(f) et par conséquent à une augmentation du déphasage à 2π près.

3-1-2 Conversion Analogique-Numérique

La transformation qui permet le passage de la forme analogique naturelle d'un signal à une représentation numérique se décompose en deux étapes :

- Une opération d'échantillonnage à la fréquence fe qui consiste à prélever une information proportionnelle à l'amplitude du signal continu s(t) tous les instants Te avec Te = 1/fe.

Le signal s(t) se trouve donc remplacé par une suite d'échantillons s(nTe). Afin d'assurer la réciprocité de cette opération, le théorème de Shannon impose à fe d'être au moins deux fois supérieure à la plus haute fréquence contenue dans le spectre de s(t).

- La deuxième étape est une opération de codage numérique de l'information analogique échantillonnée. Ce codage se fait sur des mots de Nb bits. Nb dépend de la dynamique du signal et de l'erreur de quantification que l'on peut tolérer.

Soit q la valeur de l'échelon de quantification, le bruit résultant du codage a une puissance égale à

$$P = q^2/12$$

avec une répartition spectrale uniforme dans la bande du signal : c'est un bruit blanc.

On constate donc que le rapport signal sur bruit en sortie d'un codeur linéaire est lié au nombre de bits :

$$S/B = 3/2.2^{Nb} \quad \text{(en puissance)}.$$

Nous voyons que la conversion analogique-numérique introduit de profondes modifications dans la structure de l'information.

Si le codage ne dégrade guère que le rapport signal sur bruit, l'opération d'échantillonnage temporel (échantillonnage de s(t)), introduit une modification plus profonde sous la forme d'une périodicité de la fonction S*(f) dans l'espace des fréquences.

En effet, échantillonner s(t) à fe revient à multiplier s(t, par la fonction

$$d(t) = \sum_{n=-\infty}^{+\infty} \delta(t - nTe)$$ encore appelée peigne de dirac.

Dans le domaine fréquentiel, la convolution de s(f) par d(f) soit

$S(f) * K \sum_{n=-\infty}^{+\infty} \delta(\nu - \nu/Te) = \sum_{-\infty}^{+\infty} S(f - n/Te)$ conduit à une périodicité de son spectre.

L'échantillonnage est donc une modulation en amplitude par une somme infinie de porteuses. On se retrouve donc avec une périodicité du spectre du signal échantillonné.(Fig. 3.1)

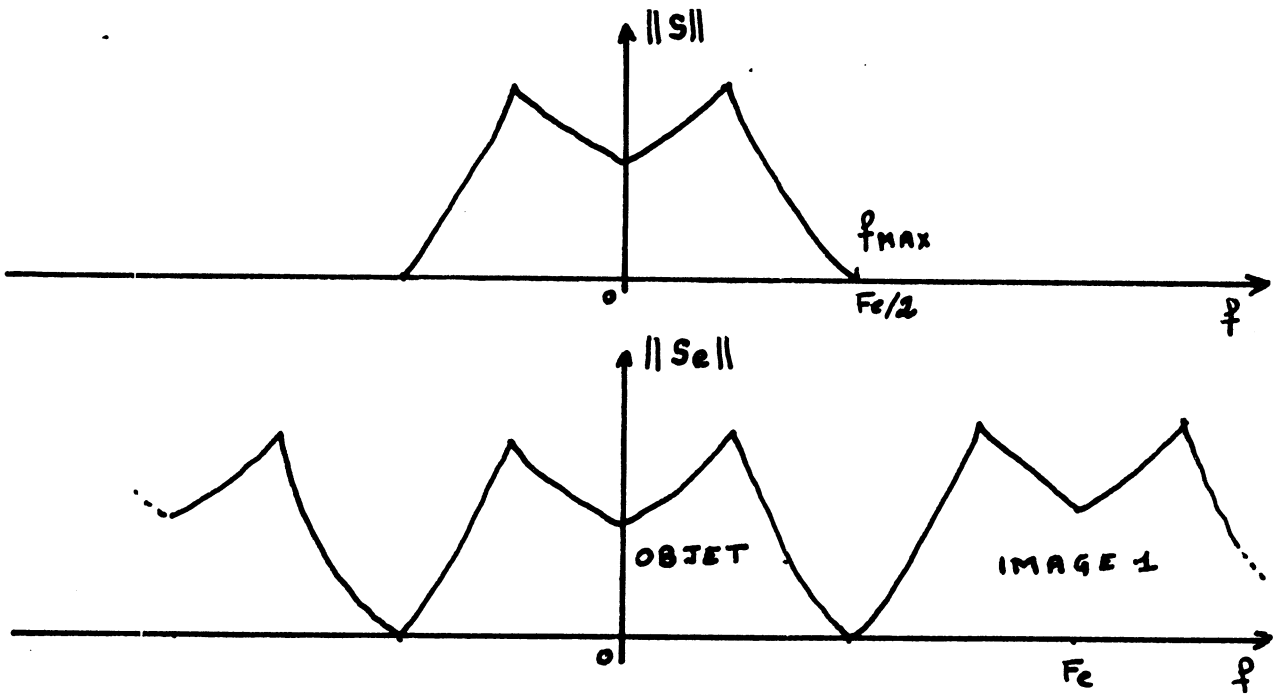


Fig. 3.1

On comprend ainsi le sens du théorème de Shannon : $f_{MAX} \leq fe/2$ car si on ne respecte pas cette condition, après échantillonnage, on a un recouvrement entre la bande objet et la lère bande image centrée sur fe et l'échantillonnage n'est plus réversible (perte d'information).

En conclusion, nous pouvons donc dire que la conversion analogique - numérique doit obéir à des règles bien précises pour être d'une part réversible, d'autre part peu bruitée. Si ces règles sont respectées, on a accès sans trop d'inconvénients aux immenses possibilités du traitement numérique.

3-1-3 Notions préliminaires de filtrage numérique.

Soient $x(n)$ et $X^*(f)$ la suite des échantillons d'entrée et son spectre en fréquence. Soit $H^*(f)$ la fonction de filtrage. Soient $y(n)$ et $Y^*(f)$ la suite des échantillons de sortie et son spectre en fréquence.

Le filtrage numérique de la suite $x(n)$ consiste à combiner au moyen d'additions, de multiplications et de retards les $x(n)$ pour obtenir une suite $y(n)$ telle que $Y^*(f) = H^*(f) \times X^*(f)$ sur tout le spectre.

Ces calculs étant faits en numérique, nous allons donc rencontrer plusieurs catégories de problèmes :

- les opérateurs sont en représentation binaire : problème de précision problème de bruit de calcul.
- les calculs sont parfois récursifs : problème de stabilité et de cycles limites observés en sortie dans certaines configurations de l'entrée.
- les échantillons arrivent à une certaine cadence : problème de rapidité de calcul.

Nous allons reprendre et développer ces questions dans le chapitre suivant consacré au filtrage numérique.

Le lecteur désireux d'appronfondir le sujet pourra avantageusement se reporter à l'ouvrage cité en bibliographie B3.1.

3-2- Le filtrage numérique d'un signal.

3-2-1 Introduction.

Lorsqu'on parle de filtrage, le paramètre essentiel est la fonction de transfert. Les différentes manières de spécifier cette fonction de transfert vont donc engendrer différentes familles de filtres.

3-2-1-1 Les différentes familles de filtres.

- Les filtres à fonction de transfert invariable dans le temps. Ce sont des filtres dont les coefficients sont constants. Ce type de filtrage regroupe 90 % des filtres existants et est le plus simple à mettre en oeuvre. Une fois la fonction de transfert déterminée en fonction de tel ou te besoin, on calcule les coefficients du filtre qui sont fixés une fois pour toute.

- Les filtres à fonction de transfert variable. On parle dans ce cas de filtrage adaptatif. Suivant la fréquence de réactualisation des coefficients, on a affaire à différentes classes de filtrage adaptatif parmi lesquelles deux sont importantes :

. les filtres à coefficients rapidement variables. Ce sont par exemple les annuleurs d'échos acoustiques où les coefficients sont fonction des données incidentes.

. et les filtres à coefficients lentement variables. Ce sont des filtres dits de compromis dont les coefficients changent en temps différé. Ils sont réservés à des vitesses de traitement plus lentes que les précédents car leur qualité d'adaptation est moins bonne.

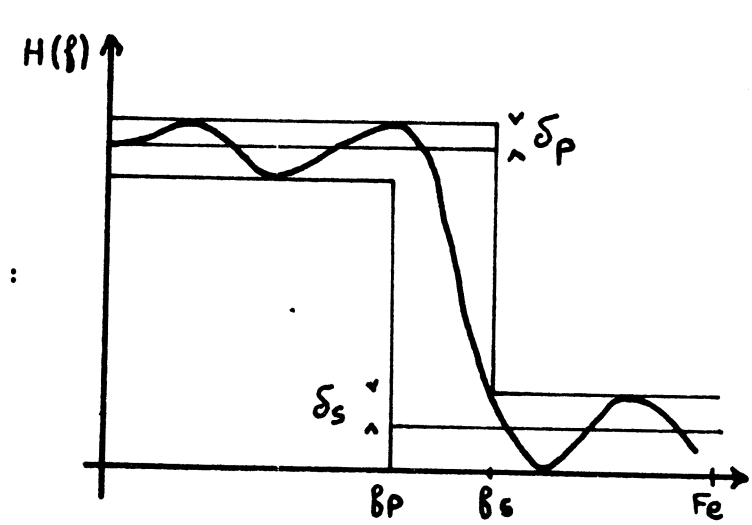
Les fréquences de réactualisation vont donc du temps réel à une fois par an.

Dans cette étude, nous nous consacrerons principalement au cas du filtrage à coefficients constants ; mais en essayant de ne jamais poser de verrou qui interdise le filtrage adaptatif.

3-2-1-2 La définition d'un filtre.

Pour définir totalement un filtre dans son environnement, il faut un certain nombre de paramètres regroupés en trois familles :

- les paramètres de filtrage : fe, fp et fs, δp et δs qui sont les marges d'erreur en bande passante et en bande atténuée et des informations : contraintes sur la phase, (Fig. 3.2)



- les paramètres liés à l'arithmétique :
  Rapport signal/bruit, maximas en matière de cycles limites,

- les paramètres liés à l'environnement :
  Nombre de bits en entrée et en sortie, modes de transmission série ou parallèle, en entrée ou en sortie.

3-2-2 Réalisation de la fonction de filtrage.

Cette réalisation laisse une large place à l'imagination, et de très nombreuses techniques sont mises en oeuvre. On trouve dans ce domaine de véritables écoles de pensée, et les concepteurs utilisent généralement un seul mode de réalisation. Un outil de CAO de filtre trouve donc ici sa pleine utilité car il peut réaliser une synthèse entre toutes les écoles.

Pour synthétiser un filtre, on dispose de deux grandes moyens :

Soit on le calcule en prenant pour modèle un filtre analogique existant. Cela présente certains avantages que nous examinerons plus loin.

Soit on le calcule directement en numérique (dans le plan des Z) par des méthodes d'analyse numérique.

On peut regrouper tous les filtres en deux grandes familles que nous allons étudier successivement.

3-2-2-1 Les filtres à réponse impulsionnelle finie (R.I.F.).

Dans un filtre RIF, si x(n) est la suite des échantillons d'entrée, la sortie s'exprime par :

$$y(n) = \sum_{i=0}^{n-1} ai\ x(n - i)$$

et la fonction de transfert en Z par : $H(Z) = \sum_{i=0}^{n-1} ai\ Z^{-i}$.

Si les coefficients sont symétriques ($a_i = a_{n-i}$), le filtre est à phase linéaire. On peut montrer que la suite des coefficients ai est la réponse impulsionnelle du filtre.

Méthodes de calcul des coefficients :

Les méthodes de calcul des filtres RIF sont presque exclusivement des méthodes numériques. On ne rencontre qu'exceptionnellement des synthèses à partir de filtres analogiques. Nous allons donc examiner les différentes méthodes numériques dont nous disposons.

* On peut développer en série de Fourier la fonction H(f) à approcher.

Des contraintes matérielles vont limiter à N le nombre des coefficients à calculer. Cela revient à multiplier la réponse impulsionnelle h(t) par une fenêtre de largeur NT centrée sur N/2.T. La fonction Hr(f) réellement obtenue sera le produit de convolution de la fonction idéale H(f) et de la transformée de Fourier de la fenêtre utilisée pour limiter à N les coefficients.(Fig. 3.3)

Du choix de la fenêtre va dépendre la qualité de la fonction réelle obtenue, on utilise généralement des fenêtres de Hamming ou de Dolf-Chebyshev.

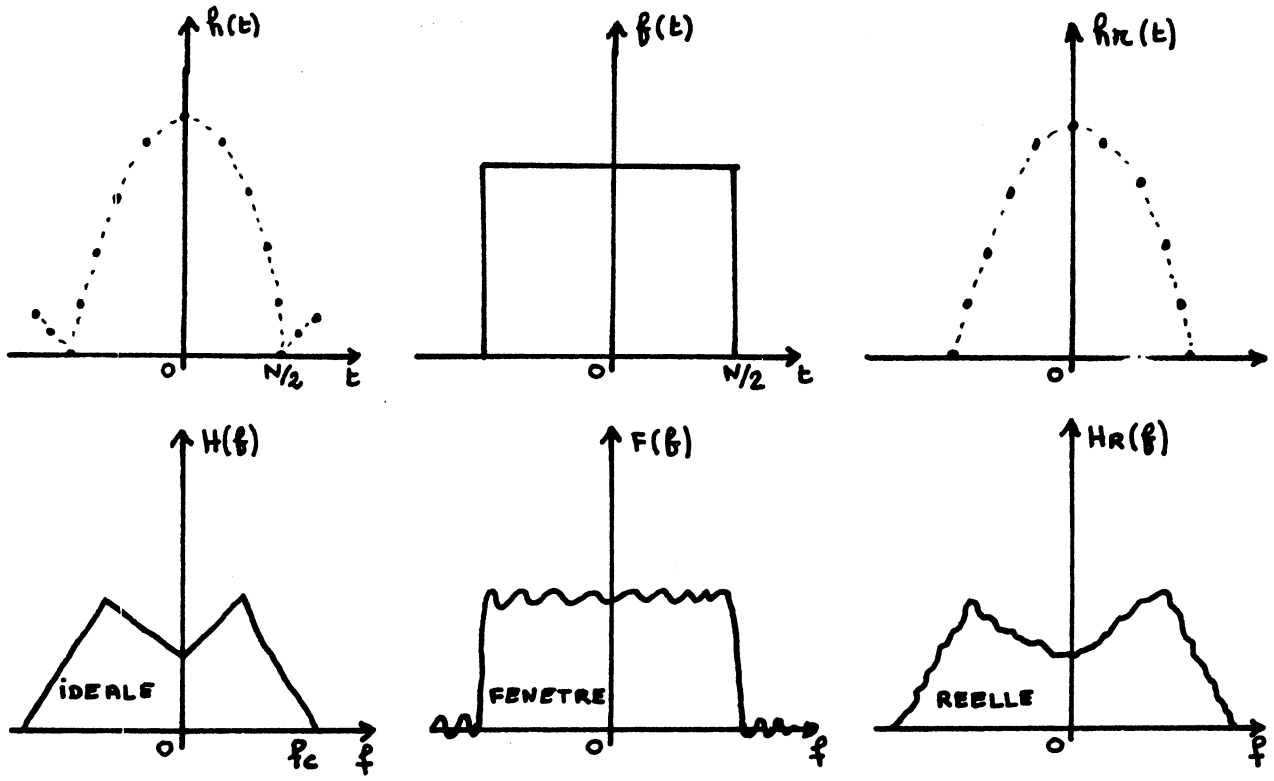


Fig. 3.3

Les limitations de cette méthode résident dans le fait que l'emploi de fenêtres ne supprime pas les effets du phénomène de Gibbs (ondulations lors de la restitution d'un échelon). Cette méthode donne des filtres à nombre de coefficients bien trop élevé.

* En se limitant à N coefficients, on peut approcher la fonction désirée par la méthode des moindres carrés. Cete manière est plus performante mais donne des ondulations d'amplitude non constante et ne permet pas de maîtriser les ondulations entre les points d'échantillonnage.

* La méthode la plus performante est itérative (algorithme de REMEZ) et donne un filtre à phase linéaire à ondulations d'amplitude constante. De plus, on maîtrise la fonction de transfert en tous points du spectre, même entre les points d'échantillonnage.

Ces filtres RIF sont stables (pas de pôles dans H(f)), mais ils nécessitent un nombre de coefficients élevé dès que la fonction H(f) devient sélective. Ils sont utilisés dès que la linéarité en phase est exigée (encore que des structures RII à phase linéaire commencent à apparaître) ou dès que la stabilité est primordiale, en particulier en filtrage adaptatif.

Si le filtre est à phase linéaire, la symétrie ou l'antisymétrie des coefficients permettent de diviser par deux le matériel.

Tous les problèmes posés par la réalisation physique des opérations arithmétiques sont communs avec les autres types de filtres, et seront donc étudiés à la fin du chapitre dans un paragraphe spécial.

3-2-2-2 Les filtres à réponse impulsionnelle infinie (R.I.I.).

La sortie des filtres RII s'exprime par :

$$y(n) = \sum_{0}^{n-1} ai\ x(n-i) - \sum_{1}^{L-1} bj\ y(n-j)$$

et la fonction de transfert en Z par : $H(Z) = \dfrac{\sum_{0}^{n-1} ai\ Z^{-i}}{1 + \sum_{1}^{L-1} bj\ Z^{-j}}$

(On a fréquemment L = N).

Pour calculer les filtres RII, contrairement aux filtres RIF, on dispose des deux méthodes énoncées au paragraphe 2-2.

3-2-2-2-1 Les méthodes de synthèse analogique.

Nous allons tout d'abord examiner les procédures qui s'inspirent de filtres analogiques existants. Elles ont un grand intérêt car certaines propriétés des filtres modèles se retrouvent dans les filtres numériques réalisés.

Par exemple, les filtres analogiques étant réalisés à partir de composants passifs R, L, C il était naturel de choisir des structures peu sensibles à des variations des paramètres R, L, ou C (vieillissement, dérives en température...). Les filtres numériques réalisés sur ce modèle seront donc peu sensibles aux variations des coefficients dues au nombre limité de bits pour les représenter.

Ceci permet donc des structures nécessitant peu de bits dans les opérateurs arithmétiques.

Ces filtres sont en général des structures en chaînes de quadripôles :

- filtres en échelles simulées (surtout employés en capas commutées)
- filtres d'ondes (Wave Digital Filters).
  Ces filtres ont beaucoup d'opérateurs, mais travaillant sur peu de bits. Leur bruit est généralement faible.
- filtres en treillis
  Très utilisés en traitement de la parole et en filtrage adaptatif. On peut remarquer que la structure treillis peut servir à faire des filtres RIF ou des filtres RII purement récursifs (pas de zéro).

Toutes les structures précitées seront examinées en détail plus loin.

Méthodes de calcul des coefficients.

* Utilisation de fonctions analogiques standard.

Des fonctions modèles du type Butterworth ou Chebyschev sont utilisables, moyennant une transformation, pour générer des filtres numériques.

Exemple : soit H(w) la fonction à réaliser

et $\left|F(w)^2\right| = \dfrac{1}{1 + w^{2n}}$ la fonction de Butterworth normalisée

On identifie $\left|F(w)^2\right| = \left|H(jw)^2\right| = \left|H(s) \times H(-s)\right| = \dfrac{1}{1 + w^{2n}}$

On affecte à H(s) les pôles à gauche de l'axe imaginaire pour avoir un filtre stable et on regroupe les cellules :

$$H(s) = \frac{1}{1 - s} \prod_{k=1}^{(n-1)/2} \frac{1}{s^2 + 2s\cos(\pi k/n) + 1}$$ (pour n impair par exemple)

Pour passer de cette fonction non périodique à H(w) périodique, on applique une transformation bilinéaire qui transforme l'axe réel en le segment 0, fe avec [partie à gauche de] (z)] ---> [intérieur cercle unité] .

Cette transformation est $T(z) = s = \frac{2}{T} \frac{1 - Z^{-1}}{1 + Z^{-1}}$

On remplace donc s par Z dans H(s) et on obtient H(z) sous forme d'un produit de fractions d'ordre 2, ce qui est une forme aisément réalisable comme nous le verrons plus loin.

Les filtres de Butterworth et plus généralement polynomiaux sont faciles à calculer, mais ils manquent de sélectivité par rapport aux filtres elliptiques.

3-2-2-2-2 Utilisation de méthodes itératives : synthèse dans le plan des Z.

Ce sont des techniques de minimisation de l'erreur quadratique moyenne entre le modèle et la fonction d'approximation, sur une distribution de points entre 0 et fe. On peut aussi minimiser ou ajuster des caratéristiques du filtre telles que le temps de propagation ou la phase.

Nous pouvons également remarquer que les filtres RII ne sont pas toujours stables, mais ils peuvent être très sélectifs avec peu de coefficients.

Après avoir calculé les coefficients idéaux réalisant une fonction de transfert donnée, nous devons examiner toutes les conséquences de l'arithmétique finie sur l'approximation initiale. Ceci conduit fréquemment à modifier les coefficients, ce qui explique que des programmes d'analyse suivent souvent les programmes de calcul de coefficients.

3-2-2-3 Modifications de H(f) apportées par l'arithmétique.

3-2-2-3-1 Influence du nombre de bits des coefficients.

Toute réduction de ce nombre est la bienvenue car elle diminue la taille et la durée des multiplications. Cette réduction modifie la forme de la courbe H(f), et, dans la limite des marges d'erreur p et s, on essaye de rendre cette réduction maximale. On remarque que tous les coefficients n'ont pas la même influence sur la courbe H(f) et que, si certains ne peuvent être que peu modifiés, d'autres admettent d'être codés sur très peu de bits.

3-2-2-3-2 Influence du nombre de bits des mémoires de données.

Après chaque multiplication, le nombre de bits de la donnée augmente et il peut être nécessaire de le limiter :

- dans les filtres RIF, on peut parfois admettre un additionneur final travaillant sur beaucoup de bits, et ne tronquer le résultat qu'à la sortie du filtre.

- dans les filtres RII, la récursivité du calcul impose une troncature peu après la multiplication, et en tous les cas avant la multiplication suivante sous peine de voir grossir indéfiniment le mot mémoire.

De plus, en présence de gains importants, il faut parfois éviter les débordements de capacité par l'emploi d'un dispositif de saturation. Ces deux dispositifs, troncature et saturation, sont des sources de bruit :

* bruit de calcul dû à l'arrondi après multiplication.

Ce bruit est assimilable à un bruit blanc de puissance $v^2 = q^2/12$, on le ramène souvent à l'entrée, pour la modélisation, où il s'ajoute au bruit du signal incident. On étudie en général surtout le bruit provenant des multiplications de la partie récursive car il est plus complexe alors que celui provenant de la partie non récursive ne fait que s'y ajouter sans re-subire la chaîne complète.

* bruit de calcul dû aux dispositifs de saturation.

Ce bruit doit être évité à tout prix car il est très important et totalement imprévisible.

En conclusion, on peut dire que le principe qui dirige la détermination du nombre de bits des mémoires est le suivant : il faut que le bruit de calcul ajouté par le filtre soit inférieur ou égal au bruit déjà présent à l'entrée.

Pour déterminer cette capacité, il faut évaluer la dynamique du signal dans le filtre et ajuster à chaque fois le nombre de bits nécessaire pour limiter le bruit au niveau que l'on s'est donné.

C'est ici, entre autres, qu'une réalisation personnalisée d'un circuit intégré VLSI de filtrage trouve tout son intérêt, car on peut, au niveau du silicium, faire ce que l'on veut quant au nombre de bits des registres. Contrairement aux circuits programmables dont les capacités de registre sont fixées définitivement.

3-2-2-3-3 Auto-oscillations ou cycles limites.

On trouve trois sortes d'oscillations dans un filtre :

- les oscillations entretenues qui font du filtre un oscillateur et qui sont donc à prohiber. Ceci est faisable moyennant des conditions de localisation des pôles.

- les oscillations de grande amplitude à la suite d'un écrêtage. Elles sont occasionnelles et sont à éviter en dimensionnant bien les mémoires.

- les oscillations de faible amplitude même en l'absence de signal à l'entrée. Elles sont dues aux troncatures et à la récursivité et ont généralement un spectre avec des raies localisées. Grâce à des procédures d'arrondi variées, on peut diminuer l'amplitude et raccourcir les transitoires de ces cycles, voire les supprimer.

3-2-2-4 Conclusion.

Nous allons maintenant examiner de nombreuses structures de filtres et les évaluer au moyen des notions développées précédemment. Les termes de la comparaison seront en effet, pour une même fonction objectif (voir "la définition d'un filtre"), la taille des mémoires, la sensibilité aux coefficients, le bruit de calcul, les cycles limites et le nombre d'opérateurs arithmétiques.

3-3- Les différentes structures de filtres numériques.

Pour étudier ces structures qui bien que traduisant exactement l'algorithme de filtrage donnent une idée de ce que pourrait être une réalisation câblée, nous avons abandonné la classification RIF, RII.

Nous en proposons une autre, plus proche de la notion de circuit : c'est une distinction entre chaînes de dipôles et chaîne de quadripôles. On remarque que les filtres synthétisés dans le plan p donnent des chaînes ou des pavages de quadripôles, et que les filtres synthétisés dans le plan des Z se rapprochent plus de chaînes de dipôles bien que l'on trouve des structures encore différentes. Les filtres en treillis constituent une exception à cette règle.

3-3-1 Les filtres en chaînes de quadripôles.

Ce sont des copies, en numérique, de filtres analogiques. Ces filtres présentent donc quatre voies d'interconnexions associées deux par deux et sur lesquelles circulent une onde incidente et une onde réfléchie.

Les extrêmités de la chaîne sont souvent dégénérées et une connexion est soit laissée en l'air, soit reliée à l'autre.

On peut également remarquer que ces filtres réalisent parfois en même temps la fonction de filtrage et son inverse suivant la connexion qui est choisie comme entrée.

3-3-1-1 Les filtres en treillis.

Ces filtres sont de type RIF (Fig. 3.4) ou de type purement RII (récursifs) (Fig. 3.5), ceci illustre ce que nous disions précédemment. Cette structure apparaît dans les études d'analyse et de synthèse de la parole, pour la simulation du conduit vocal dont elle reconstitue la forme. Les coefficients k multiplicatifs peuvent en effet simuler les différentes sections de passage de l'air.

Les filtres en treillis sont parfois associés à un processus adaptatif.

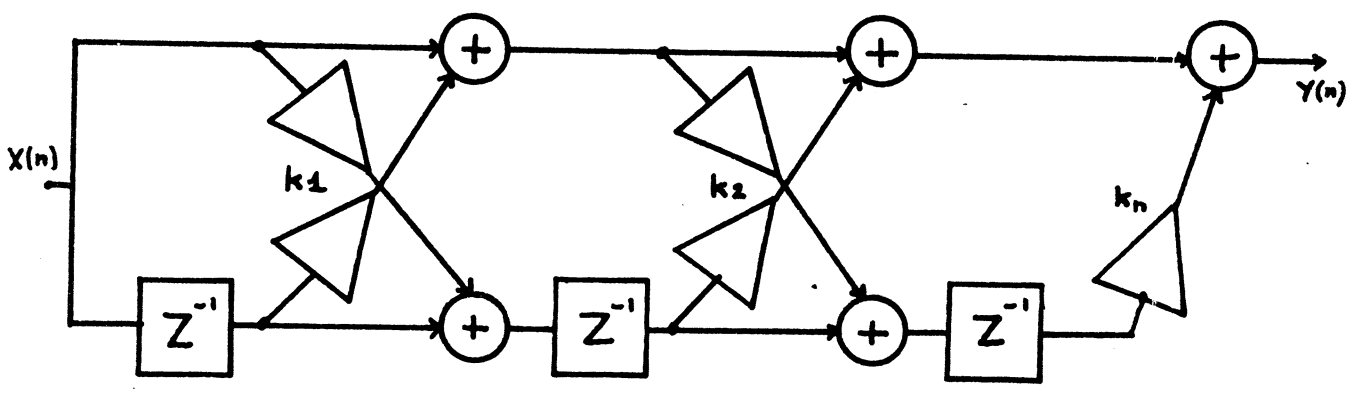


Fig. 3.4

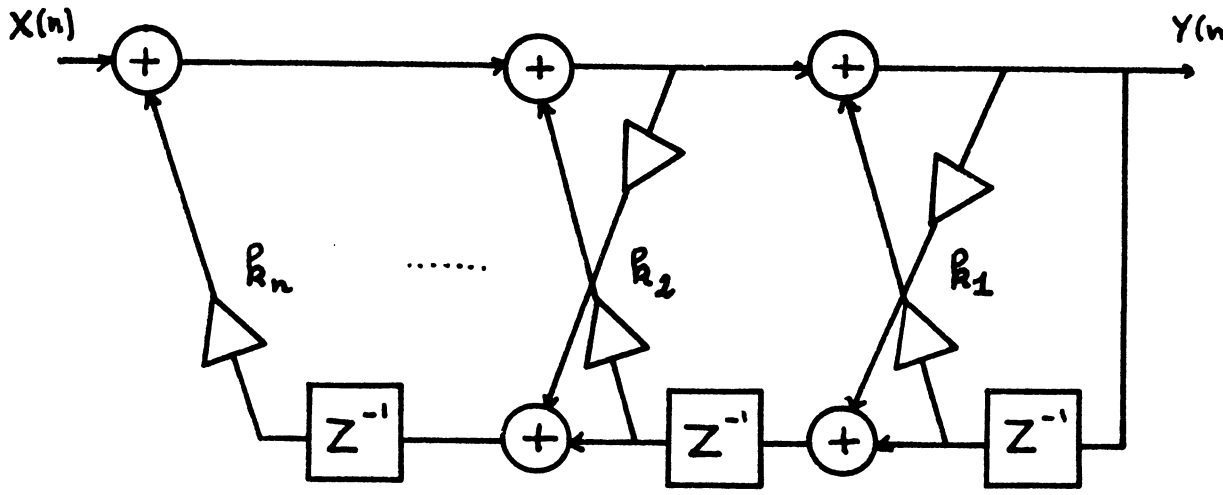


Fig. 3.5

3-1-3-1-2 Les filtres en échelle simulée.

Dans ce type de filtre, la fréquence d'échantillonnage doit être grande devant la bande passante pour minimiser les effets des terminaisons. Les coefficients multiplicatifs sont calculés directement à partir des éléments du circuit analogique et peuvent être représentés sur seulement quelques bits.

En fait, cette structure est surtout utilisée dans les dispositifs à commutation de capacités (Fig. 3.6).

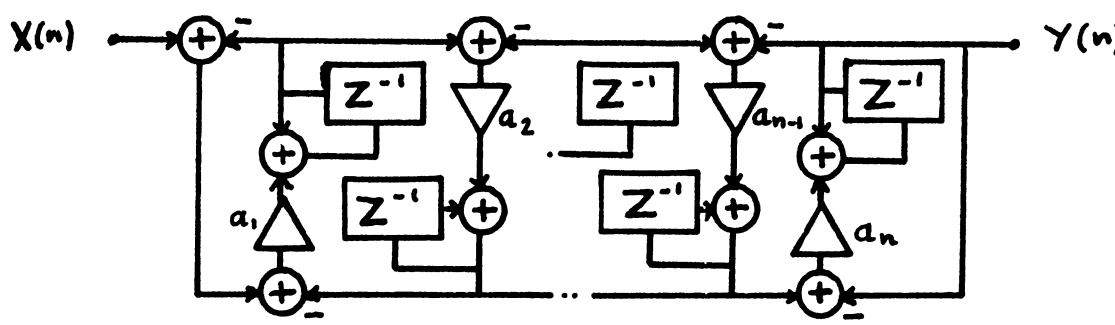


Fig. 3.6

3-3-1-3 Les filtres d'onde numériques.

Ce sont des structures très employées car elles permettent de transformer tout filtre analogique en chaîne, en un filtre numérique. Les éléments du filtre sont reliés entre eux soit en série, soit en parallèle, par des adaptateurs "série" ou "parallèle" (Fig. 3.7 à 3.13).

On trouve principalement trois sortes de filtres d'ondes :

- les cascades d'éléments unitaires
- les filtres en échelle avec insertion d'éléments unitaires
- les "lattice filters".

Nous trouverons plus loin différents schémas de structures existantes.

Il est assez difficile de porter un jugement sur ces filtres car peu d'utilisateurs se servent de plusieurs structures et peuvent donc avoir des avis comparatifs.

On peut néanmoins discerner les avantages suivants :

- un nombre de bits réduit pour faire les calculs,
- un bon rapport signal sur bruit par rapport à d'autres structures,
- une bonne stabilité en matière de cycles limites.

Mais les inconvénients ne sont pas moins affirmés :

- beaucoup d'additions et de multiplications,
- structures parfois complexes sauf pour les cascades d'éléments unitaires,
- réalisation difficile.

Les rares études comparatives existantes se montrent assez sévères pour les filtres d'ondes, mais prennent en compte des critères trop grossiers, (nombre d'opérations sans tenir compte du nombre de bits...) ou bien correspondant à des réalisations sur circuits programmables à nombre de bits fixes.

Dans certains cas de filtrage, les filtres d'ondes s'avèrent être malgré cela les plus performants.

Nous pensons donc que, dans l'hypothèse de circuits personnalisés, ces filtres présentent un réel intérêt, et ne doivent pas être négligés.

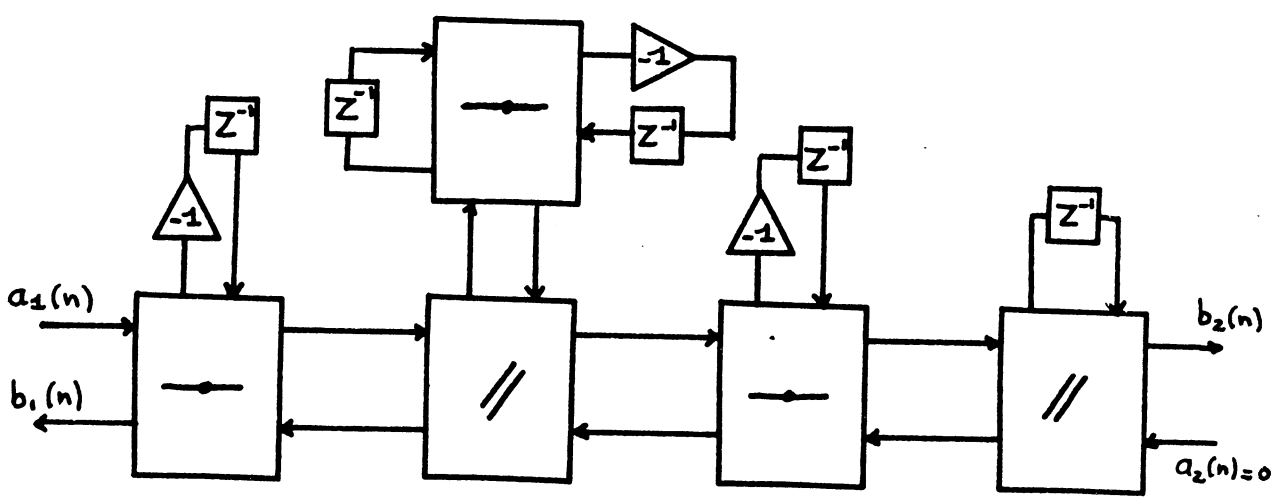
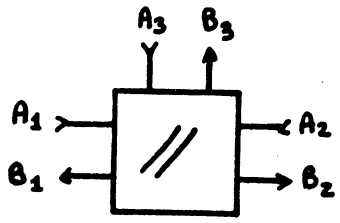


Fig. 3.7



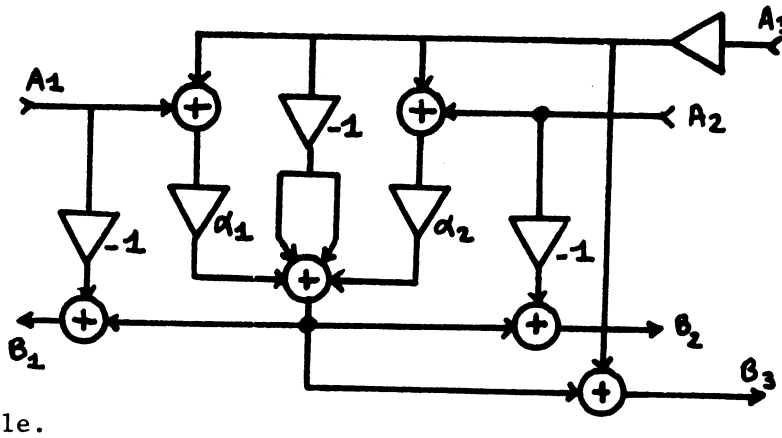
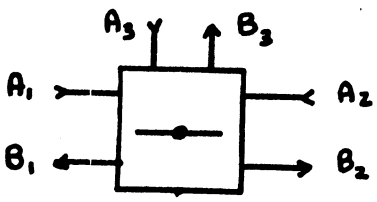


FIG 3.8 Symbole et shéma de l'adapteur parallèle.



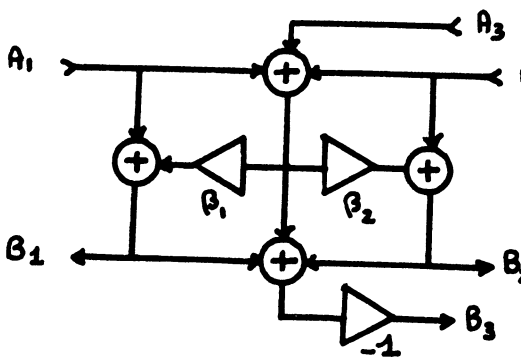


FIG 3.9 Symbole et shéma de l'adapteur série.

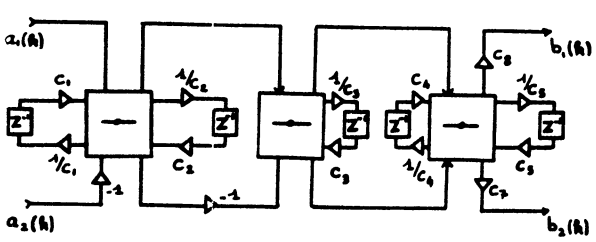


FIG 3.10

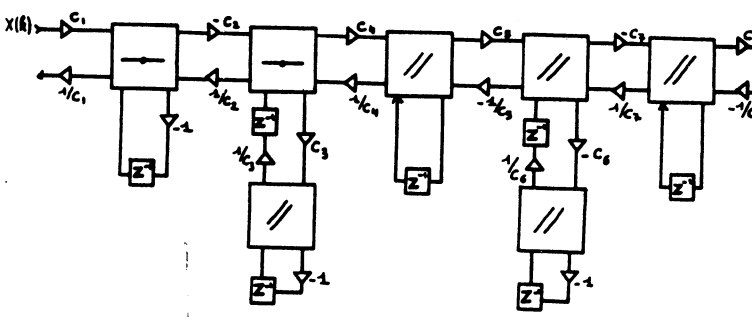


FIG 3.11

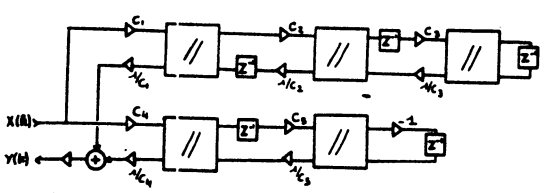


FIG 3.12

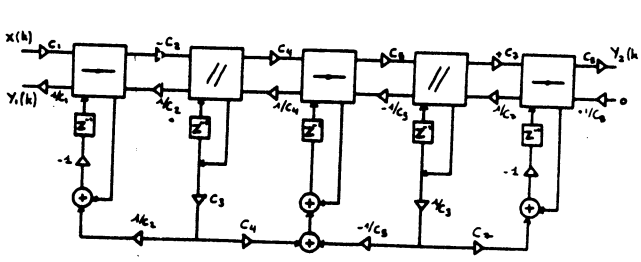


FIG 3.13

3-3-1-4 Conclusion.

Contrairement aux structures de filtres que nous allons examiner plus loin, les filtres à chaînes de quadripôles ont des topologies extrêmement variées qui peuvent devenir très complexes. Un langage de description de tels filtres devrait donc se rapprocher davantage d'un langage graphique que d'un langage équationnel. On peut certainement décrire tout filtre d'onde sous la forme d'un pavage d'adaptateurs série ou parallèle auxquels seraient connectés des opérateurs en nombre limité : retards, multiplications, additions...

3-3-2 Les filtres de type RIF.

Du fait de l'expression simple de la sortie de ces filtres en fonction de leur entrée, les structures employées sont peu nombreuses(Fig. 3.14 et 3.15). On trouve une structure directe et une structure transposée qui sont classiques, et quelques structures particulières surtout suggérées par des contraintes sur le matériel lorsqu'on réalise ces filtres avec des boîtiers du commerce (Fig. 3.17).

On peut aussi remarquer que la symétrie ou l'antisymétrie des coefficients, dans les filtres RIF à phase linéaire, permet une économie substancielle en opérateurs (Fig. 3.16).

Le nombre des coefficients est en général beaucoup plus élevé que pour un filtre RII, ce qui impose de choisir la structure soigneusement sous peine d'avoir une réalisation matérielle très lourde.

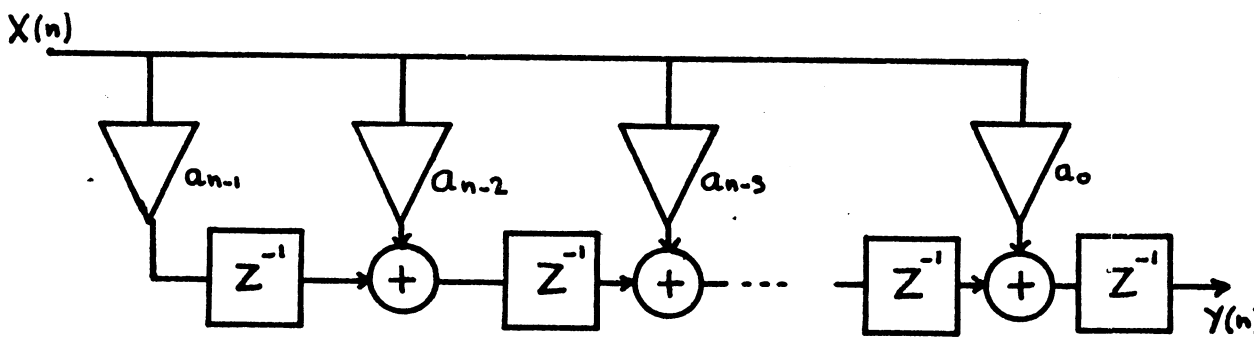


Fig. 3.14

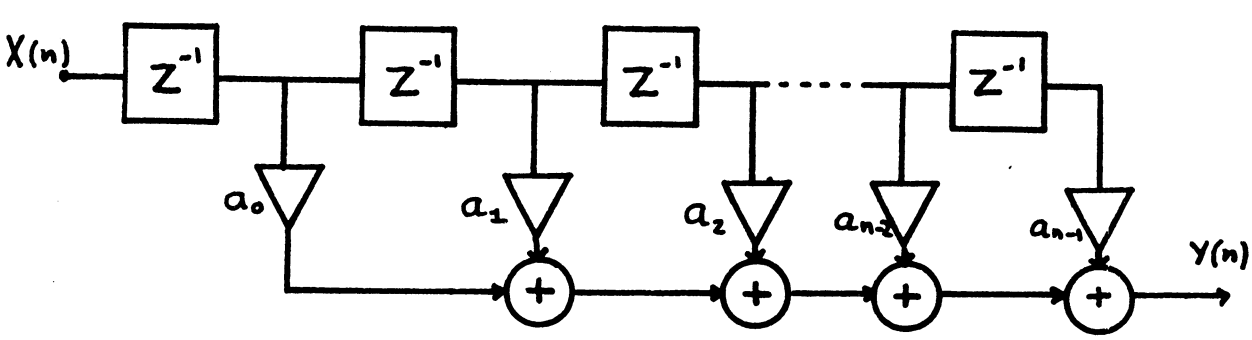


Fig. 3.15

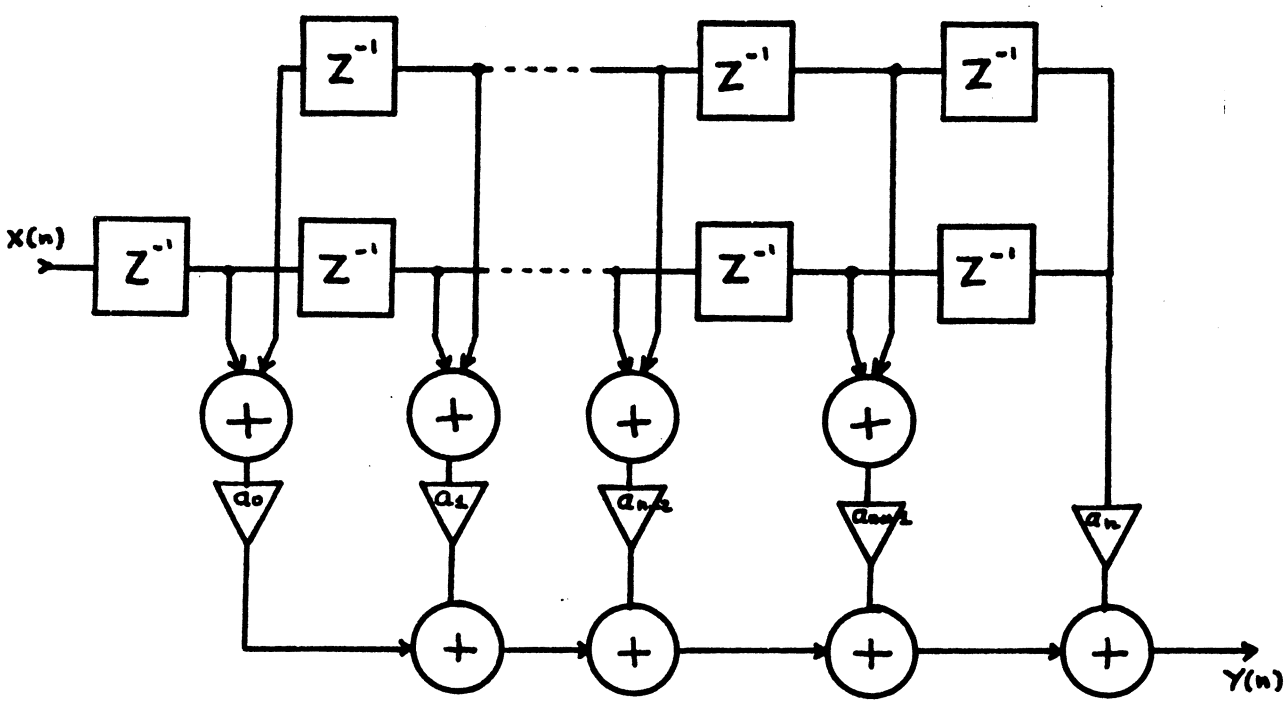


Fig. 3.16

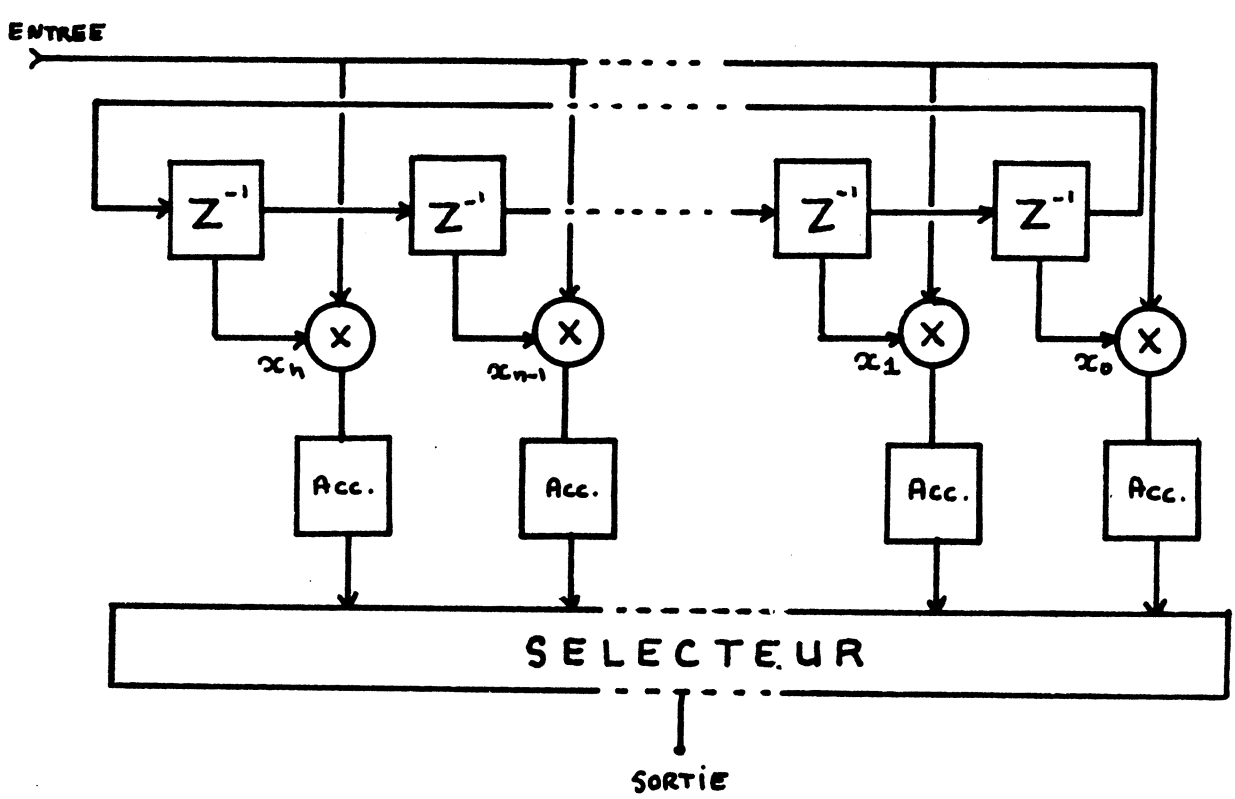


Fig. 3.17

3-3-3 Les filtres de type RII.

Ces filtres sont de loin les plus utilisés, et ont donc donné lieu à des recherches de structures très poussées. Les structures classiques ont donné lieu à de nombreux perfectionnements parfois ponctuels sur ces structures. Malgré la grande diversité des structures, nous allons essayer de les classer par famille.

3-3-3-1 Les structures directes ND et DN.

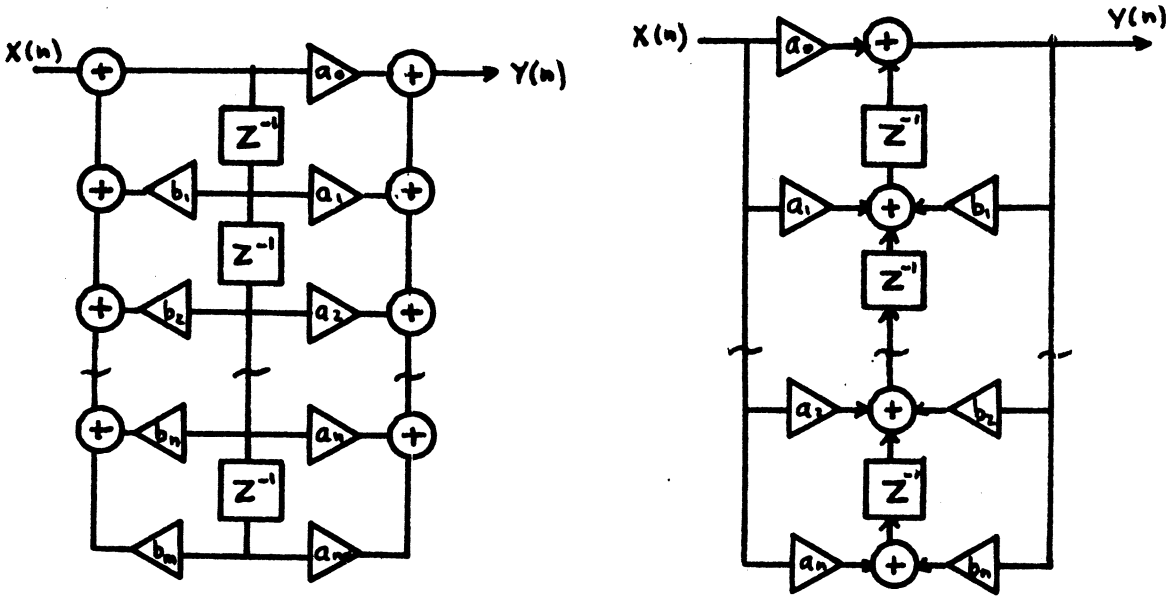


Fig. 3.18

Ces structures correspondent à une réalisation globale de la fonction de transfert en Z. Elles ne sont que rarement utilisées car bien qu'économiques en nombre d'opérations, il faut travailler sur beaucoup de bits si l'on veut un rapport S/B acceptable. Par ailleurs, la prévention des débordements de capacité n'y est pas évidente (Fig. 3.18).

Le mauvais rapport signal à bruit est dû au fait qu'aucun recadrage de la donnée n'est possible avant chaque multiplication ou chaque groupe de multiplications. De ce fait, à dynamique unique en sortie des multiplieurs, la valeur maximale de l'échantillon d'entrée sera fixée par le plus grand coefficient. En effet, si cette valeur est dépassée, il y aura débordement en sortie du multiplieur. Par contre, si un des coefficients est très petit, comme l'échantillon d'entrée n'utilise pas toute sa dynamique disponible (limitation supérieure), le résultat du produit sera repoussé vers les bits de poids faible et noyé dans le bruit.

Tous ces inconvénients sont levés par l'emploi de structures décomposées.

3-3-3-2 Les structures décomposées.

La fonction de transfert des filtres RII est une fraction rationnelle en Z qui peut donc être décomposée, soit en somme, soit en produit de fractions rationnelles en Z d'ordre 2 et d'ordre 1.

3-3-3-2-1 La structure parallèle.

Elle correspond à une décomposition de H(Z) en somme de fractions rationnelles qui sont calculées en parallèle et sommées à la sortie du filtre. Cette structure est déjà avantageuse par rapport aux formes directes car chaque cellule est isolée et les bruits de fond ne sont pas bouclés d'une cellule à l'autre. Mais elle est tout de même moins utilisée que la structure série que nous allons examiner à présent.

3-3-3-2-2 La structure série ou cascade.

La fonction H(Z) est décomposée en un produit de fractions rationnelles d'ordre deux de la forme (Fig. 3.19) :

$$Hi(Z) = \frac{ao + a1.Z^{-1} + a2.Z^{-2}}{1 + b1.Z^{-1} + b2.Z^{-2}} = \frac{Ni(Z)}{Di(Z)}$$

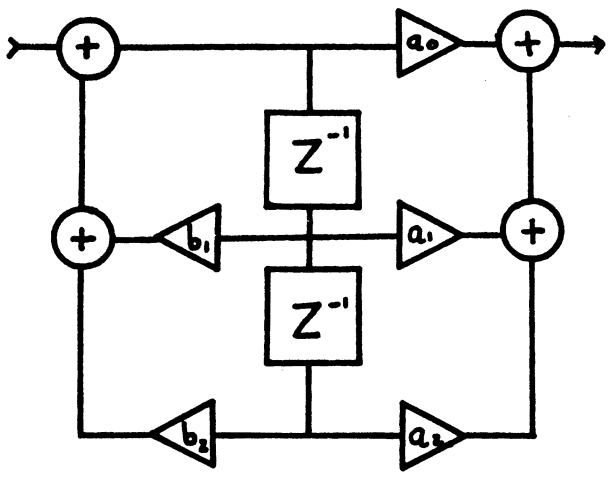


Fig. 3.19

Lors du calcul des fonctions Hi(Z), le concepteur peut jouer sur beaucoup de facteurs :

- l'appairage des zéros et des pôles a une grand influence sur le bruit. En effet, lors de la mise en cascade de N/2 cellules, le bruit a pour densité spectrale :

$$Bc = \frac{q^2}{12} \sum_{j=1}^{n/2} \int_0^1 \prod_{i=j}^{n/2} \left| \frac{Ni(f)}{Di(f)} \right|^2 df$$

L'appairage des zéros et des pôles vise à rendre $\left(\frac{Ni}{Di}\right)^2$ minimal.

- la détermination de l'ordre dans lequel seront rangées les cellules. L'expression de Bc montre que la cellule de tête n'interviendra qu'une fois dans le bruit. On met donc en tête les cellules les plus "bruyantes" qui sont souvent celles qui ont le plus fort gain.

- et enfin le facteur d'échelle associé à chaque cellule. Lors de la mise en série des cellules, on peut intercaller un multiplieur de recadrage afin d'utiliser au mieux la dynamique interne de la cellule suivante. C'est ce dispositif qui manquait dans la forme directe.

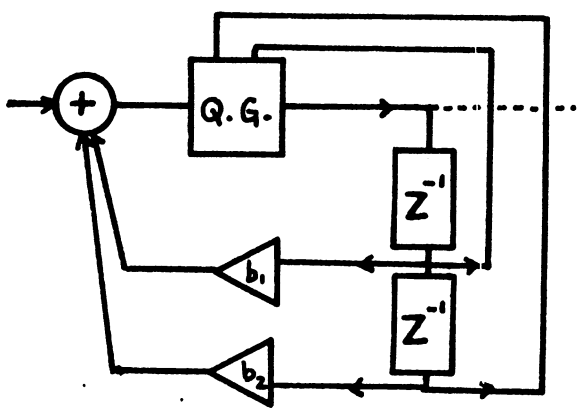
Après une utilisation optimale de ces trois techniques, le filtre obtenu est de loin supérieur à toutes les autres structures RII. Malgré cela, beaucoup d'améliorations ont encore été apportées au niveau de la cellule d'ordre deux, et qui s'ajoutent aux techniques citées ci-avant.

3-3-3-2-3 Améliorations de la cellule d'ordre 2.

Ces améliorations sont de deux types : le contrôle de la troncature en vue d'éliminer les cycles limites et de diminuer le bruit et les dispositifs divers de réduction de bruit.

* Contrôle de la troncature :

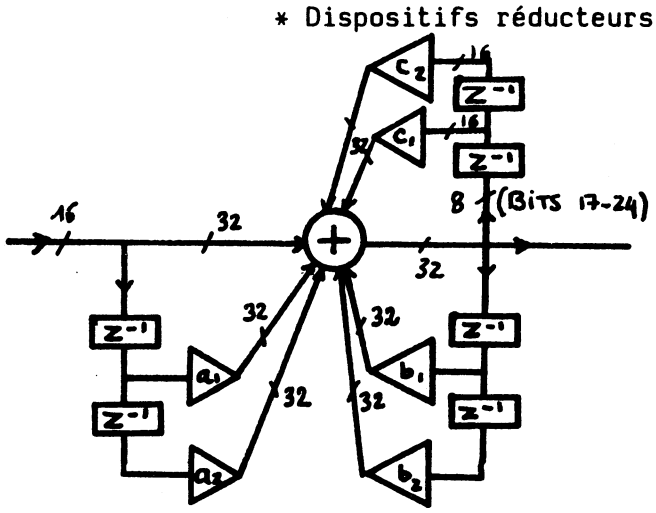
Dans une cellule d'ordre deux, on peut tronquer la sortie des multiplieurs de nombreuses manières. Ces manières ont une forte influence sur la durée et l'amplitude des cycles limites. Les dispositifs de troncature les plus sophistiqués à ce jour font appel à la manière dont ont été tronqués les deux échantillons précédents pour tronquer l'échantillon courant.



Le quantificateur généralisé est un circuit logique qui combine les bits les moins significatifs de Xn-1, Xn-2 et Yn pour générer celui de Xn.

Fig. 3.20 Schéma de principe du quantificateur généralisé

* Dispositifs réducteurs de bruit :



En établissant une fonction de transfert de bruit (sur les derniers bits), ces circuits permettent de diminuer le bruit global (Fig. 3.21).

ceci augmente la complexité du circuit, et doit être comparé à une simple augmentation de la longueur du mot interne qui aurait le même effet.

Fig. 3.21 Organisation du dispositif réducteur de bruit.

Ces dispositifs sont connus sous le dénominatif de ESS : Error Spectrum Shaping. Les coefficients multiplicatifs sont souvent des puissances de 1/2 pour être réalisés par décalage (Fig. 3.22).

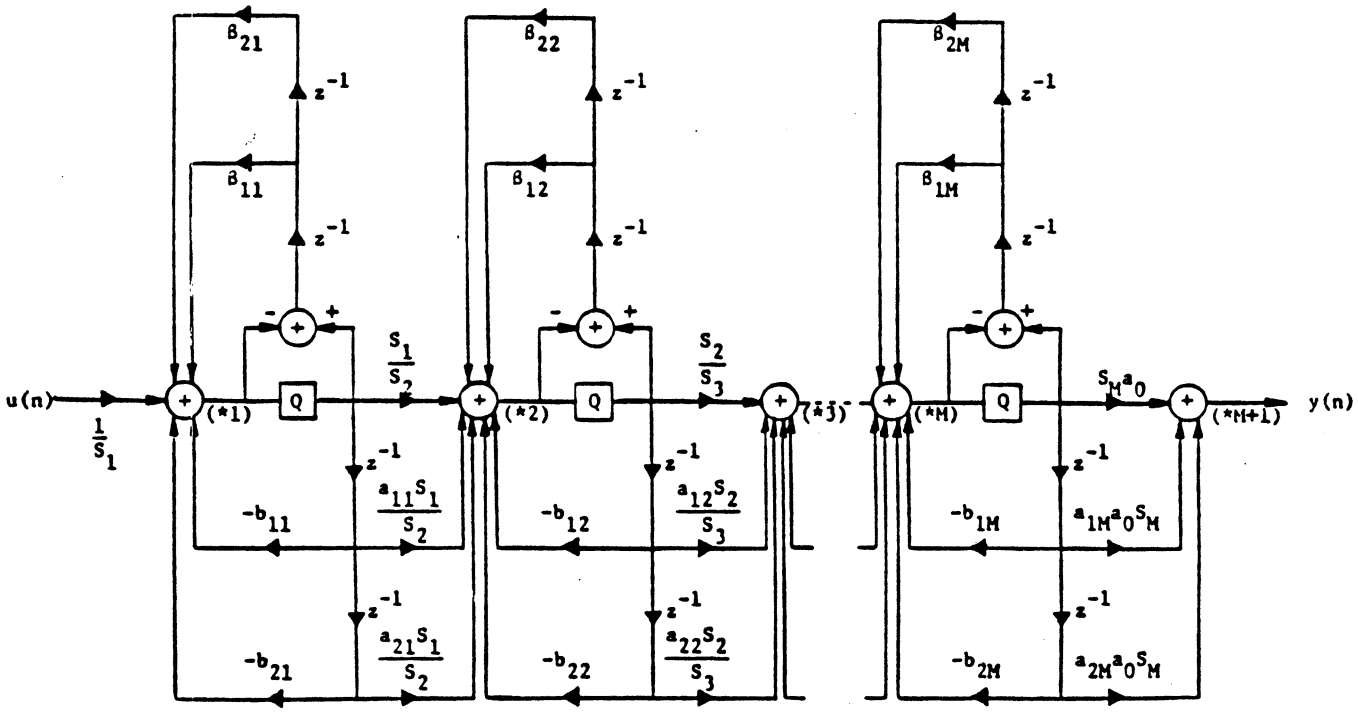


Fig. 3.22

Cascade de cellules d'ordre 2 munies d'un circuit ESS

L'efficacité de ces améliorations dépend souvent des caractéristiques de la fonction de filtrage désirée ; seul un programme de simulation ou d'analyse peut trancher quant à l'usage de tel ou tel dispositif.

3-3-3-2-4 Conclusion.

Les filtres RII que nous venons d'examiner sont parmi les plus classiques. La cascade optimisée de cellules de second ordre canoniques est très souvent employée. Nous allons à présent voir quelques autres structures de filtres qui sont le fruit de recherches récentes et dont l'emploi est loin d'être généralisé.

3-3-3-3 Les structures particulières.

Ces structures font souvent l'objet d'une publication qui en donne la formulation mathématique, le schéma, le calcul des coefficients, et une comparaison de performances avec celles de filtres classiques. La jeunesse de ces structures particulières incite à la prudence et à l'emploi d'outils de simulation.

Cette structure (Fig. 3.23) serait exempte de cycles limites quand l'entrée est constante ; et supérieure aux autres, dans certains cas, quant au bruit de fond...

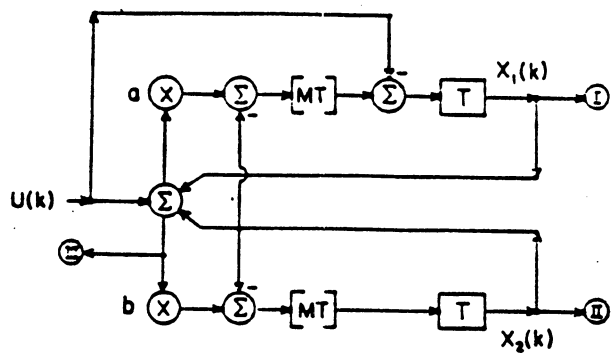


Fig. 3.23

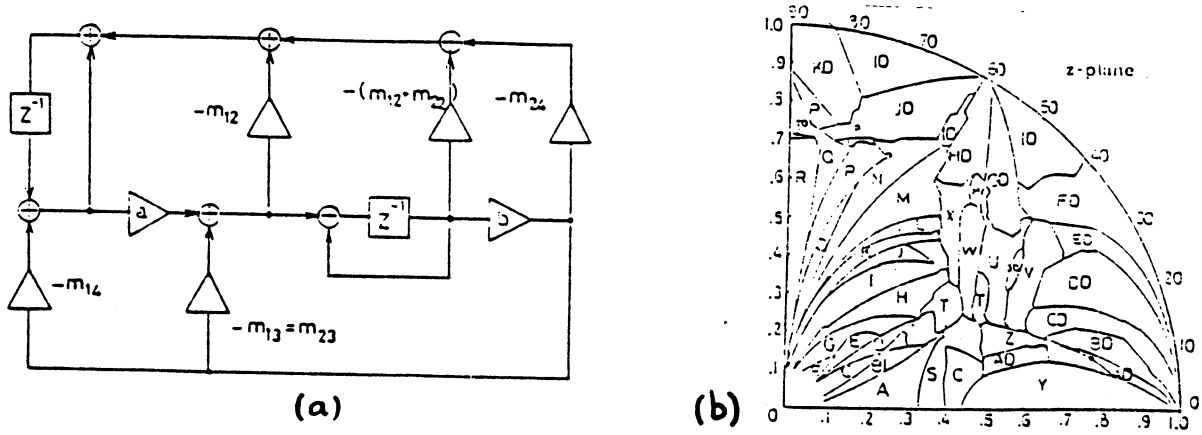
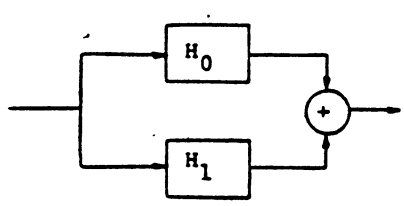


Fig. 3.24

Ce schéma (Fig. 3.24a) a pour but de diminuer la sensibilité aux pôles de la fonction. Il réalise cela en déplaçant légèrement la valeur du coefficient multiplicatif par addition, sur le schéma, de multiplieurs annexes câblés (puissance de 1/2).

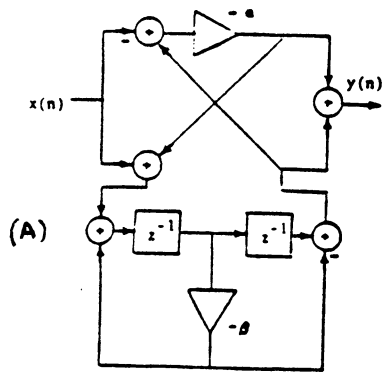
Suivant la localisation du pôle dans le plan Z (fig. 3.24b), les coefficients mij sont déterminés.

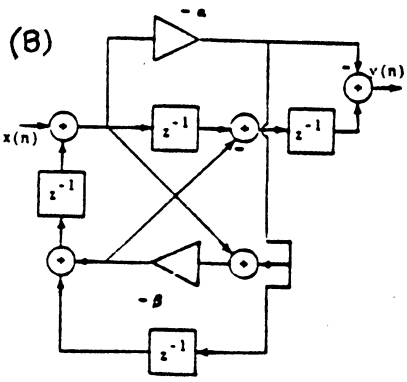
Ceci peut être utile lors de l'emploi de filtres très sélectifs quand le pôle est très près du cercle unité et que le nombre de bits du coefficient influe beaucoup sur la sensibilité du filtre à cet endroit.

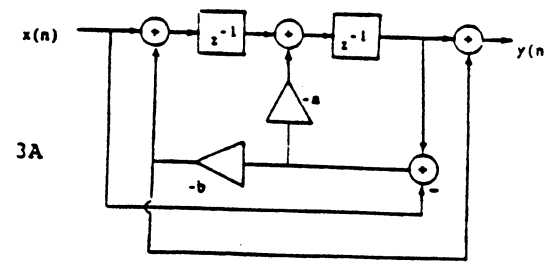


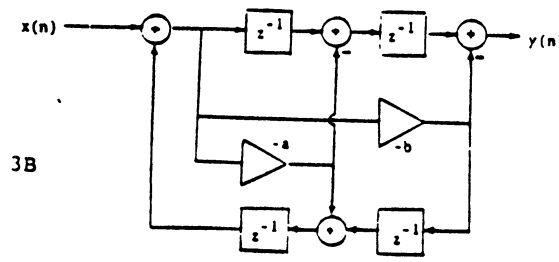
Enfin, des gens ont suggéré de réaliser des structures d'un ordre quelconque par la mise en parallèle de deux "All pass filters". Il semblerait que l'intérêt de cette structure réside dans le peu de calculs nécessaires, par rapport aux cascades de second ordre, pour la même fonction réalisée. Le bruit de fond semblerait aussi plus faible.

Nous donnons quelques exemples de ces schémas qui sont l'oeuvre de B.LIU, dont les travaux font souvent autorité dans ce domaine.









Ces structures (Fig. 3.25) sont des seconds ordres à associer A//B et qui ont chacune des caractéristiques de bruit précises.

Enfin, des perspectives intéressantes sont à espérer dans l'usage des structures sans multiplieurs :

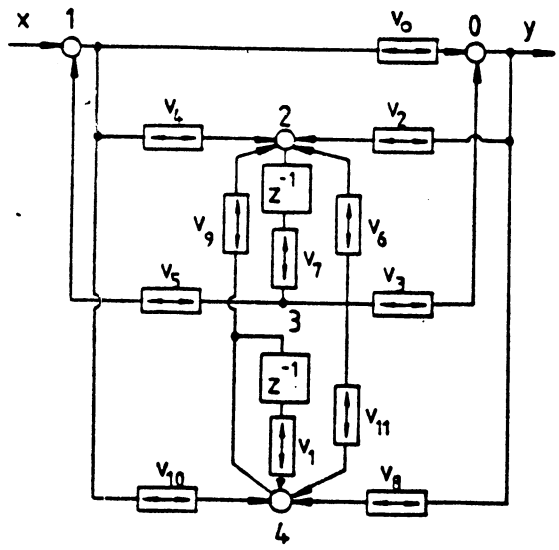


Fig. 3.26

Voici une cellule du 2ème ordre dont les multiplieurs ont été remplacés par plusieurs décalages.(Fig. 3.26)

Une réalisation câblée permettrait des vitesses de traitement considérables (vidéo).

Ceci revient à considérer des multiplieurs dont les coefficients ait un très petit nombre de 1 dans leur configuration binaire. La présence de ce 1 justifiant un décalage (X par 2 ou par 1/2).

Pour conclure quant aux structures particulières, nous dirons que seul un outil CAO de conception de filtres pourrait toutes les prendre en compte : en effet, le calcul de ces structures, puis la comparaison de leurs performances demandent des programmes très importants, et surtout intégrés dans un seul système pour en maximiser l'efficacité. Un concepteur isolé a en effet peu de chances de pouvoir appréhender toutes ces structures et les comparer.

Nous reparlerons plus loin d'une organisation possible d'un tel outil de CAO pour filtres numériques orienté "intégration à grande échelle" (VLSI).

3-3-3-4 Le formalisme et les structures à variables d'état.

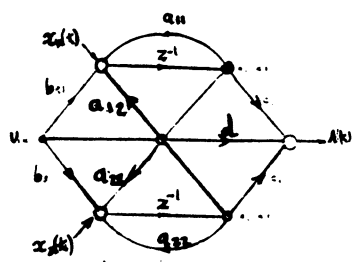
Ce formalisme, et les structures qui en découlent, représentent les filtres comme des automates d'états finis :

La formulation est la suivante :

$$x(k+1) = A\,x(k) + B\,u(k)$$
$$y(k) = C\,x(k) + D\,u(k)$$

avec u(k) = entrée
y(k) = sortie
x(k) = contexte interne

Les opérations matricielles sont réalisées dans la structure suivante :



Le calcul de x(k) impose une forte augmentation du nombre d'opérations. En contrepartie, cette structure a une faible sensibilité aux coefficients et un faible bruit.

Beaucoup de structures recensées plus haut ont une réalisation via ce formalisme, et seul un outil de simulation permettra de trancher quant à la meilleure forme, classique ou à variable d'état.

3-3-3-5 Conclusion.

Nous venons de faire le tour des différentes "structures" de filtres existantes. Celles-ci sont la définition comportementale du filtre et sont à rapprocher de la notion d'algorithme.

La manière dont cette définition comportementale du filtre sera implémentée sur des opérateurs multiplexés ou non, sera la définition structurelle de l'architecture du filtre.

Il faut insister sur ces termes afin de ne pas laisser d'ambiguités : en matière de filtres, les termes suivants sont équivalents

- algorithme = définition comportementale de l'architecture du filtre

- définition structurelle de l'architecture = réalisation physique du filtre.

Afin de ne pas compliquer les explications, le mot "structure" a été employé dans tout ce qui précède à la place du mot "algorithme". Le lecteur doit à présent re-situer chaque terme.

3-3-4 Outils de C.A.O. de filtrage numérique.

Comme on a pu le voir, le choix d'un algorithme de filtrage est une opération délicate qui demande de nombreux calculs et analyses numériques.

Il apparaît donc qu'un outil de CAO VLSI de filtres numériques qui permettrait, à partir des spécifications (voir "la définition d'un filtre") d'obtenir les masques du circuit, devrait comporter les modules suivants :

1- Un module contenant les spécifications fonctionnelles.

2- Un module de définition comportementale de l'architecture.
A partir des spécifications, ce module choisirait un algorithme parmi tous ceux qu'il connaîtrait et définirait complètement l'architecture comportementale qui réaliserait ces spécifications.

3- Un module de définition structurelle de l'architecture.
A partir de l'architecture comportementale définie au niveau le plus fin. Ce module génèrerait un schéma en terme de blocs d'opérateurs multiplexés ou non, de lignes d'interconnexions, de séquencement détaillé.

4- Un module de génération de masques.
A partir de la définition structurelle, et en puisant dans une bibliothèque de cellules, ce module génèrerait le plan de masse et les masques.

Bien évidemment, entre les modules 1, 2 et 3, de nombreuses boucles d'optimisation peuvent exister pour arriver à tenir les spécifications sur une surface de silicium raisonnable et à une consommation réaliste.

On trouvera un schéma détaillé de cet outil en page suivante.(Fig. 3.27)

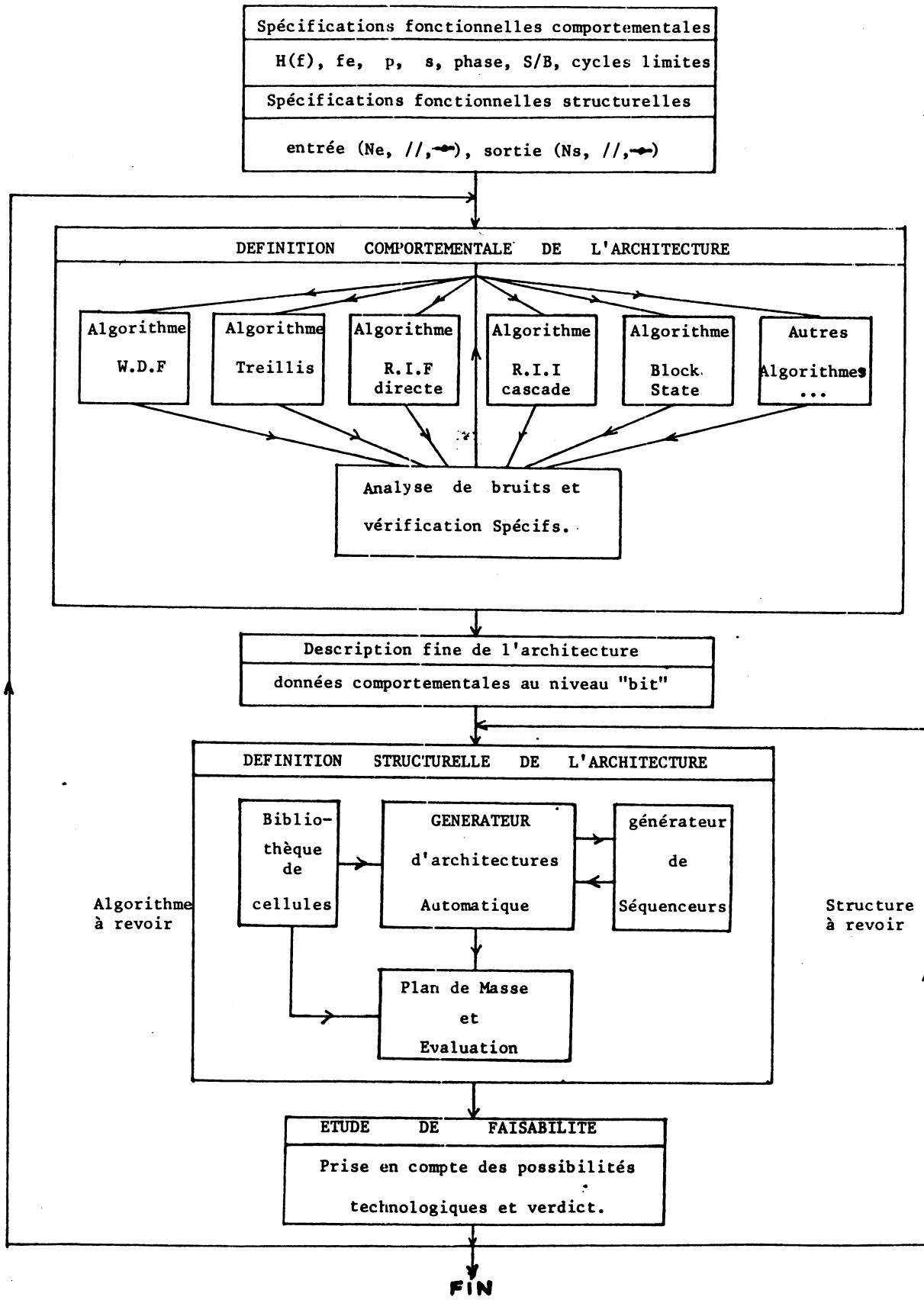

Spécifications fonctionnelles comportementales
H(f), fe, p, s, phase, S/B, cycles limites
Spécifications fonctionnelles structurelles
entrée (Ne, //,), sortie (Ns, //,)
DEFINITION COMPORTEMENTALE DE L'ARCHITECTURE
Algorithme W.D.F
Algorithme Treillis
Algorithme R.I.F directe
Algorithme R.I.I cascade
Algorithme Block State
Autres Algorithmes ...
Analyse de bruits et vérification Spécifs.
Description fine de l'architecture
données comportementales au niveau "bit"
DEFINITION STRUCTURELLE DE L'ARCHITECTURE
Biblio-thèque de cellules
GENERATEUR d'architectures Automatique
générateur de Séquenceurs
Algorithme à revoir
Structure à revoir
Plan de Masse et Evaluation
ETUDE DE FAISABILITE
Prise en compte des possibilités technologiques et verdict.
FIN

Fig. 3.27

3-4- Exemples de circuits existants spécialisés dans le filtrage.

Il est assez difficile de réunir sur ce sujet une documentation représentative : en effet, les études complètes aboutissant à une puce sont souvent menées en interne chez les fabricants et ne reçoivent aucune publicité.

Pour les architectures peu classiques, on trouve quelques publications souvent assez laconiques quant à leur réalisation physique. On trouve beaucoup de publications d'origine universitaire qui décrivent des réalisations à base de boîtiers du commerce et dont on peut s'inspirer pour des implantations sur silicium. On trouve enfin très peu de circuits monolithiques, et leur description se limite en général à des données très superficielles et aux performances, voire à un simple mode d'emploi du circuit.

3-4-1 Circuits intégrés spécialisés dans le filtrage.

3-4-1-1 An NMOS digital filter circuit (cf. bibliographie - B3.28)

Il s'agit d'une cellule du second ordre canonique réalisée sur une puce par les Télécommunications Britanniques.

Les données sont sur 16 bits série en complément à 2 et arrivent à une fréquence de 8 KHz.

La puce contient 4 multiplieurs 16 bits série - 13 bits parallèle, et a une surface de 25 mm^2 pour 8 000 transistors.

Ce circuit ne possède aucune partie multiplexée et n'est pas à proprement parlé un VLSI. Mais l'idée est originale et c'est un bon exemple de circuit à la demande pour les Télécommunications.

3-4-1-2 Une cellule pour filtres RIF (Cf. bibliographie - B3.27)

Cet article est une description succinte du circuit TDC 1028 de TRW. Ce circuit est une cellule de filtrage RIF.

Cette cellule est un filtre d'ordre 8, avec des données sur 4 bits et des coefficients sur 4 bits. Pour augmenter l'ordre ou la longueur des mots, il suffit de mettre en série et en parallèle plusieurs boîtiers TDC 1028. On peut aller jusqu'à un ordre de 36 sans problèmes de débordements.

L'architecture de cette cellule est classique et n'appelle pas de commentaires.

3-4-1-3 Cellule de filtrage du COFIDEC réalisé au CNET.

C'est un filtre récursif d'ordre 6 réalisé autour d'une ROM de produits pré-calculés. Des particularités des nombres à traiter permettent de diminuer sensiblement la taille de celle-ci. L'algorithme utilisé semble être celui des structures directes ND ou DN, ce qui impose une longueur de mots importante.(B3.26)

3-4-2 Circuits réalisés par des boîtiers standard.

L'existence de multiplieurs rapides en boîtiers commerciaux a favorisé la réalisation, spécialement dans les milieux universitaires, de nombreux circuits de filtrage. Les algorithmes choisis sont souvent classiques, d'ordre peu élevé, et les circuits réalisés ont rarement des originalités très marquées.

Les circuits utilisant des ROM de produits précalculés sont aussi très en faveur dans les situations où la rapidité est demandée : filtrage vidéo par exemple. Mais ce ne sont souvent que des simples retranscriptions de l'algorithme utilisé avec quelques astuces pour diminuer la taille des mémoires.

En marge de l'arithmétique distribuée, on trouve aussi des partisans du "Residue Number System" (RNS) dont l'emploi est resté très controversé et qui trouvera peut-être un souffle nouveau grâce à des outils de CAO VLSI pour filtres qui permettent des circuits très importants sur une seule puce.

3-4-3 Conclusion.

Le peu de réalisations spécialisées dans le filtrage face au grand nombre de circuits programmables de traitement du signal met en évidence la nécessité d'un outil de conception automatique de circuits spécialisés. En effet, comme le montrera le paragraphe suivant, les circuits programmables ne peuvent prendre en charge que du filtrage bas de gamme, et pour encore longtemps.

Le coût et la durée des études de conception freinent le développement des circuits spécialisés à la demande, et c'est pourquoi un outil de CAO-VLSI réduisant cette durée dans un rapport de 10 à 1 ou plus relancerait l'utilisation de ces circuits qui sont indispensables pour le filtrage de moyenne et haute gamme.

Enfin rappelons que le filtrage numérique occupe une place très importante parmi les techniques de traitement du signal.

Il figure en tant que "outil de base" dans la plupart des systèmes numériques de traitement de données ; et met en oeuvre des techniques de sur ou de sous-échantillonnage, et de limitation de bande passante.

En transmission de données, un filtre adapté au signal traité permet d'extraire ce signal du bruit de la ligne (filtrage adapté).

En traitement d'images, le filtrage sert à améliorer la qualité d'une image en faisant par exemple du lissage.

Citons enfin dans le domaine du traitement de la parole, le vocodeur à canaux qui produit du signal de parole à partir d'un bruit blanc et d'un banc de filtres.

Les algorithmes de filtrages sont donc des outils indispensables dans tous les systèmes numériques de traitement de données. C'est pourquoi nous nous y sommes plus particulièrement intéressés dans cette étude. En effet, sur beaucoup de circuits intégrés complexes, il se trouve une petite partie qui réalise un algorithme de filtrage ; et notre but est d'en faciliter la conception.

3-5- Limitation des processeurs de traitement du signal vis à vis du filtrage numérique.

Le développement considérable des circuits programmables de traitement du signal (Bell Labs, NEC, CNET, Philips, TEXAS...) a conduit les chercheurs à implanter sur ces machines des algorithmes de filtrage numérique.

La limite de l'utilisation de ces processeurs dans ce cas est immédiatement apparue sous la forme d'un chiffre sans appel : 2,5µs ou plus par cellule du second ordre canonique. Ce seul chiffre interdit à ces machines le filtrage de moyenne et haute gamme. Mais plusieurs autres insuffisances se dégagent d'une analyse plus approfondie.

3-5-1 Nature séquentielle des processeurs actuels.

On ne peut pas avoir de parallélisme important, tant au niveau des cellules que dans ces cellules ; donc toute augmentation de l'ordre du filtre se fait au dépend de la fréquence d'échantillonnage.

3-5-2 Nature figée de l'arithmétique.

Si la double précision est parfois possible sur les processeurs, la réduction du nombre de bits ne l'est jamais. Ainsi, le processeur trainera une structure trop lourde et lente dans certains cas où peu de bits auraient suffi.

3-5-3 Capacité mémoire limitée.

Dans certains cas de filtres RIF (annuleur d'écho acoustique) il faut beaucoup de coefficients (jusqu'à 1 000) pour avoir un filtrage efficace. Dans ce cas, la RAM interne d'un processeur sera souvent insuffisante.

En conclusion, nous pouvons dire que si ces processeurs de traitement du signal sont des outils très puissants pour le filtrage bas de gamme, ils sont irrémédiablement limités à des applications "Basse Fréquence".

Le filtrage haut de gamme va nécessiter des circuits spécialisés "collant à l'application", et utilisant des techniques qui pourront être différentes, comme l'arithmétique distribuée ou le RNS.

4 - DEFINITION D'UN LANGAGE DE DESCRIPTION POUR ALGORITHMES DE FILTRAGE

4-1- Introduction

4-2- Etude de différents langages de description d'algorithmes de traitement du signal.

4-2-1 Le langage SIGNAL
4-2-2 Le langage de SIRENA
4-2-3 Les langages graphiques
4-2-4 Conclusion

4-3- Description du langage retenu

4-3-1 Introduction
4-3-2 Un langage de description de filtres sous CASSIOPEE

4-3-2-1 les facilités offertes par CASSIOPEE
4-3-2-2 la structure de données
4-3-2-3 transformation de la description CASSIOPEE
4-3-2-4 description du programme TRADUCASS
4-3-2-5 description du programme DATAQUEST

4-4- Conclusion

4-1- Introduction

Comme nous venons de le voir, les algorithmes de filtrage numérique sont obtenus grâce à des programmes spécialisés. La description de ces algorithmes est donc contenue dans les sorties de ces programmes, mais elle peut également provenir d'autres sources telles que des schémas ou des ensembles d'équations.

Il importe donc de disposer d'un langage approprié à ces différentes descriptions ainsi que des traducteurs permettant de passer de toutes ces descriptions, au format d'entrée du générateur d'architectures.

Dans le cas où la saisie de l'algorithme est effectuée manuellement, le langage devra présenter une bonne accessibilité afin d'éviter l'apparition d'erreurs d'origine humaine.

Nous allons donc étudier différents langages existants et spécialisés dans la description d'algorithmes de traitement du signal. Mais il faut toutefois insister sur le fait que les algorithmes que nous cherchons à décrire présentent une caractéristique importante : la fonction de transfert du filtre ne dépend pas du rang de l'échantillon de signal entrant. Ce qui exclut donc la présence "d'aiguillages" dans le réseau d'opérations.

4-2- Etude des différents langages de description d'algorithmes de traitement du signal

4-2-1 Le langage SIGNAL (B4.1 à B4.4)

A l'heure ou nous commençions cette étude, aucun document officiel ne nous etait encore parvenu, décrivant ce langage. Seule une description manuscrite sommaire nous avait été fournie, et cela ne nous permettait que de cerner les grandes lignes de ce projet. Depuis, ce langage a évolué comme en témoignent les publications citées en bibliographie, mais notre choix pour un langage de description de filtres était arreté.

La classe d'algorithme visée par SIGNAL recouvre à la fois ceux qui sont de nature purement séquentielle, ceux qui traitent le signal par blocs et ceux à structure irrégulière ; et ceci par ordre de priorité. Les algorithmes de filtrage numérique dont nous nous occupons entrent dans la première catégorie.

SIGNAL est un langage du type "flot de données" où les fonctions opérant aux noeuds du réseau statique sont des blocs appelés "filtres". Le principal avantage de ce type de langage est de pouvoir explorer facilement le paralellisme d'un algorithme.

Cette représentation d'un algorithme sous la forme d'un graphe orienté est également celle que nous désirons avoir comme description d'entrée dans le générateur d'architecture.

Nous constatons donc que SIGNAL peut décrire les algorithmes qui nous intéressent dans une forme qui nous intéresse. Malheureusement, les informations dont nous disposions sur SIGNAL ne constituaient guère plus qu'une déclaration d'intention et ne nous permettaient pas encore de considérer ni d'utiliser SIGNAL comme langage d'entrée du générateur d'architecture.

En résumé, il n'y a pas, à priori d'incompatibilité entre SIGNAL, langage textuel, et le langage simplifié que nous avons développé. Ceci nous laisse espérer qu'une deuxième version de notre langage pourrait utiliser SIGNAL ou une restriction de SIGNAL en parallèle avec notre langage graphique.

4-2-2 Le langage de SIRENA (B4.5 à B4.7)

SIRENA est un outil logiciel spécialisé dans le domaine de l'automatique qui comprend un langage de description de systèmes rencontrés en automatique et traitement du signal. De nombreux outils de simulation et d'analyse sont couplés à ce langage et permettent l'étude complète d'un système. La représentation textuelle d'un bloc par sa fonction de transfert et la possibilité de construire des systèmes par assemblage de blocs nous incite à nous demander si nous pourrions utiliser ce langage pour décrire nos algorithmes et bénéficier ainsi de l'utilisation de tous les outils attenants.

Le langage de SIRENA est un langage équationnel où la fonction de transfert de chaque bloc est automatiquement calculée à l'aide des fonctions de transfert de chaque sous-bloc le constituant.

Ce langage conviendrait bien à une description de l'algorithme par ses équations, mais très mal pour une description par schéma. Il conviendrait à un concepteur qui désirerait créer un algorithme nouveau en manipulant des équations et en regardant l'effet produit sur la fonction de transfert résultante.

Malgré tout, ce genre de démarche reste assez marginal car les concepteurs préfèrent souvent faire appel à des schémas éprouvés et pour lesquels existent de puissants programmes de calcul.

Néanmoins, l'adaptation de SIRENA pourra être utile dans l'avenir, mais l'extraction de la description de l'algorithme semble difficile car SIRENA ne semble pas opérer sur une base de données ouverte vers l'extérieur.

4-2-3 Les langages graphiques

Les deux langages que nous venons d'examiner sont de forme textuelle. La description graphique d'un algorithme a aussi de nombreux avantages parmi lesquels une ergonomie appréciée des usagers : en effet, les algorithmes de filtrage sont souvent représentés par un dessin du réseau d'opérations. Dès lors que l'on dispose d'un éditeur graphique opérant sur une base de données, on peut envisager de construire des schémas de filtres en assemblant des symboles d'opérations. Le filtre ainsi construit sera donc vérifiable visuellement puisque le document de définition est très souvent un schéma accompagné d'une liste de coefficients.

Un des principaux inconvénients des langages graphiques peu puissants est la lourdeur de la description des motifs très répétitifs.

4-2-4 Conclusion

Tous les langages dont nous venons de parler sont susceptibles de répondre à notre attente, mais un problème de disponibilité nous a empêché d'utiliser les deux premiers. Nous avons donc construit un langage graphique pour l'instant rudimentaire mais ouvert à de nombreux perfectionnements. Ceci a été rendu nécessaire pour continuer à travailler sur le générateur d'architectures, mais nous pensons que l'adaptation de SIGNAL à notre problème est envisageable. Nous allons maintenant décrire le langage retenu.

4-3- Description du langage retenu

4-3-1 Introduction

Le langage que nous avons développé est principalement destiné à alimenter en exemples le générateur d'architectures. Il s'appuie sur CASSIOPEE dont il utilise l'éditeur graphique et la base de données. CASSIOPEE offre des facilités pour construire des circuits par assemblage de composants qui sont des occurences de types construits par ailleurs. Le langage graphique en question pourra aussi utiliser cette facilité mais un problème d'affectation de paramètres spécifiques à chaque composant nous a obligé dans un premier temps à construire nos filtres "à plat", c'est-à-dire sans blocs contenant eux-même des opérations. Nous allons décrire ce langage graphique et son utilisation.

4-3-2 Un langage de description de filtres sous CASSIOPEE

4-3-2-1 Les facilités offertes par CASSIOPEE (B4.8 à B4.10)

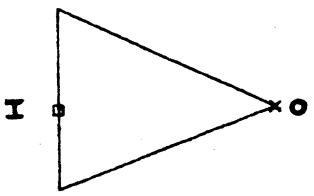
Nous avons utilisé l'éditeur graphique du niveau logique de CASSIOPEE, qui permet de construire des circuits par assemblage de composants qui sont de type prédéfinis (NAND, NOR...) ou construits (bascule...). Bien entendu, ces types étant inutiles pour construire des filtres ; nous avons dû redéfinir les types de base. Lors de l'étude des algorithmes de traitement du signal conduite précédemment, nous avons constaté qu'il suffisait de six symboles de base pour décrire "tout" type de filtre. Ces symboles sont :
multiplication, addition, registre, décalage, changement de signe et troncature.

Dès lors, en utilisant ces nouveaux types de base et en les plaçant dans un réseau d'interconnexions, nous pouvons construire tout type de filtre numérique dont nous avons alors une description dans la base de données de CASSIOPEE.

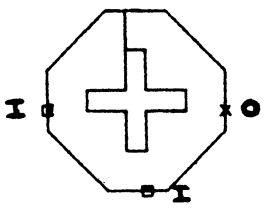
Nous donnons à la suite les symboles des opérateurs de base et des exemples de filtres décrits grâce à ce langage.

: Symbole de l'opérateur ENTREE
les entrées des opérateurs sont aussi repérées par ce signe.

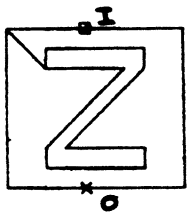
: Symbole de l'opérateur SORTIE
la sortie des opérateurs est aussi repérée par ce signe.



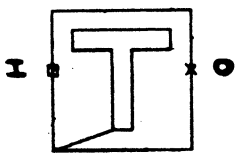
: Symbole de l'opérateur MULTIPLICATION
le coefficient est censé être stocké dans le multiplieur



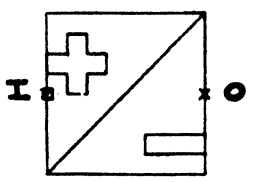
: Symbole de l'opérateur ADDITION



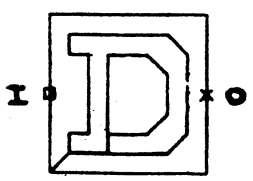
: Symbole de l'opérateur REGISTRE
Z rappelle la notation Z^{-1} des algorithmes



: Symbole de l'opérateur ARRONDI



: Symbole de l'opérateur CHANGEMENT DE SIGNE



: Symbole de l'opérateur DECALAGE

Voici à titre d'exemple une cellule de filtrage d'ordre 2 de type BIQUAD construite dans CASSIOPEE, (fig 4.1).

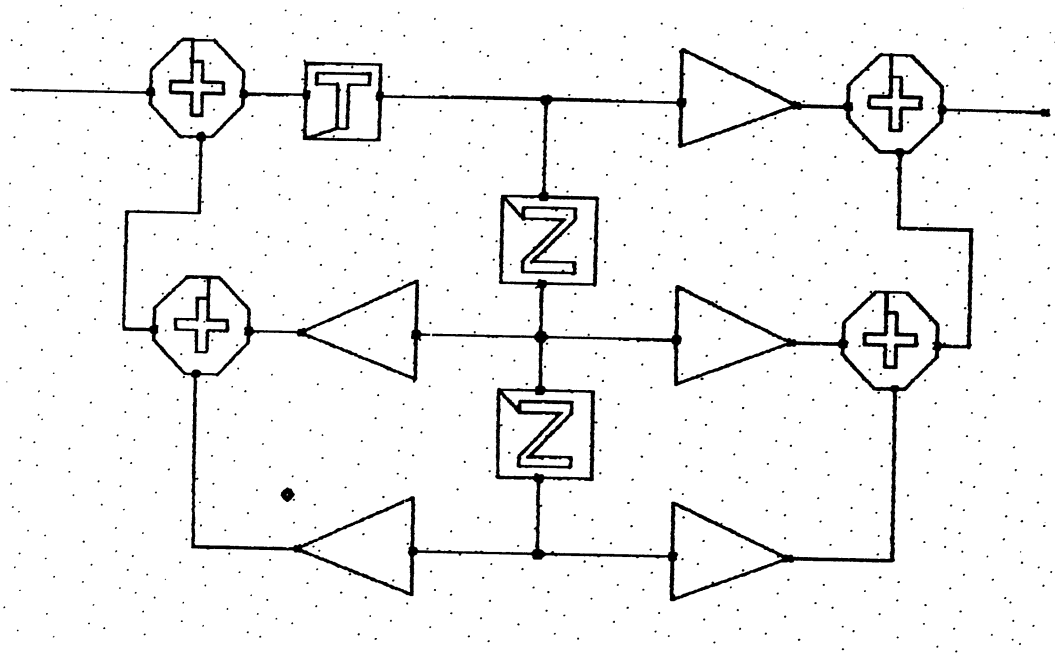


Fig : 4.1

et un circuit implementant un algorithme de type WDF (Wave Digital Filter). (fig 4.2).

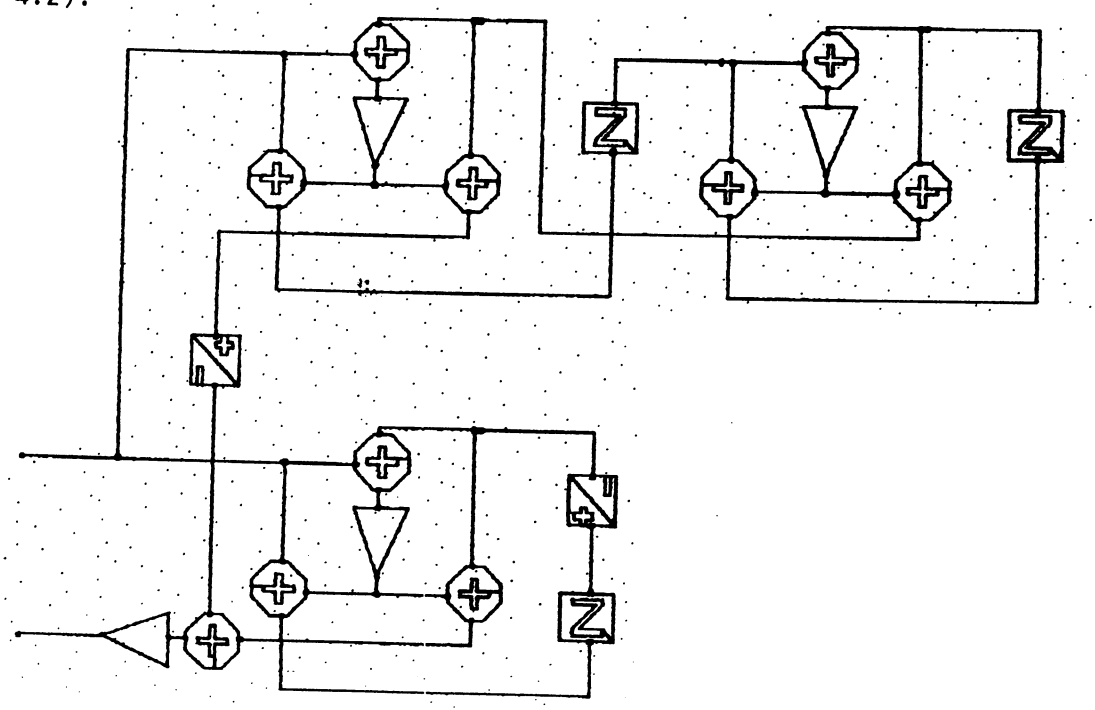


Fig : 4.2

Cette description est encore insuffisante pour décrire complétement un filtre : il manque en effet la longueur du mot binaire en tout point du circuit, la valeur des coefficients des multiplications, et des informations concernant tel ou tel opérateur. Il faut donc ajouter ces renseignements à la description CASSIOPEE.

Or, pour des raisons inhérentes à l'organisation des données dans CASSIOPEE, il est très incommode d'ajouter ces informations à l'intérieur même de la description. Nous allons donc effectuer dès maintenant la traduction dans une nouvelle structure de données qui se pretera bien aux algorithmes du générateur d'architecture.

Après avoir décrit brièvement la nouvelle structure de données, nous parlerons des programmes TRADUCASS et DATAQUEST qui effectuent cette traduction et saisissent les informations manquantes.

4-3-2-2 La structure de données

Le générateur d'architecture va manipuler le filtre d'origine décrit sous la forme d'un graphe Potentiel-tâches dans laquelle les tâches sont représentées par des sommets et les contraintes par des arcs.

Ce type de graphe se prête mieux à des modifications continuelles que le graphe Potentiel-Etapes utilisé par exemple dans la méthode PERT.

Nous allons donc devoir convertir la description CASSIOPEE du filtre en une description qui se prête bien aux manipulations que l'on doit faire sur un graphe. Afin de rester aussi près que possible de celui-ci, nous avons choisi de représenter une opération par un RECORD PASCAL contenant toute sa description, auquel on adjoint une liste d'antécédents et une liste de sucesseurs qui sont des listes de pointeurs sur les RECORD des opérations correspondantes. Pour faciliter l'accès à une opération donnée, une table de pointeurs permet l'accès direct à une opération quand on connait son numéro. On peut donc explorer le graphe rapidement en utilisant soit la table pour accès direct, soit les informations antécédent et successeur.

La description complète de la structure de données sera fournie au chapitre suivant, mais cet aperçu montre que nous avons une structure dynamique que nous pourrons modifier à volonté d'après les modifications imposées au graphe correspondant.

4-3-2-3 Transformation de la description CASSIOPEE

Un graphe Potentiel-tâches ne doit pas comporter de circuits internes pour être physiquement réalisable. Nous constatons donc que des algorithmes récursifs du type cellule BIQUAD posent un problème car une traduction directe en graphe ferait apparaitre des circuits.

Ces circuits n'ont aucune réalité physique car on s'aperçoit que chacun d'entre eux contient au moins un registre qui se charge d'effectuer la coupure du circuit : le filtre fonctionnant de manière synchrone, il convient d'examiner le déroulement des opérations sur une période d'horloge. Dès lors, les boucles n'existent plus car la sortie des registres est perçue comme une entrée du réseau, et l'entrée des registres comme une sortie du réseau.

Concrètement, la transformation de la description CASSIOPEE en description GRAPHE devra considérer les registres comme des opérateurs singuliers et leur appliquer le traitement suivant :

Chaque registre sera coupé en deux, sa sortie constituera une étape du graphe reliée - comme les entrées du circuit - à la tâche fictive "DEBUT" ; et son entrée constituera une étape du graphe reliée - comme les sorties du circuit - à la tâche fictive "FIN". Un lien nominal sera conservé entre les deux tâches crées à partir d'un même registre afin de ne pas perdre d'information de type circuit.

EXEMPLE 1 : Mécanisme général (fig 4.3).

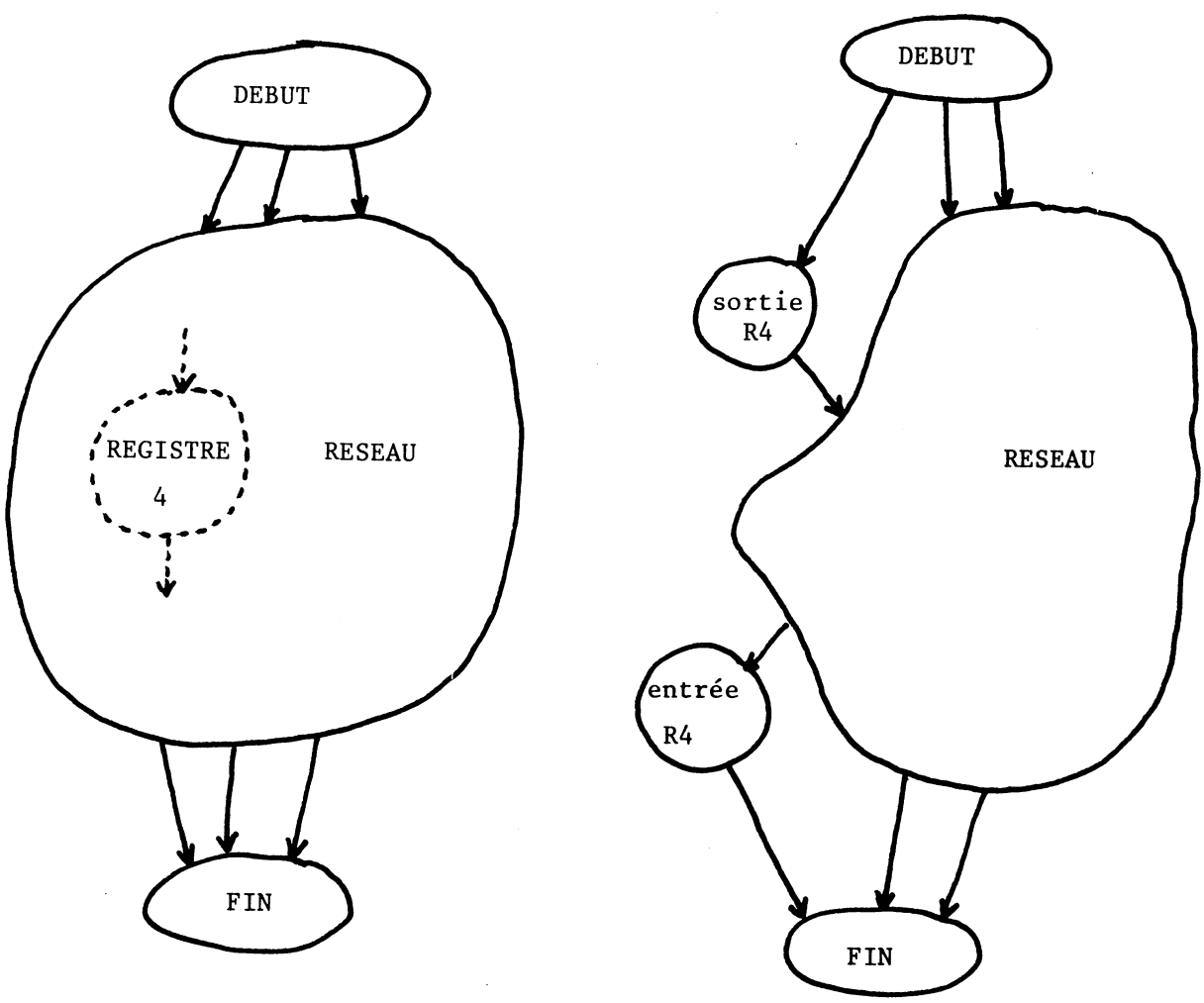


Fig 4.3

EXEMPLE 2 : La cellule BIQUAD (fig 4.4).

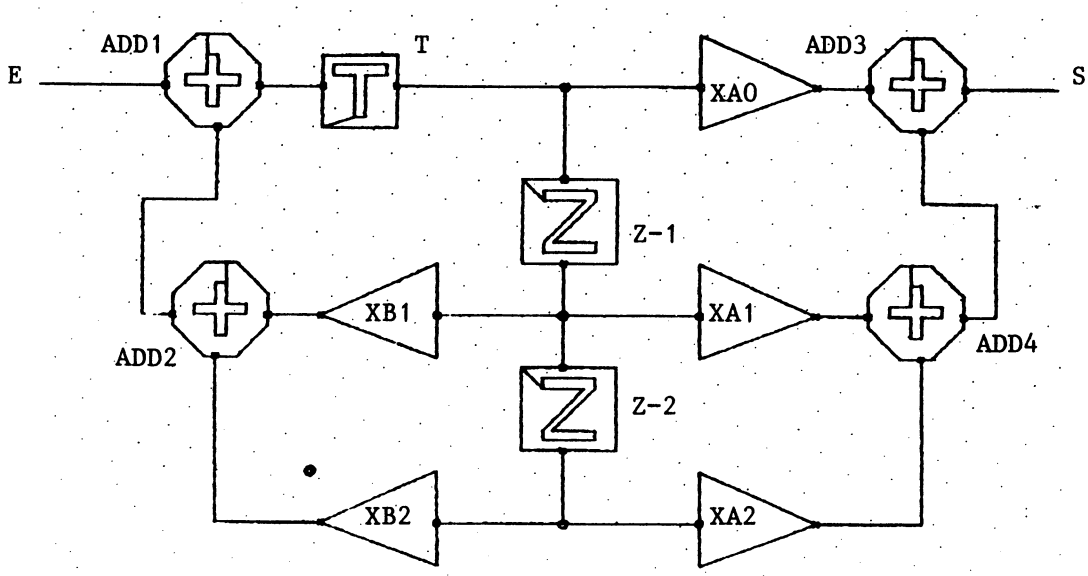


Fig : 4.4

Et le GRAPHE correspondant : (fig 4.5)

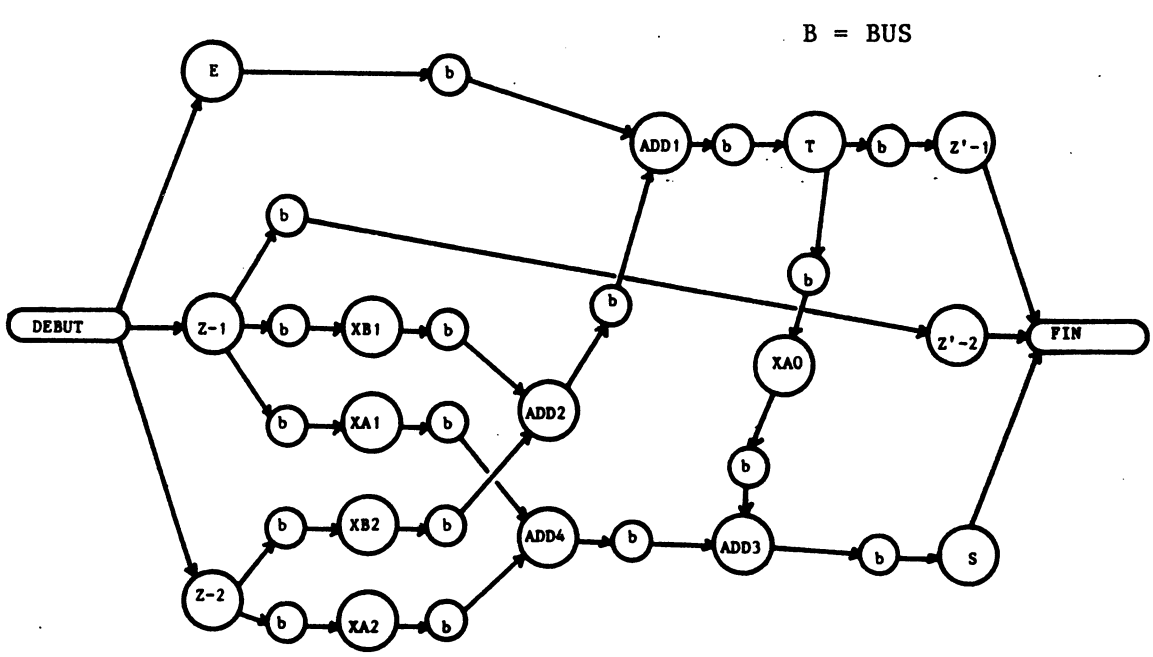


Fig : 4.5

4-3-2-4 Description du programme TRADUCASS

Ce programme est chargé de traduire une description CASSIOPEE en une description Potentiel-tâches. Il décripte donc les informations stockées dans la base de donnée de CASSIOPEE, applique aux registres le traitement décrit ci-dessus et crée le graphe correspondant.

De plus, TRADUCASS effectue un certain nombre de vérifications de cohérence et de circuiterie : il détecte les courts-circuits sur les entrées et les sorties de tout opérateur et il procède à une détection de boucles dans le graphe à titre de sécurité, bien que le cas soit très improbable.

Ce programme fournit donc un graphe comme décrit ci-dessus mais ne comportant pas encore de paramètres spécifiques à chaque opérateur, qui seront fournis par le programme DATAQUEST.

Le format du fichier MONFILTRE.TRA produit et la commande d'éxécution figurent en annexe n°1.

4-3-2-5 Description du programme DATAQUEST

Ce programme part du fichier MONFILTRE.TRA, et, pour chaque opération du filtre, effectue une saisie interactive des paramètres afférants. DATAQUEST vérifie alors la compatibilité des liaisons entre les opérations en ce qui concerne la taille du mot binaire. Les erreurs sont signalées à l'usager qui doit alors les corriger.

Dès que les vérifications sont terminées, DATAQUEST demande à l'usager de préciser la contrainte de temps qui limite la durée de fonctionnement du filtre.

DATAQUEST produit alors un fichier éditable nommé MONFILTRE.DAT qui servira d'entrée au programme CREATESOU, décrit au chapitre 7.1, afin que celui-ci y ajoute une liste d'opérateurs destinés à constituer l'architecture finale.

Les paramètres demandés par DATAQUEST pour chaque type d'opération sont explicités en annexe n°2 ainsi que le format du fichier et la commande d'éxécution.

4-4- Conclusion

Le langage que nous venons de décrire constitue une première version de ce que pourrait être un langage graphique de description de filtres. Il est suffisant pour alimenter en exemples le générateur automatique d'architectures et favoriser sa mise au point.

D'importantes modifications peuvent lui être apportées :

- Il n'utilise pas la possibilité de construction par assemblage de blocs préconstruits offerte par CASSIOPEE. On peut donc envisager d'améliorer TRADUCASS et DATAQUEST pour utiliser cette facilité qui serait utile pour les très gros filtres comportant des motifs répétitifs.

- Dès qu'un langage textuel sera disponible, il faudra essayer d'assurer la compatibilité des descriptions, au moins quant à la forme de sortie.

Nous considérons qu'il convient d'arreter à ce point, la première version du langage, quitte à la reprendre et à la développer au vu d'autres outils et de l'évolution de l'ensemble du projet.

5 - PRESENTATION DE LA BIBLIOTHEQUE D'OPERATEURS

5-1- Introduction

5-2- Contenu de la bibliothèque

5-3- Modes d'utilisation de la bibliothèque

5-3-1 Utilisation optimale
5-3-2 Utilisation actuelle
5-3-3 Nature des échanges avec l'extérieur

5-3-3-1 Echanges avec l'utilisateur humain
5-3-3-2 Echanges avec IMHOTEP

5-4- Présentation de la bibliothèque

5 - PRESENTATION DE LA BIBLIOTHEQUE D'OPERATEURS

Dès qu'il a défini le travail à réaliser, l'utilisateur du système doit préciser les moyens matériels qu'il compte mettre en oeuvre pour faire ce travail. Dans le cas qui nous intéresse, les moyens sont les différents opérateurs qui implémenteront les opérations constituant l'algorithme de départ.

Ces opérateurs sont caractérisés par leur fonction (additionneur, multiplieurs...) et par un certain nombre de paramètres comme la taille, la surface, la consommation, le temps d'opération... etc.

Dès que l'utilisation ou la manipulation par ordinateur de ces opérateurs est envisagée, il convient de créer une bibliothèque permettant de les stocker, de les manipuler, bref de les exploiter.

Nous allons donc présenter la bibliothèque d'opérateurs retenue, ainsi que son contenu.

5-1- Introduction

Un générateur d'architectures ne manipule que des informations d'assez haut niveau concernant les opérateurs qu'il utilise : ce sont des informations fonctionnelles (nature de l'opération, conditions d'accès), des informations électriques (temps de traversée sur capacité donnée, consommation), et des informations géométriques (dimensions, surface). Nous aurions donc pu nous contenter d'une bibliothèque très rudimentaire, qui n'aurait pas, en particulier, contenu les descriptions des masques des opérateurs. Cependant, le générateur d'architecture n'étant qu'un maillon d'un futur compilateur de silicium pour filtres numériques ; il est apparu souhaitable de réaliser une vraie bibliothèque.

Cette bibliothèque permet l'enrichissement progressif du système grâce au stockage de nouveaux opérateurs dont pourront profiter non seulement le générateur d'architectures, mais aussi tout système y ayant accès.

Nous allons décrire les fonctionnalités que nous attendons de cette bibliothèque dans l'optique d'une utilisation strictement limitée au générateur d'architectures, puis nous décrirons brièvement la bibliothèque qui a été réalisée par les chercheurs du CNET Grenoble (CCI/AMS) ; d'après un cahier des charges établi conjointement avec d'autres utilisateurs potentiels.

5-2- Contenu de la bibliothèque

La bibliothèque doit bien évidemment contenir les types d'opérateurs correspondant aux opérations que l'on trouve dans les filtres que nous nous proposons de réaliser. Ce sont donc des plots d'entrée (avec latch), des plots de sortie (avec amplificateur), des multiplieurs, des additionneurs, des registres, des opérateurs d'arrondi, des opérateurs de changement de signe et des opérateurs de décalage.

Les voies de communication ou bus sont aussi des opérateurs particuliers : leur fonction est de véhiculer une donnée d'un point à un autre en un temps donné.

On remarque donc que la notion de bus intègre les accès trois états au canal de communication, le canal lui-même, et les multiplexeurs d'accès aux opérateurs destinataires. Ceci explique que l'on puisse donner ou imposer un temps de fonctionnement au bus, puisqu'il suffit d'ajuster la dimension (et donc la sortance) du ou des accès 3 états correspondants.

Par contre, la surface d'un bus et sa consommation sont impossibles à définir à priori.

Il nous faut également revenir sur l'opérateur registre dont nous dirons qu'il peut être de deux sortes : on distingue en effet les registres maître-esclave (double-latch) et les registres à un seul étage type simple-latch.

Tous ces opérateurs sont en fait des types paramètrables en nombre de bits, et dont au moins une instanciation a été testée sur silicium.

De fait, au lieu d'opérateur figés, la bibliothèque contient des recettes pour construire des opérateurs réels dans une base de donnée pour circuits VLSI type CASSIOPEE, ou pour fournir des renseignements sur ces opérateurs sans avoir besoin de les construire.

La souplesse d'une telle réalisation est évidente puisqu'elle fournira l'opérateur exactement adapté à la demande, et non l'opérateur surabondant le plus proche.

5-3- Modes d'utilisation de la bibliothèque

Le générateur d'architectures étant en phase de développement, il a été nécessaire de garder un certain degré d'interactivité pour apprécier les résultats et pour maitriser le temps de calcul. Cela nous a conduit à adopter un mode d'utilisation de la bibliothèque différent du mode "idéal" que nous décrivons en premier.

5-3-1 Utilisation optimale

Cette utilisation minimiserait l'intervention de l'opérateur humain :

Dès que l'algorithme à implémenter serait décrit, un module spécial du générateur d'architecture (appelons le Superviseur) analyserait les besoins en opérateurs pour cet algorithme. Puis, il consulterait la bibliothèque sur son contenu et déterminerait un ensemble d'opérateurs appelé ressources. Il activerait alors le générateur d'architecture IMHOTEP qui produirait ou non la meilleure architecture possible utilisant ces ressources.

Au vu et après analyse des résultats, le Superviseur modifierait les ressources et ré-activerait IMHOTEP jusqu'à obtenir la meilleure solution.

Ce mode de fonctionnement du système s'avère d'une part trop gourmand en temps machine, et suppose d'autre part que l'on sache apprécier automatiquement une solution et modifier automatiquement les ressources qui l'ont engendrée pour améliorer encore le résultat.

Ceci dépasse largement le cadre de cette étude, et doit être réservé pour des développements ultérieurs.

5-3-2 Utilisation actuelle

Elle consiste à confier la tâche du Superviseur à un opérateur humain. Il doit donc consulter la bibliothèque au vu d'une liste d'opérations à effectuer fournie par le système, et déterminer les ressources qu'il va fournir à IMHOTEP en vue de trouver une architecture.

Cependant, un accès automatique à la bibliothèque a encore lieu à l'initiative de IMHOTEP : en effet, lors de l'élaboration de l'architecture, des besoins en registres de travail (simple latch) peuvent apparaitre. Ces registres n'existant pas dans l'algorithme initial, ils n'ont pas de ressources propres ; et il convient de les demander à la bibliothèque.

De même, des multiplexeurs et des accès 3-états apparaissent aux entrées et aux sorties de certains opérateurs utilisés par l'architecture finale sans avoir pu être prévus au départ. Ils doivent être demandés à la bibliothèque au cours du travail d'IMHOTEP.

Nous venons de préciser les circonstances des différents accès à la bibliothèque d'opérateurs, et nous devons à présent donner la nature des informations échangées.

5-3-3 Nature des échanges entre la bibliothèque et l'extérieur

5-3-3-1 Echanges avec l'utilisateur humain

En vue de déterminer les ressources en opérateurs qu'il va confier à IMHOTEP, l'utilisateur va consulter la bibliothèque : il lui pose une question sur son contenu en matière de tel type d'opérateur possédant telles caractéristiques, et la bibliothèque lui renvoie une liste de réponses.

Exemple de question : Quels sont les multiplieurs que vous possédez, qui ont tel lot de caractéristiques ? (ce lot est le même que celui qui est demandé à l'utilisateur lors de la définition de l'algorithme, et figure aux annexes 2 et 3).

Exemple de réponse : 1er multiplieur :
- lot de caractéristiques (peut être surabondant)
- identificateur
- lot de paramètres (surface, temps,...)

2ème multiplieur : idem

etc

fin.

La nature des paramètres concernant chaque opérateur est donnée à l'annexe 3.

5-3-3-2 Echanges avec IMHOTEP

Ce sont les mêmes échanges, en se limitant au cas des registres simple latch, des mutiplexeurs et des accès 3-états. Ces échanges sont transparents à l'utilisateur et ont lieu chaque fois qu'un besoin en registre apparait lors de l'élaboration de l'architecture.

5-4- Présentation de la bibliothèque

Cette bibliothèque d'opérateurs flexibles dédiés au traitement du signal est créée et exploitée grâce au langage LOF (B5.1 à B5.3).

Le langage LOF permet de décrire des modes d'assemblages de cellules et de sous-cellules hiérarchisées, ainsi que de fournir des calculs et des évaluations sur leurs caractéristiques. Il offre la possibilité d'écrire des textes LOF appelés modules qui contiennent des "recettes" d'assemblage de cellules en vue d'obtenir tel ou tel opérateur.

LOF intègre un système de communication entre les différents modules par l'intermédiaire de boîtes aux lettres, ce qui permet à un module de niveau "supérieur" d'appeler un module "inférieur" lors de la construction d'un opérateur. A titre d'exemple, un module MULTIPLIEUR peut appeler un module ADDITIONNEUR pour disposer d'une cellule ADDITIONNEUR 1 BIT.

Une grande partie de l'intérêt des bibliothèques crées par LOF réside dans le fait que les opérateurs qu'elles contiennent sous forme de recettes sont paramètrables tant au niveau fonctionnel qu'au niveau géométrique, d'où le terme "flexibles".

Pour nous résumer, disons qu'une bibliothèque LOF est un ensemble hiérarchisé de modules communicants. Chaque module contient une recette permettant de construire une cellule, plus où moins évoluée, à partir d'un matériel qui se trouve soit dans le module lui-même, soit dans des modules inférieurs.

Les produits du module sont de différentes catégories :

Il peut s'agir bien sûr des masques de la cellule générée mais également de renseignements chiffrés sur cette cellule. Cette dernière catégorie de produits convient particulièrement aux générateurs d'architectures et autres outils travaillant au niveau logico-fonctionnel.

Ainsi IMHOTEP disposera t-il des données appropriées à son niveau de travail, tandis qu'un identificateur permettra à un générateur de plan de masse de retrouver l'opérateur choisi par IMHOTEP et d'en faire alors générer les masques par LOF.

Nous décrivons à la figure (5.1) l'arborescence de la bibliothèque sur laquelle opère IMHOTEP.

La prise en charge des échanges entre la bibliothèque et l'extérieur tels qu'ils ont été définis en 5-3-3 est assurée par un module de haut niveau. Ce module offre des services du genre catalogue, recherche sur critère et édition de résultats. Il permet donc à l'usager humain de conserver un bon confort d'utilisation, et à IMHOTEP de ne pas avoir à gérer le tri parmi les registres simple latch proposés.

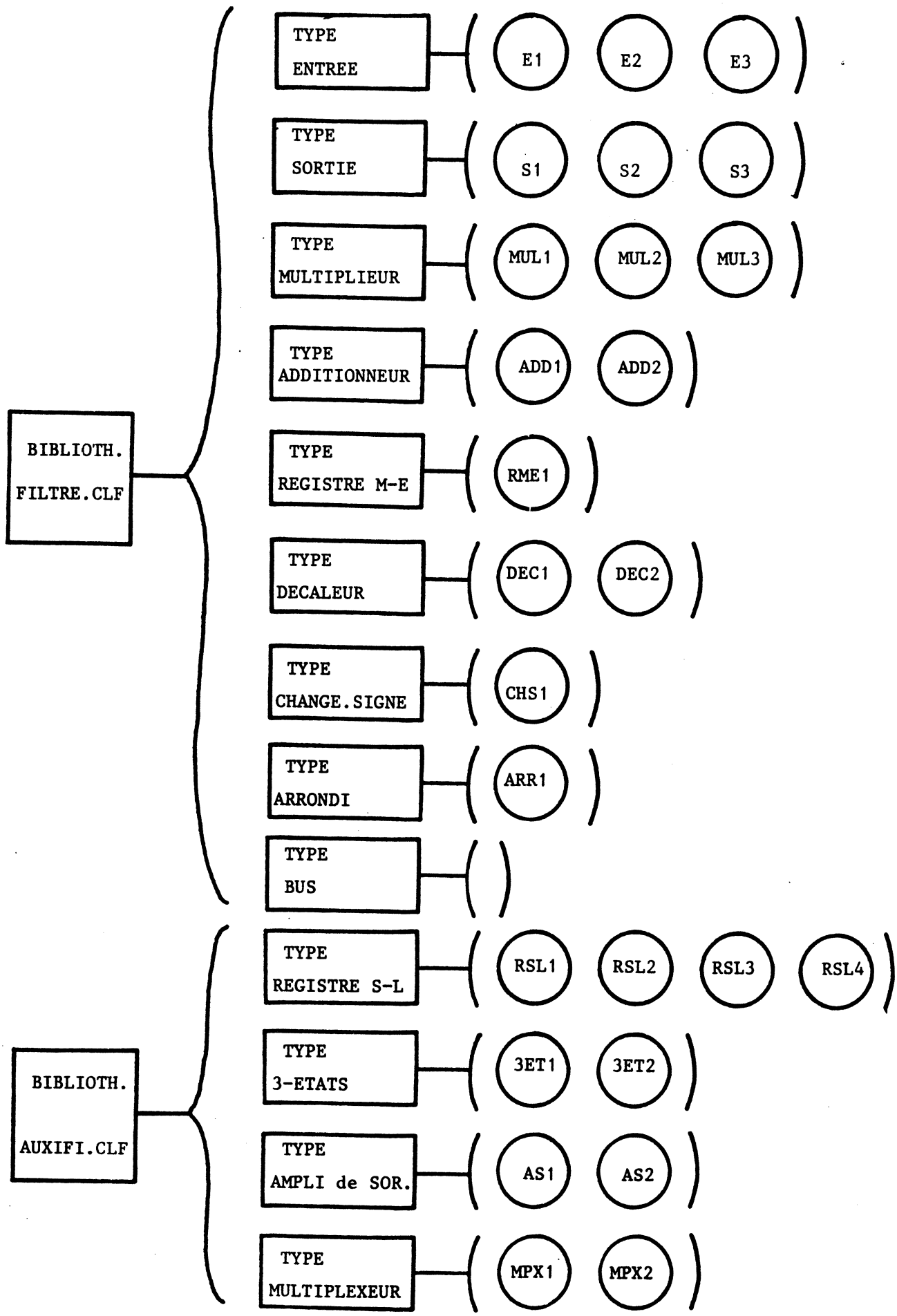

BIBLIOTH.
FILTRE.CLF
TYPE ENTREE
E1
E2
E3
TYPE SORTIE
S1
S2
S3
TYPE MULTIPLIEUR
MUL1
MUL2
MUL3
TYPE ADDITIONNEUR
ADD1
ADD2
TYPE REGISTRE M-E
RME1
TYPE DECALEUR
DEC1
DEC2
TYPE CHANGE.SIGNE
CHS1
TYPE ARRONDI
ARR1
TYPE BUS
BIBLIOTH.
AUXIFI.CLF
TYPE REGISTRE S-L
RSL1
RSL2
RSL3
RSL4
TYPE 3-ETATS
3ET1
3ET2
TYPE AMPLI de SOR.
AS1
AS2
TYPE MULTIPLEXEUR
MPX1
MPX2

Fig : 5.1

6 - Problèmes posés par l'ordonnancement de tâches sous contraintes de ressources

6- Problèmes posés par l'ordonnancement de tâches sous contraintes de ressources

6-1 Introduction

Nous allons reprendre ici ce qui avait été évoqué au chapitre 5-3 et définir en détail la manière dont nous concevons la génération d'architectures. Ce problème très complexe peut se décomposer en deux sous-problèmes, eux-même étant difficiles à résoudre. Néanmoins, cette dichotomie permet un exposé et un traitement simplificateurs.

Le problème général peut se résumer ainsi : comment choisir un ensemble d'opérateurs, et quelles opérations leur confier, pour obtenir un circuit à surface minimale respectant une contrainte temporelle ?

La division en deux sous-problèmes est immédiate, et ces deux parties sont :

- Etant donné un jeu d'opérateurs, quelles tâches confier à chacun d'eux et dans quel ordre, pour minimiser la surface totale (interconnexions incluses) tout en respectant la contrainte de temps.

- Comment choisir ce jeu d'opérateurs, et comment le modifier au vu des résultats de la partie précédente.

Nous avons choisi, dans un premier temps, de n'automatiser que la première partie car elle présente un degré de complexité déjà très appréciable.

Ce problème est du genre : ordonnancement de tâches avec contraintes de ressources d'une part et de succession d'autre part. En effet, les tâches doivent être éxécutées dans un certain ordre non total qui est décrit par le graphe initial. La limitation des ressources introduit d'autres contraintes, et le problème est donc de trouver le meilleur ordonnancement qui respecte la contrainte temporelle extérieure et qui fournisse un circuit de surface minimale.

La deuxième partie qui consiste à choisir et faire varier le jeu d'opérateur est laissée aux soins de l'utilisateur, tout en lui fournissant certains moyens d'évaluer l'architecture produite.

6-2 Classification des problèmes d'ordonnancement sous contraintes de ressources

La classification de ce genre de problèmes a été tout d'abord proposée par R.L Graham (B6.6) et consiste en une notation à trois champs a,b,c.

Le premier champ "a" spécifie l'environnement machine du problème et se décompose en deux parties $a = a_1, a_2$.

- a1 $\in$ (P, Q, R, O, F, J)
- a2 est un entier positif ou l'ensemble vide o.

Si al ∈ (P,Q,R) chaque tâche Tj consiste en une seule opération qui peut être effectuée par une quelconque des machines Mi. Le temps de passage de Tj sur Mi est Pij.

- al = P : (machines parallèles identique) : pij = pj
- al = Q : (machines parallèles uniformes) : pij = pj/qi
  (qi=vitesse de Mi)
- al = R : (machines parallèles décorrelés) pij est arbitraire.

Si al ∈ (O,F,J) chaque tâche Tj est un ensemble de mj opérations Oij et Oij doit être traitée sur une machine nij durant pij unités de temps.

- al = O : (traitement ouvert) : mj=m et nij=Mi
- al = F : (traitement séquentiel) : mj=m, Nij=Mi, Oi-1,j doit être terminée avant que Oi,j ne commence.
- al = J : (traitement par tâche) : mj et nij sont arbitraires, ni-i,j et nij sont différents et Oi-i,j doit être terminée avant que Oi,j ne commence.

Si A2 est un entier positif, A2 est le nombre de machines m; sinon m fait partie des données.

Le deuxième champ "b" donne les caractéristiques du travail, et se décompose en quatre indicateurs : b = (b1, b2, b3, b4).

- b1 ∈ (pmtn,o) : pmtn signifie que la commutation en cours de tâche est autorisée. Les tâches peuvent être interrompues et reprises sur d'autres machines.

- b2 = RES LST : exprime les contraintes sur les ressources.

Si L est un entier positif, il exprime le nombre de ressources qui est alors constant, sinon L=o et le nombre de ressources fait partie des données.

Si S est un entier positif, il exprime la taille des ressources qui est alors constante, sinon S=o et la taille de chaque ressource fait partie des données.

Enfin, si T est un entier positif, il exprime la valeur maximale de la proportion de chaque ressource consommée par chaque tâche, sinon T=o et aucune valeur maximale n'est spécifiée.

- b3 ∈ (prec, tree, chain, o) : exprime les autres contraintes.

Si b3 = prec (contraintes de précédence arbitraires) : un graphe acyclique orienté exprime les contraintes entre tâches.

Si b3 = tree (contraintes de précédences en arbre)

Si b3 = chain (contraintes de précédences en chaines)

Si b3 = o il n'existe pas de contraintes de précédences

- b4∈(pij = 1, o)

Si b4 = pij = 1, chaque opération est unitaire

Si b4 = o, les temps de calcul sont quelconques

Remarquons ici que si A1 (P,Q) alors pij vaut pj et
si A1 = R alors b4 = o

Le troisième et dernier champ "c" indique le critère que l'on a choisi d'optimiser. Pour chaque tâche, on peut en effet définir une date d'achèvement Cj, et, ayant donné une date au plus tard dj, une marge Lj = Cj - Dj.

Les critères les plus souvent utilisés sont :

- CMAX = MAX (C1, C2, ... Cn) : date d'achèvement maximale

- Cj = SOM (C1, C2, ... Ck) : somme des dates d'achèvement

- LMAX = MAX (L1, L2, ... Ln) : marge maximale

6-3 Evaluation de la complexité des problèmes classifiés

A partir de la classe du problème considéré, des rapprochements sont faits avec des problèmes classés et résolus. Il est ainsi possible de démontrer la solvabilité d'un problème en un temps polynomial, ou au contraire, son aspect NP complet.

Ainsi, il a été démontré (Garey et Johnson (B6.7)) que :

P2 (res ooo, pj=1) Cmax est solvable en O ($ln^2 + n^{s/2}$)
ou que : Q2 (res loo, pj=1) Cmax est solvable en O (n log n)

mais également que :

P3 (res loo, pj=1) Cmax est NP complet au sens strict.

La question que nous nous sommes posés est de savoir si la génération automatique d'architecture, envisagée comme un problème d'ordonnancement sous contraintes de ressources, était ou non exprimable dans cette classification. Un échange de correspondance avec Mr J.K. Lenstra du centre de recherches mathématiques d'AMSTERDAM (annexe 6) nous a révélé que ce problème n'avait encore jamais été abordé dans la littérature, et qu'il était vraisemblablement d'une grande complexité.

Les travaux de Mr Lenstra sur la classification des problèmes d'ordonnancement peuvent être utilement consultés d'après la référence B6.1.

Néanmoins, nous allons essayer de montrer la nature de cette complexité, et justifier par là notre approche heuristique.

6-4 Analyse du problème de la génération d'architectures

Au vu de l'énoncé du problème fait en 6-1 et de l'exposé de la classification fait en 6-2, force nous est de constater qu'aucune catégorie de problèmes ne correspond directement au notre. En effet, dans tous les cas représentés par le paramètre "a", les "machines", peuvent exécuter tout ou partie de n'importe quelle tâche. Ceci n'est pas le cas en ce qui nous concerne car les opérations (tâches) ne peuvent s'éxécuter que sur certains opérateurs (machines).

La première idée consiste donc à se demander si il est possible de découper le problème global en autant de sous-problèmes que de type d'opération. S'il était possible de simplifier le graphe global pour créer successivement un sous-graphe pour chaque type d'opération, les sous-problèmes pourraient s'exprimer ainsi :

Rn (res ool, prec) Cmax

avec n = nombre d'opérateurs de cette catégorie.

La bibliographie indique que les problèmes Pn (res ool, prec) Cmax avec n > 2 sont souvent NP complets, et ils sont de nature plus simple que le sous-problème que nous nous posons.

Si nous remarquons de plus que le critère que nous minimisons est la surface, sous contrainte temporelle (tendances antagonistes), et que tous les sous-problèmes ne sont pas indépendants, le lecteur se convaincra aisément de la complexité du problème global.

Nous avons donc développé une méthode heuristique qui exploite les particularités du problème pratique à résoudre afin de réduire le plus possible le nombre des solutions à explorer.

Le dénombrement des solutions reste toutefois un exercice utile afin de discerner les causes de l'explosion combinatoire du nombre des solutions.

6-5 Dénombrement des solutions

Pour ce faire, nous introduisons les notations suivantes :

- Nous avons G genres d'opérations, indexées par i.

- Pour chaque genre, il y a Ti tâches indexées par j
  et il y a Oi opérateurs indexés par k

Nous caractérisons par Pijk la possibilité pour l'opérateur k du genre i de réaliser ou non la tâche j de genre i. Si c'est possible Pijk = 1 sinon Pijk = 0.

Si nous ne considérons que les opérations du genre i, le nombre total Ci d'associations opérateurs-opération est donné par :

$$C_i = \prod_{j=1}^{T_i} \left(\sum_{k=1}^{O_i} P_{ijk} \right)$$

Comme le dénombrement pour un genre d'opération est indépendant de celui fait pour un autre genre, le nombre total des solutions possibles est :

$$C = \prod_{i=1}^{G} C_i = \prod_{i=1}^{G} \prod_{j=1}^{T_i} \sum_{k=1}^{O_i} P_{ijk}$$

Nous vérifions donc que, comme prévu, c'est le nombre d'opérateurs à essayer par opération $\sum_{k=1}^{O_i} P_{ijk}$ qui est à l'origine de l'explosion combinatoire. Donnons l'expression des bornes de C :

- Si chaque opération a un seul opérateur qui lui est dédié et qui ne peut faire aucune autre opération :

$$C = \prod_{i=1}^{G} \prod_{j=1}^{T_i} 1 \quad = 1 \text{ seule solution}$$

- Si l'on donne des opérateurs généraux, qui puissent faire toutes les opérations de leur catégorie :

$$C = \prod_{i=1}^{G} \prod_{j=1}^{T_i} O_i = \prod_{i=1}^{G} O_i^{T_i} = O_i^{T_i \cdot G}$$

à titre d'exemple, soit un circuit ayant :

- 5 multiplications
- 4 additions
- 1 entrée
- 10 bus

auquel nous fournissons le matériel suivant :

- 3 multiplieurs
- 2 additionneurs
- 1 entrée
- 1 sortie
- 3 bus

qui sont des opérateurs généraux

Le nombre total de combinaisons est donc :

$C = 3^5 \times 2^4 \times 1^1 \times 1^1 \times 3^{10} = 229582512$ solutions.

Ces résultats nous ont incités à développer une méthode heuristique qui évite d'explorer toutes les solutions pour trouver la meilleure. En effet, de nombreuses solutions sont éliminables rapidement en fonction des considérations suivantes :

- existence d'une contrainte temporelle : cela élimine les ordonnancements trop lents.

- essai d'un nombre limité d'opérateurs pour chaque opération : cela annule certains Pijk.

- recherche de la surface minimale : cela interrompt les ordonnancements donnant des circuits trop gros.

Mais la complexité est dynamique car les nombres Ti et Oi peuvent varier au cours de l'ordonnancement pour certains types d'opérations. Ceci sera exposé plus loin, mais cela rend impossible le calcul exact du nombre des solutions à essayer, et par là du temps de calcul de l'algorithme.

6-6 Conclusion

Cette étude de complexité nous permet d'avancer l'idée que le problème de la génération d'architectures est NP-COMPLET. Cette hypothèse ,souvent suggérée à propos de la conception des circuits, veut dire qu'aucun algorithme polynomial (en O(f(n))) ne peut exister qui résolve le problème.

Le dénombrement des solutions possibles sur un cas simple, nous fait par ailleurs craindre une explosion combinatoire de ce nombre quand la taille du circuit augmente, ou quand le jeu d'opérateurs grossit.

Notre approche sera donc fondée sur une méthode heuristique qui tirera parti au maximum de la spécificité du problème pour éviter ou retarder cette explosion combinatoire. Nous espérons par là rendre l'outil efficace sur des circuits de complexité moyenne (100 opérations), et en tirer suffisamment d'enseignements pour une éventuelle deuxième version qui pourrait alors utiliser des méthodes proches de l'Intelligence Artificielle.

7 - ORGANISATION GENERALE DU GENERATEUR D'ARCHITECTURES IMHOTEP

Introduction

7-1- Rappels sur l'environnement d'IMHOTEP

7-1-1 Le fichier SOURCE et son mode de création
7-1-2 Liaisons avec LOF pendant le traitement
7-1-3 Conclusion

7-2- Choix et stratégies pour l'élaboration des architectures

7-2-1 Introduction
7-2-2 Des opérateurs parallèles latchés sur leurs entrées
7-2-3 Cas des opérateurs ENTREE et SORTIE
7-2-4 Les registres fonctionnels
7-2-5 La notion de bus
7-2-6 Choix de l'horloge
7-2-7 Conclusion

7-3- Pré-allocation des opérateurs aux opérations

7-3-1 Introduction
7-3-2 La liste des candidats
7-3-3 Les critères d'admission
7-3-4 Les critères d'adéquation

7-3-4-1 Le critère statique %S
7-3-4-2 Le critère dynamique %D
7-3-4-3 Le critère global %G

7-3-5 Conclusion

7-4- Structuration des données

7-5- Les guides de la génération d'architecture : les listes hiérarchisées

7-6- Mécanismes d'élaboration de l'architecture

7-7- Organisation du programme IMHOTEP

7-8- Gestion des solutions architecturales trouvées pour le circuit

7-9- Conclusion

7 - ORGANISATION GENERALE DU GENERATEUR D'ARCHITECTURES IMHOTEP

Introduction

Comme nous l'avons précisé au chapitre 4, le problème de la génération automatique d'architectures est ici identifié à celui de l'ordonnancement de tâches sous contraintes de précédence et de ressources.

La méthode de résolution développée dans ce chapitre s'appuie sur des techniques de Project Planning, dont la méthode PERT est certainement la plus connue. Certaines particularités du traitement nous ont fait lui préférer une méthode s'appuyant sur un graphe potentiel-tâches plutôt que potentiel-étapes (cf 2-3-2-2).

Ce graphe, dont les noeuds sont des tâches et les arcs des contraintes, va évoluer au cours du traitement : au début du travail, les seules contraintes sont celles qui découlent de l'algorithme de filtrage. Au fur et à mesure de l'affectation des tâches aux opérateurs, de nouvelles tâches vont apparaitre (latchs, registres, bus) ainsi que de nouvelles contraintes de précédence. Le graphe final contiendra la description complète (parties opérative et contrôle) de l'architecture ainsi élaborée.

Nous supposons que les notions couramment utilisées en théorie des graphes sont connues (dates au plus tôt, au plus tard, marges, rang, chemin critique). Dans le cas contraire, le lecteur pourra se référer aux ouvrages en bibliographie (B7.1 et suivantes).

Le plan de ce chapitre central est le suivant : après un bref rappel sur l'environnement de IMHOTEP, nous décrirons les principes de base choisis pour l'élaboration des architectures.

Puis, nous préciserons la structure de données adoptée pour poursuivre avec l'exposé des méthodes heuristique retenues. Enfin, après avoir parlé de l'organisation du programme, nous évoquerons la gestion des solutions retenues avant de conclure.

Plan du chapitre 7.1

7-1 Rappels sur l'environnement d'IMHOTEP

7-1-1 Le fichier SOURCE et son mode de création

7-1-2 Liaisons avec LOF pendant le traitement

7-1-3 Conclusion

7-1- Rappels sur l'environnement d'IMHOTEP

IMHOTEP requiert comme base de travail la description d'un filtre d'une part et une liste d'opérateurs d'autre part. Il est donc en liaison avec un éditeur de descriptions adapté à des filtres numériques et avec une bibliothèque d'opérateurs. La liaison avec l'éditeur de filtre a été largement décrite au chapitre 2, et celle avec la bibliothèque d'opérateurs a été évoquée au chapitre 3.

Nous allons expliquer dans le détail cette deuxième liaison ainsi que la constitution du fichier SOURCE de IMHOTEP.

7-1-1 Le fichier SOURCE et son mode de création

Le fichier SOURCE contient tous les paramètres et données décrivant le travail confié à IMHOTEP : ce sont la description du filtre, le temps de fonctionnement du circuit, la liste des opérateurs fournis, et la période d'horloge disponible.

Ce fichier est élaboré par le programmme CREATESOU à partir du fichier NOMFILTRE.DAT (cf 4-3-2) et grâce à un dialogue avec la bibliothèque d'opérateurs. CREATESOU autorise l'usager à constituer, catégorie d'opération par catégorie d'opération, un jeu d'opérateurs pour réaliser le filtre.

Pour chaque catégorie d'opération, la procédure est la même, et nous allons la décrire pour la catégorie "addition". CREATESOU est organisé autour d'un menu de cinq choix qui sont (pour la catégorie en cours) :

- AFFICHAGE DES BESOINS EN OPERATEURS DU CIRCUIT.
- ENVOI D'UNE QUESTION A LA BIBLIOTHEQUE.
- AFFICHAGE DES REPONSES DE LA BIBLIOTHEQUE.
- COMPOSITION D'UN JEU D'OPERATEURS.
- PASSAGE A LA CATEGORIE SUIVANTE.

A la fin de l'éxécution de chaque choix, le menu principal est ré-affiché. Nous allons détailler les choix proposés.

- AFFICHAGE DES BESOINS DU CIRCUIT EN ADDITIONNEURS :

Cette option affiche la liste des additions du filtre avec toutes leurs caractéristiques. L'usager peut ainsi se rendre compte des besoins du circuit en additionneurs.

- ENVOI D'UNE QUESTION A LA BIBLIOTHEQUE :

Ce choix permet, via le système LOF, d'interroger la bibliothèque sur son contenu en matière d'additionneurs répondant à telle spécification. CREATESOU affiche l'écran suivant :

- ENVOI D'UNE QUESTION A LOF :

Format de la question :

- nombre de bits en entrée 1
- nombre de bits en entrée 2
- nombre de bits en sortie
- mode de contrôle d'overflow

Votre question ?

L'usager doit alors taper les paramètres de sa question qui est traitée en mode interactif, et la réponse est stockée dans un fichier BUFFER.

- AFFICHAGE DES REPONSES DE LA BIBLIOTHEQUE.

Cette option produit à l'écran, avec un format précisé, les réponses à toutes les questions posées à la bibliothèque pour la catégorie d'opération concernée. Une possibilité de supprimer certaines répones existe afin de ne garder que les plus intéressantes.

L'usager peut alors repérer les opérateurs susceptibles de l'intéresser.

- COMPOSITION D'UN JEU D'OPERATEURS

Ce choix aide l'utilisateur à constituer une liste d'opérateurs, pris parmi les réponses de LOF, qui sera donnée à IMHOTEP pour composer l'architecture. Les réponses de LOF concernent les types d'opérateurs contenus dans la bibliothèque et les opérateurs de la liste sont des instanciations de ces types et possèdent un nom qui permet de les reconnaître. Chaque type d'opérateur possède un code d'identification interne à LOF qui permettra au générateur de plan de masse PROTO de retrouver d'où viennent les opérateurs utilisés par IMHOTEP.

En effet, si IMHOTEP ne manipule que certaines caractéristiques fonctionnelles de l'opérateur (plus la surface), PROTO aura besoin de caractéristiques topologiques qui n'apparaissent pas dans la description de l'architecture fournie par IMHOTEP. Dès lors, il lui faudra questionner LOF en lui donnant l'identificateur et certains paramètres de l'opérateur pour obtenir les renseignements manquants.

- PASSAGE A LA CATEGORIE SUIVANTE

Cette option permet d'utiliser le menu principal sur la catégorie d'opérateur suivante.

Chacune des options évoquées peut être utilisée plusieurs fois, les résultats successifs se cumulant en règle générale. Ceci offre une souplesse indéniable qui rend CREATESOU amical à la plupart des utilisateurs.

Une fois le lot d'opérateurs défini, CREATESOU demande la période de l'horloge qui est disponible. A ce stade, il est possible de rendre toutes les durées des opérateurs multiples de la période d'horloge, ce qui facilitera l'élaboration du séquencement (cf 7-8).

Le fichier NOMFILTRE.SOU crée est éditable, ce qui permet des modifications manuelles rapides sans avoir à dérouler à nouveau CREATE SOU.

7-1-2 Liaisons avec LOF pendant le traitement

Le principe de ces échanges ayant été expliqué au chapitre 5-3-3-2, nous nous contentons d'en donner le mécanisme.

Au vu du temps nécessaire à chaque accès à la bibliothèque pour obtenir un latch, nous avons mis en place une "mémoire locale" afin de réduire ces accès au minimum. A chaque demande de REGISTRE de la part de IMHOTEP, une scrutation d'une table contenant toutes les réponses de LOF à cette date est entreprise. Cette table renferme des TYPES de registres différents.

Si un type de registre nous convient, nous en créeons une instanciation dans la structure de donnée, et nous fournissons un nom à ce nouvel opérateur. Si aucun type de la table n'est satisfaisant, une question est posée à LOF, et la réponse est stockée dans la table.

Ce processus est identique pour les REGISTRES SIMPLE-LATCH, les MULTIPLEXEURS et les 3-états. Il permet de réduire considerablement le nombre d'accès à LOF et la durée du traitement.

7-1-3 Conclusion

Nous avons terminé la description de l'environnement amont de IMHOTEP, ainsi que son environnement "latéral" (liaisons avec LOF pendant le traitement). Les outils situés en aval n'étant pas terminés à cette date, ils n'ont été décrits que partiellement, et de manière répartie sur l'ensemble des chapitres de cet ouvrage. Néanmoins, les principales caractéristiques en ont été définies.

Le reste du chapitre 7 décrit en détail les mécanismes et les algorithmes internes de IMHOTEP.

PLAN DU CHAPITRE 7.2

7-2- Choix et stratégies pour l'élaboration des architectures

7-2-1 Introduction

Ainsi que nous l'avons énoncé au chapitre 2, les outils de conception automatique ont besoin de s'appuyer sur des idées directrices voire des schémas simplificateurs pour être efficaces.

Nous allons développer dans ce chapitre ceux qui ont présidé à la réalisation de la première version d'IMHOTEP. Ces choix s'avèrent parfois pénalisants par rapport à ceux qu'aurait adopté un concepteur humain, mais ils ont le mérite d'avoir permis à une version d'IMHOTEP de voir le jour avec un produit homme-années d'environ 3. L'examen des résultats obtenus permettra de juger de la validité des hypothèses de départ.

7-2-2 Des opérateurs parallèles latchés sur leurs entrées

Si des architectures séries et parallèles, voire mixtes ont été envisagées au début, seul le cas des architectures parallèles a été traité à fond par la suite. Les opérateurs manipulables par IMHOTEP sont donc à structure parallèle. De plus, un latch est accolé sur chacune de leurs entrées. Ces latchs peuvent servir ou non, mais le temps de traversée des opérateurs doit tenir compte de leur présence, et ne sera pas considéré comme diminuable d'autant si on les enlève.

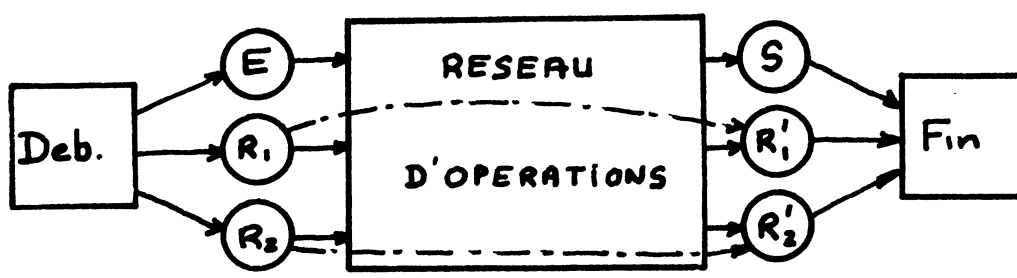
7-2-3 Cas des opérateurs ENTREE et SORTIE

Ces opérateurs ont été différenciés des autres, car, en contact avec l'extérieur, ils peuvent avoir des formes très variées. Leurs seules caractéristiques, au vu d'IMHOTEP, est d'incorporer une fonction de latch et d'avoir une durée de fonctionnement connue.

Chacun de ces deux opérateurs peut communiquer soit avec l'extérieur, soit avec un autre circuit implanté sur la même puce. Si des fonctionnalités plus élaborées sont nécessaires (signaux de DATA-TAKEN ou de DATA-READY) il convient que l'utilisateur du système les rajoute manuellement dans la liste des commandes.

7-2-4 Les registres "fonctionnels"

Ces registres figurent dans la description comportementale du filtre. Ils sont réalisés par des registres maître-esclaves ou double-latch, avec un opérateur par tâche de type registre. La partie "sortie" du registre est considérée comme une entrée du filtre, et la partie "entrée" comme une sortie du filtre (formalisme transfert de registre). A chaque échantillon entrant, le passage de l'information stockée depuis la partie maître (R') jusque vers la partie esclave (R) se fait au début du cycle.



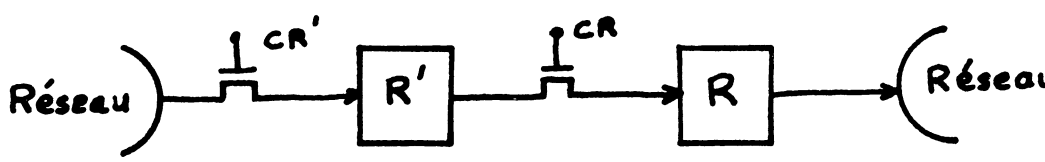


Fig : 7.1

la flèche pointillée entre R et R' (fig 7.1) exprime que dans un même intervalle de temps, R' ne doit pas être chargé avant que R n'ait été libéré, c'est-à-dire avant que le transfert de maître vers esclave ne soit terminé.

7-2-5 La notion de bus

La manière de manipuler les bus est une des options fondamentales d'IMHOTEP. La notion de bus recouvre un ensemble d'éléments qui comprend :

- le canal de communication lui-même sous la forme d'une nappe de fils.

- les accès à ce canal, qui possèdent une fonction 3-états si le bus a plusieurs antécédents.

- les multiplexeurs à l'entrée des opérateurs atteints si l'entrée considérée est atteinte par plus qu'un bus.

Chaque accès 3-états et multiplexeur possède une ou plusieurs commandes. Ainsi, dans le cas général, pour établir une liaison entre la sortie de l'opérateur op1 et l'entrée E de l'opérateur op2 il faut : (fig 7.2)

- activer la commande du 3-états situé en sortie de op1,

- activer la commande du multiplexeur situé sur l'entrée E de l'opérateur op2 pour sélecter le bon bus.

Bien entendu, pour certaines liaisons, ces commandes peuvent être dégénérées (bus toujours connecté).

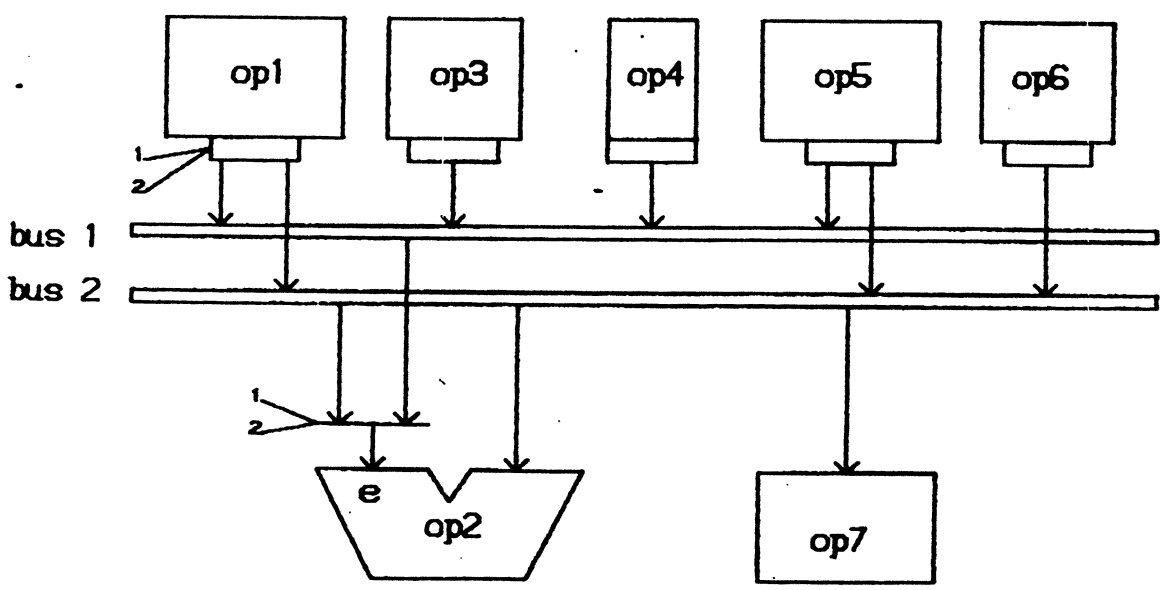


Fig : 7.2

7-2-6 Choix de l'horloge

Nous avons fait ici le choix d'une horloge unique. Toutes les commandes sont activées sur un même front de cette horloge (ce peut être le front montant ou descendant suivant le matériel utilisé).

Cette horloge conditionnera donc toutes les transitions dans le graphe du séquencement de la structure obtenue.

L'utilisation d'une horloge à deux phases non recouvrantes permettrait de gagner en rapidité dans le cas des latches par exemple, dont la commande présente deux transitions rapides. Il serait envisageable de synchroniser la première transition sur une phase, et la deuxième sur l'autre, ce qui permettrait de gagner une période d'horloge par rapport au cas d'une horloge unique.

7-2-7 Conclusion

Nous avons exposé l'ensemble des choix et des hypothèses qui permettent à IMHOTEP de résoudre de manière efficace le niveau 3 de la chaîne de conception. Il faudra vérifier la validité de ces hypothèses en examinant les architectures produites.

PLAN DU CHAPITRE 7.3

7-3- Pré-allocation des opérateurs aux opérations

7-3-1 Introduction

Lorsque l'on désire répartir un ensemble d'opérations sur un jeu d'opérateurs, on peut décomposer le problème en deux étapes.

La première consiste à éliminer les cas aberrants pour ne pas demander, par exemple, à un inverseur de faire une addition. Chaque opération doit donc posséder une liste des opérateurs capables de la réaliser.

La deuxième étape consiste à ordonner cette liste en attribuant à chaque opérateur un critère d'adéquation à l'opération concernée.

7-3-2 La liste des candidats

Chaque opération du graphe va recevoir une liste chaînée de candidats qui sont des opérateurs susceptibles de la réaliser.

Les différentes simplifications et optimisations ayant été réalisées au niveau du fichier source (§ 7-1) chaque opération doit être éxécutée sur un opérateur ayant le même type qu'elle.

Chaque élément de la liste chaînée est un RECORD PASCAL qui a les champs suivants :

- SP-NUMOPER : numéro de l'opérateur candidat
- SP-POPER : pointeur sur l'opérateur
- SP-CRITERE-S : critère statique de l'opérateur
- SP-CRITERE : critère global de l'opérateur
- SP-PSPER : pointeur sur le candidat suivant

Nous devons à présent expliquer la signification et le calcul des critères d'adéquation, mais auparavant, il faut expliciter quels sont les critères d'admission dans la liste des candidats.

7-3-3 Les critères d'admission

Ces critères varient selon le type d'opérateur concerné. Nous en donnons ici la liste. Les abréviations suivantes seront utilisées :

- NBENT = nombre de bits en entrée
- NBSOR = nombre de bits en sortie
- RANG = valeur du décalage d'un décaleur (-1 = variable)
- ARMODE = mode d'arrondi en sortie
- OVERFL = contrôle d'overflow
- COEFF = nombre de bits du coefficient d'un multiplieur
- CVAL = valeur du coefficient d'un multiplieur (0 si variable)

De plus, le suffixe .TA rapportera le terme à une tâche et .OP à un opérateur. Voici à présent la condition d'admission dans la liste des candidats de telle tâche pour un opérateur de même type.

- ENTREE :
```
NBENT.OP >= NBENT.TA
```

- SORTIE :
```
NBENT.OP >= NBENT.TA
```

- ADDITION :
```
(((NBENT1.OP >= NBENT1.TA) AND
(NBENT2.OP >= NBENT2.TA)) OR
((NBENT1.OP >= NBENT2.TA) AND
(NBENT2.OP >= NBENT1.TA))) AND
(NBSOR.OP >= NBSOR.TA) AND
(OVERFL.OP = OVERFL.TA).
```

- MULTIPLICATION :
```
((NBENT.OP >= NBENT.TA) AND
(NBCOEFF.OP >= NBCOEFF.TA) AND
(NBSOR.OP >= NB SORTA) AND
((CVAL.OP = 0) OR (CVAL.OP = CVAL.TA))) AND
(NBSOR.OP = NBSOR.TA) AND
(ARMODE.OP = ARMODE.TA)
```

- REGISTRE :
```
NBENT.OP >= NBENT.TA
```

- CHANGEMENT DE SIGNE :
```
NBENT.OP >= NBENT.TA
```

- ARRONDI :
```
(NBENT.OP >= NBENT.TA) AND
(NBSOR.OP >= NBSOR.TA) AND
(ARMODE.OP = ARMODE.TA)
```

- DECALAGE :
```
(NBENT.OP >= NBENT.TA) AND
(NBSOR.OP >= NBSOR.TA) AND
((RANG.OP = RANG.TA) OR
(RANG.OP = - 1))
```

- BUS :
```
NBENT.OP >= NBENT.TA
```

Il est aisé de reconnaître que cette condition d'admission se réduit pour la plupart des types d'opérateurs à accepter des nombres supérieurs ou égaux à ceux concernés par l'opération. Pour les autres opérateurs, des conditions supplémentaires, qui peuvent encore être raffinées, viennent s'ajouter à la condition de capacité.

7-3-4 Les critères d'adéquation

Plusieurs types de critères peuvent mesurer l'adéquation d'un opérateur à une tâche :

Le premier sera dit CRITERE STATIQUE, et, exprimé en pourcentage, nous le noterons %S. Il ne prend en compte que la plus ou moins bonne adaptation matérielle de l'opérateur à la tâche. A titre d'exemple, un multiplieur 12 x 12 sera mieux adapté à une multiplication 12 x 10 qu'un multiplieur 12 x 16.

Ce pourcentage est calculé pour chaque opérateur, et peut être modifié par une fonction ajustable par l'utilisateur, afin de donner plus de poids à ce critère pour tel ou tel type d'opérateur.

Le deuxième critère sera dit CRITERE DYNAMIQUE et noté %D car il prend en compte le "passé" de l'opérateur. En effet, si l'opérateur a déjà réalisé telle tâche, il est lié à certains bus et sera donc plus ou moins adapté à réaliser une autre tâche précédée par ses propres bus. Ce critère est bien dynamique puisqu'il évolue au cours de l'élaboration de l'architecture. Si il est bien choisi, il conduit à ossifier progressivement le circuit dans une bonne configuration, et il évite à l'algorithme d'affectation de se perdre dans des exploitations hasardeuses.

Remarquons enfin que ces deux critères, %S et %D, sont assez indépendants. Nous allons devoir les unifier en un seul qui guidera IMHOTEP. Le critère global %G fera l'objet d'un paragraphe spécial.

7-3-4-1 Le critère statique %S

Ce critère est grosso modo un pourcentage d'utilisation physique de l'opérateur ; ceci de manière spécifique à chaque type d'opérateur, mais de manière uniforme et sans jugement de valeur. %S peut être modifié par une fonction spécifique à chaque type d'opérateur comme évoqué plus haut (fig 7.3).

En reprenant les notations du paragraphe précédent, nous donnons le calcul de %S :

- ENTREE, SORTIE, REGISTRE, CHANGEMENT DE SIGNE et BUS :

  %S = 1 - (NBENT.OP - NBENT.TA)/NBENT.OP

- MULTIPLIEUR :

  Si il est dédié, FLAGD=1 sinon FLAGD=0

  %S = 1 - (NBSOR.OP - NBSOR.TA)/NBSOR.OP
  - (COEFF.OP - COEFF.TA)*FLAGD/(COEFF.OP*10)
  - (NBENT.OP - NBENT.TA)/(NBENT.OP*10)

- ADDITIONNEUR :

  %S = 1 - (NBSOR.OP - NBSOR.TA)/NBSOR.OP
  - (NBENT1.OP + NBENT2.OP - NBENT1.TA - NBENT2.TA)/
  ((NBENT1.OP + NBENT2.OP)*10)

- ARRONDI :

  %S = 1 - (NBENT.OP - NBENT.TA)/(NBENT.OP-NBSOR.OP)

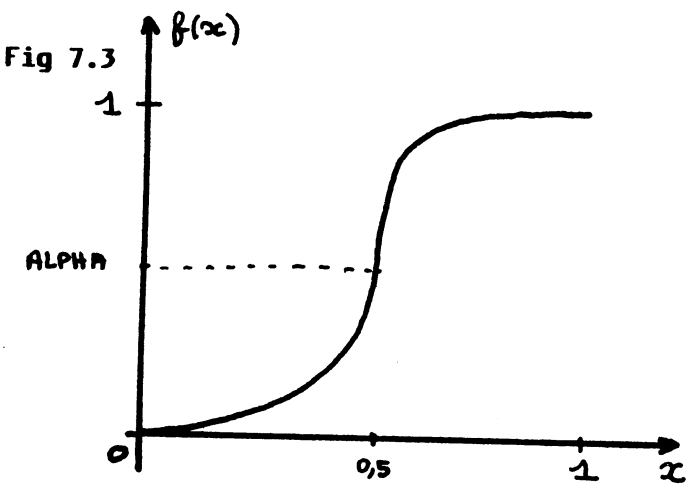
- DECALEUR :

  %S = 1 - (NBSOR.OP-NBSOR.TA)/NBSOR.OP (si il est cablé)
  = 1 - (NBENT.OP + NBSOR.OP - NBENT.TA - NBSOR.TA)/
  (NBENT.OP + NBSOR.TA) (si programmable)

Le pourcentage obtenu tient évidemment compte de la nature de chaque opérateur, mais comme un multiplieur utilisé à 50 % est plus coûteux qu'un décaleur utilisé à 50 %, on corrige %S par la fonction suivante : (n dépendant du type d'opérateur).

f(x) = (n+1)*x**n-n*x**(n+1)
avec ALPHA = (n+2)/2**(n+1)
si ALPHA = f(0,5)

Fig 7.3



7-3-4-2 Le critère dynamique %D

Comme nous l'avons exprimé précédemment, %D incorpore le passé de l'opérateur, c'est-à-dire les tâches qu'il a déjà effectuées.

Le but est d'obtenir un jeu d'interconnexions réaliste. Nous allons expliquer, sur un exemple, comment est calculé ce critère.

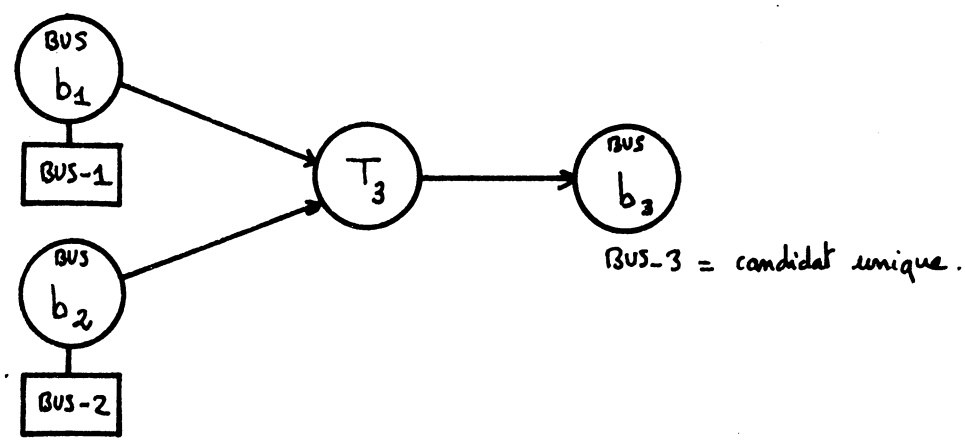
Soit une tâche T3 et un opérateur OP pour lequel on veut calculer le critère %D d'adaptation à T3. Si OP n'a jamais effectué d'opérations, le critère %D est neutre et seul compte %S. Si OP a déjà effectué deux tâches T1 et T2, nous allons examiner les similitudes entre T1 et T3, puis entre T2 et T3.

Le nombre maximum de similitudes servira à calculer le critère %D.

Considérons à présent le couple (T1, T3) : nous définissons comme une "similitude" le fait qu'une entrée (ou une sortie) de T3 soit réalisée (ou obligée d'être réalisée) par le même opérateur qui a réalisé l'entrée correspondante sur T1. Le critère %D sera la valeur maximum du rapport entre les similitudes trouvées sur les similitudes possibles ; ceci pour les tâches T1 et T2.

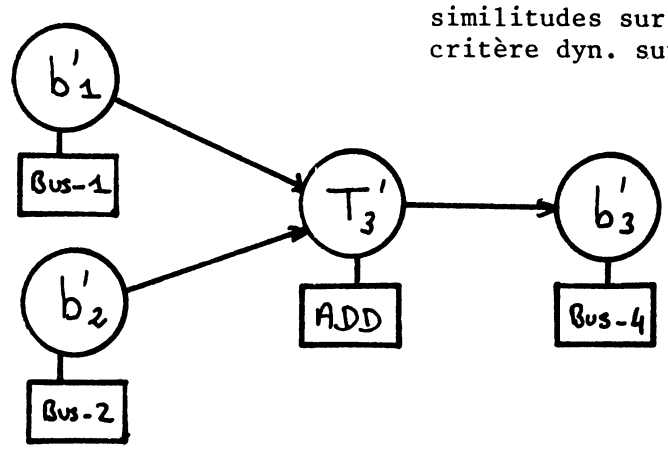
La figure (7.4) traite un exemple pour un opérateur de type addition qui a déjà réalisé 2 additions auparavant.

Situation à évaluer : ADD sur l'addition T3



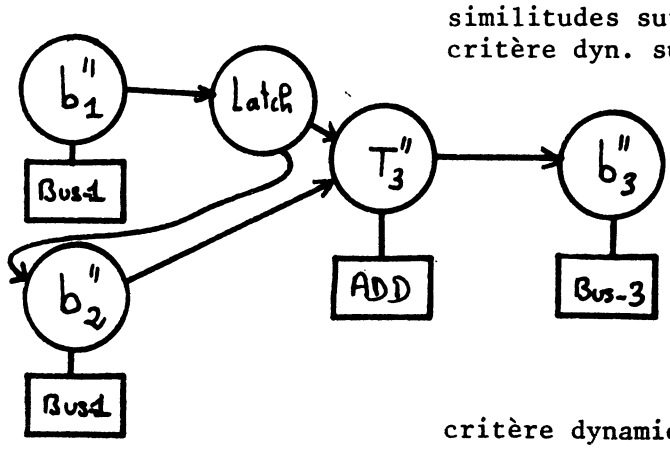
Première situation déjà réalisée : ADD sur T'3

similitudes sur les entrées : 2 sur 2
similitudes sur la sortie : 0 sur 1
critère dyn. sur ce cas = (2 +0)/3=0.66



Deuxième situation déjà réalisée : ADD sur T"3

similitudes sur les entrées : 1 sur 2
similitudes sur la sortie : 1 sur 1
critère dyn. sur ce cas = (1 + 1)/3=0.66



critère dynamique final = 0.66

Fig : 7.4

7-3-4-3 Le critère globale %G

%G est une combinaison de %S et %D. Ces deux critères étant indépendants, car ayant chacun leur spécificité, il n'est pas aisé de les combiner. En effet, ils expriment le compromis fondamental entre les opérateurs et les interconnexions qu'il faut résoudre habilement si l'on veut un circuit efficace. L'exemple suivant peut aider à comprendre la nécessité de ce compromis : pour faire une multiplication 12 x12 alimentée par le bus B1 de 16 bits, vaut-il mieux utiliser le multiplieur M1 de 16 x 16 déjà connecté à B1, ou le multiplieur M2 de 12 x 12 connecté au bus B2 ? Utiliser M1 serait pénalisant si l'on se réfère à %S ; et utiliser M2 sera pénalisant pour %D. (fig 7.5).

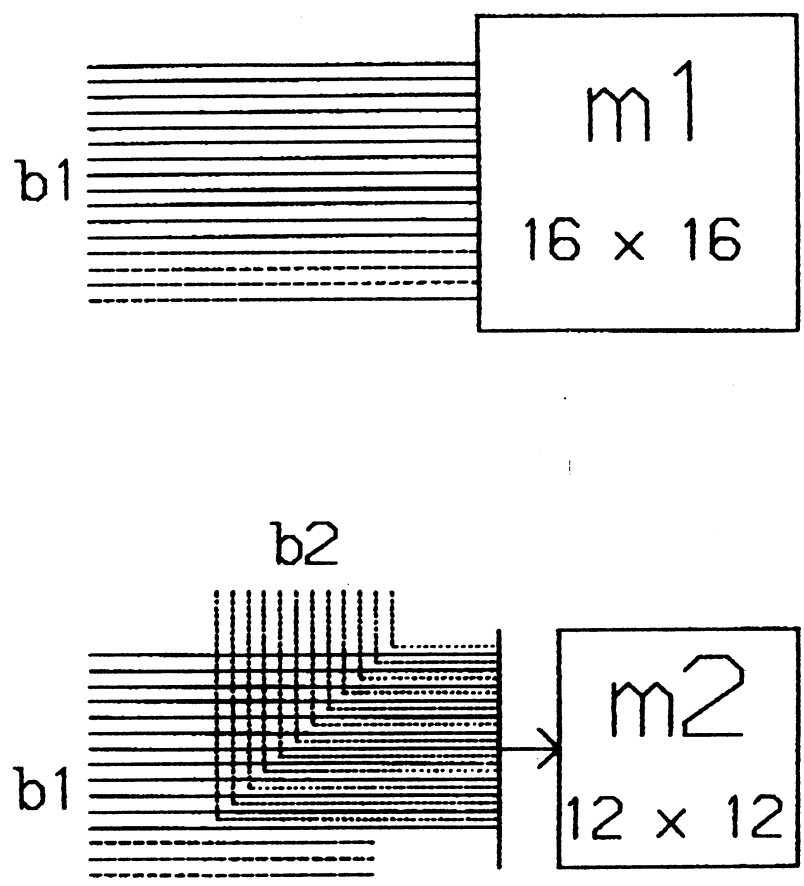


Fig : 7.5

Ce critère dynamique est donc un pourcentage, comme %S, et peut également être modifié par une fonction spécifique pour chaque type d'opérateur.

Nous avons isolé certaines caractéristiques d'une fonction %G = F(%S, %D), puis nous avons extrapolé ces caractéristiques pour trouver F.

Ces caractéristiques sont les suivantes :

- Si %S = 0 alors %G = 0 aussi.
- Si %S = 1 alors %G >= 0,5
- Si %D = 0 alors %G <= 0,5
- Si %D = 1 alors %G devra être élevé si %S > 0,5
  et %G devra rester bas si %S < 0,5

Ces caractéristiques nous fournissent les quatre plans verticaux du cube qui contient la surface décrivant %G. La figure 7.6 donne les représentations graphiques sur les quatre plans, et la figure 7.7 représente la surface du critère %G.

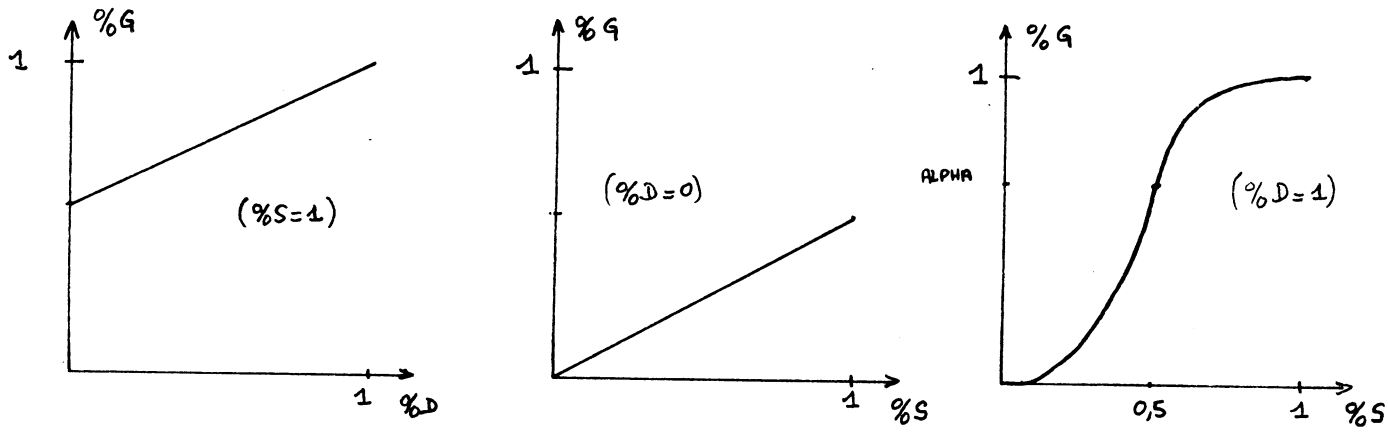
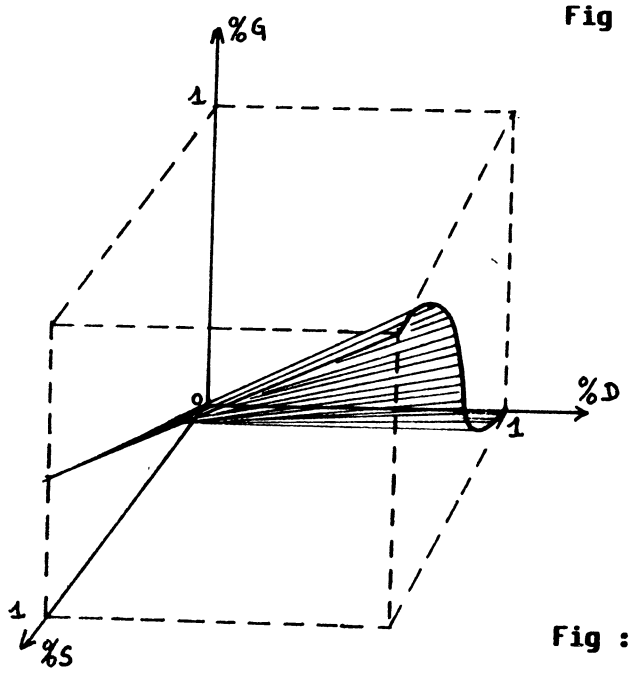


Fig : 7.6



$$\%G = \frac{\%S}{2} + \%D\left((n+1)\,\%S^{n} - n\,\%S^{n+1} - \frac{\%S}{2}\right)$$

$$\text{ALPHA} = f(0{,}5) = (n+2)/2^{n+1}.$$

Fig : 7.7

7-3-5 Conclusion

L'élaboration de la liste des candidats est incontestablement un des points clef de IMHOTEP, et en conditionne l'efficacité.

Un grand nombre de paramètres sont ajustables, tout en possédant des valeurs par défaut qui ont été déterminées par le simple bon sens, puis ajustées au vu des premiers essais. Néanmoins, pour un circuit particulier, il devrait être intéressant de faire varier ces paramètres pour privilégier tel ou tel style d'architecture.

PLAN DU CHAPITRE 7-4

7-4- Structuration des données

7-4-1 Introduction

La représentation en mémoire du problème de la génération d'une architecture et de son évolution devait présenter certaines caractéristiques. D'une part, la structure adoptée devait être totalement dynamique pour ne pas introduire de problèmes de dépassements de capacité pour des circuits importants. D'autre part, il convenait de minimiser le temps d'accès à l'information, quitte à introduire de la redondance et augmenter le volume des données stockées.

Nous avons donc fait un usage intensif des POINTEURS PASCAL qui se prètent bien à la représentation d'un graphe dynamique. La description des composants principales de la structure de données fait l'objet des quatre prochains paragraphes.

7-4-2 La structure de table variable

Cette table de longueur variable et de contenu variable est très utilisée, et nous allons la décrire en détail. Par la suite, la dénomination "table de X" se rapportera à une table de longueur variable contenant des éléments de type X.

La propriété "longueur variable" est réalisée de la manière suivante : la table est composée de tronçons de longueur fixe, liés entre eux par des pointeurs. Chaque tronçon contient MAX-CASES éléments qui peuvent être de type différents. Un élément est un mot de 32 bits défini par un CASE PASCAL. Ainsi, si ELEM est un élément du tableau, ELEM.PT sera un pointeur sur une tâche, et ELEM.PO sera un pointeur sur un opérateur, etc... Deux procédures TABLE-L et TABLE-E permettent d'écrire dans la table voulue, à l'endroit voulu, l'élément voulu.

Lors de l'écriture, TABLE-E génère autant de tronçons que nécessaire pour atteindre la place spécifiée. Le parcours des tronçons chainés est donc totalement transparent à l'utilisateur.

7-4-3 La table des travaux et leurs candidats

Ce paragraphe reprend et complète le 4-3-2-2 qui décrivait succintement la structure de données, les tâches du graphe, opérations et bus, sont liées entre elles par des flèches (contraintes) de sucessions.

Chaque tâche est représenté par un RECORD PASCAL qui contient :

- son nom, son numéro, son espèce, ses dates au plus tôt et plus tard
- un pointeur sur la liste de ses antécédents
- un pointeur sur la liste de ses successeurs
- une liste de paramètres dépendant de la nature de la tâche.

Un élément de la liste des antécédents ou des successeurs contient :

- le numéro du voisin
- un pointeur sur le RECORD PASCAL du voisin
- le numéro logique de l'entrée atteinte (si nul, c'est une contrainte dûe au multiplexage).

Un TRAVAIL est un RECORD PASCAL contenant :

- un pointeur sur la tâche à faire
- son activité (à faire, terminée, suspens, active)
- sa durée (réelle ou la plus probable)
- un témoin de latchage de la tâche, et le tableau des entrées latchées
- un pointeur sur la liste des candidats

Ces travaux sont donc rangés dans une table, au numéro qui est le numéro de la tâche correspondante (fig 7.8).

La durée d'une tâche et calculée de la manière suivante : si la tâche est terminée, c'est la durée connue de l'opérateur qui l'a réalisée. Si la tâche est encore à faire, sa durée est la moyenne des durées des candidats opérateurs pondérées par le critère %G de chaque opérateur.

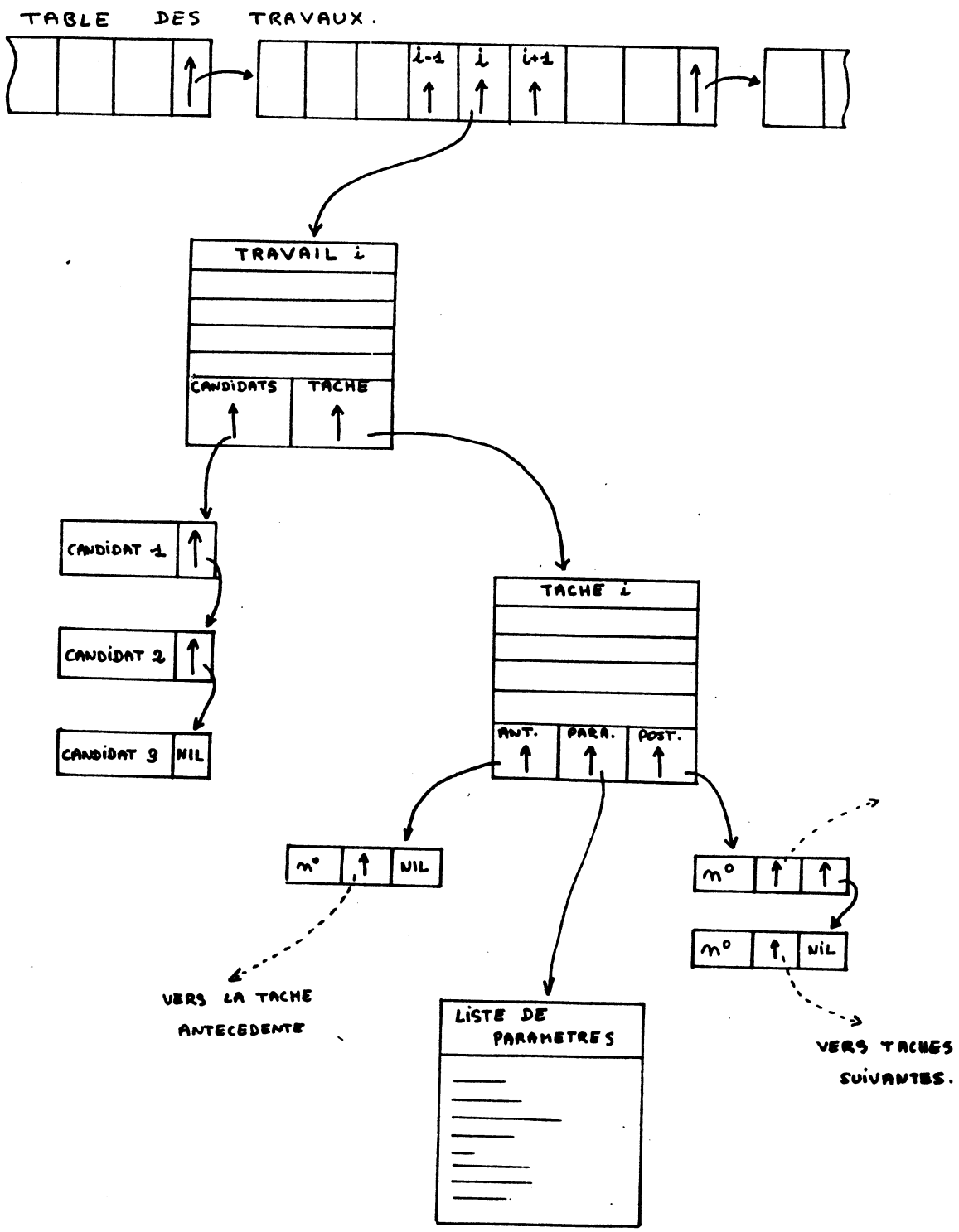


TABLE DES TRAVAUX.
i-1
i
i+1
TRAVAIL i
CANDIDATS
TACHE
CANDIDAT 1
CANDIDAT 2
CANDIDAT 3
NIL
TACHE i
ANT.
PARA.
POST.
n°
NIL
n°
n°
NIL
VERS LA TACHE ANTECEDENTE
LISTE DE PARAMETRES
VERS TACHES SUIVANTES.

Fig : 7.8

7-4-4 La table des témoins et leurs preuves

Un TEMOIN représente l'activité d'un opérateur, et une PREUVE est une trace de l'utilisation de cet opérateur. Le témoin et ses preuves sont donc à un opérateur ce que le travail et ses candidats sont à une opération.

Chaque opérateur est représenté par un RECORD PASCAL qui contient :

- son nom, son numéro, son espèce et son identificateur base de donnée
- un pointeur sur la liste de ses paramètres fonctionnels qui dépendent de son espèce.
- un pointeur sur ses paramètres matériels (durée, surface...).

Le TEMOIN est un RECORD PASCAL qui contient :

- le pointeur sur l'opérateur correspondant
- les dates de début et de fin de la dernière utilisation
- un indicateur d'occupation
- la liste des entrées physiques latchées
- des pointeurs sur le début et la fin du tableau de GANTT de cet opérateur (voir 7-4-5)
- un pointeur sur la liste des PREUVES d'utilisation

De même, une PREUVE est un RECORD contenant :

- un pointeur sur la tâche qui a été réalisée
- le numéro de cette tâche
- le tableau de correspondance entre entrées logiques de la tâche et entrées physiques de l'opérateur.

Les témoins sont rangés dans une table, par ordre de numéro d'opérateur croissant. (fig 7.9).

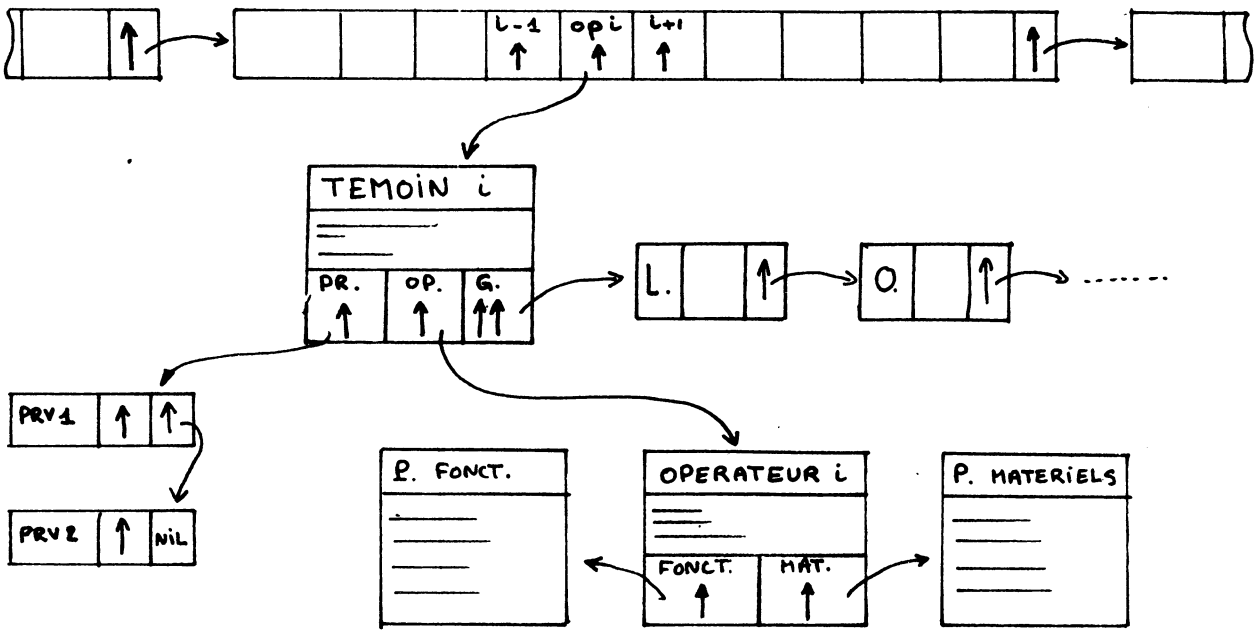


Fig : 7.9

7-4-5 Le tableau de GANTT

Le tableau de GANTT est une représentation très utilisée en ordonnancement de tâches. C'est un échéancier qui donne, pour chaque ressource, les dates séparant des périodes d'occupation et de non-occupation.

Nous avons utilisé cette représentation (fig 7.10) pour visualiser l'occupation de chaque opérateur au cours du temps. Chaque "tranche de temps" est un RECORD qui, outre un double chaînage avant et arrière, contient les informations suivantes.

- Etat de la tranche : libre ou occupée
- Date de début de la tranche
- Date de fin de la tranche
- Tâche clef = tâche occupante si la tranche est occupée
  = liste des tâches libérantes si la tranche est libre.

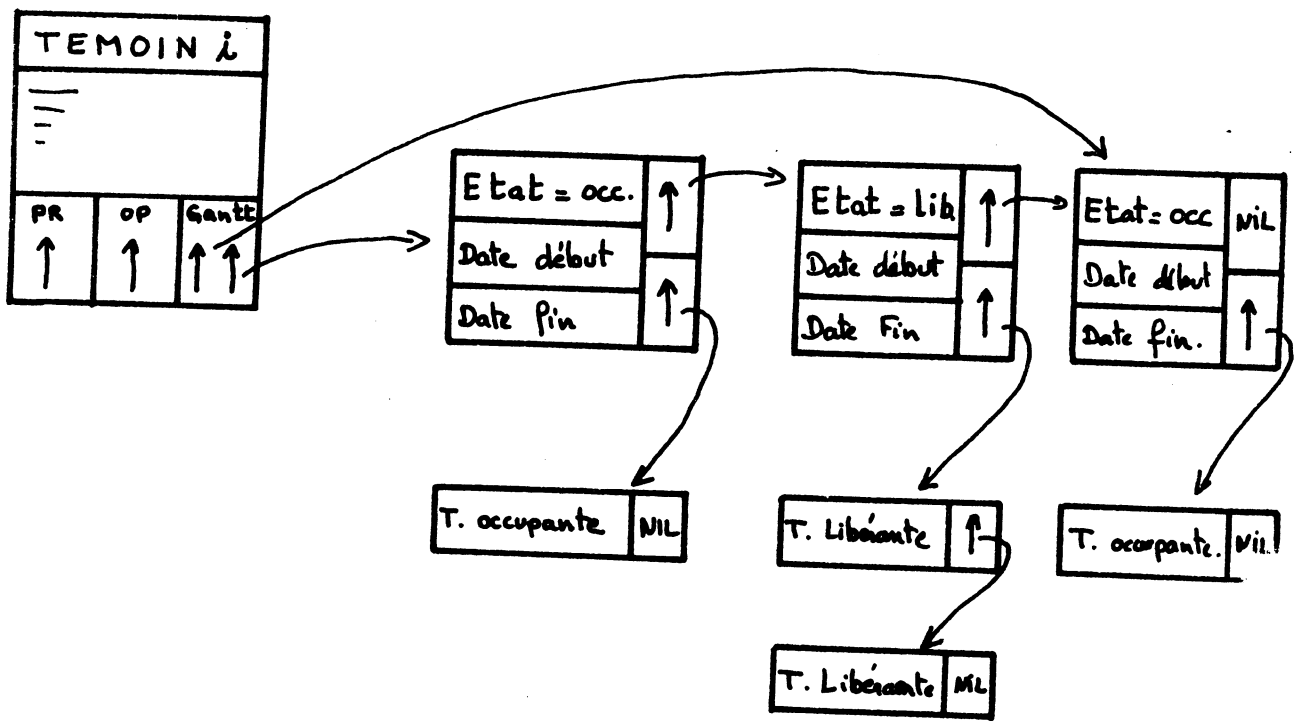


Fig : 7.10

7-4-6 Conclusion

L'ensemble de ces tableaux sert à IMHOTEP à la fois de tableau de bord pour conduire la génération d'architecture, et de carnet de bord pour y inscrire toutes les modifications.

Ces tables sont donc modifiées durant toute la conception, et lorsqu'IMHOTEP arrive à une solution, elles contiennent toutes les informations nécessaires à l'extraction de la partie opérative et de la partie contrôle.

PLAN DU CHAPITRE 7-5

guides de la génération d'architecture : les listes hiérarchisées

7-5-1 Introduction

Depuis le début du chapitre 7, nous avons défini l'environnement dans lequel IMHOTEP va travailler. Nous précisons à présent les heuristiques d'élaboration de l'architecture qui s'appuient sur cet environnement.

IMHOTEP utilise le graphe des tâches pour déterminer la tâche qu'il est le plus opportun de réaliser à cet instant. Pour cette tâche, il disposera de plusieurs opérateurs à essayer. Là encore, il faudra faire un choix pour n'en essayer qu'un nombre minimum et judicieux.

Nous constatons qu'IMHOTEP utilise des listes de tâches à faire et des listes d'opérateurs pour chaque tâche. L'élaboration de ces listes contient une grande partie de "l'intelligence" de IMHOTEP, et nous allons consacrer un paragraphe à chacune d'elles.

7-5-2 La liste des tâches activées

7-5-2-1 Définition

A un instant quelconque de l'élaboration de l'architecture, certaines tâches du graphe sont terminées, c'est-à-dire qu'un opérateur leur a été affecté pendant un certain temps, et d'autres sont encore à faire.

Parmi les tâches à faire, certaines sont activées ; c'est-à-dire que leur exécution peut commencer immédiatement. Ce sont toutes celles dont tous les antécédents sont terminés.

La liste des tâches activées regroupe donc ces tâches dans un ordre qui est loin d'être sans influence sur l'efficacité de l'architecture finale. Examinons donc le mode de constitution de la liste des tâches activées.

7-5-2-2 Hiérarchisation stricte

Dans un premier temps, toutes les tâches terminées du graphe voient leur durée mise à zéro. Pour toutes les tâches restantes, on calcule leur marge qui est la différence entre la date de fin du graphe et leur date au plus tard. On met dans la liste toutes les tâches validées sous forme d'un RECORD qui contient :

- le numéro de la tâche
- le pointeur sur cette tâche
- la marge de la tâche
- un paramètre de classement ORDRE
- un indicateur particulier

A la suite de ce premier traitement, on recalcule le graphe de manière réaliste, chaque tâche ayant une durée non nulle.

Dans un deuxième temps, nous calculons le paramètre ORDRE qui va dépendre du type de la tâche d'une part, et de configurations spéciales du graphe d'autre part.

Si la tâche est un bus, on regarde si un des bus-physiques candidat porte déjà la donnée destinée à être acheminée sur le bus-tâche. Si c'est le cas, l'indicateur particulier est positionné. De plus le paramètre ORDRE est chargé avec une priorité : si le bus-tâche ne sort pas d'un registre, la priorité est plus forte que dans le cas contraire. En effet, si la donnée est stockée dans un registre, il est moins "urgent" de profiter du fait qu'elle est déjà présente sur un bus, puisque son maintien n'immobilise pas un opérateur "couteux". (fig 7.11).

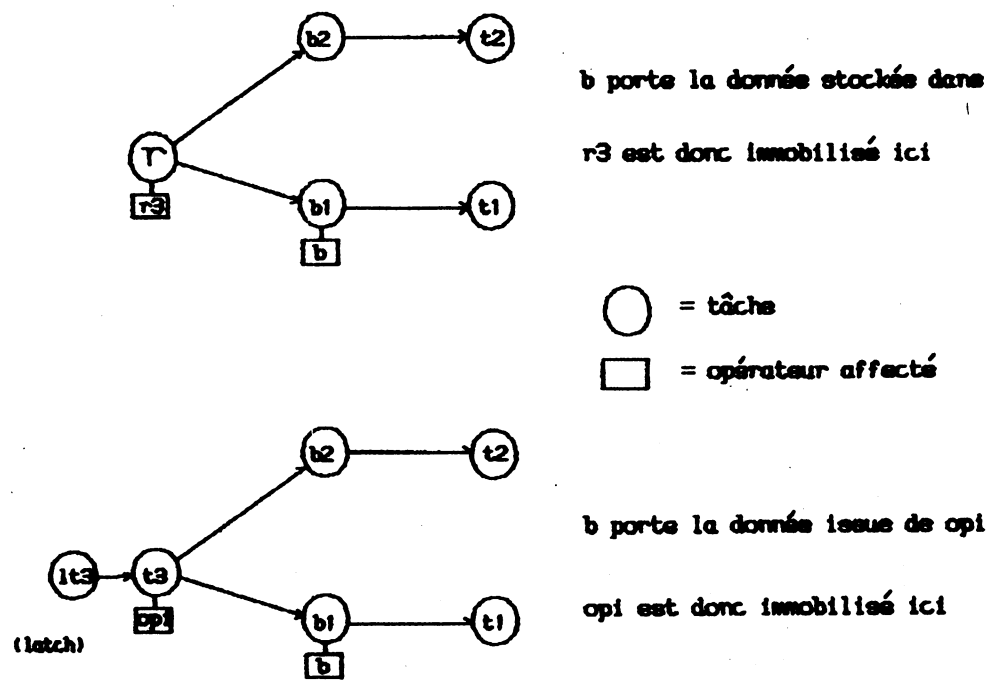


Fig : 7.11

Si le bus n'a aucun bus-physique porteur de la donnée, la calcul de ORDRE est différent : si le successeur du bus a une seule entrée ORDRE prend une certaine valeur, si ce successeur a deux entrées, ORDRE sera différent suivant que le bus alimentant l'autre entrée est terminé ou non.

Chacune des valeurs de ORDRE est encore modulée par le fait que l'antécédent du bus est ou non un registre.

Si la tâche n'est pas un bus, le calcul des différentes valeurs que peut prendre ORDRE est différent. Une distinction est faite entre les tâches "SUSPENS" et les autres. La définition précise de l'attribut SUSPENS sera donné au chapitre 7-6-2, mais nous allons résumer sa signification comme suit :

Une tâche SUSPENS est une tâche pour laquelle IMHOTEP a fait le pari suivant : elle aura un opérateur à sa disposition avant telle date, et sera donc capable de latcher la donnée présente sur son entrée, libérant ce faisant telle autre ressource.

Dans le cas des tâches non suspens, ORDRE prendra des valeurs différentes si cette tâche est un registre ou non. Si la tâche n'a pas d'opérateur libre ou libérable (cf chapitre 7-6), elle est supprimée de la liste des tâches activées.

Dans le cas des tâches SUSPENS, ORDRE prend la valeur 0 si aucun opérateur n'est disponible (pari perdu), et une autre valeur si un opérateur est disponible à temps.

A l'issu de ce travail, toutes les tâches ont un paramètre ORDRE affecté d'une valeur. Ces différentes valeurs sont choisies pour introduire une gamme de priorité entre les tâches, afin de tenir compte de configurations locales du graphe.

On classe alors la liste par marge décroissante, puis, comme sous-critère, par ORDRE décroissant.

Cet ordre strict, à critères hiérarchisés, masque parfois la meilleure stratégie à suivre. A titre d'exemple, une tâche qui aurait une marge légèrement inférieure, mais une priorité ORDRE très élevée, sera reléguée à la deuxième place. Nous allons donc atténuer cet ordre en introduisant une tolérance sur la tête de la liste.

7-5-2-3 Hiérarchisation douce

Nous définissons deux pourcentages : "tolérance mini" et "tolérance maxi". Dans un premier temps, les marges qui sont supérieures ou égales à (1 - tolérance mini)* marge maximum sont mises à marge maximum et la liste est classée à nouveau avec les mêmes critères. Pour les tâches en tête de liste, le critère ORDRE sera donc prééminent.

Nous avons constaté une amélioration très nette de la qualité des architectures générée en utilisant ces pourcentages.

Le pourcentage "tolérance maxi" est utile dans un cas plus rarement rencontré.

Si, dans la liste hiérarchisé avec "tolérance mini", la première tâche a une priorité zéro, on convient d'étendre la tolérance jusqu'à "tolérance maxi" ou jusqu'à ce qu'une tâche à priorité non nulle apparaisse en tête.

Le cas ORDRE=0 représente un traitement pénalisant pour l'architecture, qui peut de plus être résolu simplement en poursuivant l'affectation des tâches, ce qui peut supprimer le problème posé.

Par contre, le cas ORDRE=0 ne peut être systèmatiquement rejeté en fin de liste, car, si le problème n'est pas de nature à se résoudre simplement en attendant, on risque de désoptimiser l'architecture finale.

La liste ainsi hiérarchisée est prète à être exploitée par IMHOTEP.

7-5-2-4 Conclusion

A l'exposé de ce qui précède, il apparait que le classement de la liste des tâches validées est un travail très subjectif, difficile à traiter de manière purement algorithmique.

Il s'agit d'avantage de recettes paramètrables qu'il faut mettre en oeuvre, que de critères stricts.

Nous avons donc isolé toutes ces procédures sensibles, afin de pouvoir les ré-écrire par la suite dans un langage plus proche de ceux utilisés en intelligence artificielle.

Néanmoins, de bons résultats ont été obtenus dès à présent avec un jeu de valeurs et de coefficients ajustés petit à petit. On peut par contre espèrer diminuer le temps nécessaire à la génération de l'architecture en employant des méthodes d'intelligence artificielle pour classer les listes.

7-5-3 La liste des opérateurs

7-5-3-1 Introduction

Une fois la liste des tâches activées établie et classée, IMHOTEP va prendre la première tâche et lui affecter un opérateur. Le problème du choix d'un tel opérateur parmi tous ceux disponibles dans la liste des candidats se pose donc. En effet, si tous les opérateurs de la liste des candidats sont adaptés à réaliser la tâche choisie, certains peuvent être occupés sur d'autres tâches, où bien libres mais pas à temps. Il convient donc de pondérer le critère d'adaptation globale %G en fonction d'autres considérations afin de réaliser une liste d'opérateurs à essayer dans un certain ordre. Cette liste est différente suivant que la tâche est une opération de type bus ou non, mais ses éléments sont de toutes façons des RECORD qui contiennent :

- le numéro de l'opérateur
- un pointeur sur l'opérateur
- les critères %S et %G
- la marge temporelle de l'opérateur
- la priorité dans la liste

7-5-3-2 Constitution de la liste des opérateurs

La priorité de chaque opérateur de la liste est déterminée par plusieurs facteurs qui sont :

- être un opérateur de type registre ou non
- la tâche est SUSPENS ou non
- l'opérateur est libre ou non
- être un opérateur registre maître-esclave ou simple-latch

La combinaison de tous ces facteurs détermine la priorité donnée à l'opérateur concerné. Nous retrouvons donc la constatation, déjà énoncée plus haut, que la liste est établie en suivant des recettes pouvant changer en fonction de la famille de circuits considérée, et que des méthodes non algorithmiques conviendraient probablement mieux.

La liste des opérateurs est ensuite classée par priorités décroissantes et rendue à IMHOTEP.

7-5-3-3 Constitution de la liste des bus

Bien que les bus soient structurellement considérés comme des opérateurs au même titre que les multiplieurs, par exemple, la liste des bus-physiques s'établit sur des critères différents.

Le cas du bus portant déjà la donnée est bien entendu exploité, mais on distingue aussi certains cas susceptibles d'améliorer la connectique (diminution du nombre de multiplexeurs) pour calculer la priorité de chaque bus dans la liste.

La liste des bus est classée par priorités décroissantes et rendue à IMHOTEP.

7-5-3-4 Détermination du nombre d'opérateurs à essayer

Chaque tâche ayant un ou plusieurs opérateurs à sa disposition, le nombre total d'architectures générées en essayant chaque cas n'est pas en rapport avec les temps de calcul admissibles. Il convient donc de limiter, pour chaque tâche, le nombre d'opérateurs à essayer dans la liste.

Imposer un nombre maximum fixe et indépendant de la tâche considéré est une manière trop brutale pour être efficace. Nous avons gardé ce nombre maximum, fixé par défaut ou par l'utilisateur, mais nous avons ajouté une fonction pouvant le minorer pour tirer parti des particularités des listes d'opérateurs rencontrées. Quatre seuils ont été fixés, deux seuils hauts et deux seuils bas, un pour les opérateurs et un pour les bus. La fonction en question compare les critères des opérateurs ou bus de la liste avec les seuils qui lui sont fournis, et décide de ne garder que certains opérateurs qui feront l'objet d'un essai.

A titre d'exemple, si deux opérateurs ont un critère supérieur au seuil haut, on ne gardera qu'eux dans la liste.

7-5-4 Conclusion

Ce chapitre a exposé les principes dont se sert IMHOTEP dans son élaboration de l'architecture. Ces procédures se basent sur des indications (critères d'adaptation, marges, configurations spéciales) et les comparent à des seuils pour en déduire des choix qui guideront la génération d'architecture.

Leur caractère peu algorithmique mais proche des raisonnements utilisés en intelligence artificielle, nous a poussé à les regrouper dans un même module, et à leur consacrer un chapitre entier, afin de les isoler du reste de IMHOTEP. Il pourra être souhaitable de les modifier ou les rendre automatiquement modifiables en fonction de telle ou telle classe d'architecture visée. Cette modification se fera manuellement en ajustant certains coefficients ou de manière automatique par d'autres voies.

PLAN DU CHAPITRE 7.6

7-6- Mécanismes d'élaboration de l'architecture

7-6-1 Introduction

Le chapitre précédent a développé la manière dont étaient prises les décisions concernant quelle tâche allait être traitée d'une part, et quel opérateur allait lui être affecté d'autre part. Ces procédures constituent la "partie opérative" d'IMHOTEP, qui opèrent sous le "contrôle" des procédures qui ont fait l'objet du chapitre précédent.

Nous allons tout d'abord donner quelques définitions de termes utilisés plus loin, mais dont certains, comme le terme SUSPENS, ont déjà été évoqués plus haut.

7-6-2 Définition des états pour les tâches et les opérateurs

Le terme "tâche" désigne une étape du graphe représentant le circuit, et vaut aussi bien pour une opération que pour un bus. De même, le terme "opérateur" désigne également un additionneur ou un bus-opérateur comme nous l'avons défini au § 7-2-5.

7-6-2-1 Définitions concernant les tâches

- Tâches "A FAIRE" (fig 7.12).

Ces tâches ont deux caractéristiques : d'une part aucun opérateur ne les a encore réalisées ; et d'autre part, aucune de leurs entrées n'ont reçu de latches. En effet, une tâche de type latch est insérée entre le bus et la tâche sur l'entrée voulue, quand le besoin s'en fait sentir. La présence du latch dans le graphe signifie que, dans le futur circuit, une action de mémorisation réelle existera, à cet instant, sur l'entrée de l'opérateur qui réalisera cette tâche.

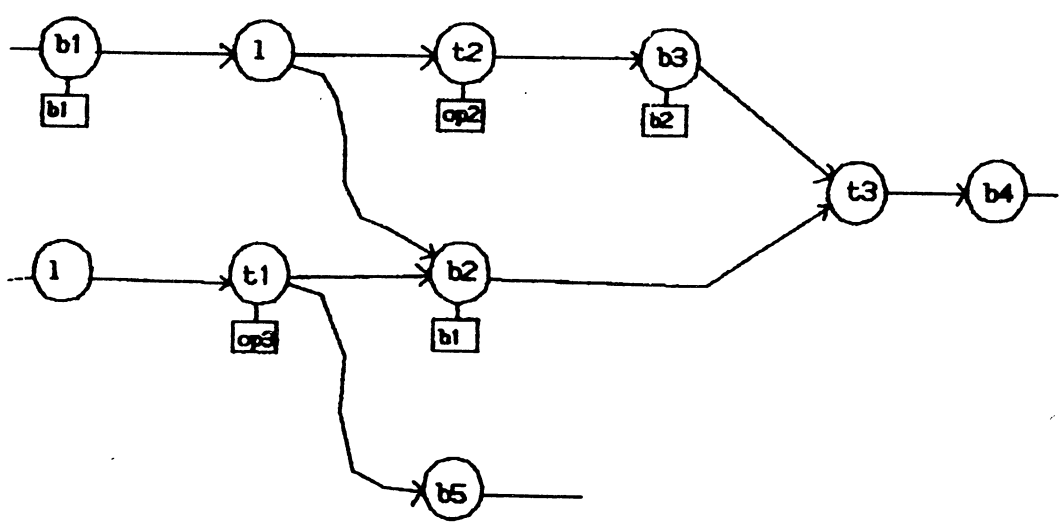


Fig : 7.12

- Taches "SUSPENS" (fig 7.13)

Ces tâches ne sont encore réalisées par aucun opérateur, et possèdent au moins un latch sur une de leurs entrées. ce latch (L) a DEJA libéré le bus-opérateur (B) qui réalise le bus-tâche (b_1) le précédant. Plus encore, ce bus-opérateur a déjà été REUTILISE ailleurs comme l'indique la flèche issue du latch et allant vers le bus-tâche utilisateur (b2). Nous apercevons ici la nature du PARI que fait IMHOTEP lorsqu'il déclare une tâche SUSPENS : il suppose que l'opérateur qui réalisera la tâche SUSPENS sera présent à temps pour assurer le latchage de la donnée présente sur le bus, autorisant ainsi non seulement la libération de ce bus, mais aussi sa réutilisation A UNE DATE BIEN PRECISE. Si l'opérateur n'est pas capable, à cette date précise, de réaliser la tâche SUSPENS, le pari est perdu, et IMHOTEP aura une stratégie de dégagement expliquée plus loin.

L'existence des tâches SUSPENS, qui sont uniquement de type opération, est due au fait suivant : IMHOTEP a souvent besoin de libérer un bus pour le réutiliser ailleurs, sans attendre que la tâche destinataire de la donnée du bus soit nantie d'un opérateur.

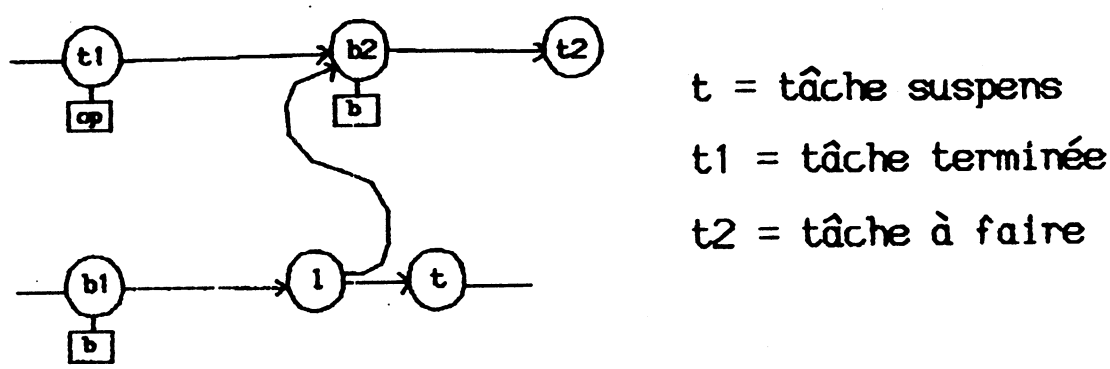


Fig : 7.13

- Tâches "TERMINEES".

Un opérateur a été affecté à ces tâches. Elles ont donc une durée précise, et non plus probable comme c'était le cas pour les autres catégories.

De plus, deux uniformisations ont été faites entre cette tâche et son opérateur : la première consiste à établir la correspondance entre les entrées logiques de la tâche et les entrées physiques de l'opérateur. Ceci prend tout son sens dans le cas de tâches à plusieurs entrées. La deuxième est d'uniformiser les latches entre la tache et l'opérateur, et sera développée plus loin.

7-6-2-2 Définitions concernant les opérateurs

- Opérateur "LIBRE" (fig 7.14).

Un opérateur est considéré comme libre dans deux cas : soit il n'a jamais réalisé de tâche, soit tous les chemins issus de la dernière tâche qu'il a réalisé aboutissent à un dispositif de mémorisation (latch, registre, sortie).

Ceci nous montre que, pour qu'un opérateur soit libérable, il faut que tous les bus issus de la dernière tâche qu'il a réalisé soient affectés. En effet, il ne peut y avoir de latch devant un bus.

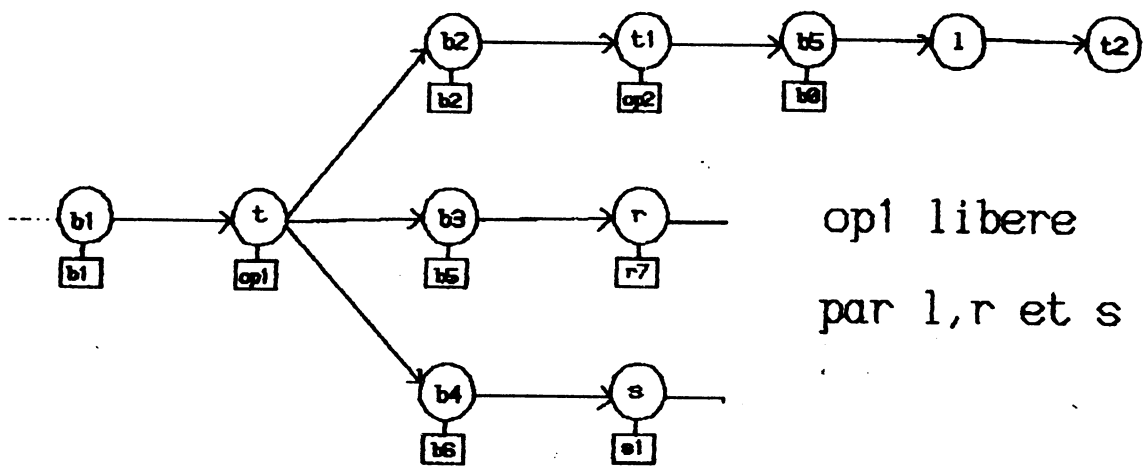


Fig : 7.14

- opérateur "OCCUPE" (fig 7.15).

C'est simplement le contraire d'un opérateur libre. Remarquons que, sur les chemins issus de la tâche occupant l'opérateur, l'on peut rencontrer des tâches terminées. Si pourtant aucun élément de mémorisation ne s'intercalle à un moment derrière une tâche terminée, l'opérateur restera occupé.

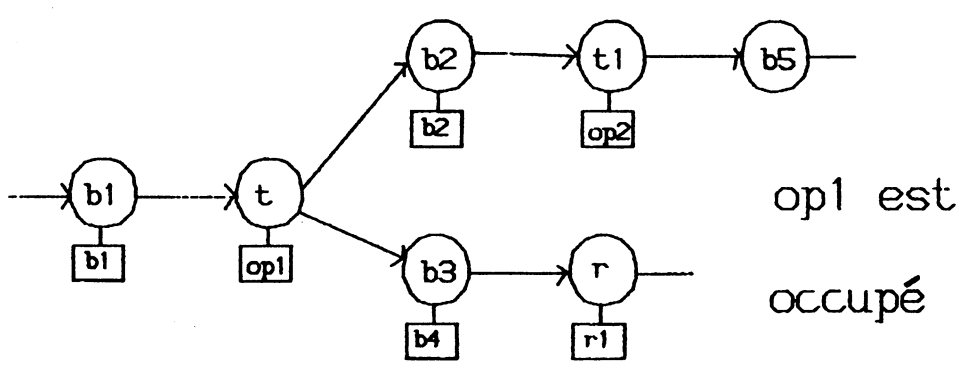


Fig : 7.15

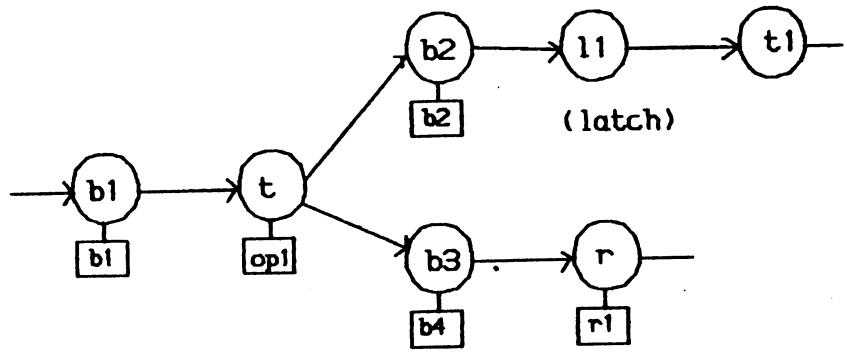
7-6-3 Les mécanismes d'affectation

L'affectation d'un opérateur à une tâche met en jeu un certain nombre de mécanismes. Ils varient légèrement selon le type de la tâche concernée, c'est pourquoi nous devons expliciter les différents cas, quitte à n'expliquer qu'une seule fois les procédures communes.

7-6-3-1 Affectation d'un opérateur

- Cas d'une tache non SUSPENS.

L'opérateur proposé est libre ou libérable : dans le deuxième cas, il faut d'abord le libérer en créant des latchs à la suite des bus, sortant de la tâche qui l'occupe (fig 7.16).



l1 crée pour libérer op1

Fig : 7.16

Dans un deuxième temps, IMHOTEP met les liens de succession entre ce qui libère l'opérateur et la tâche qui va l'utiliser. Ces flèches de succession permettent de calculer les dates au plus tard et au plus tôt de chaque tâche, en prenant en compte le multiplexage des opérateurs (fig 7.17).

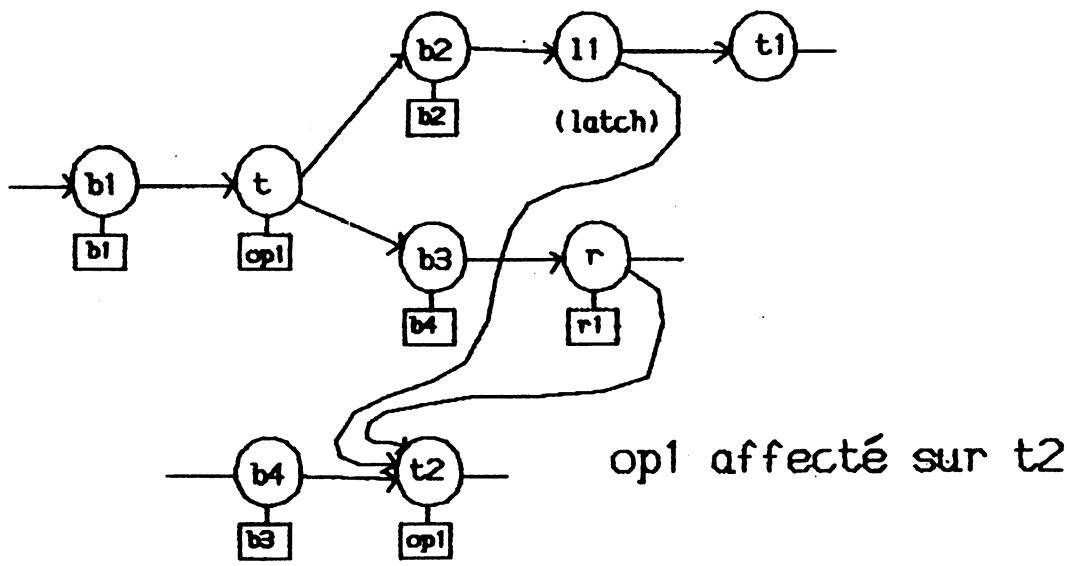


Fig : 7.17

Nous constatons donc que, au fur et à mesure de l'avancement de l'architecture, le graphe va s'étoffer en prenant en compte le multiplexage.

En troisième lieu, IMHOTEP met à jour les tables TRAVAIL et TEMOIN de la tâche et de l'opérateur, en rajoutant la nouvelle PREUVE à celle-ci. Le tableau de GANTT est ré-actualisé, et les critères %G des opérateurs bus et des opérateurs du même type que celui de la tâche affectée sont recalculés et reclassés dans les listes de candidats.

Enfin, IMHOTEP va uniformiser les latchs entre les tâches faites par l'opérateur choisi. En effet, si au moins une tâche faite par l'opérateur est latchée, cela suppose qu'un latch est physiquement présent sur l'opérateur. En conséquence, toutes les tâches faites par l'opérateur profiteront de ce latch qui pourra servir à libérer des opérateurs occupés sur les tâches amonts de celle-ci.

Ceci étant terminé, IMHOTEP passe à la tâche suivante en recalculant toutes les listes.

- Cas d'une tâche SUSPENS

Les tâches SUSPENS, tant par leur apparition que par leur traitement, constituent une originalité d'IMHOTEP. Elles sont la conséquence d'un pari, qui, réussi, améliore l'efficacité de l'architecture, et qui, perdu, offre tout de même une sortie de secours raisonnable.

Si un opérateur est libre avant la date au plus tard de la tâche, le pari est réussi et l'affectation se fait comme dans le cas des tâches non SUSPENS.

Si aucun opérateur ne présente cette caractéristique, la liste des opérateurs à essayer est vide. Il faut alors adopter une solution de secours : créer un registre simple latch entre le bus amenant la donnée critique et l'entrée atteinte sur l'opération.

Cette stratégie de dégagement revient à dire : puisqu'aucun opérateur ne sera disponible pour stocker telle donnée avant une date déterminée (contrairement à l'hypothèse initiale), nous créeons un registre simple latch spécial pour le faire à la place de l'opérateur. (fig 7.18).

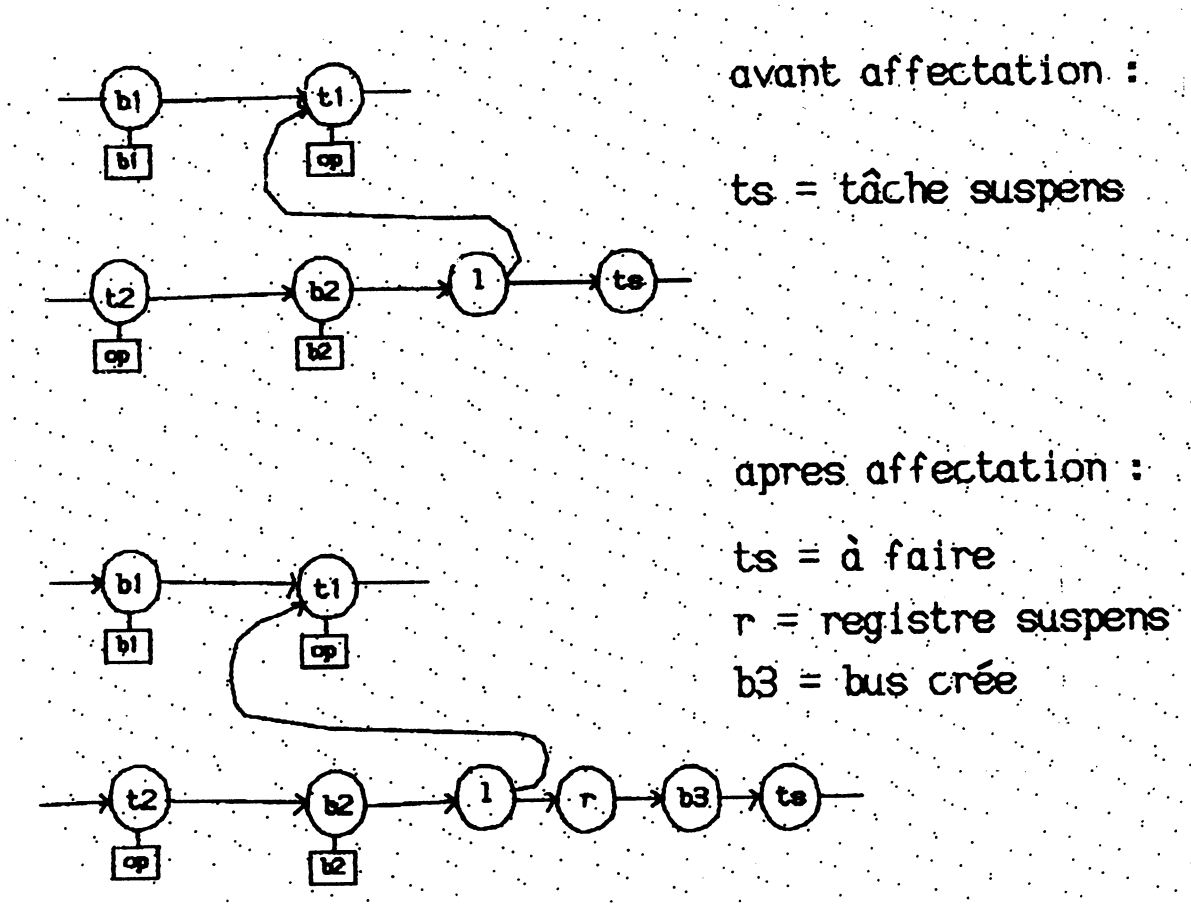


Fig : 7.18

IMHOTEP crée donc deux tâches : une tâche registre et une tâche bus, puis passe à l'étape suivante en recalculant toutes les listes.

7-6-3-2 Affectation d'un registre

- Cas du registre fonctionnel (Maitre-esclave).

Ce registre possède un opérateur réservé puisqu'il y a un opérateur registre ME par tâche registre ME dans le graphe. Il suffit donc soit de prendre le premier de la liste si l'autre extrémité du registre n'est pas terminée, soit de prendre la deuxième moitié du registre ME dont la première moitié a déjà été utilisée.

Puis on met à jour les champs du TRAVAIL correspondant, et on crée une PREUVE supplémentaire pour le TEMOIN de l'opérateur.

Le tableau de GANTT du circuit est recalculé, de même que les critères %G de tous les bus du circuits. En effet, un bus se trouve connecté à un nouveau registre, ce qui peut modifier son critère %G, ainsi que celui de certains autres bus.

A la suite du calcul des critères, les listes de candidats sont reclassées pour toutes les opérations bus.

L'affectation est alors terminée.

- Cas du registre simple-latch.

Ce genre de registre est obligatoirement SUSPENS, suite à son mode d'apparition. Il faut donc dans un premier temps passiver son latch antécédent. A la suite de quoi, la liste des opérateurs à essayer est examinée, et, comme elle peut être vide, deux cas se présentent :

Si la liste est vide, IMHOTEP demande à LOF un opérateur de type registre simple latch. Sinon IMHOTEP affecte le registre disponible.

L'affectation d'un registre n'appellant pas de commentaires particuliers et se faisant comme pour toute opération SUSPENS, précisons plutôt la demande à LOF d'un nouvel opérateur.

Le registre étant latché, il doit pouvoir stocker la donnée au moins aussi vite que le latch qui le précède. En effet, le latch étant passivé à l'affectation, les flèches de succession issues du latch seront reportées sur le registre (fig 7.19). Si d'aventure, la tâche registre durait plus longtemps que la tâche latch, les dates du graphe pourraient être retardées et ceci pourrait nuire à l'optimisation globale.

IMHOTEP crée donc deux tâches : une tâche registre et une tâche bus, puis passe à l'étape suivante en recalculant toutes les listes.

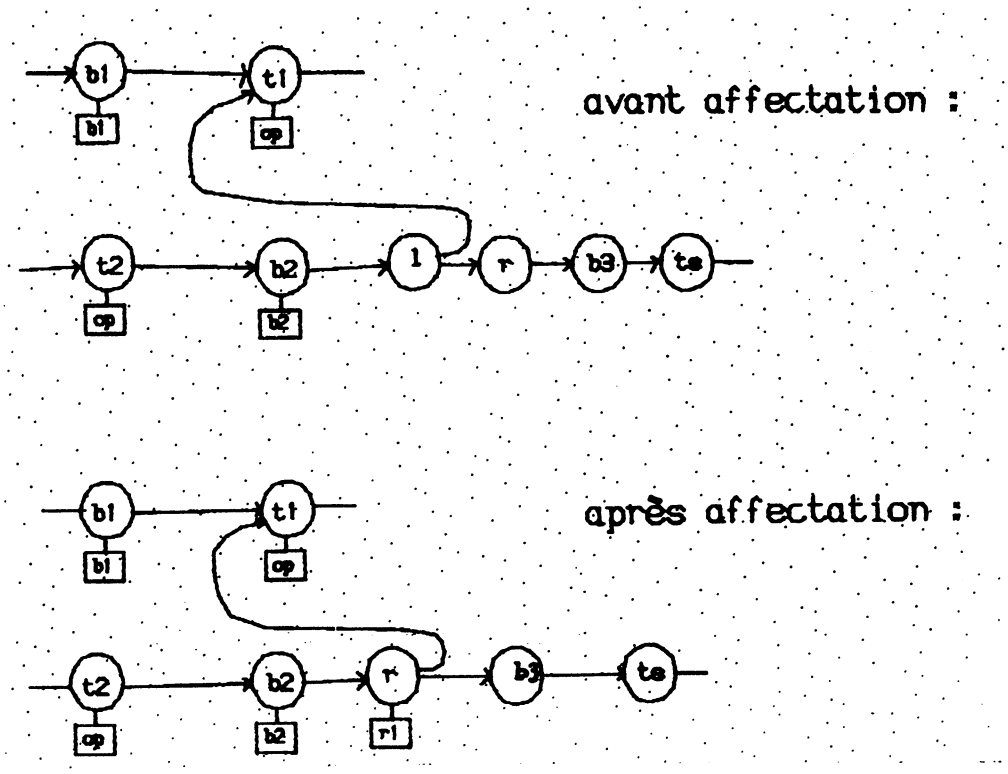


Fig : 7.19

7-6-3-3 Affectation d'un bus

- Cas du bus opérateur portant déjà la donnée.

Dans ce cas, IMHOTEP va supprimer du graphe la tâche bus en question, et reporter ses successeurs sur la tâche bus occupant actuellement l'opérateur et portant la même donnée (fig 7.20).

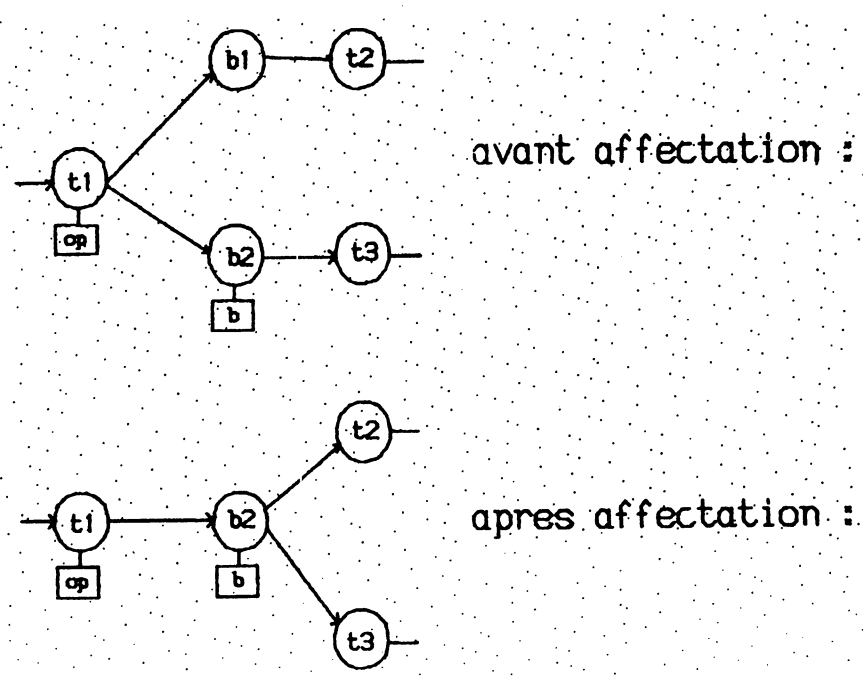


Fig : 7.20

A la suite de quoi, IMHOTEP passe à l'étape suivante.

- Cas du bus opérateur non porteur.

Le bus opérateur est éventuellement libéré, puis affecté à la tâche en mettant les flèches de succession appropriées. Le tableau de GANTT est recalculé, et les critères de tous les opérateurs sont recalculés, et les listes de candidats classées.

7-6-3-4 Conclusion

Les mécanismes d'affectation comprennent toujours un noyau commun :

- libération éventuelle de l'opérateur
- création des flèches d'affectation
- mise à jour du TRAVAIL de la tâche
- création de la nouvelle PREUVE et mise à jour du TEMOIN
- calcul du tableau de GANTT
- recalcul de certains critères et classement de la liste des candidats

Pour certains opérateurs, des mécanismes particuliers pouvant s'ajouter ont été recensés dans les rubriques précédentes.

7-6-4 La mise en correspondance des passages

Dans le cas des opérateurs à plusieurs entrées, réalisant plusieurs tâches, le problème de mise en correspondance des entrées logiques des tâches avec les entrées physiques de l'opérateur se pose.

Pour certaines configurations, il n'y a pas le choix ; nous pensons notamment au cas où les nombres de bits sont différents sur les entrées, ce qui limite le nombre des possibilités de permutation.

Les circonstances présentant des ambiguïtés font l'objet d'un traitement spécial que nous allons exposer.

Pour départager plusieurs associations (entrée logique, entrée physique), nous avons considéré deux critères : le premier est de prendre en compte les antécédents existants sur les entrées ; ce sera un critère dominant. Le deuxième est de tenir compte du fait que les entrées sont latchées ou non ; ce sera un critère récessif.

Nous établissons donc les correspondances entre les entrées et sorties de la tâche considérée et les entrées-sorties de l'opérateur qui la réalise. Pour cela, les tâches déjà réalisées par l'opérateur, et dont les correspondances E/S ont été précisées (elles peuvent rester indifférentes), sont comparées une par une à la tâche courante.

Si une affinité forte ou une affinité faible entre E/S de la tâche et E/S de l'opérateur se dégage de ces comparaisons, la correspondance est établie sous forme d'un tableau. Dans le cas contraire, la correspondance est laissée indifférente.

A chaque fois qu'une mise en correspondance est tentée et réussie, on essaye à nouveau d'établir les correspondances pour toutes les tâches faites par cet opérateur et laissées indifférentes jusqu'alors.

En effet, une nouvelle correspondance établie peut permettre l'apparition d'une préférence là ou il n'y avait qu'indétermination.

7-6-5 Conclusion

Dans ce chapitre, nous avons montré comment IMHOTEP mettait en oeuvre les décisions prises dans les listes. Les procédures utilisées assurent une affectation progressive et totale des tâches du graphe sans possibilité de dead-lock. En effet, un opérateur-bus est toujours libérable, donc une tâche bus sera toujours affectable. Il reste donc à montrer qu'une opération ne peut se trouver indéfiniment sans opérateur : si c'est une tâche SUSPENS, la création d'un registre viendra pallier au manque d'opérateur ; et si la tâche est non suspens, l'opérateur sera libérable dès que les bus de sortie de la tâche qui l'occupe seront affectés.

Le seul cas dangereux réside dans la création d'une succession de registres à la file devant une tâche SUSPENS qui n'arriverait pas à avoir d'opérateurs. Ce cas est détecté, et l'organisation du programme IMHOTEP, qui sera expliqué au chapitre suivant, permet aisément de supprimer ce comportement aberrant.

7-7- Organisation du programme IMHOTEP

7-7-1 Introduction

Le dénombrement des solutions possibles pour l'ordonnancement des tâches sur les opérateurs nous a montré que les possibilités étaient très nombreuses.

Par ailleurs, un certain nombre d'opérateurs étant candidats pour chaque opérations, il convient de les essayer successivement. Le cheminement adopté lors de la recherche des solutions est représentable par un arbre dont les feuilles sont les solutions et la racine le début du traitement (fig 7.21). Chaque noeud de l'arbre représente le traitement d'une tâche, et il est à noter que tous les noeuds de même niveau ne concernent pas forcément la même tâche.

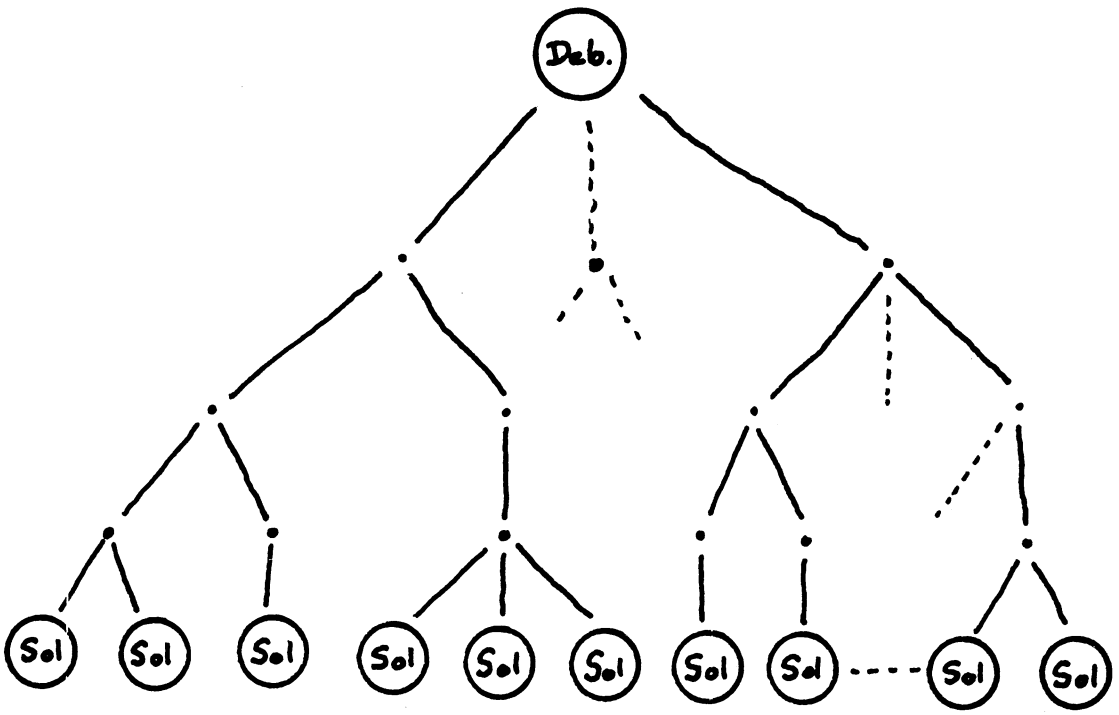


Fig : 7.21

7-7-2 Gestion du "back-tracking"

La gestion du back-tracking se fait grâce à une procédure récursive qui s'appelle elle-même à chaque noeud de l'arbre. Le premier problème qui se posa alors, fut de décider de la manière dont seraient gérées les modifications dans la structure de données. En effet, chaque affectation d'un opérateur à une tâche laisse des traces dans la structure de donnée (cf 7-4 et 7-6).

Deux solutions se sont présentées à nous, l'une agissant en mode commun, et l'autre en mode différentiel.

Méthode du mode commun

La solution dite de mode commun consistait à dupliquer, à chaque appel de la procédure, l'ensemble de la structure de donnée ; puis à continuer la descente dans l'arbre avec cette structure jumelle.

Lors de la remontée, il suffisait dès lors de supprimer à chaque noeud la structure jumelle pour remonter avec la structure originelle qui avait servi pour la descente. Cette solution était pénalisante quant au volume des données, car pour un arbre ayant N niveaux, le volume des données présentes en mémoire au niveau le plus bas était donc N fois celui que l'on avait au niveau le plus haut. Ceci est d'ailleurs une approximation minorante car le volume des données manipulées croit avec le nombre de tâches affectées dans le graphe.

De plus, cette solution demandait la duplication de la structure du niveau appelant, ce qui est une action pénible eut égard au fait que la structure est totalement dynamique et utilise au maximum les pointeurs PASCAL.

Cette solution aboutissait à un traitement considérablement ralenti par les actions de pagination mémoire, même pour des circuits petits, et n'a pas été retenue.

Méthode de mode différentiel

Cette deuxième méthode consiste à ne mémoriser, à chaque noeud de l'arbre, que les modifications apportées à la structure. Elle a donc l'avantage de minimiser le volume des données, par rapport à l'autre solution.

La procédure s'organise donc ainsi :

```
PROCEDURE MODE-DIF ;
BEGIN
DECISION SUR LE TRAITEMENT A EFFECTUER ;
TRAITEMENT ;
STOCKAGE DES MODIFICATIONS DE LA STRUCTURE ;
MODE-DIF ; (appel suivant)
REMISE EN ETAT DE LA STRUCTURE ;
END ;
```

Cette méthode, bien que pénalisée par la nécessité de remettre en état la structure de données, est beaucoup plus dense et rapide que la précédente.

7-7-3 Ajustement dynamique de la taille de l'arbre

La taille de l'arbre et le nombre des solutions possibles dépendent du nombre de branches issues de chaque noeud. Ce nombre de branches n'est pas constant pour tous les noeuds car il correspond au nombre d'opérateurs à essayer pour la tâche dont s'occupe le noeud considéré. La "DECISION SUR LE TRAITEMENT A EFFECTUER" que comprend MODE-DIF consiste donc à décider : quelle tâche va être traitée d'une part, et quels opérateurs vont être essayés d'autre part. Ces décisions sont prises par des procédures dont le fonctionnement a été décrit au chapitre 7-5.

7-7-4 Mécanismes d'abandon des branches non optimales

Le nombre des solutions possibles étant particulièrement élevé pour certains circuits, les performances de l'algorithme sont très dépendantes de sa capacité à détecter tôt les branches conduisant à des solutions non optimales. La définition d'une solution non optimale est la suivante : soit elle est moins bonne que la meilleure des solutions déjà trouvées, soit elle contient une "aberration architecturale" qui rend inutile la poursuite de son traitement.

IMHOTEP possède quatre critères lui permettant d'interrompre l'exploration d'une branche de l'arbre, mais seuls trois d'entre eux sont réellement des critères architecturaux.

7-7-4-1 Le critère "contrainte de temps"

L'architecture désirée devant satisfaire une contrainte de temps, il est possible d'évaluer le temps de fonctionnement d'une architecture en cours d'élaboration. Cette estimation sera d'autant plus précise que le degré d'achèvement de l'architecture sera élevé, et cette estimation sera toujours optimiste afin de ne pas écarter de bonnes architectures.

La manière d'évaluer le temps de fonctionnement du circuit est la suivante :

En annulant la durée des tâches non pourvues d'opérateurs, la durée de fonctionnement du circuit est la durée exacte de fonctionnement des seules tâches affectées. Il faut donc ajouter à cela une estimation du temps de fonctionnement des tâches non affectées pour obtenir celui du circuit entier.

Cette deuxième estimation est difficile dans la mesure où l'on ne sait pas ce que sera le multiplexage pour les tâches non affectées. Cela dépend en effet du type des tâches restant à faire et du nombre d'opérateurs de chaque catégorie disponibles.

Nous avons donc eu recours à l'approximation suivante. Nous considèrons que les multiplications sont les tâches prépondérantes au point de vue durée, et nous négligeons les autres tâches. Le problème de l'estimation du temps de fonctionnement revient alors à calculer combien de temps faut-il pour faire M multiplications avec N multiplieurs de caractéristiques données. Ce mode de calcul nous donne une estimation globale satisfaisante.

Dès que l'exploration d'une branche produit une estimation trop lente à un noeud de l'arbre, cette branche est abandonnée.

7-7-4-2 Le critère "vraissemblance architecturale"

Les aléas de l'ordonnancement, bien que maîtrisés localement, peuvent conduire à des aberrations architecturales. La seule qui soit détectée est le cas où se crée une chaîne de registres "simple latch" (fig 7.22). Ceci est du au fait que l'opérateur qui aurait du se trouver à la place du registre aval n'a pas été disponible à temps, et a du être remplacé par un registre simple-latch. Dans ce cas, la branche correspondante est abandonnée et le registre amont est marqué. Quand l'algorithme reviendra au noeud où a été affecté le registre amont, une tentative sera faite pour s'occuper non pas du registre, mais de la tâche qui le suit dans la liste hiérarchique des tâches à satisfaire.

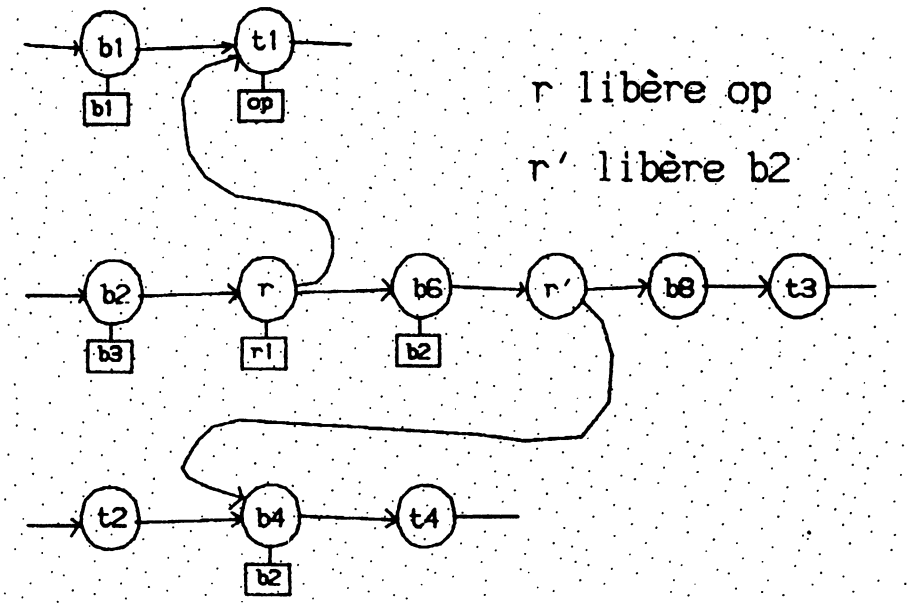


Fig : 7.22

7-7-4-3 Le critère "temps CPU maximum"

Ce critère n'est pas architectural. Il permet d'interrompre proprement l'exploration de l'arbre quand le temps CPU imparti est écoulé. Le programme a ainsi le temps de remonter au noeud le plus élevé et de créer les fichiers de résultats, sans être stoppé brutalement par une action système extérieure.

7-7-4-4 Le critère "surface minimale"

Lors de la découverte de la première solution, IMHOTEP fait une estimation fine de la surface obtenue (cf 7-8). Puis, lors de la recherche des autres solutions, à chaque noeud de l'arbre, une estimation de surface grossière est faite (somme des surfaces des opérateurs utilisés). Cette estimation est comparée à la meilleure surface fine trouvée, et, en cas de comparaison défavorable, la branche est abandonnée. Remarquons que cette méthode ne permet pas d'interrompre l'élaboration d'une solution qui utiliserait les mêmes opérateurs que la solution optimale ; en effet le calcul fin de la surface tient compte des multiplexeurs et des bus, et son résultat sera toujours supérieur à celui de l'estimation grossière. Mais dans le cas où les opérateurs sont différents (cas d'un opérateur jamais utilisé), cette méthode permet de couper certaines branches.

7-7-5 Algorithme de IMHOTEP

Nous donnons cet algorithme à ce stade de l'exposé, afin de permettre au lecteur de situer les nombreuses notions données auparavant. Certaines données mineures ne seront développées qu'aux chapitres suivants, mais l'essentiel permettant de comprendre les mécanismes d'ordonnancement a déjà été expliqué.

Tout ce qui concerne la mise en forme et la gestion des solutions est regroupé au chapitre suivant.

```
IMHOTEP (Partie ordonnancement)

BEGIN
Evaluation du temps de fonctionnement du circuit = DATE_FIN;
IF (DATE_FIN < TEMPS_MAX)           AND
   ((SURF_CUM < SURF_BEST) OR (SURF_BEST = 0))  AND
   (FLAG_CONTINUER)                      THEN
 BEGIN
 Etablissement de la liste hiérarchique des taches à traiter;
 WHILE on doit explorer la liste DO
  BEGIN
  On prend une tache dans la liste;
  IF TACHE <> tache de fin de graphe THEN
   BEGIN
   Etablissement de la liste des opérateurs à essayer;
   ASSEZ_ESSAI:=Nombre maxi d'opérateurs à essayer;
   ESSAI_COURANT:=Numéro de l'operateur en cours d'essai;
   OPER_RETENU:=Nombre d'operateurs déjà essayés et retenus;
   IF ESSAI_COURANT <> fin de liste THEN
     REPEAT
     On affecte cet opérateur à cette tache;
     IF pas de boucles détectées THEN
      BEGIN
      Mise à jour de la structure de données;
      IMHOTEP; (on relance la procedure)
      Restauration de la structure en l'état initial;
      END
     On désaffecte l'opérateur de la tache;
     OPER_RETENU:=OPER_RETENU+1;
     ESSAI_COURANT:=Opérateur suivant dans la liste;
     IF (OPER_RETENU > ASSEZ_ESSAI)    OR
        (ESSAI_COURANT = FIN DE LISTE) OR
         NOT FLAG_CONTINUER            THEN STOP_ESSAI:=TRUE
                                       ELSE STOP_ESSAI:=FALSE;
     UNTIL STOP_ESSAI;
   END;
  IF TACHE = tache de fin de graphe THEN
   BEGIN
   EVALUATION DE LA SURFACE DU CIRCUIT;
   IF la surface est meilleure que celle de la meilleure solution actuelle THEN
    IF Durée du circuit < TEMPS_MAX THEN
     BEGIN
     On génere et stocke la solution;
     IF (DUREE CPU ECOULEE) >= CPU_MAX THEN FLAG_CONTINUER:=FALSE;
     END;
   END;

  IF OPER_RETENU = 0 THEN
   IF la liste des taches à faire est vide THEN EXPLOR_LIST:=FALSE
                                           ELSE EXPLOR_LIST:=TRUE
                     ELSE
   BEGIN
   EXPLOR_LIST:=FALSE;
   IF la tache est un registre à l'origine d'une chaine de registres AND
      il reste une tache à faire dans la liste THEN EXPLOR_LISTE:=TRUE;
   END;
  IF NOT FLAG_CONTINUER THEN EXPLOR_LIST:=FALSE;
  END;
 END;
END;
```

PLAN DU CHAPITRE 7.8

7-8- Gestion des solutions architecturales trouvées pour le circuit

7-8-1 Introduction

Lors de ses déplacements dans l'arbre de recherche des solutions, IMHOTEP rencontre des noeuds sans successeurs, qui correspondent à des situations où toutes les tâches ont été affectées. Ces noeuds terminaux, ou feuilles, font l'objet d'un traitement particulier. Le graphe, à ce stade, contient toute la description de l'architecture avec parties opérative et contrôle.

Il convient donc d'évaluer la solution trouvée, et de la stocker si elle en vaut la peine, avant de remonter dans l'arbre. La description des traitements successifs que l'on effectue aux noeuds terminaux fait l'objet de ce paragraphe.

7-8-2 L'évaluation fine de la surface

Tout d'abord, les correspondances indéterminées entre entrées logiques des opérations et entrées physiques des opérateurs les réalisant sont précisées.

A la suite de quoi, IMHOTEP procède au calcul fin de la surface du circuit. Ce calcul doit tenir compte non seulement de la surface des opérateurs utilisés, mais également de la surface des interconnexions qui est difficile à évaluer. La surface des interconnexions comprend, suivant le modèle de bus que nous nous sommes donnés, la surface des nappes de fils, celle des 3-états d'accès au bus, et celle des multiplexeurs situés sur les entrées des opérateurs. Si les deux dernières surfaces sont faciles à estimer, celle des nappes de fils va dépendre du placement et du routage des opérateurs.

C'est à ce stade que l'utilité d'une architecture cible pour le plan de masse du circuit apparait pleinement. Elle va permettre de calculer plus précisement la surface des nappes de fils, ou tout au moins, permettre d'en trouver une estimation fiable.

Dans notre cas, le plan de masse utilisé disposera les opérateurs en tranches de bits, avec routage vertical entre les tranches, et passage horizontal des alimentations et des commandes (fig 7.23).

Les multiplexeurs et les 3-états accroîtront donc les dimensions du circuit dans le sens vertical, tandis que le nombre de bus augmentera la largeur du circuit.

Nous avons décidé, pour l'instant, de ne prendre en compte pour évaluer la surface des bus, que les multiplexeurs et 3-états. En effet, les algorithmes du générateur de plans de masse ne sont pas suffisamment stabilisés pour en tirer un estimateur rapide de la largeur des canaux de routage.

Ceci revient à dire que seule la surface transistorisée sera prise en compte pour départager les différentes architectures, la largeur des canaux de routage étant supposée à peu près constante quelquesoit l'architecture.

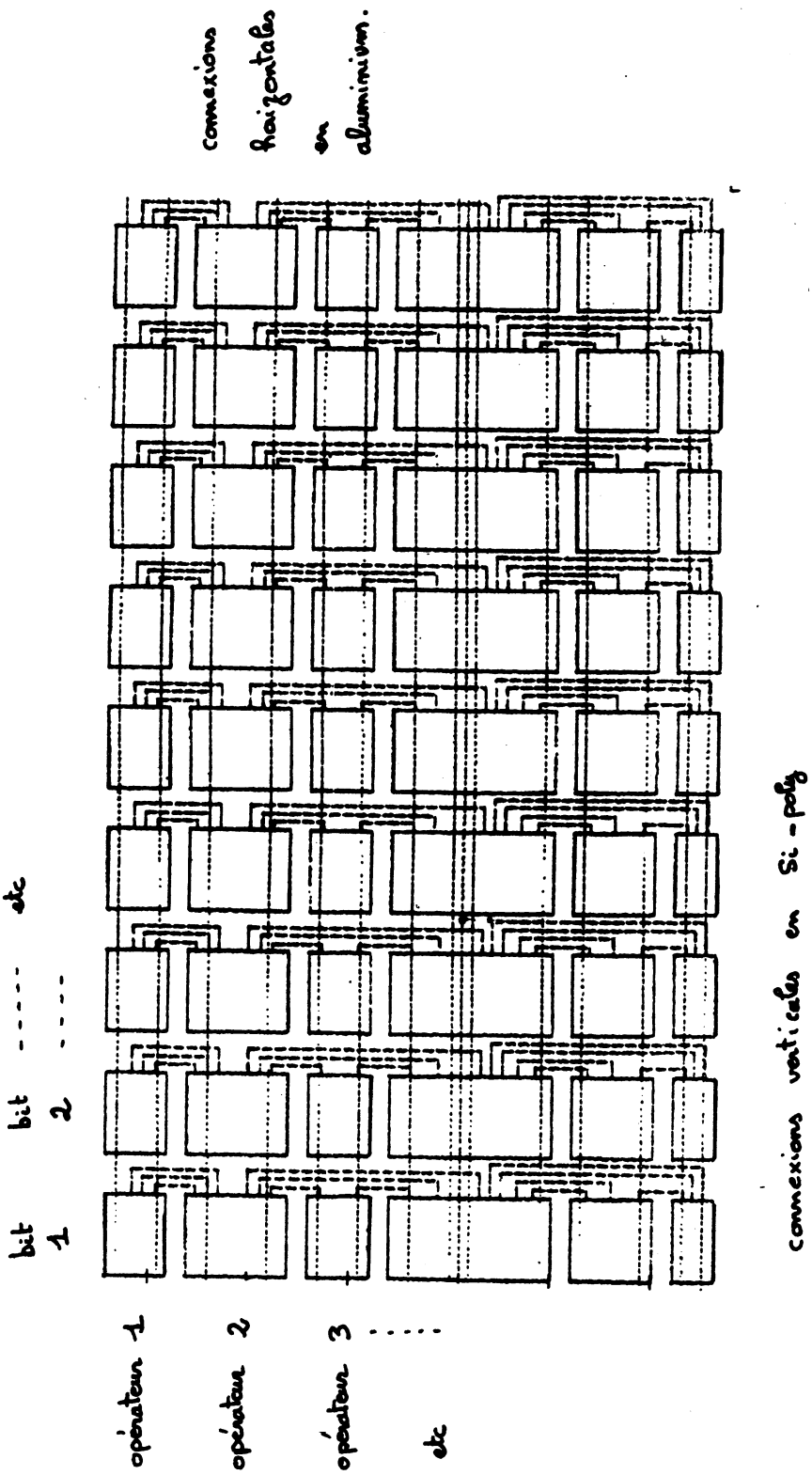

connexions horizontales en aluminium.
bit 1
bit 2 ----
etc
opérateur 1
opérateur 2
opérateur 3 ----
etc
connexions verticales en Si - poly

Fig : 7.23

Le chiffrage de la surface des 3-états d'accès aux bus se fait en comptant le nombre d'opérateurs arrivant sur ce bus. Si, il n'y en a qu'un seul, aucun 3-états n'est nécessaire ; sinon le circuit requiert un 3-états par antécédent de ce bus. La surface d'un 3-états sera demandée à LOF connaissant la capacité à charger et la rapidité de transfert requise pour ce bus. La capacité à charger est la somme des capacités des nappes de fils plus celles des entrées des opérateurs reliés sur le bus (fig 7.24).

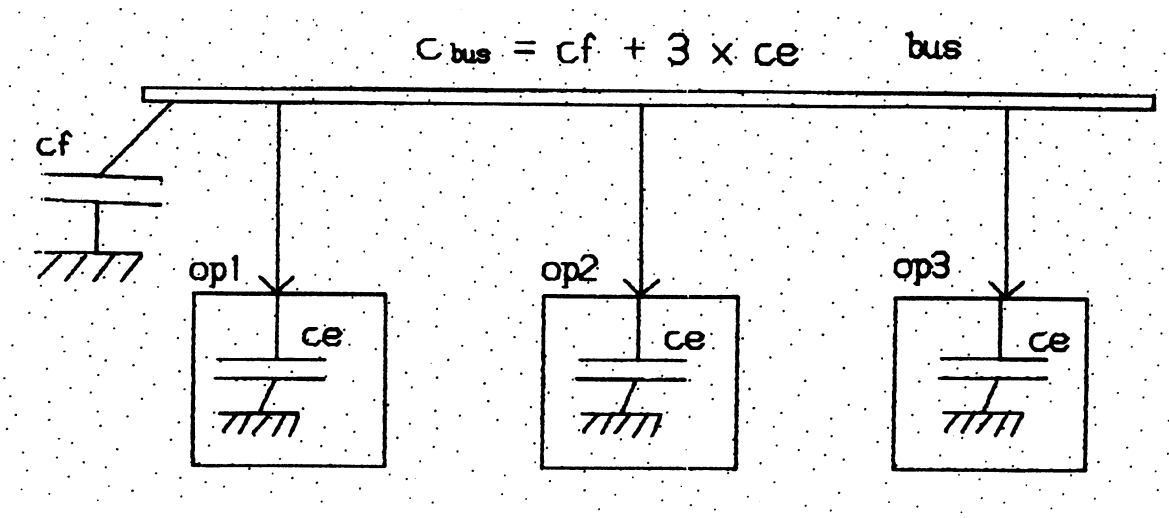


Fig : 7.24

En première approximation, nous négligerons la capacité de la nappe de fil elle-même dont l'estimation nécessiterait d'en connaître la longueur. Nous ne prendrons en compte que les capacités à l'entrée des opérateurs connectés au bus. Ces capacités étant elles-mêmes normalisées pour l'ensemble des opérateurs de la bibliothèque, seul compte le nombre d'opérateurs connectés au bus. IMHOTEP demandera donc à LOF quelle est la surface d'un trois-états capable d'assurer un temps de propagation, donné sur le bus chargé par n opérateurs.

L'estimation de la surface des multiplexeurs se fait en inspectant les opérateurs entrée par entrée, et en déterminant le nombre et la fonction des multiplexeurs nécessaires. A ce stade, pour chaque fonctionnalité différente (multiplexeur n vers 1) LOF fournit la surface correspondante.

La surface totale du circuit est ainsi estimée de manière assez précise. Seule la surface des canaux de routage est inconnue, mais elle ne semble pas pouvoir servir à départager différentes solutions entre elles dans la mesure où toutes les solutions utilisent le même nombre de bus.

Les exemples donnés au chapitre 6 illustrent ces propos.

Dès qu'une estimation de la surface du circuit est calculée, cette surface est comparée à la surface de la meilleure solution trouvée à ce jour. Si la comparaison est favorable, l'examen de cette solution peut continuer ; dans le cas contraire elle est abandonnée et IMHOTEP remonte d'un niveau dans l'arbre après avoir restauré le circuit.

7-8-3 Calcul du temps de fonctionnement du circuit

Le circuit est représenté par un graphe dont toutes les tâches ont une durée connue. Le temps de fonctionnement du circuit est donc la date au plus tôt de la dernière tâche qui est la tâche fictive FIN. Ce temps correspond à un fonctionnement asynchrone du circuit car la durée de chaque tâche est quelconque.

Pour calculer le temps synchrone du circuit séquencé par une horloge unique de période Th, il faut caler les commandes du circuit sur un front montant de l'horloge. Cela va modifier le temps de fonctionnement global du circuit. Nous constatons donc qu'il nous faut à ce stade extraire du graphe la description de la partie opérative du circuit, puis décrire son séquencement, et enfin synchroniser le circuit définitif.

7-8-3-1 Extraction de la partie opérative

La description de la partie opérative consiste en une liste d'opérateurs munis de leurs commandes propres, de leurs commandes de latch, de leur signal de RAZ, et de leurs interconnexions avec les bus disponibles (fig 7.25).

Cette description s'établit en scrutant le graphe, et, pour chaque opérateur, en regardant quelles tâches ont été réalisées. Ceci fournit les informations de connectique par l'intermédiaire des bus antécédents des tâches; et les informations de commandes de latches. Dans le cas des multiplieurs et des décaleurs généraux, la commande de sélection du coefficient est également calculable.

PARTIE OPERATIVE DU CIRCUIT

```
Opérateur numéro : 1 nommé ENTREE   d'espèce ENTREE

Commande propre numéro :          1
Commande de RAZ        :         53

Opérateur ayant          0 antécédents :
             et          1 successeurs :
- successeur numéro :       15
(*************************************************************************)
Opérateur numéro : 2 nommé SORTIE   d'espèce SORTIE

Commande propre numéro :          2
Commande de RAZ        :         53

Opérateur ayant          1 antécédents :
- antécédent numero :       13
             et          0 successeurs :
(*************************************************************************)
Opérateur numéro : 3 nommé MULT     d'espèce MULTIPLI

l'entrée :          1 est latchée avec la commande numéro :          3

Commande de RAZ :         53

Ce multiplieur général fait          3 multiplications
la multiplication numéro            22 avec la commande numéro :          4
la multiplication numéro            14 avec la commande numéro :          5
la multiplication numéro             5 avec la commande numéro :          6

Opérateur ayant :          2 antécédents sur l'entrée :          1
- antécédent numéro :        14
- antécédent numéro :        13
             et :          1 successeurs
- successeur numéro :        15
(*************************************************************************)
Opérateur numéro : 4 nommé ADDIT    d'espèce ADDITION

l'entrée :          1 est latchée avec la commande numéro :          7
l'entrée :          2 est latchée avec la commande numéro :          8

Commande de RAZ :         53

Opérateur ayant :          3 antécédents sur l'entrée :          1
- antécédent numéro :        15
- antécédent numéro :        13
- antécédent numéro :        14
Opérateur ayant :          3 antécédents sur l'entrée :          2
- antécédent numéro :        15
- antécédent numéro :        14
- antécédent numéro :        13
             et :          3 successeurs
- successeur numéro :        14
- successeur numéro :        13
- successeur numéro :        15
```

Fig : 7.25

7-8-3-2 Description de la partie contrôle

Le graphe décrivant cette partie contrôle est purement séquentiel (Fig 7.26) ,chaque étage ayant une liste de commandes, et une transition toujours vraie avec l'étape suivante.

Il existe plusieurs possibilités pour implémenter ce graphe. Nous utilisons pour l'instant un séquenceur mono-PLA dans lequel chaque étape est susceptible de durer plusieurs périodes d'horloge. La réalisation pouvait donc prendre deux formes, soit un PLA doté d'un compteur permettant de rester plusieurs périodes dans la même étape, soit un PLA où les étapes de durée supérieure à une période auraient été éclatées en autant d'étapes que de périodes. Nous avons pour l'instant choisi cette deuxième solution.

La solution mono-PLA peut toutefois s'avérer avoir un temps de calcul des commandes trop long par rapport à la période d'horloge. Nous pourrions envisager dans ce cas de séquencer le circuit par un registre à décalage associé à un demi PLA, étant donné l'aspect figé du séquencement. Ces choix étant reportés au niveau de la génération du plan de masse, nous ne ferons pas ici de définition plus précise du mode de fonctionnement de la partie controle.

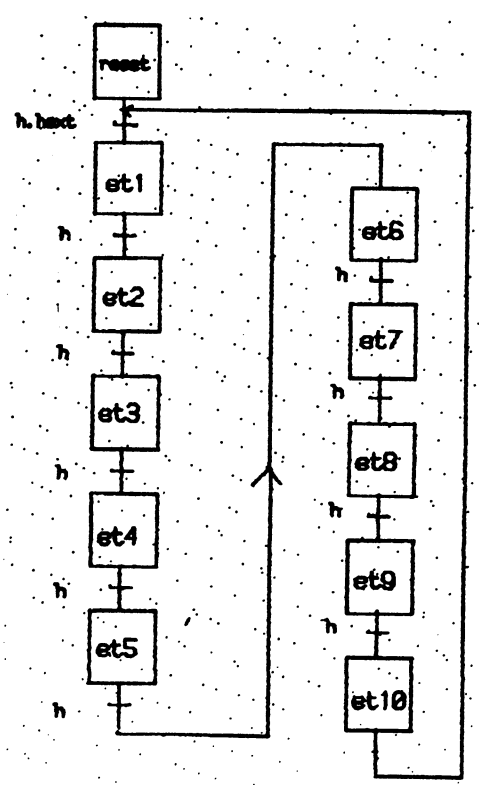


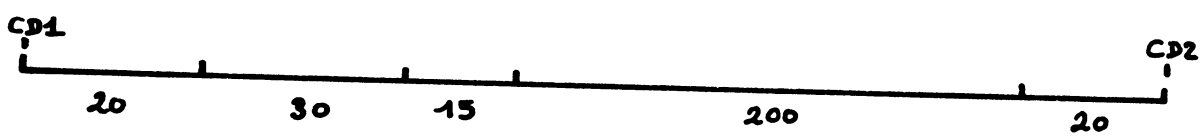
Fig : 7.26

L'extraction de la partie contrôle consiste dès lors à examiner, dans le graphe du circuit, les tâches possédant une commande. Les dates au plus tôt de ces tâches sont donc répertoriées ainsi que le nom de la commande et sa transition. Nous possèdons à l'issue de ce traitement une liste de commandes avec leur date et leur sens de transition ; cela constitue une description asynchrone du séquencement du circuit dans la mesure où les dates des commandes ne correspondent pas forcément à des multiples de la période d'horloge.

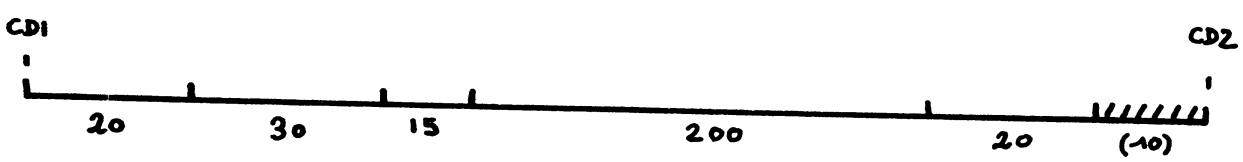
Le problème posé est alors de synchroniser cette description sur l'horloge disponible en perdant le moins de temps possible.

La meilleure méthode possible consiste à respecter le parallèlisme du circuit, c'est-à-dire à synchroniser chaque branche sans introduire de facto, de pénalités sur les branches qui lui sont parallèles. Il suffit dès lors de prendre les tâches possédant une commande dans l'ordre de leur rang dans le graphe, de synchroniser la première sur la période d'horloge juste suivante, et de recalculer TOUTES les dates du graphe avant de passer à la tâche suivante.

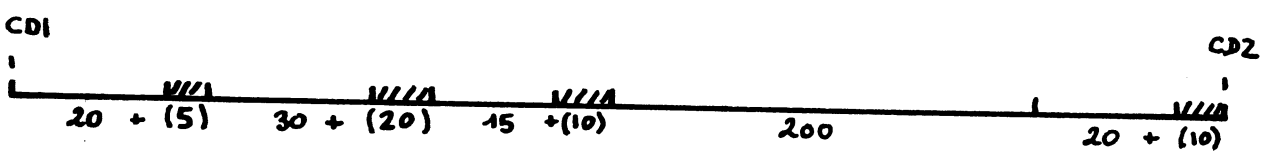
Cette méthode étant assez longue à mettre en oeuvre, nous avons adopté une deuxième méthode beaucoup plus rapide mais moins performante. Elle consiste à dire que les durées de tous les opérateurs sont multiples de la période d'horloge. Dès lors, les dates du graphe seront obligatoirement calées sur une période. En revanche, ceci augmente le temps de fonctionnement du circuit de la manière suivante : dans le graphe, toutes les tâches n'ont pas de commandes. C'est-à-dire que si une branche du graphe contient plusieurs tâches sans commandes, l'accroissement artificiel de la durée de chaque tâche pour la rendre multiple d'une période d'horloge, peut introduire un délai de une ou plusieurs périodes d'horloges inutiles sur cette branche (fig 7.27).



Circuit asynchrone, durée = 265 ns.



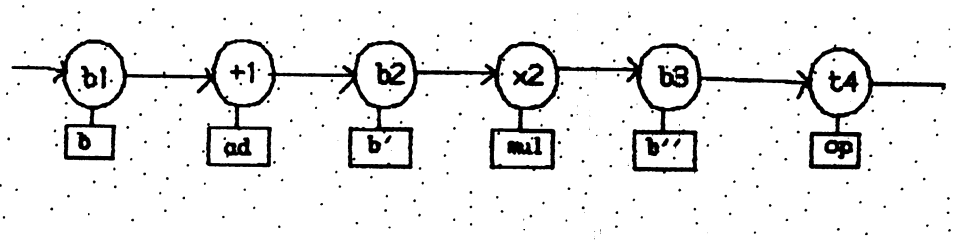
Circuit synchronisé 1, horloge de 25 ns, durée = 275 ns.



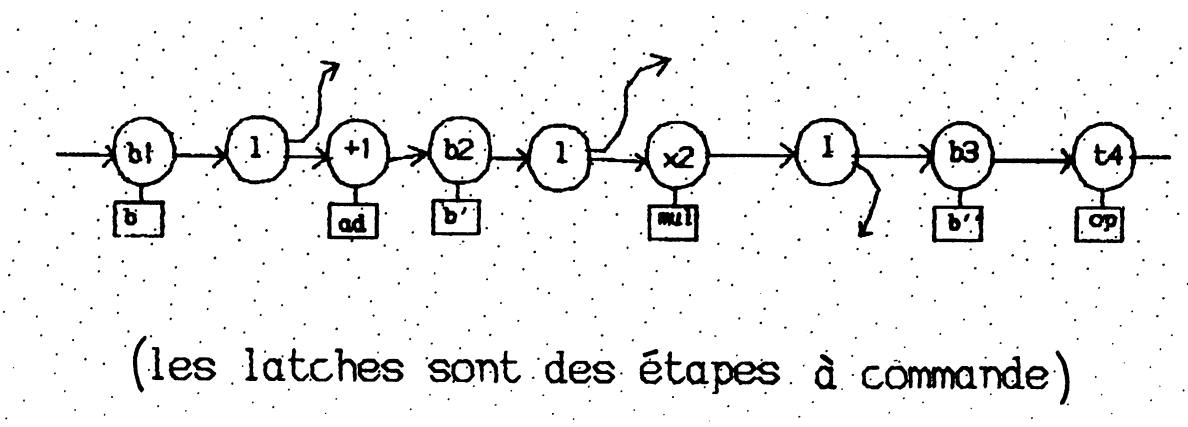
Circuit synchronisé 2, horloge de 25 ns, durée = 325 ns.

Fig : 7.27

Ce risque diminue quand le degré de multiplexage du circuit augmente. On trouve en effet dans ce cas beaucoup de commandes supplémentaires induites par les latches, et la longueur moyenne entre deux étapes à commande diminue sur toutes les branches du graphe (fig 7.28).



Pas de commandes sur cette chaine = coût élevé.



Beaucoup de commandes = coût faible.

Fig : 7.28

Remarquons toutefois, que bien que la synchronisation du contrôle ne se fasse pas de manière optimale, de nombreuses optimisations sont possibles qui visent à minimiser le nombre d'étapes du graphe. La plupart des commandes ont une marge temporelle (différence entre date au plus tard et date au plus tôt) bien supérieure à la marge de la tâche à laquelle elles sont rattachées.

Ceci permet de ne créer les étapes du contrôle qu'aux instants situés à l'intersection d'un maximum de marges temporelles (fig 7.29).

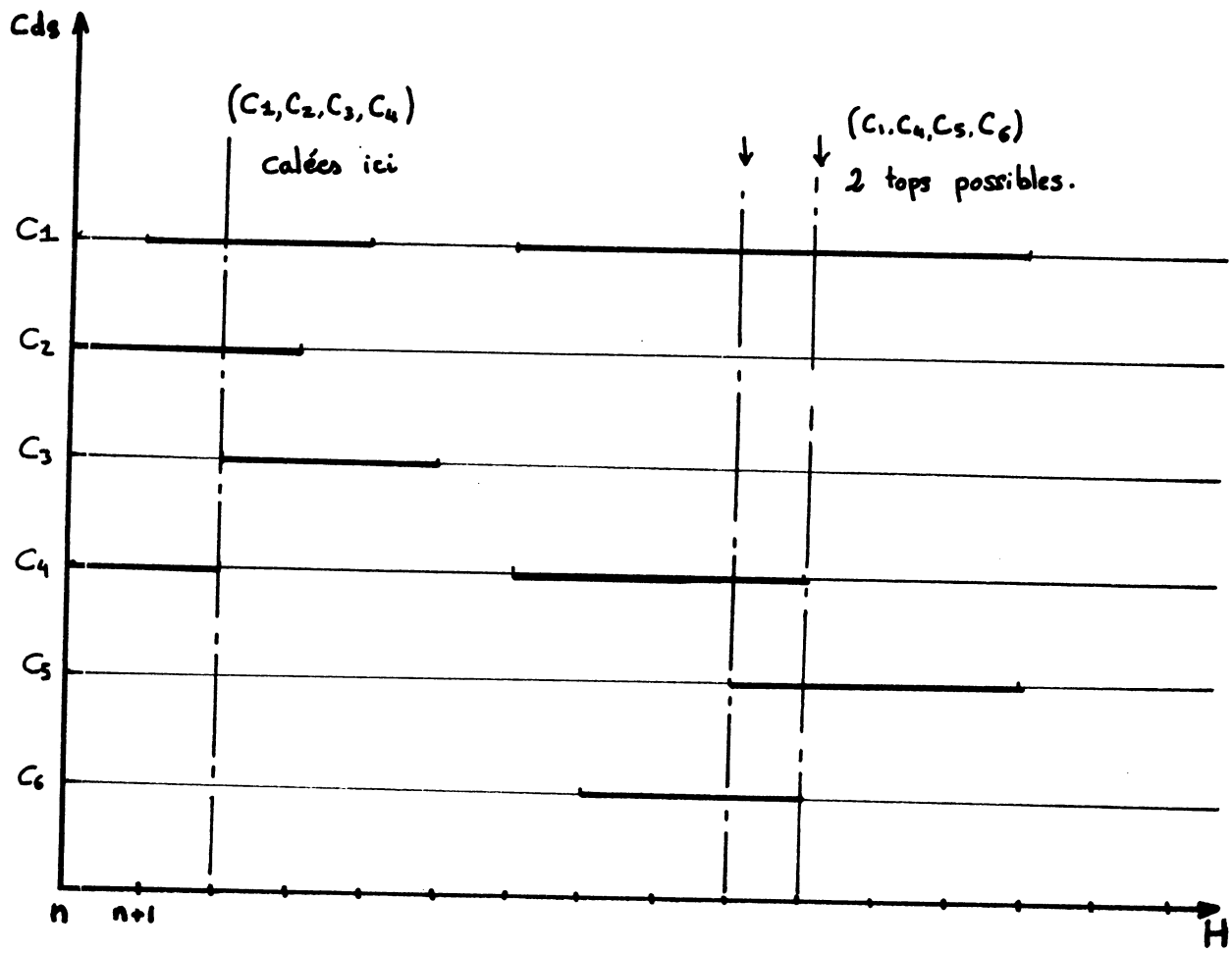


Fig : 7.29

7-8-4 Mise en forme et archivage des solutions

Dès que le graphe du séquencement est synchronisé, le temps de fonctionnement du circuit est connu en termes de nombre de période d'horloges.

Si ce temps satisfait la contrainte temporelle, la solution trouvée devient la meilleure trouvée à ce jour et repousse l'ancienne à la deuxième place. Le tableau des solutions stocke les cinq meilleures jamais trouvées sous la forme d'un RECORD PASCAL contenant les champs suivants :

- SO-TREP = temps de réponse;
- SO-T-HORLO = période de l'horloge;
- SO-SURF-G = estimation grossière de surface;
- SO-SURF-F = estimation fine de surface;

- SO-NBRTACH = nombre de tâches du graphe;
- SO-NBROPER = nombre d'opérateurs du circuit;
- SO-DATECREA = temps CPU nécessaire à la découverte;
- SO-P-CIRCUIT = pointeur sur la table des tâches;
- SO-P-HARDWAR = pointeur sur la table des opérateurs;
- SO-P-CONTROLE = pointeur sur le graphe de séquencement;

Dès que la solution est archivée, le programme restaure la structure de données décrivant le graphe dans l'état où elle était avant ce traitement et remonte dans l'arbre à la recherche d'autres solutions.

7-8-5 Traitement final des solutions

Ce traitement est effectué juste avant la fin de la session : soit l'arbre des solutions est complètement exploré, soit le temps CPU imparti est écoulé. A ce stade, le tableau des solutions contient ou non les meilleures architectures trouvées, et il convient de créer les fichiers permettant de stocker les architectures et leurs indicateurs de performances.

- Le stockage des solutions.

Il se compose de quatre fichiers de type texte qui sont :

. FILTRE.SOL
. FILTRE.OPT et FILTRE.PRO
. FILTRE.CTR

FILTRE étant la racine du nom du fichier SOURCE traité. Le fichier FILTRE.SOL contient la description sommaire des cinq meilleures architectures produites, sous la forme indiquée en 7-8-4.

Le fichier FILTRE.OPT renferme la description de la partie opérative dans une forme lisible par un utilisateur humain tandis que FILTRE.PRO fournit la même description au générateur de plan de masse PROTO sous une forme condensée.

Enfin FILTRE.CTR donne la description de la partie contrôle sous forme d'une liste d'étapes et de leurs commandes associées.

Les fichiers .OPT et .CTR peuvent concerner soit la meilleure, soit un nombre quelconque (maximum 5) d'architectures produites ; et ceci à la volonté de l'utilisateur.

- Les indicateurs de performances.

Ces fichiers fournissent à l'utilisateur des moyens d'apprécier les architectures produites avec un minimum d'effort. Une étude plus approfondie permettra de juger exactement de la valeur de telle ou telle solution mais nécessitera un travail beaucoup plus important.

Le fichier FILTRE.IMA donne une image de la partie opérative sous la forme suivante : les bus sont regroupés dans un canal central, et les opérateurs, sous la forme de rectangles identiques, sont disposés autour. Cela permet surtout d'évaluer la complexité des interconnexions, le nombre des multiplexeurs et éventuellement le degré d'utilisation, ou surtout de sous-utilisation de certains bus.

Le fichier FILTRE.GNT permet d'apprécier le degré de multiplexage des opérateurs et leur pourcentage d'utilisation sur le temps de fonctionnement total du circuit. Ceci est fait grâce à un tableau caractérisant l'ordonnancement du graphe et appelé "tableau de Gantt".

Ce diagramme possède un axe horizontal gradué en unités de temps et un axe vertical ayant autant de graduations qu'il y a d'opérateurs dans le circuit. Dès qu'un opérateur est utilisé entre les dates t1 et t2, un segment horizontal apparait à la ligne et entre les abcisses correspondants (fig 7.30).

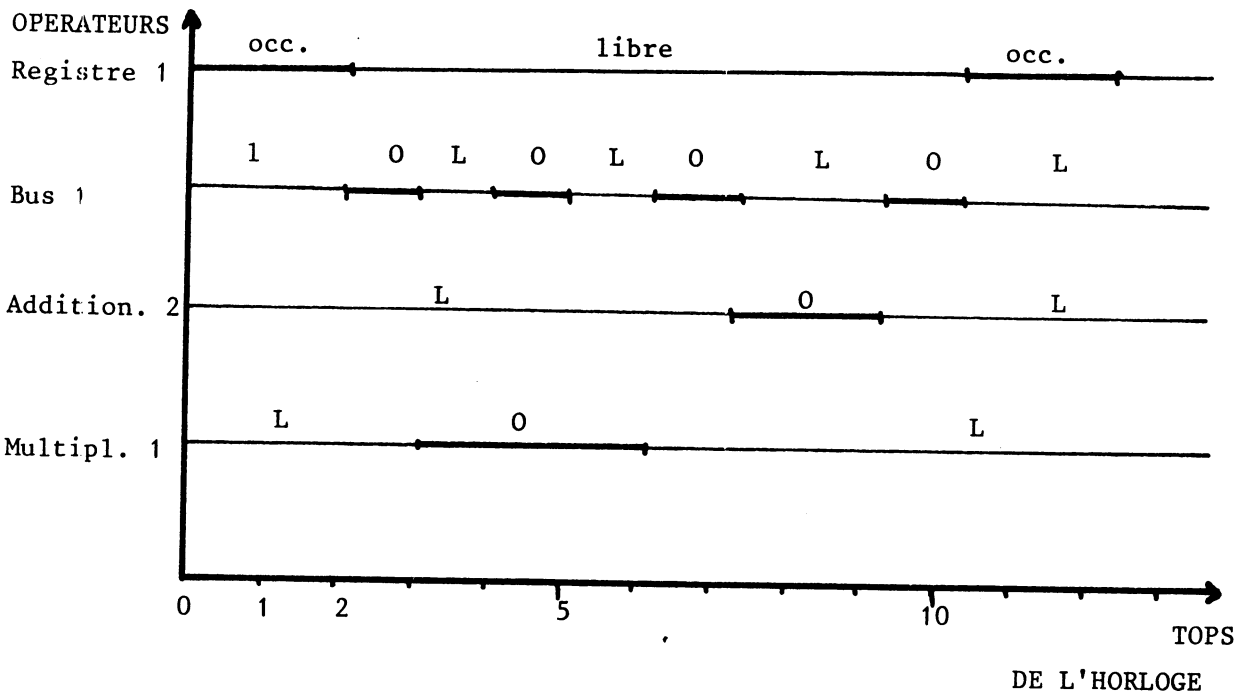


Fig : 7.30

Le tableau de Gantt de l'architecture est stocké dans le fichier FILTRE.GNT et utilise les trois symboles suivants :

. Le point "." quand l'opérateur n'est pas utilisé
. Le "%" ou le "a rond" quand il est utilisé

Les deux derniers symboles permettent de distinguer deux utilisations différentes contigûes non séparées par des ".".

Ce tableau permet de saisir visuellement et rapidement la "densité de travail" dans une architecture. Un tableau rempli de "%" et de " " indiquera une forte utilisation des opérateurs, et une architecture de bonne qualité. Les opérateurs sous-utilisés seront aisément répérables et l'utilisateur pourra envisager leur suppression du lot d'opérateurs initial et un nouvel essai d'IMHOTEP.

Des exemples sont donnés à la figure 7.31.

```
****************************************
 TABLEAU DE GANTT DE L'ARCHITECTURE
****************************************
    0 E S M M A R R R R D B B : :
    0 N O U U D E E E E E U U O O
    0 T R L L D G G G G C S S O 6
    0 R T T T N I I I I A _ _ : :
    0 E I _ _ E S S S S L 1 2 1 2
    0 E E S G U _ _ _ _ E 2 4 1 9
    0 _ _ P E R 1 1 2 2 U     . .
    0 B B E N           R     6 7
    0*****************************
    1 % . . . . % . % . . . . . .
    2 % . . . . % . % . . . . . .
    3 % . . . . % . % . . % . . .
    4 % . . . . % . % . . % . . .
    5 % . . % . % . % . . % . . .
    6 % . . % . % . % . . % . . .
    7 % . . % . % . % . . à . . .
    8 % . . % . % . % . . à . . .
    9 % . . % . % . % . . à . . .
   10 % . . % . % . % . . à . . .
   11 % . . % . % . % . % % . . .
   12 % . . % . % . % . % % . . .
   13 % . . % . % . % . % % . . .
   14 % . . % . % . % . % % . . .
   15 % . . % . % . % . . . . . .
   16 % . . % . % . % . . . . . .
   17 % . . % . % . % . . . . . .
   18 % . . % . % . % . . . . . .
   19 % . . % . % . % . . . . . .
   20 % . . % . % . % . . . . . .
   21 % . . % . % . % . . . . . .
   22 % . . % . % . % . . . . . .
   23 % . . % . % . % . . . . . .
   24 % . . % . % . % . . . . . .
   25 % . . % . % . % . . . % . .
   26 % . . % . % . % . . . % . .
   27 % . . % % % . % . . . % . .
   28 % . . % % % . % . . . % . .
   29 % . . . % % . % . . . à . .
   30 % . . . % % . % . . . à . .
   31 % . . . % % . % . . . à % .
   32 % . . . % % . % . . . à % .
   33 % . . . . % . % . . % % % .
   34 % . . . . % . % . . % % % .
   35 % . . % . % . % . . % % % .
   36 % . . % . % . % . . % % % .
   37 . . . % . % . % . . à à % .
   38 . . . % . % . % . . à à % .
   39 . . % % . % . % . . à à % .
   40 . . % % . % . % . . à à % .
   41 . . % % % % . % . . à % . .
   42 . . % % % % . % . . à % . .
   43 . . % % % % . % . . à % % .
   44 . . % % % % . % . . à % % .
   45 . . % % % % . % . . à . % .
   46 . . % % % % . % . . à . % .
   47 . . % % % % . % . . à . % .
```

Fig : 7.31

PLAN DU CHAPITRE 7-9

7-9- Conclusion

A ce stade de l'exposé, nous avons décrit en profondeur les mécanismes de IMHOTEP. L'idée directrice consiste à ramener le problème de la génération d'architecture à un problème d'ordonnancement sous contraintes de précédences et de ressources. Une fois cette étape franchie, l'extraction du circuit est un travail aisé.

La complexité de la résolution de l'ordonnancement est largement diminuée par le fait de profiter de tous les aspects particuliers de l'application, mais cette résolution demeure très longue si elle doit être exhaustive.

Pour fixer les idées, nous concluons ce chapitre 7 en donnant l'organisation générale de IMHOTEP, ainsi que le déroulement d'une session de travail vécue par l'utilisateur.

7-9-1 IMHOTEP

Nous le décrivons sous la forme d'un listing simplifié :

```
IMHOTEP;

BEGIN
LECTURE DU FICHIER SOURCE;
CREATION DE LA STRUCTURE DE DONNEES;
ACQUISITION DES PARAMETRES DU JOB
     - temps CPU maxi autorisé;
     - niveau de commentaires
RESOLUTION DE L'ORDONNANCEMENT; (procédure récursive)
CREATION DES FICHIERS DE RESULTATS;
END.
```

Le listing détaillé de IMHOTEP figure à l'annexe n°4.

7-9-2 Déroulement d'une session

Nous regroupons sous ce nom l'ensemble des actions enchaînées par l'utilisateur entre le moment où il possède le document de définition du filtre et celui où il reçoit la description de l'architecture de son circuit sous forme de fichiers traitables par les outils situés en aval.

L'utilisateur commence par entrer son filtre sous forme graphique dans la base de données de CASSIOPEE. Puis il exécute le programme TRADUCASS qui lui fournit un fichier MONFILTRE.TRA. MONFILTRE est le nom supposé du filtre à réaliser.

Vient alors le moment d'incorporer à la description graphique, les données numériques supplémentaires telles que les valeurs des coefficients des multiplications ou les largeurs des bus. Ceci est fait en exécutant DATAQUEST qui, à partir de MONFILTRE.TRA et d'un dialogue interactif avec l'utilisation produit le fichier MONFILTRE.DAT. Ce fichier contient, sous une forme vérifiée et correcte, l'équivalent du document de définition du filtre.

L'utilisateur doit alors choisir le jeu d'opérateurs dont il désire voir son filtre composé. Ceci se fait, bien sur, au vu du fichier MONFILTRE.DAT et grâce à l'accès à la bibliothèque d'opérateurs gérée par LOF.

La pertinence du choix de ce jeu d'opérateurs dépend de l'expérience qu'à l'utilisateur de tout le compilateur de silicium, ainsi que de ses essais antérieurs sur d'autres jeux d'opérateurs. Le programme CREATESOU gère le dialogue avec LOF et produit le fichier MONFILTRE.SOU.

Enfin, l'utilisateur peut utiliser IMHOTEP de deux manières :

La première est interactive et représente l'équivalent d'un essai "pour voir". Elle permet de valider un choix de jeu d'opérateurs car une solution architecturale peut être obtenue rapidement, dans un temps qui varie entre quelques secondes et quelques minutes CPU selon la complexité du filtre. Nous avons tout fait pour que la première solution trouvée soit aussi la meilleure, mais nous sommes conscients du fait qu'un enchainement de choix optimaux localement ne conduisent pas assurément à un optimum global. C'est pourquoi IMHOTEP fait un certain "balayage" de l'ensemble des solutions. Néanmoins, l'expérience prouve que la première solution trouvée est souvent très proche de la meilleure trouvée avec une recherche bien plus longue.

Cette première démarche interactive permet donc d'affiner le jeu d'opérateurs ainsi que de modifier les coefficients et les seuils fixés par défaut afin d'adapter le raisonnement d'IMHOTEP aux particularités du circuit.

Dès que l'utilisateur est satisfait de ses "réglages", il peut lancer en "batch" une exploration longue de l'arbre des solutions.

A l'issue de ces traitements, (fig 7.32) l'utilisateur possède les fichiers décrivant l'architecture désirée et permettant de l'évaluer. La suite du travail consistera à exécuter le programme PROTO qui génèrera le plan de masse du circuit, en liaison avec la bibliothèque d'opérateurs gérée par LOF.

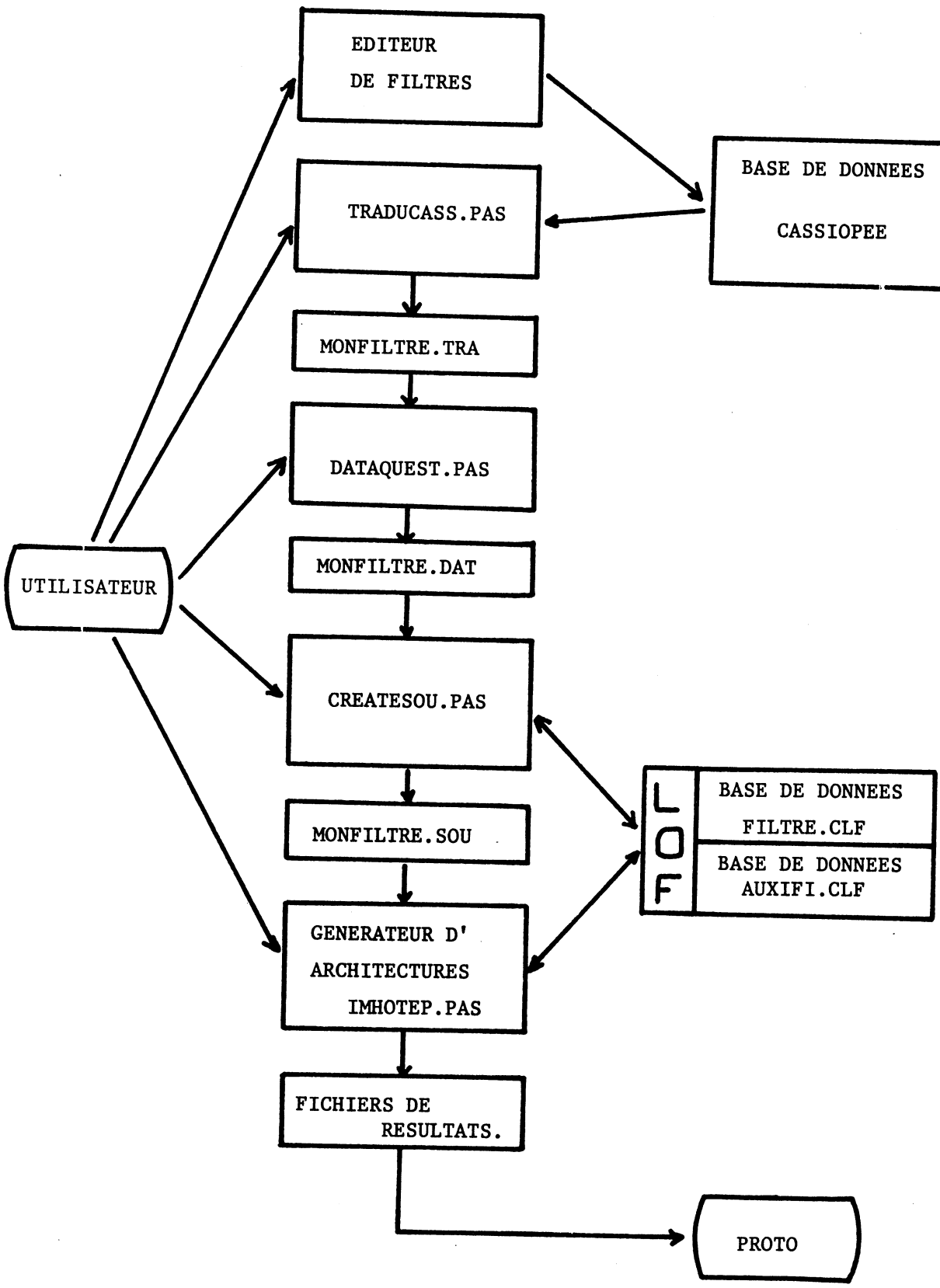

EDITEUR
DE FILTRES
BASE DE DONNEES
CASSIOPEE
TRADUCASS.PAS
MONFILTRE.TRA
DATAQUEST.PAS
MONFILTRE.DAT
UTILISATEUR
CREATESOU.PAS
LOF
BASE DE DONNEES
FILTRE.CLF
BASE DE DONNEES
AUXIFI.CLF
MONFILTRE.SOU
GENERATEUR D'
ARCHITECTURES
IMHOTEP.PAS
FICHIERS DE
RESULTATS.
PROTO

Fig : 7.32

8 - EXEMPLES ET TESTS : LES ARCHITECTURES PRODUITES

8-1 Introduction

Les tests destinés à corriger le code d'IMHOTEP ont consisté à dérouler des exemples de circuits possèdant chacun une spécifité faite pour éprouver une partie précise des algorithmes d'IMHOTEP. Les erreurs corrigeables l'ont été, mais certaines résultaient de configurations spéciales du graphe des tâches. Elles n'étaient pas détectables localement, sauf à y consacrer un temps considérable.

Dans ces cas, nous avons été obligés d'instaurer une stratégie de détection des erreurs (par exemple des boucles dans le graphe), et des stratégies de dégagement. Celles-ci ont été facilitées par la structure récursive du programme qui permet de revenir en arrière à la détection d'une anomalie.

IMHOTEP est à présent un prototype logiciel relativement fiable qui a travaillé sur plusieurs circuits pendant une centaine d'heures CPU sans incidents. Nous présentons ici plusieurs filtres que l'on peut tenir pour réalistes, et les architectures trouvées pour eux par IMHOTEP.

8-2 Les exemples traités

Ces filtres sont représentatifs des principaux algorithmes recensés au chapitre 3.

Il s'agit :

- d'un filtre non récursif à coefficients symétriques et d'ordre 11 nommé FIR-11.
- d'un filtre d'ondes (WDF) nommé LOUVAIN.
- d'un filtre d'ordre 4 constitué de deux cellules BIQUAD et nommé BIQUAD 2.

Les opérateurs utilisés pour produire les architectures des machines réalisant ces algorithmes ne sont pour l'instant que des approximations réalistes d'opérateurs NMOS 3.5. En effet, les bibliothèques d'opérateurs flexibles FILTRE.CLF et AUXIFI.CLF ne sont pas complètement remplies à ce jour, et les cellules manquantes ont été remplacées par des modules fictifs.

Les surfaces et les temps d'opération sont donnés ici pour la technologie NMOS 3.5 microns du CNET GRENOBLE.

Pour chaque filtre, IMHOTEP a travaillé en nocturne pendant environ 6 heures CPU sur un VAX 782. L'architecture présentée est la meilleure trouvée dans ce temps.

8-3 L'exploitation des résultats

Pour chaque filtre, nous donnons les renseignements suivants :

- Schéma du filtre de départ et renseignements non graphiques
- Jeu d'opérateurs utilisé
- Schémas bloc de la partie opérative
- Tableau de GANTT de la machine.

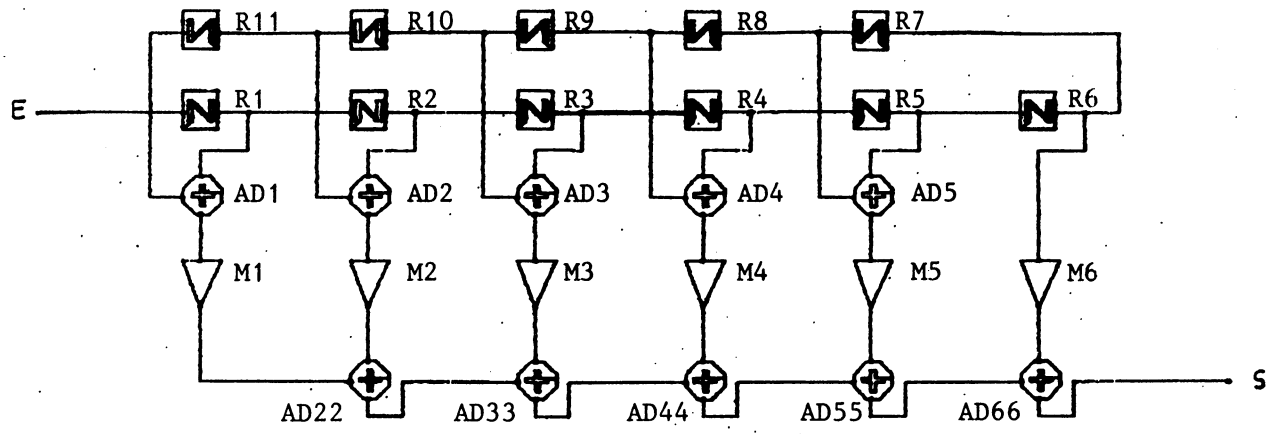
La répartition des tâches dans la partie opérative et la description de la partie contrôle sont reportées en annexe 7.

8-3-1- Le filtre FIR-11

C'est un filtre à réponse impulsionnelle finie, et à phase linéaire possédant 11 coefficients. Ce genre de schéma est très utilisé dans les applications où la stabilité et la linéarité en phase sont indispensables. De plus, l'ordre de ce type de filtre peut être très élevé (jusqu'à 1000 coefficients) et sa régularité permet aux concepteurs de trouver des réalisations économiques.

Il était donc intéressant de voir comment IMHOTEP traitait les architectures très régulières qui seraient donc pénalisantes pour lui, car il ne peut tirer parti de cette répétitivité du motif.

L'examen de l'architecture trouvée nous montre que IMHOTEP s'en tire de manière honorable car le taux d'occupation des principaux opérateurs est élevé. Un reproche peut être fait sur la manière dont la circulation des valeurs se passe dans la chaîne des registres : ceci est fait à la fin du traitement à cause de la faible urgence de ces tâches. En fait, cela ralenti le circuit d'environ 200 ns, alors que cette circulation pourrait être masquée durant le traitement principal.



LISTE DES PARAMETRES :

Chemins de données sur 12 bits en parallèle.
Additions 12 + 12 sur 12 bits sans contrôle d'overflow.
Multiplications 12 x 12 sur 12 bits
Arrondi = Troncature en complément à 2.

M1 : coefficient = 0.234
M2 : " = 0.345
M3 : " = 0.456
M4 : " = 0.321
M5 : " = 0.212
M6 : " = 0.123

CONTRAINTE DE TEMPS : 2500 ns.

Plusieurs essais nous ont conduits à adopter le jeu d'opérateurs décrit ci-après. Il comporte principalement un multiplieur général (coefficients en ROM), deux additionneurs et trois bus. Le tout séquencé par une horloge de période 25 ns.

L'architecture obtenue est satisfaisante, tant sur le plan du taux d'occupation des principaux opérateurs, que sur celui du nombre de registres auxilliaires nécessaires (seulement 2). Une réserve est à faire sur la présence de nombreux multiplexeurs dans le schéma final. Néanmoins, cette solution fut trouvée en 5 mn CPU, et aucune ne fut de meilleure qualité pendant les 300 mn restantes imparties à ce travail !

Les caractéristiques de cette solution sont les suivantes :

- découverte en 4,6 mn
- temps de réponse = 2350 ns avec une horloge de 25 ns
- surface des opérateurs seuls = 6,7 mm^2
- surface totale de la partie opérative : 9,15 mm^2
- durée totale du travail = 300 mn
- nombre de solutions explorées = 264

La description détaillée des parties opératives et contrôle est reportée à l'annexe 7.

JEU D'OPERATEURS PROPOSE POUR L'ARCHITECTURE :

OPERATEURS DE TYPE "ENTREE" :

- Opérateur numéro : 1, nommé : ENTREE F
  Généré par le module LOF nommé : ENTREE DE TYPE 2
  Paramètres fonctionnels :
  - 12 bits en PARALLELE
  Paramètres matériels :
  - Durée totale = 50 (ns).
  - Durée de set-up = 25 (ns).
  - Surface = 1.0E+05 (microns 2).

OPERATEURS DE TYPE "SORTIE" :

- Opérateur numéro : 2, nommé : SORTIE F
  Généré par le module LOF nommé : SORTIE DE TYPE 1
  Paramètres fonctionnels :
  - 12 bits en PARALLELE
  Paramètres matériels :
  - Durée totale = 50 (ns).
  - Durée de set-up = 25 (ns).
  - Surface = 1.0E+05 (microns 2).

OPERATEURS DE TYPE "MULTIPLIEUR" :

- Opérateur numéro : 3, nommé : MULT1
  Généré par le module LOF nommé : MULTIPLIEUR DE TYPE BOOTH
  Paramètres fonctionnels :
  - Entrée sur 12 bits.
  - Coefficient sur 12 bits.
  - Valeur du coefficient = 0 (multiplieur général)
  - Sortie sur 12 bits.
  - Mode d'arrondi = TRONCATURE_COMPL_A_2
  Paramètres matériels :
  - Durée totale = 200 (ns).
  - Durée de latchage = 50 (ns).
  - Durée de set-up = 25 (ns).
  - Surface = 3.0E+06 (microns 2).

OPERATEURS DE TYPE "ADDITIONNEUR" :

- Opérateur numéro : 4, nommé : ADD_1
  Généré par le module LOF nommé : ADDITIONNEUR DE TYPE ANTICIP
  Paramètres fonctionnels :
  - Entrée 1 sur 12 bits.
  - Entrée 2 sur 12 bits.
  - Sortie sur 12 bits.
  - Mode de contrôle d'overflow = RIEN
  Paramètres matériels :
  - Durée totale = 100 (ns).
  - Durée de latchage = 50 (ns).
  - Durée de set-up = 25 (ns).
  - Surface = 5.0E+05 (microns 2).

- Opérateur numéro : 5, nommé : ADD_2
  Généré par le module LOF nommé : ADDITIONNEUR DE TYPE ANTICIP
  Paramètres fonctionnels :
  - Entrée 1 sur 12 bits.
  - Entrée 2 sur 12 bits.
  - Sortie sur 12 bits.
  - Mode de contrôle d'overflow = RIEN
  Paramètres matériels :
  - Durée totale = 100 (ns).
  - Durée de latchage = 50 (ns).
  - Durée de set-up = 25 (ns).
  - Surface = 5.0E+05 (microns 2).

OPERATEURS DE TYPE "REGISTRE MAITRE-ESCLAVE" :

- Opérateur numéro : 6, nommé :REG1
  Généré par le module LOF nommé : REGISTRE DE TYPE STATIQUE ET RAZ
  Paramètres fonctionnels :
  - Stockage sur 12 bits.
  Paramètres matériels :
  - Durée totale = 50 (ns).
  - Durée de set-up = 25 (ns).
  - Surface = 1.0E+05 (microns 2).

- Opérateur numéro : 8, nommé :REG2
  Généré par le module LOF nommé : REGISTRE DE TYPE STATIQUE ET RAZ
  Paramètres fonctionnels :
  - Stockage sur 12 bits.
  Paramètres matériels :
  - Durée totale = 50 (ns).
  - Durée de set-up = 25 (ns).
  - Surface = 1.0E+05 (microns 2).

- Opérateur numéro : 10, nommé :REG3
  Généré par le module LOF nommé : REGISTRE DE TYPE STATIQUE ET RAZ
  Paramètres fonctionnels :
  - Stockage sur 12 bits.
  Paramètres matériels :
  - Durée totale = 50 (ns).
  - Durée de set-up = 25 (ns).
  - Surface = 1.0E+05 (microns 2).

- Opérateur numéro : 12, nommé :REG4
  Généré par le module LOF nommé : REGISTRE DE TYPE STATIQUE ET RAZ
  Paramètres fonctionnels :
  - Stockage sur 12 bits.
  Paramètres matériels :
  - Durée totale = 50 (ns).
  - Durée de set-up = 25 (ns).
  - Surface = 1.0E+05 (microns 2).

- Opérateur numéro : 14, nommé :REG5
  Généré par le module LOF nommé : REGISTRE DE TYPE STATIQUE ET RAZ
  Paramètres fonctionnels :
  - Stockage sur 12 bits.
  Paramètres matériels :
  - Durée totale = 50 (ns).
  - Durée de set-up = 25 (ns).
  - Surface = 1.0E+05 (microns 2).

- Opérateur numéro : 16, nommé :REG6
  Généré par le module LOF nommé : REGISTRE DE TYPE STATIQUE ET RAZ
  Paramètres fonctionnels :
  - Stockage sur 12 bits.
  Paramètres matériels :
  - Durée totale = 50 (ns).
  - Durée de set-up = 25 (ns).
  - Surface = 1.0E+05 (microns 2).

- Opérateur numéro : 18, nommé :REG7
  Généré par le module LOF nommé : REGISTRE DE TYPE STATIQUE ET RAZ
  Paramètres fonctionnels :
  - Stockage sur 12 bits.
  Paramètres matériels :
  - Durée totale = 50 (ns).
  - Durée de set-up = 25 (ns).
  - Surface = 1.0E+05 (microns 2).

- Opérateur numéro : 20, nommé :REG8
  Généré par le module LOF nommé : REGISTRE DE TYPE STATIQUE ET RAZ
  Paramètres fonctionnels :
  - Stockage sur 12 bits.
  Paramètres matériels :
  - Durée totale = 50 (ns).
  - Durée de set-up = 25 (ns).
  - Surface = 1.0E+05 (microns 2).

- Opérateur numéro : 22, nommé :REG9
  Généré par le module LOF nommé : REGISTRE DE TYPE STATIQUE ET RAZ
  Paramètres fonctionnels :
  - Stockage sur 12 bits.
  Paramètres matériels :
  - Durée totale = 50 (ns).
  - Durée de set-up = 25 (ns).
  - Surface = 1.0E+05 (microns 2).

- Opérateur numéro : 24, nommé :REG10
  Généré par le module LOF nommé : REGISTRE DE TYPE STATIQUE ET RAZ
  Paramètres fonctionnels :
  - Stockage sur 12 bits.
  Paramètres matériels :
  - Durée totale = 50 (ns).
  - Durée de set-up = 25 (ns).
  - Surface = 1.0E+05 (microns 2).

- Opérateur numéro : 26, nommé :REG11
  Généré par le module LOF nommé : REGISTRE DE TYPE STATIQUE ET RAZ
  Paramètres fonctionnels :
  - Stockage sur 12 bits.
  Paramètres matériels :
  - Durée totale = 50 (ns).
  - Durée de set-up = 25 (ns).
  - Surface = 1.0E+05 (microns 2).

OPERATEURS DE TYPE "BUS" :

- Opérateur numéro : 28, nommé : BUS1
  Renseigné par le module LOF nommé : BUS
  Paramètres matériels :
  - Durée totale garantie = 50 (ns).
  - Surface inconnue a priori.

- Opérateur numéro : 29, nommé : BUS2
  Renseigné par le module LOF nommé : BUS
  Paramètres matériels :
  - Durée totale garantie = 50 (ns).
  - Surface inconnue a priori.

- Opérateur numéro : 30, nommé : BUS3
  Renseigné par le module LOF nommé : BUS
  Paramètres matériels :
  - Durée totale garantie = 50 (ns).
  - Surface inconnue a priori.

SHEMA-BLOC DE L'ARCHITECTURE TROUVEE POUR " FIR-11 "

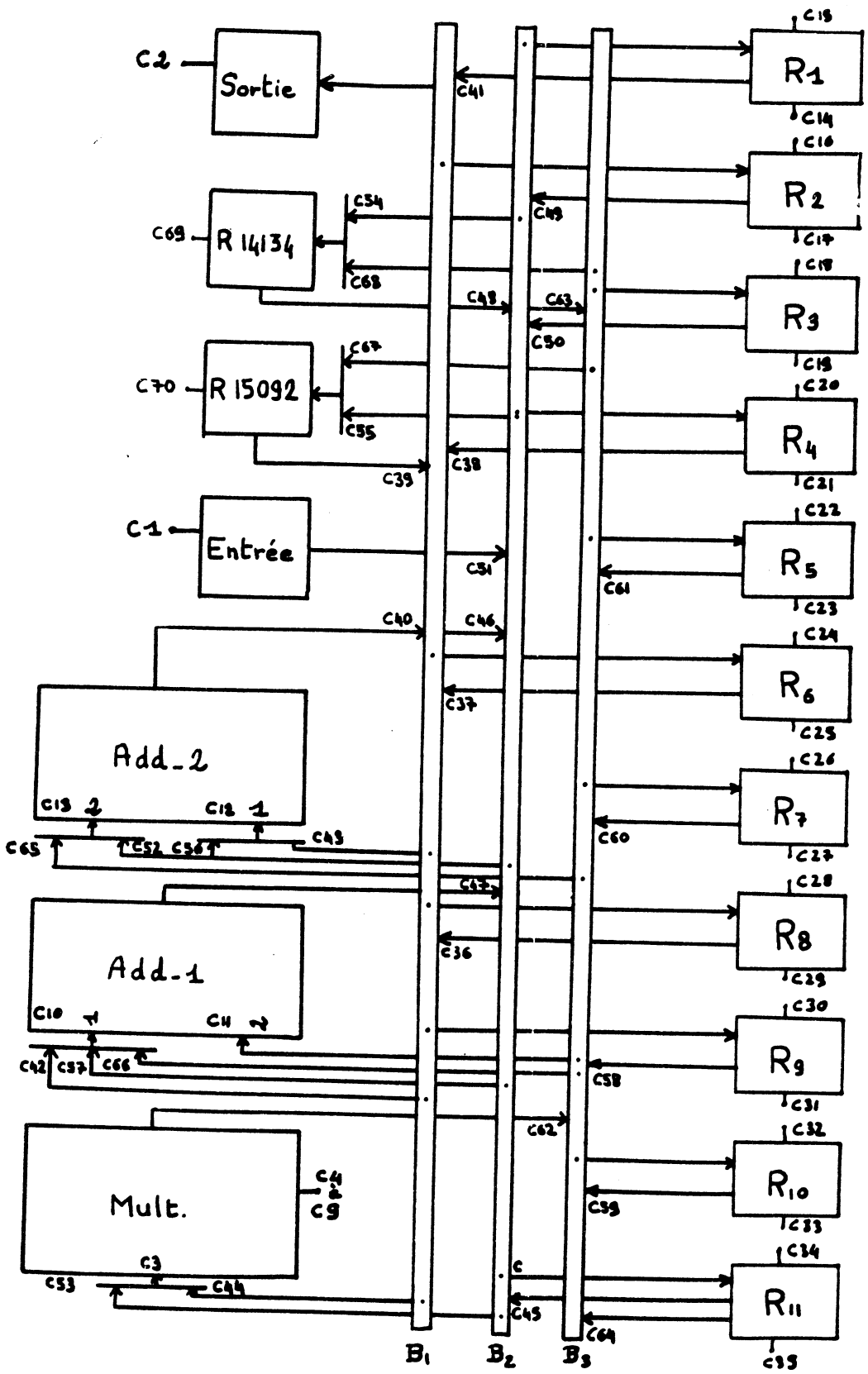
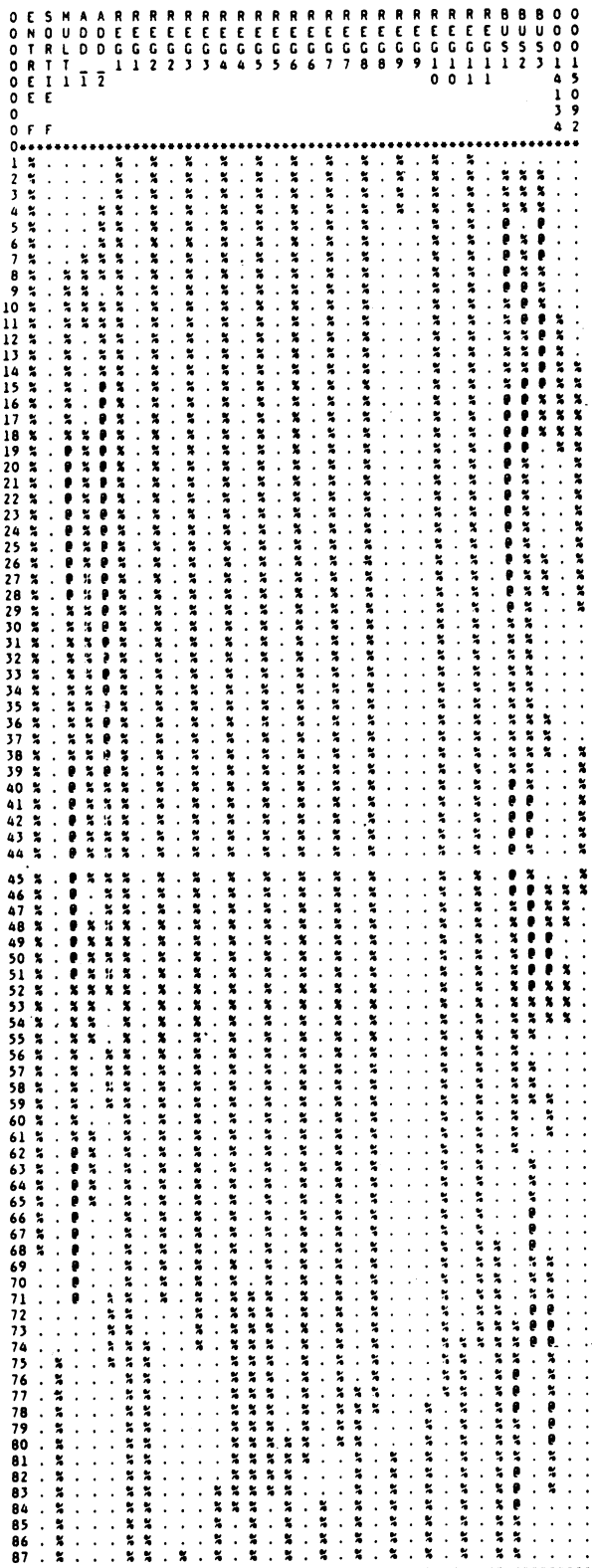


TABLEAU DE GANTT DE L'ARCHITECTURE DE "FIR-11"



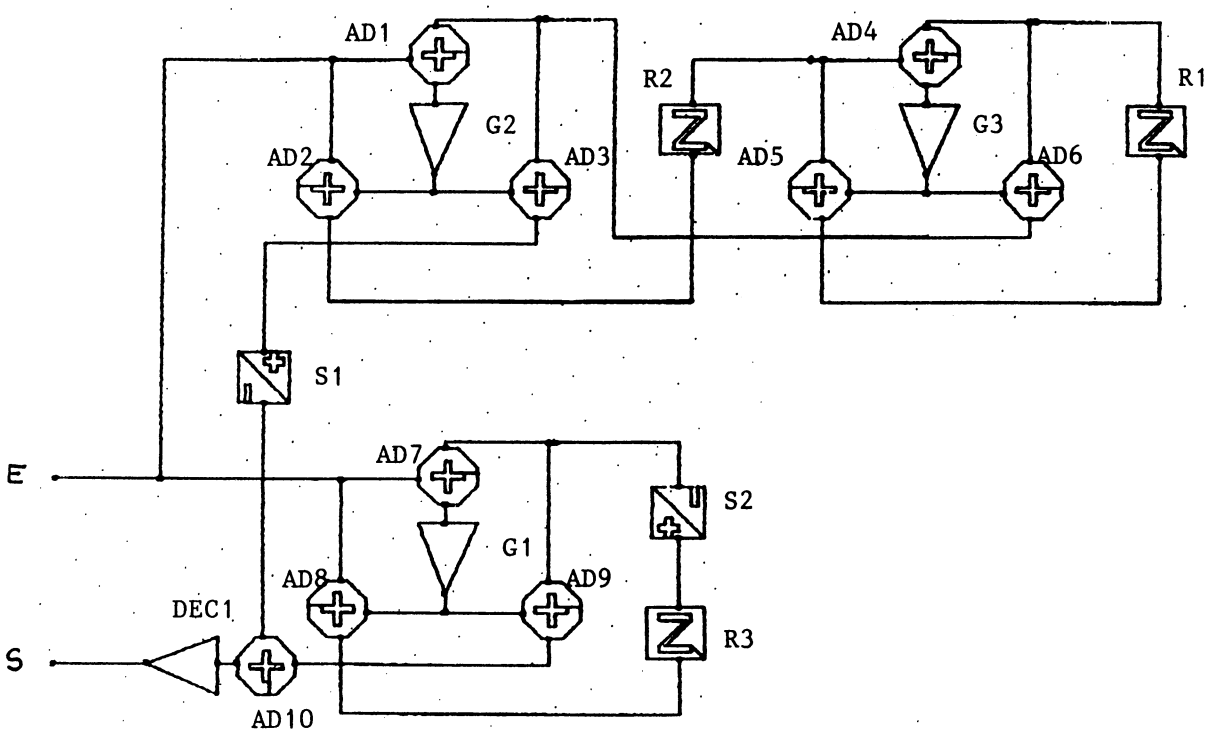
POURCENTAGES D'UTILISATION DES OPERATEURS DE L'ARCHITECTURE

Opérateur nommé : ENTREE F utilisé à 7.81609E+01 pour cent
Opérateur nommé : SORTIE F utilisé à 1.49425E+01 pour cent
Opérateur nommé : MULT1 utilisé à 7.35632E+01 pour cent
Opérateur nommé : ADD_1 utilisé à 5.28736E+01 pour cent
Opérateur nommé : ADD_2 utilisé à 6.55172E+01 pour cent
Opérateur nommé : BUS1 utilisé à 8.73563E+01 pour cent
Opérateur nommé : BUS2 utilisé à 7.81609E+01 pour cent
Opérateur nommé : BUS3 utilisé à 5.74713E+01 pour cent
Opérateur nommé : 00014134 utilisé à 1.83908E+01 pour cent
Opérateur nommé : 00015092 utilisé à 2.87356E+01 pour cent

8-3-2- Le filtre LOUVAIN

Ce filtre est à structure récursive, il est calculé en utilisant un algorithme de Wave Digital Filter (WDF). Le principal avantage de ce genre de structure est de manipuler des mots binaires de faible longueur, en comparaison avec une structure BIQUAD par exemple qui réaliserait la même fonction de transfert. Cette caractéristique étant compensée par un nombre d'opérations plus important.

La structure de cet algorithme est assez irrégulière comparativement à l'exemple précédent, et IMHOTEP produit une architecture performante avec un taux moyen d'occupation de 78% pour les trois principaux opérateurs.



LISTE DES PARAMETRES :

Chemins de données sur 16 bits en parallèle.
Additions 16 bits + 16 bits sans contrôle d'overflow.
Décalage Dec1 sur un mot de 16 bits, 1 rang en sens négatif,
Arrondi = troncature en complément à 2.
Multiplications : entrées sur 16 bits,
Arrondis = troncatures en complément à 2.

G1 : coefficient valant 0,1 codé sur 12 bits
G2 : coefficient valant 0,01 codé sur 10 bits
G3 : coefficient valant 0,001 codé sur 7 bits

Sorties sur 16 bits.

CONTRAINTE DE TEMPS : 3200 ns.

Dans le traitement de cet exemple, la meilleure solution ne fut pas la première trouvée. La première solution fut trouvée au bout de 4 minutes et avait les caractéristiques suivantes :

Temps de réponse : 2950 ns
Surface de la partie opérative : 62,2 mm^2

La deuxième fut trouvée après 7 minutes CPU :

Temps de réponse : 2975 ns
Surface de la partie opérative : 62,0 mm^2

La troisième, et meilleure pendant tout le temps restant, fut découverte après 36 mn CPU :

Temps de réponse = 3025 ns (horloge de 25 ns)
Surface de la partie opérative : 61,9 mm^2
Surface des opérateurs seuls : 59 mm^2
Durée totale du travail 120 mm CPU
Nombre de solutions explorées : 505

Ceci met en lumière un aspect du fonctionnement de IMHOTEP qui, orienté "temps-réel", cherche les solutions les plus rapides. Ceci est souvent en conflit avec la recherche de la plus petite architecture.

Ceci est particulièrement vrai lorsque l'on peut se passer d'un opérateur qui avait pourtant été fourni : IMHOTEP ne trouve ceci qu'au bout d'un certain nombre d'essais. L'importance du choix du jeu d'opérateur est ici mise en évidence.

LISTE DES OPERATEURS UTILISES PAR L'ARCHITECTURE PRODUITE.

OPERATEURS DE TYPE "ENTREE" :

- Opérateur numéro : 1, nommé : ENTREE
  Généré par le module LOF nommé : ENTREE DE TYPE 1
  Paramètres fonctionnels :
  - 16 bits en PARALLELE
  Paramètres matériels :
  - Durée totale = 37 (ns).
  - Durée de set - up = 20 (ns).
  - Surface = 1.12E + 06 (microns 2).

OPERATEURS DE TYPE "SORTIE" :

- Opérateur numéro : 2, nommé : SORTIE
  Généré par le module LOF nommé : SORTIE DE TYPE 2
  Paramètres fonctionnels :
  - 16 bits en PARALLELE
  Paramètres matériels :
  - Durée totale = 37 (ns).
  - Durée de set - up = 20 (ns).
  - Surface = 1.12E + 06 (microns 2).

OPERATEURS DE TYPE "MULTIPLIEUR" :

- Opérateur numéro : 3, nommé : MULT
  Généré par le module LOF nommé : MULTIPLIEUR DE TYPE BOOTH GENERAL
  Paramètres fonctionnels :
  - Entrée sur 16 bits.
  - Coefficient sur 12 bits.
  - Valeur du coefficient = 0 (multiplieur général).
  - Sortie sur 16 bits.
  - Mode d'arrondi = TRONCATURE_COMPL_A_2
  Paramètres matériels :
  - Durée totale = 232 (ns).
  - Durée de latchage = 80 (ns).
  - Durée de set - up = 40 (ns).
  - Surface = 5.04E + 07 (microns 2).

OPERATEURS DE TYPE "ADDITIONNEUR" :

- Opérateur numéro : 4, nommé : ADDIT_1
  Généré par le module LOF nommé : ADDITIONNEUR DE TYPE CARRY LOOK AHEAD
  Paramètres fonctionnels :
  - Entrée 1 sur 16 bits.
  - Entrée 2 sur 16 bits.
  - Sortie sur 16 bits.
  - Mode de contrôle d'overflow = RIEN
  Paramètres matériels :
  - Durée totale = 75 (ns).
  - Durée de latchage = 11 (ns).
  - Durée de set - up = 5 (ns).
  - Surface = 2.88E + 06 (microns 2).

- Opérateur numéro : 5, nommé : ADDIT_2
  Généré par le module LOF nommé : ADDITIONNEUR DE TYPE CARRY LOOK AHEAD
  Paramètres fonctionnels :
  - Entrée 1 sur 16 bits.
  - Entrée 2 sur 16 bits.
  - Sortie sur 16 bits.
  - Mode de contrôle d'overflow = RIEN
  Paramètres matériels :
  - Durée totale = 75 (ns).
  - Durée de latchage = 11 (ns).
  - Durée de set - up = 5 (ns).
  - Surface = 2.88E + 06 (microns 2).

OPERATEURS DE TYPE "REGISTRE MAITRE - ESCLAVE" :

- Opérateur numéro : 6, nommé :REG1
  Généré par le module LOF nommé : REGISTRE DE TYPE STATIQUE AVEC RAZ
  Paramètres fonctionnels :
  - Stockage sur 16 bits.
  Paramètres matériels :
  - Durée totale = 13 (ns).
  - Durée de set - up = 6 (ns).
  - Surface = 2.4E + 05 (microns 2).

- Opérateur numéro : 8, nommé :REG2
  Généré par le module LOF nommé : REGISTRE DE TYPE STATIQUE AVEC RAZ
  Paramètres fonctionnels :
  - Stockage sur 16 bits.
  Paramètres matériels :
  - Durée totale = 13 (ns).
  - Durée de set - up = 6 (ns).
  - Surface = 2.4E + 05 (microns 2).

- Opérateur numéro : 10, nommé :REG3
  Généré par le module LOF nommé : REGISTRE DE TYPE STATIQUE AVEC RAZ
  Paramètres fonctionnels :
  - Stockage sur 16 bits.
  Paramètres matériels :
  - Durée totale = 13 (ns).
  - Durée de set - up = 6 (ns).
  - Surface = 2.4E + 05 (microns 2).

OPERATEURS DE TYPE "CHANGEMENT DE SIGNE" :

- Opérateur numéro : 12, nommé : CHSGN
  Généré par le module LOF nommé : CHANGEMENT DE SIGNE DE TYPE 1
  Paramètres fonctionnels :
  - Entrée sur 16 bits.
  Paramètres matériels :
  - Durée totale = 66 (ns).
  - Durée de latchage = 13 (ns).
  - Durée de set - up = 4 (ns).
  - Surface = 5.6E + 06 (microns 2).

OPERATEURS DE TYPE "DECALAGE" :

- Opérateur numéro : 13, nommé : DECA
  Généré par le module LOF nommé : DECALEUR DE TYPE 1 CABLE
  Paramètres fonctionnels :
  - Entrée sur 16 bits.
  - Décalage de 1 bits.
  - Dans le sens NEGATIF
  - Sortie sur 16 bits.
  - Mode d'arrondi = TRONCATURE_COMPL_A_2
  Paramètres matériels :
  - Durée totale = 37 (ns).
  - Durée de latchage = 37 (ns).
  - Durée de set - up = 20 (ns).
  - Surface = 1.12E + 05 (microns 2).

OPERATEURS DE TYPE "BUS" :

- Opérateur numéro : 14, nommé : BUS_1
  Renseigné par le module LOF nommé : BUS
  Paramètres matériels :
  - Durée totale garantie = 50 (ns).
  - Surface inconnue a priori.

- Opérateur numéro : 15, nommé : BUS_2
  Renseigné par le module LOF nommé : BUS
  Paramètres matériels :
  - Durée totale garantie = 50 (ns).
  - Surface inconnue a priori.

- Opérateur numéro : 16, nommé : BUS_3
  Renseigné par le module LOF nommé : BUS
  Paramètres matériels :
  - Durée totale garantie = 50 (ns).
  - Surface inconnue a priori.

SHEMA-BLOC DE L'ARCHITECTURE TROUVEE POUR " LOUVAIN "

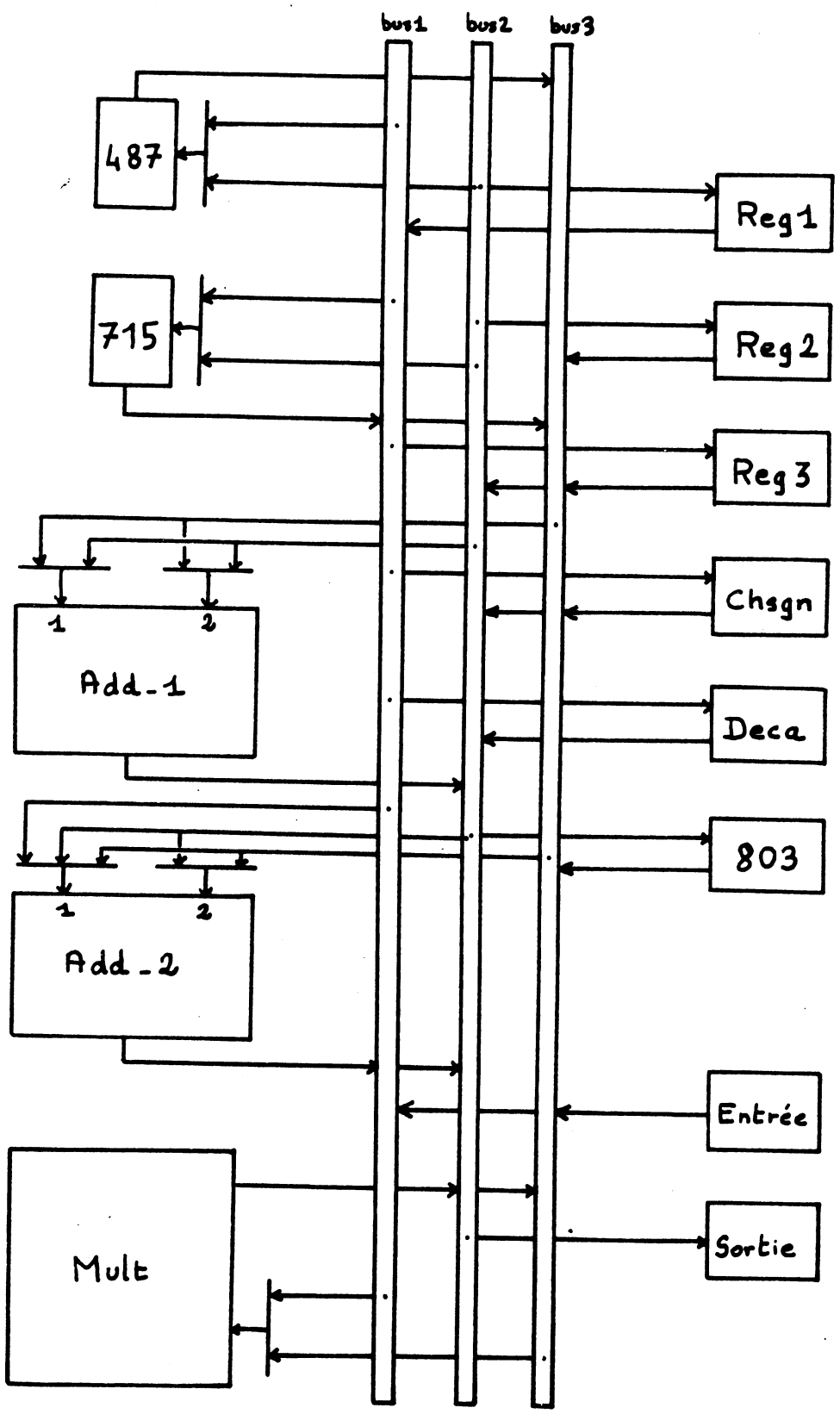
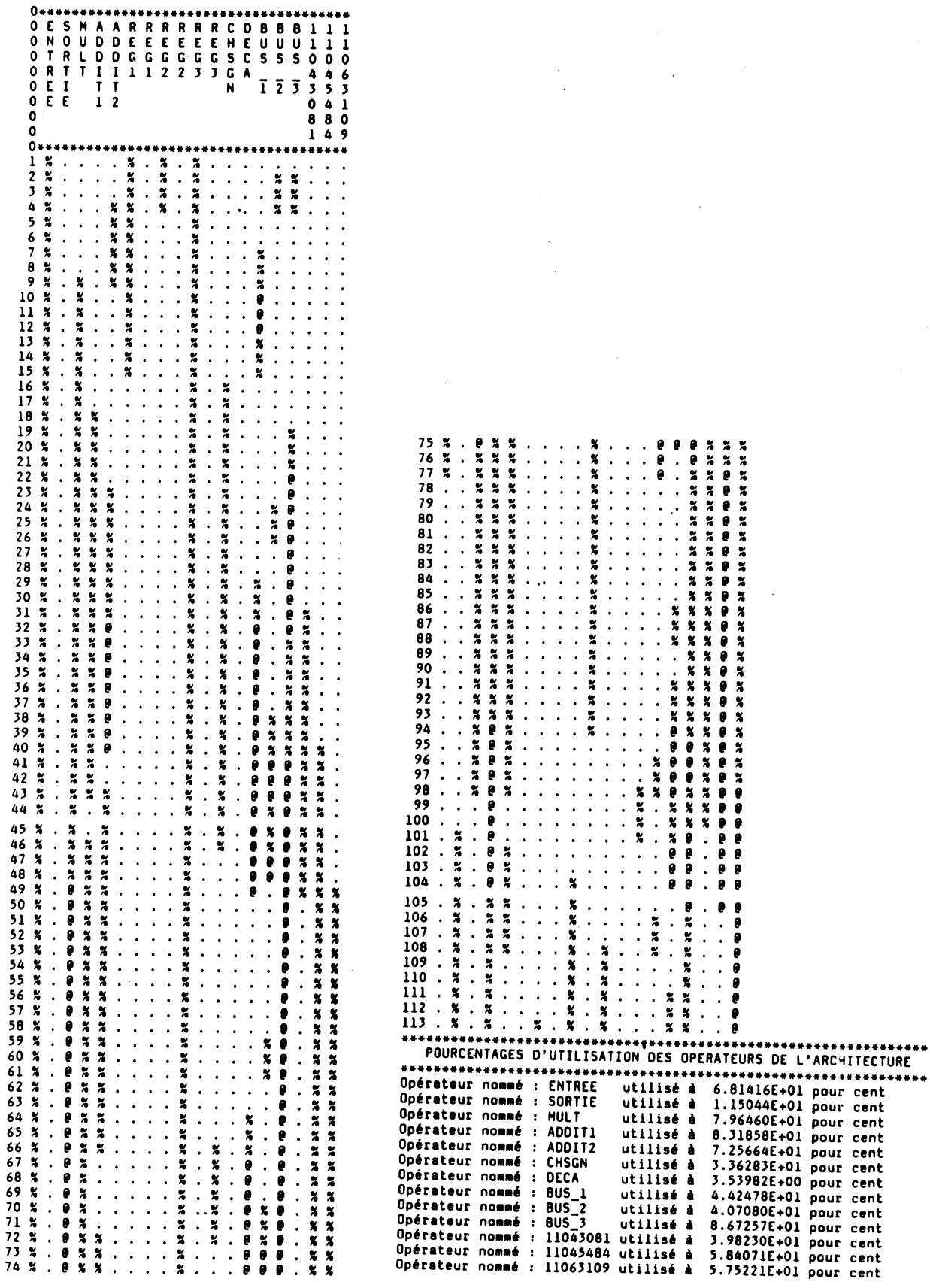


TABLEAU DE GANTT DE L'ARCHITECTURE DE "LOUVAIN"



POURCENTAGES D'UTILISATION DES OPERATEURS DE L'ARCHITECTURE

Opérateur nommé : ENTREE utilisé à 6.81416E+01 pour cent
Opérateur nommé : SORTIE utilisé à 1.15044E+01 pour cent
Opérateur nommé : MULT utilisé à 7.96460E+01 pour cent
Opérateur nommé : ADDIT1 utilisé à 8.31858E+01 pour cent
Opérateur nommé : ADDIT2 utilisé à 7.25664E+01 pour cent
Opérateur nommé : CHSGN utilisé à 3.36283E+01 pour cent
Opérateur nommé : DECA utilisé à 3.53982E+00 pour cent
Opérateur nommé : BUS_1 utilisé à 4.42478E+01 pour cent
Opérateur nommé : BUS_2 utilisé à 4.07080E+01 pour cent
Opérateur nommé : BUS_3 utilisé à 8.67257E+01 pour cent
Opérateur nommé : 11043081 utilisé à 3.98230E+01 pour cent
Opérateur nommé : 11045484 utilisé à 5.84071E+01 pour cent
Opérateur nommé : 11063109 utilisé à 5.75221E+01 pour cent

8-3-3- Le filtre BIQUAD 2

Cette structure de filtre est très classique, et comporte souvent plusieurs cellules BIQUAD en série. Nous avons choisi un exemple n'en ayant que deux pour simplifier l'analyse du lecteur désireux de comprendre l'architecture produite.

Afin de montrer l'importance du choix du jeu d'opérateurs initial, nous donnons deux exemples : le premier jeu contient deux multiplieurs et un additionneur, le deuxième jeu un seul multiplieur et deux additionneurs.

Dans le premier cas, il est aisé de voir que les deux multiplieurs ne sont pas utilisés efficacement, et ne travaillent que rarement en parallèle. D'autres parts, les cascades d'additions sont plus faciles à réaliser avec deux additionneurs qu'avec un seul.

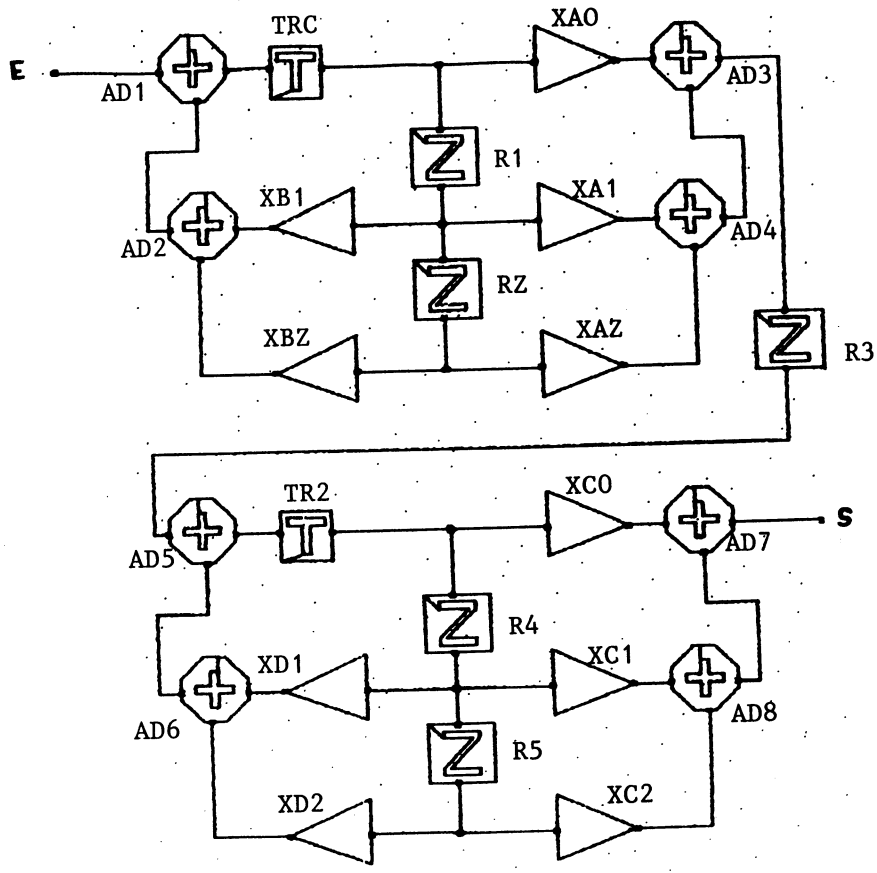
Dans le deuxième cas, les taux d'utilisation sont beaucoup mieux équilibrés ; le résultat est probant : le circuit est à la fois moins gros et plus rapide. Un ajustage plus fin des paramètres de contrôle de l'algorithme devrait en outre permettre de diminuer le nombre des registres auxilliaires du deuxième exemple.

Première architecture :

- temps de réponse : 3725 ns
- surface des opérateurs seuls : 19,88 mm^2
- surface de la partie opérative : 24,50mm^2
- Durée totale du travail : 300 mm CPU
- Nombre de solutions explorées : 250
- Meilleure solution trouvée en 74 mn.

Deuxième architecture :

- temps de réponse : 3050 ns
- surface des opérateurs seuls : 11,16 mm^2
- surface de la partie opérative : 17,15 mm^2
- durée totale du travail : 300 mm
- nombre de solutions explorées : 270
- meilleure solution trouvée en 68 mn.



LISTE DES PARAMETRES :

Entrée et sortie se font sur 24 bits en parallèle.
Chemins de données sur 16 bits près des registres,
et sur 24 bits ailleurs.
TRC et TR2 font un arrondi du complément à 2 de 24 vers 16 bits.
Les multiplications ont des données sur 16 bits,
des résultats sur 24 bits,
et un arrondi en troncature si nécessaire.

Coefficients :

XA0 --> 0,01627 sur 10 bits, XA1 --> 1,4573 sur 12 bits
XA2 --> 0,2365 sur 10 bits, XB1 --> 0,18403 sur 8 bits
XB2 --> 1 , XC0 --> 1
XC1 --> 1,00564 sur 10 bits, XC2 --> 1,48375 sur 13 bits
XD1 --> 0,2654 sur 8 bits , XD2 --> 0,017384 sur 8 bits.

CONTRAINTE TEMPORELLE : 4000 ns

JEU D'OPERATEURS PROPOSE POUR L'ARCHITECTURE :

OPERATEURS DE TYPE "ENTREE" :

- Opérateur numéro : 1, nommé : ENTREE F
  Généré par le module LOF nommé : ENTREE2
  Paramètres fonctionnels :
  - 24 bits en PARALLELE
  Paramètres matériels :
  - Durée totale = 30 (ns).
  - Durée de set-up = 14 (ns).
  - Surface = 2.1294E+05 (microns 2).

OPERATEURS DE TYPE "SORTIE" :

- Opérateur numéro : 2, nommé : SORTIE F
  Généré par le module LOF nommé : SORTIE2
  Paramètres fonctionnels :
  - 24 bits en PARALLELE
  Paramètres matériels :
  - Durée totale = 35 (ns).
  - Durée de set-up = 19 (ns).
  - Surface = 2.1294E+05 (microns 2).

OPERATEURS DE TYPE "MULTIPLIEUR" :

- Opérateur numéro : 3, nommé : MULT1
  Généré par le module LOF nommé : MULTIPLI1
  Paramètres fonctionnels :
  - Entrée sur 16 bits.
  - Coefficient sur 13 bits.
  - Valeur du coefficient = 0 (multiplieur général)
  - Sortie sur 24 bits.
  - Mode d'arrondi = TRONCATURE_COMPL_A_2
  Paramètres matériels :
  - Durée totale = 232 (ns).
  - Durée de latchage = 8 (ns).
  - Durée de set-up = 4 (ns).
  - Surface = 3.48E+06 (microns 2).

- Opérateur numéro : 4, nommé : MULT2
  Généré par le module LOF nommé : MULTIPLI1
  Paramètres fonctionnels :
  - Entrée sur 16 bits.
  - Coefficient sur 13 bits.
  - Valeur du coefficient = 0 (multiplieur général)
  - Sortie sur 24 bits.
  - Mode d'arrondi = TRONCATURE_COMPL_A_2
  Paramètres matériels :
  - Durée totale = 232 (ns).
  - Durée de latchage = 8 (ns).
  - Durée de set-up = 4 (ns).
  - Surface = 3.48E+06 (microns 2).

OPERATEURS DE TYPE "ADDITIONNEUR" :

- Opérateur numéro : 5, nommé : ADD_1
  Généré par le module LOF nommé : ADDITION
  Paramètres fonctionnels :
  - Entrée 1 sur 24 bits.
  - Entrée 2 sur 24 bits.
  - Sortie sur 24 bits.
  - Mode de contrôle d'overflow = RIEN
  Paramètres matériels :
  - Durée totale = 80 (ns).
  - Durée de latchage = 8 (ns).
  - Durée de set-up = 4 (ns).
  - Surface = 4.6846E+05 (microns 2).

OPERATEURS DE TYPE "REGISTRE MAITRE-ESCLAVE" :

- Opérateur numéro : 6, nommé :REG1
  Généré par le module LOF nommé : REGISTRE DE TYPE STATIQUE ET RAZ
  Paramètres fonctionnels :
  - Stockage sur 16 bits.
  Paramètres matériels :
  - Durée totale = 80 (ns).
  - Durée de set-up = 4 (ns).
  - Surface = 4.6846E+05 (microns 2).

- Opérateur numéro : 8, nommé :REG2
  Généré par le module LOF nommé : REGISTRE DE TYPE STATIQUE ET RAZ
  Paramètres fonctionnels :
  - Stockage sur 16 bits.
  Paramètres matériels :
  - Durée totale = 80 (ns).
  - Durée de set-up = 4 (ns).
  - Surface = 4.6864E+05 (microns 2).

- Opérateur numéro : 10, nommé :REG3
  Généré par le module LOF nommé : REGISTRE DE TYPE STATIQUE ET RAZ
  Paramètres fonctionnels :
  - Stockage sur 16 bits.
  Paramètres matériels :
  - Durée totale = 80 (ns).
  - Durée de set-up = 4 (ns).
  - Surface = 4.6864E+05 (microns 2).

- Opérateur numéro : 12, nommé :REG4
  Généré par le module LOF nommé : REGISTRE DE TYPE STATIQUE ET RAZ
  Paramètres fonctionnels :
  - Stockage sur 16 bits.
  Paramètres matériels :
  - Durée totale = 80 (ns).
  - Durée de set-up = 4 (ns).
  - Surface = 4.6864E+05 (microns 2).

- Opérateur numéro : 14, nommé :REG5
  Généré par le module LOF nommé : REGISTRE DE TYPE STATIQUE ET RAZ
  Paramètres fonctionnels :
  - Stockage sur 16 bits.
  Paramètres matériels :
  - Durée totale = 80 (ns).
  - Durée de set-up = 4 (ns).
  - Surface = 4.6864E+05 (microns 2).

OPERATEURS DE TYPE "BUS" :

- Opérateur numéro : 16, nommé : BUS1
  Renseigné par le module LOF nommé : BUS
  Paramètres matériels :
  - Durée totale garantie = 15 (ns).
  - Surface inconnue a priori.

- Opérateur numéro : 17, nommé : BUS2
  Renseigné par le module LOF nommé : BUS
  Paramètres matériels :
  - Durée totale garantie = 15 (ns).
  - Surface inconnue a priori.

- Opérateur numéro : 18, nommé : BUS3
  Renseigné par le module LOF nommé : BUS
  Paramètres matériels :
  - Durée totale garantie = 15 (ns).
  - Surface inconnue a priori.

SHEMA BLOC DE L'ARCHITECTURE DU FILTRE BIQUAD 2

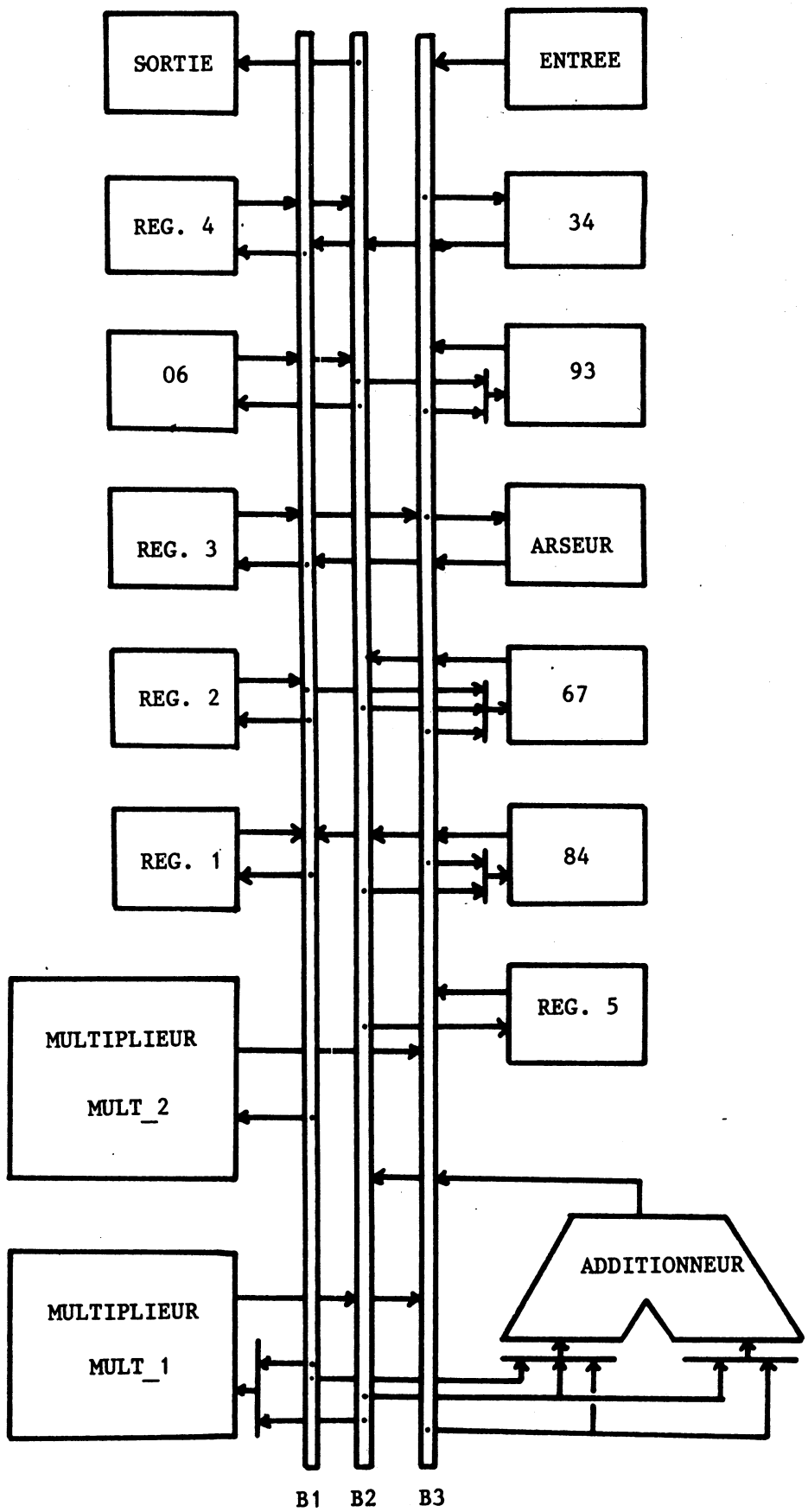
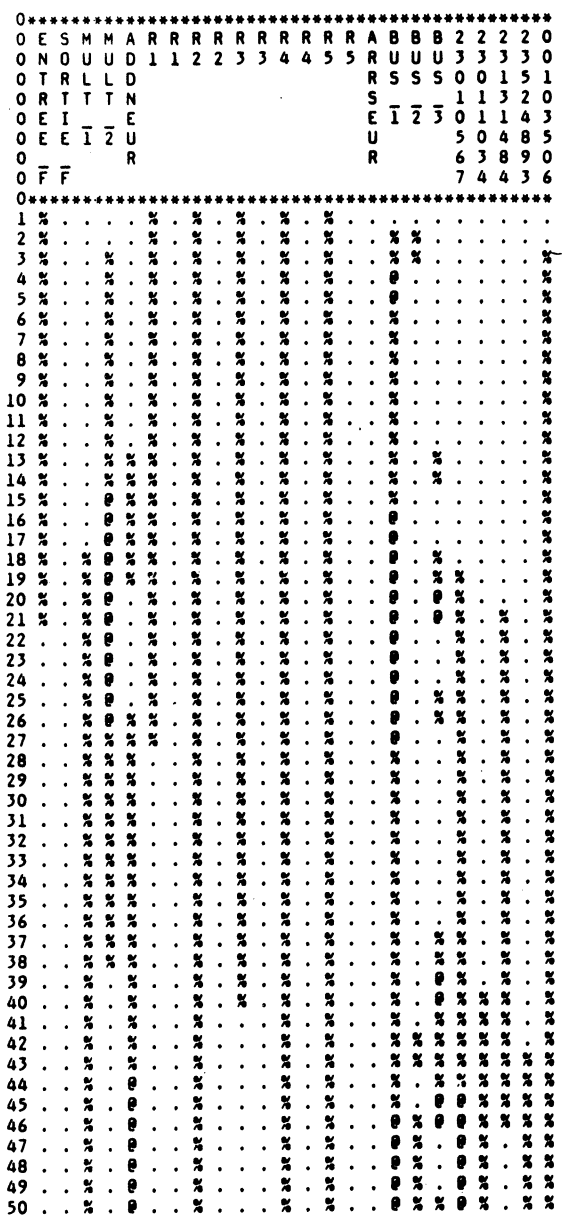
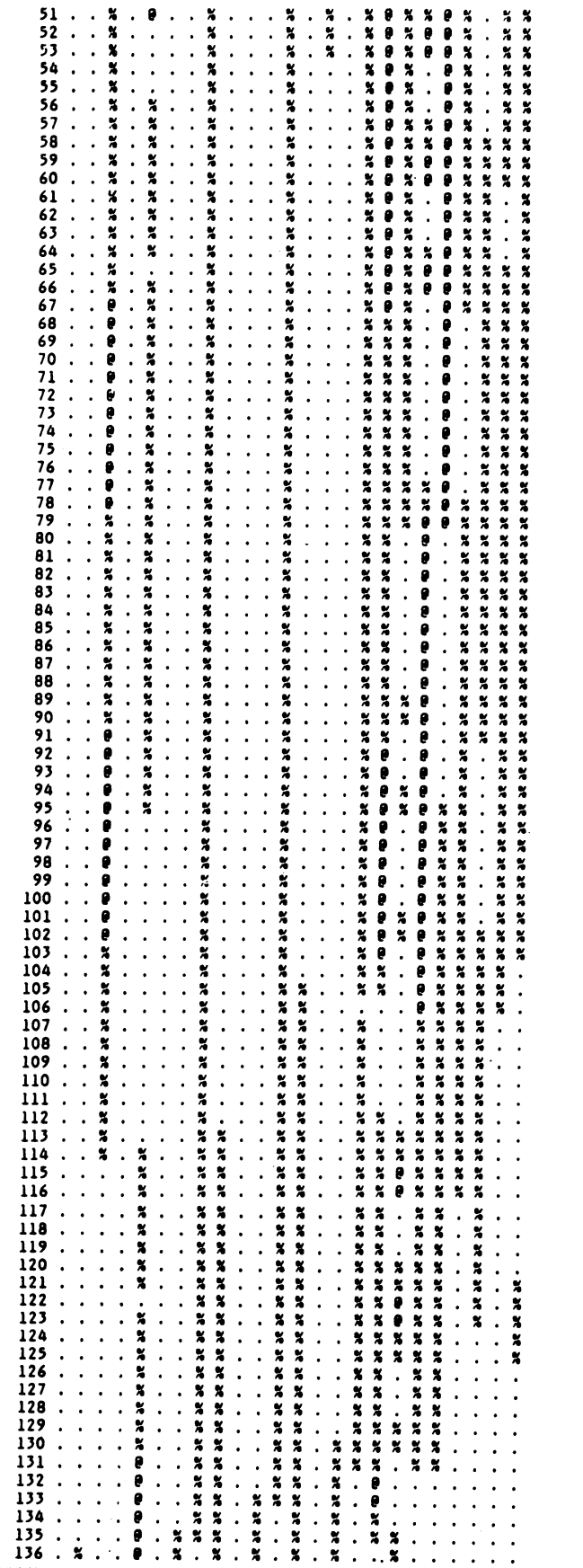


TABLEAU DE GANTT DE L'ARCHITECTURE DE "BIQUAD 2"





```
**********************************************************************
  POURCENTAGES D'UTILISATION DES OPERATEURS DE L'ARCHITECTURE
**********************************************************************
Opérateur nommé : ENTREE_F utilisé à  1.54412E+01 pour cent
Opérateur nommé : SORTIE_F utilisé à  7.35294E-01 pour cent
Opérateur nommé : MULT_1   utilisé à  7.13235E+01 pour cent
Opérateur nommé : MULT_2   utilisé à  2.64706E+01 pour cent
Opérateur nommé : ADDNEUR  utilisé à  6.91176E+01 pour cent
Opérateur nommé : ARRSEUR  utilisé à  5.88235E+01 pour cent
Opérateur nommé : BUS_1    utilisé à  9.41176E+01 pour cent
Opérateur nommé : BUS_2    utilisé à  4.26471E+01 pour cent
Opérateur nommé : BUS_3    utilisé à  6.17647E+01 pour cent
Opérateur nommé : 23010567 utilisé à  7.20588E+01 pour cent
Opérateur nommé : 23011034 utilisé à  4.92647E+01 pour cent
Opérateur nommé : 23131484 utilisé à  6.02941E+01 pour cent
```

JEU D'OPERATEURS PROPOSE POUR L'ARCHITECTURE :

OPERATEURS DE TYPE "ENTREE" :

- Opérateur numéro : 1, nommé : ENTREE F
  Généré par le module LOF nommé : ENTREE2
  Paramètres fonctionnels :
  - 24 bits en PARALLELE
  Paramètres matériels :
  - Durée totale = 30 (ns).
  - Durée de set-up = 14 (ns).
  - Surface = 2.1294E+05 (microns 2).

OPERATEURS DE TYPE "SORTIE" :

- Opérateur numéro : 2, nommé : SORTIE F
  Généré par le module LOF nommé : SORTIE2
  Paramètres fonctionnels :
  - 24 bits en PARALLELE
  Paramètres matériels :
  - Durée totale = 35 (ns).
  - Durée de set-up = 19 (ns).
  - Surface = 2.1294E+05 (microns 2).

OPERATEURS DE TYPE "MULTIPLIEUR" :

- Opérateur numéro : 3, nommé : MULT1
  Généré par le module LOF nommé : MULTIPLI1
  Paramètres fonctionnels :
  - Entrée sur 16 bits.
  - Coefficient sur 13 bits.
  - Valeur du coefficient = 0 (multiplieur général)
  - Sortie sur 24 bits.
  - Mode d'arrondi = TRONCATURE_COMPL_A_2
  Paramètres matériels :
  - Durée totale = 232 (ns).
  - Durée de latchage = 8 (ns).
  - Durée de set-up = 4 (ns).
  - Surface = 3.46E+06 (microns 2).

OPERATEURS DE TYPE "ADDITIONNEUR" :

- Opérateur numéro : 4, nommé : ADD_1
  Généré par le module LOF nommé : ADDITION
  Paramètres fonctionnels :
  - Entrée 1 sur 24 bits.
  - Entrée 2 sur 24 bits.
  - Sortie sur 24 bits.
  - Mode de contrôle d'overflow = RIEN
  Paramètres matériels :
  - Durée totale = 80 (ns).
  - Durée de latchage = 8 (ns).
  - Durée de set-up = 4 (ns).
  - Surface = 4.6846E+05 (microns 2).

- Opérateur numéro : 5, nommé : ADD_2
  Généré par le module LOF nommé : ADDITION
  Paramètres fonctionnels :
  - Entrée 1 sur 24 bits.
  - Entrée 2 sur 24 bits.
  - Sortie sur 24 bits.
  - Mode de contrôle d'overflow = RIEN
  Paramètres matériels :
  - Durée totale = 80 (ns).
  - Durée de latchage = 8 (ns).
  - Durée de set-up = 4 (ns).
  - Surface = 4.6846E+05 (microns 2).

OPERATEURS DE TYPE "REGISTRE MAITRE-ESCLAVE" :

- Opérateur numéro : 6, nommé :REG1
  Généré par le module LOF nommé : REGISTRE DE TYPE STATIQUE ET RAZ
  Paramètres fonctionnels :
  - Stockage sur 16 bits.
  Paramètres matériels :
  - Durée totale = 80 (ns).
  - Durée de set-up = 4 (ns).
  - Surface = 4.6846E+05 (microns 2).

- Opérateur numéro : 8, nommé :REG2
  Généré par le module LOF nommé : REGISTRE DE TYPE STATIQUE ET RAZ
  Paramètres fonctionnels :
  - Stockage sur 16 bits.
  Paramètres matériels :
  - Durée totale = 80 (ns).
  - Durée de set-up = 4 (ns).
  - Surface = 4.6864E+05 (microns 2).

- Opérateur numéro : 10, nommé :REG3
  Généré par le module LOF nommé : REGISTRE DE TYPE STATIQUE ET RAZ
  Paramètres fonctionnels :
  - Stockage sur 16 bits.
  Paramètres matériels :
  - Durée totale = 80 (ns).
  - Durée de set-up = 4 (ns).
  - Surface = 4.6864E+05 (microns 2).

- Opérateur numéro : 12, nommé :REG4
  Généré par le module LOF nommé : REGISTRE DE TYPE STATIQUE ET RAZ
  Paramètres fonctionnels :
  - Stockage sur 16 bits.
  Paramètres matériels :
  - Durée totale = 80 (ns).
  - Durée de set-up = 4 (ns).
  - Surface = 4.6864E+05 (microns 2).

- Opérateur numéro : 14, nommé :REG5
  Généré par le module LOF nommé : REGISTRE DE TYPE STATIQUE ET RAZ
  Paramètres fonctionnels :
  - Stockage sur 16 bits.
  Paramètres matériels :
  - Durée totale = 80 (ns).
  - Durée de set-up = 4 (ns).
  - Surface = 4.6864E+05 (microns 2).

OPERATEURS DE TYPE "BUS" :

- Opérateur numéro : 16, nommé : BUS1
  Renseigné par le module LOF nommé : BUS
  Paramètres matériels :
  - Durée totale garantie = 15 (ns).
  - Surface inconnue a priori.

- Opérateur numéro : 17, nommé : BUS2
  Renseigné par le module LOF nommé : BUS
  Paramètres matériels :
  - Durée totale garantie = 15 (ns).
  - Surface inconnue a priori.

- Opérateur numéro : 18, nommé : BUS3
  Renseigné par le module LOF nommé : BUS
  Paramètres matériels :
  - Durée totale garantie = 15 (ns).
  - Surface inconnue a priori.

SHEMA BLOC DE L'ARCHITECTURE DU FILTRE " BIQUAD 2 (bis)"

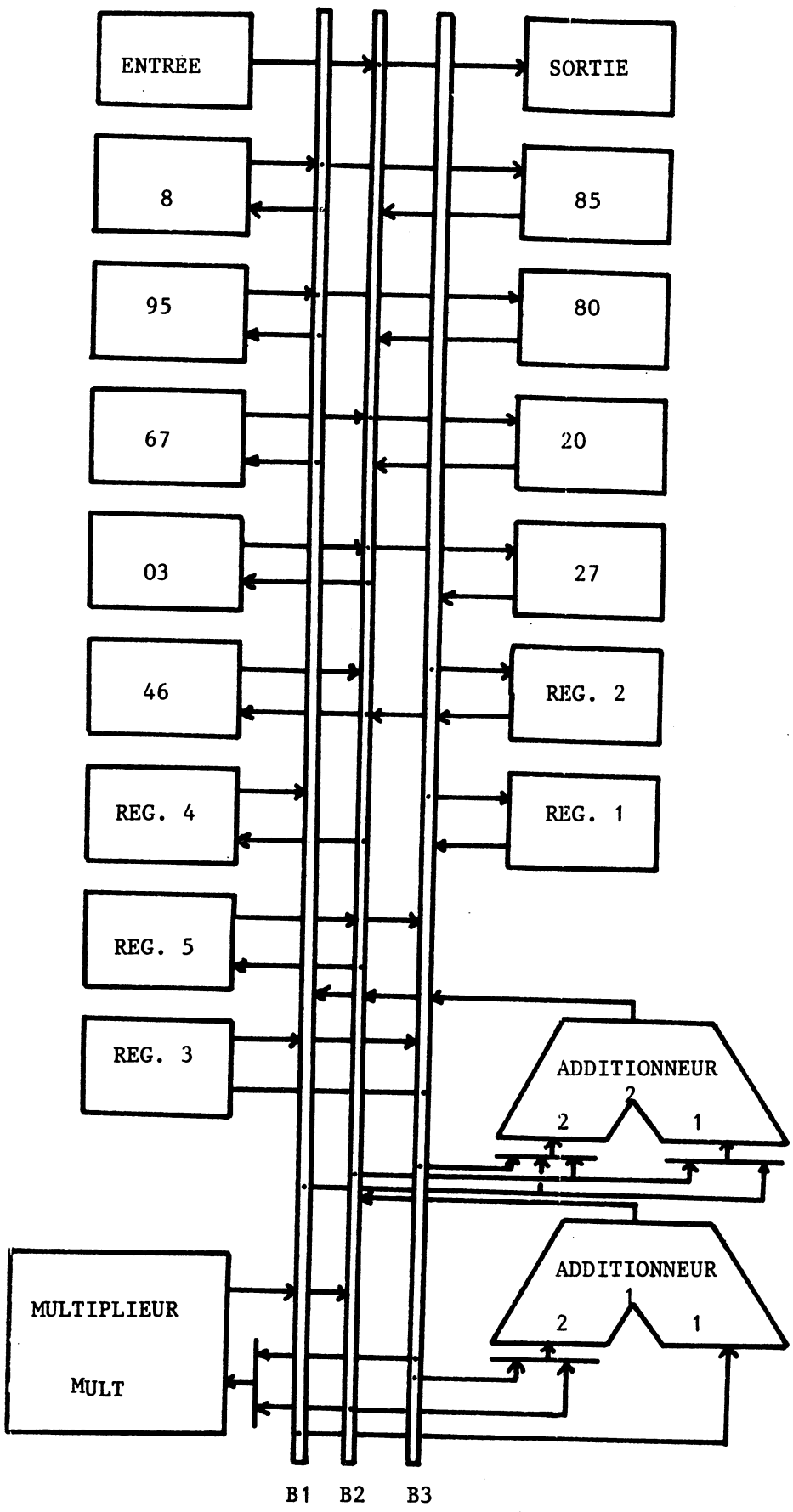
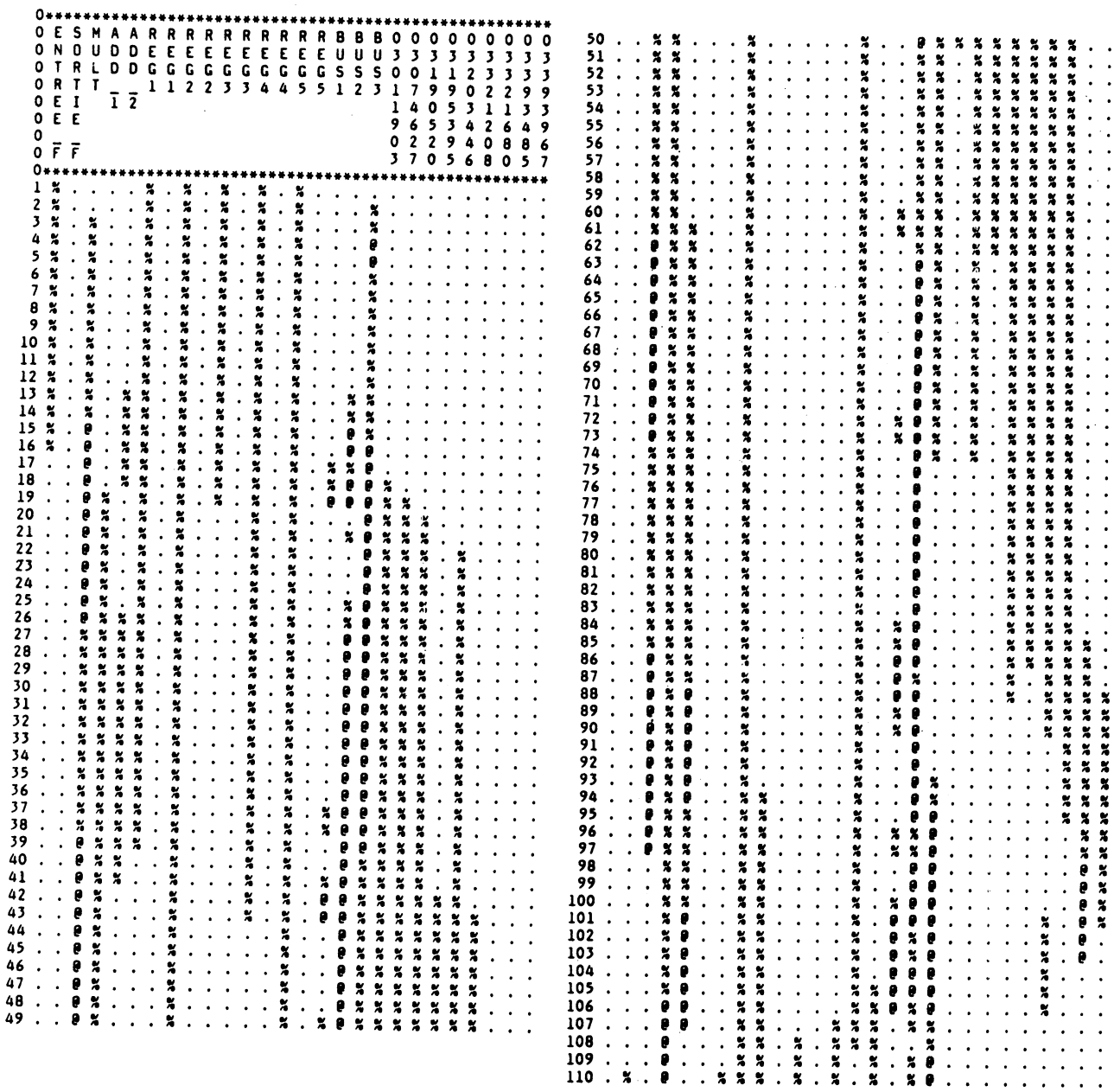


TABLEAU DE GANTT DE L'ARCHITECTURE DE "BIQUAD 2(v2)"




```
**************************************************************
  POURCENTAGES D'UTILISATION DES OPERATEURS DE L'ARCHITECTURE
**************************************************************
Opérateur nommé : ENTREE_F utilisé à  1.45455E+01 pour cent
Opérateur nommé : SORTIE_F utilisé à  9.09091E-01 pour cent
Opérateur nommé : MULT     utilisé à  8.63636E+01 pour cent
Opérateur nommé : ADD_1    utilisé à  8.36364E+01 pour cent
Opérateur nommé : ADD_2    utilisé à  6.27273E+01 pour cent
Opérateur nommé : BUS1     utilisé à  2.63636E+01 pour cent
Opérateur nommé : BUS2     utilisé à  8.45455E+01 pour cent
Opérateur nommé : BUS3     utilisé à  8.27273E+01 pour cent
Opérateur nommé : 03011903 utilisé à  3.00000E+01 pour cent
Opérateur nommé : 03074627 utilisé à  5.09091E+01 pour cent
Opérateur nommé : 03190520 utilisé à  3.90909E+01 pour cent
Opérateur nommé : 03195395 utilisé à  4.27273E+01 pour cent
Opérateur nommé : 03203446 utilisé à  5.90909E+01 pour cent
Opérateur nommé : 03321208 utilisé à  4.90909E+01 pour cent
Opérateur nommé : 0332168U utilisé à  4.18182E+01 pour cent
Opérateur nommé : 03393485 utilisé à  1.72727E+01 pour cent
Opérateur nommé : 03393967 utilisé à  1.27273E+01 pour cent
```

8-4 Conclusion

Les trois exemples traités ici nous permettent de juger les performances d'IMHOTEP sur des circuits de caractéristiques différentes.

Si FIR-11 possède un fort taux de répétitivité dans le réseau d'opérations, BIQUAD2 et LOUVAIN sont constitués de motifs plus importants, répétés peu de fois, et entourés d'opérations singulières.

De par son mode de fonctionnement, IMHOTEP est incapable de tirer parti de la répétitivité d'un motif, et il se contente de réaliser les opérations selon leur degré d'urgence et leur priorité relative.

Par contre, IMHOTEP est très sensible au nombre de registres fonctionnels présents dans la description du filtre, ainsi qu'à leur situation dans le réseau : Nous avons étudié un exemple de filtre en treillis dans lequel nous avions l'alternance RETARD---ADDITIONNEUR répétée une dizaine de fois. La configuration était donc différente de celle de FIR-11 pour lequel tous les registres sont bout à bout. Dans le cas du filtre en treillis, l'apparition d'une suite "Registre-fonctionnel, Registre simple latch" était fréquente, et ceci était refusé par IMHOTEP qui mettait donc longtemps à trouver un chemin dans l'arbre ne contenant pas de telle chaîne. La solution dans ce cas consistait à diminuer la priorité de ces chaînes (paramètre PLAGE-TEMPS-MAX).

Nous constatons donc que l'usager d'IMHOTEP doit savoir régler les paramètres à sa disposition en fonction des caractéristiques du filtre.

Remarquons également que le nombre considérable de tentatives faites par IMHOTEP est dû au manque de perception globale du circuit, perception qui est un atout des concepteurs humains. Des techniques de programmation permettant d'augmenter cette perception au moyen de listes de règles scrutées automatiquement devraient améliorer spectaculairement les temps de calcul.

Néanmoins, les règles câblées et paramétrées qui guident actuellement IMHOTEP sont assez satisfaisantes si elles sont bien connues et bien utilisées par l'usager.

Nous ne pourrons malheureusement pas donner de schéma d'implantation des architectures produites, car la bibliothèque d'opérateurs n'est pas encore complètement réalisée. Un travail est actuellement fait pour la remplir de cellules vides mais possédant une enveloppe et des positions d'entrée-sortie réalistes. Nous pourrons alors présenter une résultat complet du compilateur de silicium associé à IMHOTEP.

9 - CONCLUSION

Nous venons d'exposer l'étude et la réalisation d'un générateur automatique d'architectures pour circuits intégrés de filtrage numérique. Cet outil logiciel automatise le passage d'une description comportementale d'un filtre à celle de la structure de la machine réalisant l'algorithme de filtrage désiré. Il accepte de plus une contrainte de temps exprimée par la fréquence d'échantillonnage du filtre, et il produit une machine à surface de silicium minimale.

Nous avons proposé une approche du problème par des techniques d'ordonnancement de tâches sous contraintes de précédence et de ressources. La modélisation de la génération d'architectures en ces termes étant raisonnablement considérée comme un problème NP-complet, nous avons adopté une méthode heuristique.

Cette approche est elle-même décomposée en deux parties ; la première prend une décision quant à la prochaine tâche à traiter et aux moyens à lui consacrer, et la deuxième met en oeuvre cette décision sur la structure de données.

La programmation en PASCAL a permis une réalisation performante de l'exécution de la décision, mais se prête mal à l'élaboration de celle-ci. En effet, l'expression de nombreuses règles aux priorités changeantes se fait mal via des langages de programmation algorithmique. Nous avons partiellement pallié à cette difficulté en procédant à une exploration coûteuse d'un nombre important de possibilités. Ceci se traduit évidemment par des temps de calcul très importants.

Malgré ces réserves, le fonctionnement du prototype IMHOTEP est satisfaisant (il explore environ une solution par minute pour un circuit de 100 tâches). Une poursuite de cette étude pourra porter sur l'amélioration des mécanismes d'élaboration de la décision, et sur l'utilisation de techniques non algorithmiques proches de l'intelligence artificielle. De nombreuses autres possibilités existent pour rendre les architectures produites plus performantes. Nous pensons en particulier à l'utilisation de bancs de mémoire à la place de registres maitre-esclaves dans les cas où ceux-ci sont très nombreux. D'autres solutions architecturales (architectures séries ou distribuées) méritent également d'êtres utilisées dans IMHOTEP.

Certaines optimisations concernant l'élaboration de la partie contrôle du circuit devraient être payantes, notamment sur la synchronisation à une horloge choisie automatiquement, à une ou plusieurs phases.

Ce travail montre néanmoins que des voies existent vers l'automatisation de la conception d'architectures d'après une description fonctionnelle des systèmes.

BIBLIOGRAPHIE DU CHAPITRE 1

B1.1 - VLSI CIRCUIT DESIGN REACHES THE LEVEL OF ARCHITECTURAL DESCRIPTION.
Stephens c. Johnson (Silicon compilers inc.)
Electronics, May 3, 1984. pp 121-128

B1.2 - THE MICRO-VAX DATA-PATH CHIP.
Glenn Louie, Tom Ho, Ed Cheng (Silicon compilers inc.)
VLSI design Dec 1983 pp 14-21

B1.3 - SUPERCHIPS FACE DESIGN CHALLENGE.
John G. POSA
HIGH TECHNOLOGY Jan 1983

B1.4 - LES CIRCUITS A LA DEMANDE ATTEIGNENT LA MATURITE.
Electronique industrielle no 46 Févr 1983

BIBLIOGRAPHIE DU CHAPITRE 2

B2.1 - BRISTLE BLOCKS - A SILICON COMPILER.
D. L. Johannsen
IEEE proc of 16th D. A. Conf. Jun 1979 pp 310-313

B2.2 - GENERATING CUSTOM HIGH PERFORMANCE VLSI DESIGN FROM SUCCINT ALGORITHMIC DESCRIPTIONS.
Siskind, Southard, Crouch
Conf. on advanced research MIT Jan 1982

B2.3 - THE MACPITTS SILICON COMPILER - A VIEW FROM THE TELECOMMUNICATION INDUSTRY.
Jeffrey R. Fox
VLSI Design May-June 1983 pp 30-37

B2.4 - A UNIFIED BOX OF CAD TOOLS FOR THE DESIGN OF DEDICATED SIGNAL PROCESSING CHIPS.
H. de Man, J. Vandewalle
IEEE proc. of ICCD Oct 1984 pp 838-844

B2.5 - A SYSTEMATIC APPROACH TO DESIGN AND SPEED COMPARAISON OF SIGNAL PROCESSOR ARCHITECTURE FOR DIGITAL FILTERING.
J. Zeman, G.S. Moschytz
IEEE proc. of ISCAS 1981 pp 729-732

B2.6 - THE STRUCTURE OF A SILICON COMPILER.
D. Gajki
IEEE proc. of D.A. conf. 1982 pp 272-276

B2.7 - A SILICON COMPILER FOR VLSI SIGNAL PROCESSORS.
P.B. Denyer, D. Renshaw, N. Bergmann
proc. of ESSIRC 1982 pp 215-218

B2.8 - A TOOL FOR THE AUTOMATED SYSTHESIS OF DIGITAL FILTERS.
G. Michel, J-F Reyss-brion
2nd Int. Workshop on VLSI in telecommunications (NY Jun 1983)

B2.9 - A FLEXIBLE SILICON COMPILER FOR DIGITAL PROCESSING CIRCUITS.
M. Glesner, J. Schuck, H. Joepen
IEEE proc. of ICCD 1984 pp 845-850

B2.10- SAGA : AN EXPERIMENTAL SILICON ASSEMBLER.
Antoni A. Szepieniec
IEEE proc. of D.A. conf. 1982 pp 365-370

B2.11- STEPWISE LAYOUT REFINIMENT.
Lukas P.P.P. Van Ginneken, R. Otten
IEEE proc. of ICCD 1984 pp 30-36

BIBLIOGRAPHIE DU CHAPITRE 3

Théorie générale du filtrage

B3.1 - M.BELLANGER "TRAITEMENT NUMERIQUE DU SIGNAL".
(EDITIONS ENST-CNET)
B3.2 - ANALYSIS OF LINEAR DIGITAL NETWORKS.
R. E. CROCHIERES A. OPPENHEIM (PROC. IEEE AVR. 75)
B3.3 - A GENERAL THEORIE OF BLOCK-STATE DIGITAL FILTERS.
S. K. MITRA (ISCAS 82)

Filtres en chaines de quadripôles

B3.4 - SOME ASPECTS OF THE LSI IMPLEMENTATION OF WAVE DIGITAL FILTERS.
BJORN SIKSTROM (ISCAS 82)
B3.5 - AN IN-DEPTH STUDY OF THE SENSITIVITY OF WDF.
SYDNEY R. PARKER (ASILOMAR 79)
B3.6 - TRANSFERT FUNCTION OF WAVE DIGITAL FILTERS.
E. C. TAN (ELECTRON. LETTERS AUG 82)

Filtres RII - études théoriques

B3.7 - SYNTHESIS OF MINIMUM ROUNDOFF NOISE FIXED POINT DIGITAL FILTERS.
C. T. MULLIS R.A. ROBERTS (IEEE TR. ON C&C SEP 76)
B3.8 - DIGITAL FILTERS REALISATIONS WITHOUT OWERFLOW OSCILLATIONS.
C. T. MULLIS R.A. ROBERTS (IEEE TR. ASSP 4 AOU 78)
B3.9 - FAST DIGITAL FILTERS WITH LOW ROUNDOFF NOISE.
J. ZEMAN (ASILOMAR 80)

Sensibilité aux pôles

B3.10- A SYNTHESIS OF DIGITAL FILTERS WITH MINIMUN POLE SENSITIVITY.
A. NISHIHARA K. SUGAHARA (ISCAS 82)

Etude des effets de la quantification

B3.11- QUANTIZATION SHEMES FOR RECURSIVES DIGITAL FILTERS.
V.B. LAWRENCE (BELL) (ISCAS 82)
B3.12- QUANTIZATION EFFECTS IN COMPUTATIONNALLY EFFICIENT REALIZATIONS OF INFINITE RESPONSE FILTERS.
B. LIU & R. ANSARI (ISCAS 82)
B3.13- TWO METHODS FOR REDUCTION OF QUANTIZATION EFFECTS IN IIR FILTERS.
A. ANTONIOU (ISCAS 82)
B3.14- A PRACTICAL DIGITAL FILTER WITH NEAR OPTIMAL ROUNDING NOISE CANCELLATION.
L. MINTZER & J.P. STRAUSS (ASILOMAR 80)
B3.15- LOW ROUNDOFF NOISE AND NORMAL REALISATIONS OF FIXED POINT INFINITE RESPONSE DIGITAL FILTERS.
C.T. MULLIS & R.A. ROBERTS (IEEE TR. ASSP 4 AUG 81)
B3.16- SYNTHESIS OF MINIMUM ROUNDOFF NOISE FIXED POINT DIGITAL FILTERS.
C. T. MULLIS R.A. ROBERTS (IEEE TR. ON C&C SEP 76)
B3.17- QUANTIZATION SHEMES FOR RECURSIVES DIGITAL FILTERS.
V.B. LAWRENCE (BELL) (ISCAS 82)
B3.18- A NEW STATISTICAL APPROACH TO THE COEFFICIENT WORDLENGH PROBLEM FOR DIGITAL FILTERS.
R. E. CROCHIERE (IEEE TRANS C&C MARS 75)

Cycles limites et oscillations

B3.19- DIGITAL FILTERS REALISATIONS WITHOUT OWERFLOW OSCILLATIONS.
C. T. MULLIS R.A. ROBERTS (IEEE TR. ASSP 4 AOU 78)
B3.20- SECOND ORDER IIR FILTER FREE FROM CONSTANT INPUT LIMIT CYCLES.
L. E. TURNER (ELECTR. LETTERS AUG 82)

Comparaison entre les différents types de filtres

B3.21- MONOLITHIC INTEGRATED FILTERS.
E. LUEDER (FROM ELECT TO MICROELECTRONICS)
B3.22- COMPARAISON OF THREE TYPES OF DIGITAL FILTER STRUCTURES.
M.G. REZK A. ANTONIOU (ISCAS 79)

Structures particulières

Structures sans multiplieurs

B3.23- FAST DIGITAL FILTERS WITHOUT MULTIPLIERS.
E. LUEDER (ISCAS 82)

Architectures data-flow

B3.24- PERFORMANCE OF AN EXPERIMENTAL DATA FLOW ARCHITECTURE FOR SIGNAL PROCESSING.
K. KRONLOF (ICASSP 82)
B3.25- SIMULATION OF A DIGITAL SIGNAL PROCESSING ARCHITECTURE BASED ON DATA FLOW.
K. KRONLOF (ISCAS 82)

Réalisation de circuits spécialisés dans le filtrage numérique

Circuits monolithiques

B3.26- REALISATION D'UN COFIDEC NUMERIQUE.
P. SENN (NT/CNET/GRE/31)
B3.27- AN EXPANDABLE SINGLE IC DIGITAL FILTER/CORRELATOR.
F. A. WILLIAMS TRW (ICASSP 82)
B3.28- AN NMOS DIGITAL FILTER CIRCUIT.
BRITSH POST OFFICE (EUROCON 80)

Circuits à base de boitiers commerciaux

B3.29- AN ARCHITECTURE FOR HIGH SPEED SIGNAL PROCESSING DEVICE.
S. A. WHITE ROCKWELL INT. (ISCAS 81)
B3.30- EFFICIENT HARDWARE IMPLEMENTATION FOR FIXED POINT DIGITAL FILTERS.
R. A. ROBERTS (ISCAS 80)
B3.31- HIGH SPEED STORED PRODUCT RECURSIVE DIGITAL FILTERS.
D. DUBOIS & W. STEENAART (IEEE TR. ON C&C JUN 82)

Circuits de traitement du signal programmables

B3.32- A DSP FOR TELECOMMUNICATIONS APPLICATIONS.
JAMES R. BODDIE (ISSCC 80)
B3.33- NOVEL STRUCTURE OF A USER-PROGRAMMABLE INTEGRATED DSP.
R. GEPPERT (ICASSP 82)
B3.34- THE DSP ARCHITECTURE AND APPLICATIONS OWERWIEW.
ISMAIL I. ELDUMIATI (ISCAS 81)
B3.35- PROGRAMMABLE SIGNAL PROCESSOR LSI RIVALS ANALOG-CIRCUITS FILTERS.
GWYN P. EDWARDS (ELECTRON. DESIGN JUN 80)
B3.36- A SINGLE CHIP DSP FOR VOICE-BAND APPLICATIONS.
YUICHI KAWAKAMI (ISSCC 80)

BIBLIOGRAPHIE DU CHAPITRE 4

LE LANGAGE SIGNAL :

B4.1 - ALGEBRA OF EVENTS, A MODEL FOR PARALLELE AND REAL TIME SYSTEMS.
Caspi & Halbwachs
proc. of int. conf. on parallele processing.
1982 pp 150-159

B4.2 - REAL TIME LANGUAGES, DESIGN AND DEVELOPPEMENT.
S. J. Young
Elis Horwood Publishers 1982.

B4.3 - SIGNAL : DESCRIPTION ALGEBRIQUE DES FLOTS DE SIGNAUX.
P. Le-Guernic
AFCET, Nov 1982 pp 243-252

B4.4 - SIGNAL : A DATA-FLOW ORIENTED LANGUAGE FOR SIGNAL PROCESSING.
T. Gauthier
IRISA Technical report 1984

LE SYSTEME SIRENA :

B4.5 - SYSTEME CONVERSATIONNEL AVEC VISUALISATION GRAPHIQUE D'AIDE A L'ENSEIGNEMENT ET A LA RECHERCHE EN AUTOMATIQUE.
J. P. Calvez, Y. Thomas, R. Gerber
Annales ENSM 1972

B4.6 - INTERACTIVE SYSTEM AS AID TO TEACHING AUTOMATIC CONTROL.
R. Gerber, Y. Quenec Hdu, Y. Thomas
Proc. of IFAC symp. Barcelone 1977 pp 106-125

B4.7 - COMPUTED AIDED DESIGN IN AUTOMATIC CONTROL.
J. P. Le Baron, C. Brie, R. Gerber
Sem. on real time process control
Varsovie 1978 pp 259-271

LE SYSTEME CASSIOPEE :

B4.8 - CASSIOPEE : AN INTEGRATED CAD SYSTEM FOR INTEGRATED CIRCUITS.
B. Hennion, G. Mazaré
Proc. of ESSIRC 1980

B4.9 - A DESIGN METHODOLOGY BASED UPON SYMBOLIC LAYOUT AND INTEGRATED CAD TOOLS.
A.M. Beyls, B. Hennion, J. Lecourvoisier, G. Mazaré, A. Puissochet
IEEE proc. of D.A. Conf. 1982 pp 872-878

B4.10- CASSIOPEE - A-PRESENTATION GENERALE NT/CNS/CCI/27
- B-MANUEL D'UTILISATION NT/CNS/CCI/28
- C-STRUCTURATION, REALISATION ET METHODES D'ACCES A LA BASE DE DONNEES. NT/CNS/CCI/29
M. Albareil, J. Lecourvoisier, A. Puissochet, C. Julien, D. Sallée
1984

BIBLIOGRAPHIE DU CHAPITRE 5

B5.1 - LOF : A BUILDING TOOL FOR FLEXIBLE BLOCKS LIBRARIES.
J.M. Berge, L.O. Donzelle, G. Michel, J. Rouillard, D. Rouquier.
IEEE proc. of ICCD 1984 pp 851-856

B5.2 - PICTURES WITH PARENTHESES, COMBINING GRAPHIC & PROCEDURES IN A VLSI LAYOUT TOOL.
R.N. Mayo, J.K. Ousterhout.
IEEE proc. of D.A. conf. 1983 pp 270-276

B5.3 - BIBLIOTHEQUES DE CELLULES.
J-F Reyss-brion
Rapport IMAG de contrat DAII 82PE215 Lot no 2 1983

BIBLIOGRAPHIE DU CHAPITRE 6

B6.1 - SCHEDULING SUBJECT TO RESOURCE CONSTRAINTS : CLASSIFICATION AND COMPLEXITY.
J. Blazewicz, J.K. Lenstra, A.H.G. Rinnooy Kan
Discrete Applied Math. 5, 1983 pp 11-24

B6.2 - CRITICAL PATH SCHEDULING WITH RESOURCE AND PROCESSOR CONSTRAINTS.
E.L. Lloyd
Journ. Assoc. comput. mach. 29, 1982 pp 781-811

B6.3 - CONCURRENT TASK SYSTEMS.
E.L. Lloyd
Oper. Research 29, 1981 pp 189-201

B6.4 - SCHEDULING SUBJECT TO NONRENEWABLE RESOURCE CONSTRAINTS.
J. Carlier, A.H.G. Rinnooy Kan
Oper. Research Letter 1, 1982 pp 52-55

B6.5 - OPTIMIZATION AND APPROXIMATION IN DETERMINISTIC SEQUENCING AND SCHEDULING : A SURVEY.
R.L. Graham, E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan
Ann. of Discrete Math. 5, 1979 pp 287-326

B6.6 - COMPLEXITY RESULTS FOR MULTIPROCESSOR SCHEDULING UNDER RESOURCE CONSTRAINTS.
M.R. Garey, D.S. Johnson
SIAM Journ. Comput. 4, 1975 pp 397-411

B6.7 - A SYSTEMATIC APPROACH TO DESIGN AND SPEED COMPARAISON OF SIGNAL PROCESSOR ARCHITECTURE FOR DIGITAL FILTERING.
J. Zeman, G.S. Moschytz
IEEE proc. of ISCAS 1981 pp 729-732

BIBLIOGRAPHIE DU CHAPITRE 7

B7.1 - CHEMINEMENT ET CONNEXITE DANS LES GRAPHES.
B. Roy
Thèse université de Paris METRA 1962

B7.2 - EXERCICES ET PROBLEMES DE RECHERCHE OPERATIONNELLE.
G. Desbazeilles
DUNOD 1972

B7.3 - PLUS COURTS CHEMINS AVEC CONTRAINTES
M. Minoux
Annales des télécoms. 1975

B7.4 - GRAPHES ET ALGORITHMES.
M. Gondran, M. Minoux
Coll. de la dir. des Etudes et Recherches EDF - EYROLLES

ANNEXE 1

PROGRAMME TRADUCASS.PAS-----

Le circuit décrivant le filtre est extrait de la base CASSIOPEE nommée

DISQUE:[REYSS.FILTRE.CASSIO]FILT.TYP;
DISQUE:[REYSS.FILTRE.CASSIO]FILT.COP;
DISQUE:[REYSS.FILTRE.CASSIO]FILT.EQI;

et est stocké dans le fichier nommé MONFILTRE.TRA si NOMFILTRE est le nom que vous avez donné en réponse à la question posée.

FORMAT DU FICHIER :

Ce fichier est un fichier de type TEXT , et est donc éditable.
Il contient ligne par ligne les renseignements suivants :

(début du fichier)

NOM DU FILTRE (sur 8 lettres),
DATE DE CREATION (sur 11 lettres),
HEURE DE CREATION (sur 11 lettres),
(CR)
NOMBRE DE TACHES DU CIRCUIT (entier),
(CR)
(puis, pour chaque tache répéter :)
  *NUMERO DE LA TACHE (entier),
  *NOM DE LA TACHE (sur 8 lettres),
  *ESPECE DE LA TACHE (sur 8 lettres),
  *NUMERO D'ESPECE DE LA TACHE (entier),
  *RANG DE LA TACHE DANS LE GRAPHE (entier),
  *NOMBRE D'ANTECEDENTS DE LA TACHE (entier),
  *(CR)
  *(pour chaque tache antécédante :)
  * *NUMERO DE LA TACHE ANTECEDANTE (entier),
  * *(CR)
  *NOMBRE DE SUCCESSEURS DE LA TACHE (entier),
  *(CR)
  *(pour chaque tache succédante :)
  * *NUMERO DE LA TACHE SUCCEDANTE (entier),
  * *(CR)
(fin de la tache)

(fin du fichier).

COMMANDE DE RUN : = TRADUCASS

TRADUCASS:==@DISQUE:[REYSS.FILTRE.LANG]TRADUCASS.COM

ANNEXE 2

PROGRAMME DATAQUEST.PAS-----

Le fichier décrivant le filtre est nommé MONFILTRE.TRA, et il a été produit par le programme TRADUCASS.PAS à partir de la description CASSIOPEE du filtre.

COMMANDE DE RUN : = DATAQUEST

DATAQUEST:==@DISQUE:[REYSS.FILTRE.LANG]DATAQUEST.COM

FORMAT DU FICHIER :

Ce fichier est un fichier de type TEXT , et est donc éditable. Il contient ligne par ligne les renseignements suivants :

(début du fichier)

NOM DU FILTRE (sur 8 lettres),
DATE DE CREATION (sur 11 lettres),
HEURE DE CREATION (sur 11 lettres),
(CR)
DUREE ENTRE CHAQUE ARRIVEE D'UN ECHANTILLON (réel),
(CR)
NOMBRE DE TACHES DU CIRCUIT (entier),
(CR)
(puis, pour chaque tache répéter :)
*NUMERO DE LA TACHE (entier),
*NOM DE LA TACHE (sur 8 lettres),
*ESPECE DE LA TACHE (sur 8 lettres),
*NUMERO D'ESPECE DE LA TACHE (entier),
*(CR)
*RANG DE LA TACHE DANS LE GRAPHE (entier),
*NOMBRE D'ANTECEDENTS DE LA TACHE (entier),
*NOMBRE DE SUCCESSEURS DE LA TACHE (entier),
*(CR)
*(paramètres de la tache : (par espèce))
* *(entrée :)
* * MODE DE TRANSMISSION (type énuméré ECHANGETYP ...cf déclarations),
* * NOMBRE DE BITS EN ENTREE (entier),
* * (CR)
* *(sortie :)
* * MODE DE TRANSMISSION (type énuméré ECHANGETYP ...cf déclarations),
* * NOMBRE DE BITS EN SORTIE (entier),
* * (CR)

```
  * *(multiplication : )
  * *  NOMBRE DE BITS EN ENTREE (entier),
  * *  NOMBRE DE BITS DE COEFFICIENT (entier),
  * *  NOMBRE DE BITS EN SORTIE (entier),
  * *  VALEUR DU COEFFICIENT (réel),
  * *  MODE D'ARRONDI EN SORTIE (type énuméré ARRONTYP ...cf déclarations),
  * *  (CR)
  * *(additions : )
  * *  NOMBRE DE BITS EN ENTREE 1 (entier),
  * *  NOMBRE DE BITS EN ENTREE 2 (entier),
  * *  NOMBRE DE BITS EN SORTIE   (entier),
  * *  MODE DE CONTROLE DU DEBORDEMENT (type énuméré OVFTYP ...cf déclarations),
  * *  (CR)
  * *(registre : )
  * *  NOMBRE DE BITS A STOCKER (entier),
  * *  GENRE DE L'EXTREMITE DE REGISTRE M-E (1=esclave , 2=maitre),
  * *  (CR)
  * *(opérateur d'arrondi : )
  * *  NOMBRE DE BITS EN ENTREE (entier),
  * *  NOMBRE DE BITS EN SORTIE (entier),
  * *  MODE D'ARRONDI (type énuméré ARRONTYP ...cf déclarations),
  * *  (CR)
  * *(opérateur de changement de signe : )
  * *  NOMBRE DE BITS EN ENTREE (entier),
  * *  (CR)
  * *(décaleur : )
  * *  NOMBRE DE BITS EN ENTREE (entier),
  * *  NOMBRE DE RANGS DE DECALAGE (entier),
  * *  NOMBRE DE BITS EN SORTIE (entier),
  * *  MODE D'ARRONDI EN SORTIE (type énuméré ARRONTYP ...cf déclarations),
  * *  SENS DU DECALAGE         (type énuméré SENSTYP ...cf déclarations),
  * *  (CR)
  *(fin des paramètres)
  *(pour chaque tache antécédante : )
  * *NUMERO DE LA TACHE ANTECEDANTE (entier),
  * *NUMERO DE L'ENTREE ATTEINTE (entier),
  * *(CR)
  *(pour chaque tache succédante : )
  * *NUMERO DE LA TACHE SUCCEDANTE (entier),
  * *NUMERO DE L'ENTREE ATTEINTE SUR CETTE TACHE (entier),
  * *(CR)
(fin de la tache courante)

(fin du fichier).
```

LISTE DES PARAMETRES DEMANDES PAR DATAQUEST.PAS :

Ces paramètres dépendent de la nature de l'opérateur. Ils ont déjà été évoqués dans le format du fichier produit par DATAQUEST, mais nous allons les expliciter davantage :

Le temps qui s'écoule entre l'arrivée de 2 échantillons consécutifs et qui est la période d'échantillonnage du filtre est demandé à ce niveau.
C'est un nombre réel exprimé en nanosecondes.

Les paramètres suivants dépendent de l'opérateur :

Opérateurs de type "ENTREE" :

- EN_TRANSMI = Mode de transmission de l'échantillon.
  Il appartient au type énuméré ECHANGETYP qui contient les modes suivants : PARALLELE et SERIE
  Il convient de taper l'un ou l'autre de ces modes.
- EN_ENTREE = Nombre de bits en entrée.
  Entrer un entier.

Opérateur de type "SORTIE" :

- SO_TRANSMI = Mode de transmission de l'échantillon.
  Il appartient au type énuméré ECHANGETYP qui contient les modes suivants : PARALLELE et SERIE
  Il convient de taper l'un ou l'autre de ces modes.
- SO_ENTREE = Nombre de bits en sortie.
  Entrer un entier.

Opérateur de type "MULTIPLICATION" :

- MU_ENTREE = nombre de bits en entrée de la multiplication (entier).
- MU_COEFF = nombre de bits du coefficient de la multiplication (entier).
- MU_SORTIE = nombre de bits en sortie de la multiplication (entier).
- MU_COEFFVAL = valeur réelle du coefficient de la multiplication (réel).
- MU_ARMODE = mode d'arrondi éventuel du résultat.
  Il appartient au type énuméré ARRONTYP qui contient les modes suivants :
  AUCUN,ARRONDI_COMPL_A_2,ARRONDI_SIGNE_AMPL,ARRONDI_MATH,
  TRONCATURE_COMPL_A_2,TRONC_NS_SIGNE_AMPL,TRONC_S_SIGNE_AMPL,ALEATOIRE.

Opérateur de type "ADDITION" :

- AD_ENTREE1 = nombre de bits sur l'entrée numéro 1 (entier).
- AD_ENTREE2 = nombre de bits sur l'entrée numéro 2 (entier).
- AD_SORTIE = nombre de bits en sortie (entier).
- AD_OVERFL = mode de contrôle d'un éventuel débordement.
  Il appartient au type énuméré ARRONTYP qui contient les modes suivants :
  SATURATION,OVERFLOW_DETECTION,DETECTION_SATURATION,BUS_WIDENING,RIEN.

Opérateur de type "REGISTRE" :

- RE_BITSTOCK = nombre de bits de capacité du registre (entier).

Opérateur de type "ARRONDI" :

- AR_ENTREE = nombre de bits sur l'entrée (entier).
- AR_SORTIE = nombre de bits en sortie (entier).
- AR_ARMODE = mode d'arrondi éventuel du résultat.
  Il appartient au type énuméré ARRONTYP qui contient les modes suivants :
  AUCUN,ARRONDI_COMPL_A_2,ARRONDI_SIGNE_AMPL,ARRONDI_MATH,
  TRONCATURE_COMPL_A_2,TRONC_NS_SIGNE_AMPL,TRONC_S_SIGNE_AMPL,ALEATOIRE.

Opérateur de type "CHANGEMENT DE SIGNE" :

- CH_MOTBITS = nombre de bits en entrée (entier).

Opérateur de type "DECALAGE" :

- DE_ENTREE = nombre de bits en entrée du décalage (entier).
- DE_NBRANGS = nombre de rangs de décalage sur l'entrée (entier).
- DE_SORTIE = nombre de bits en sortie du décalage (entier).
- DE_SENS = sens de ce décalage.
  Il appartient au type énuméré SENSTYP qui contient les modes suivants :
  POSITIF,NEGATIF,GENERAL.
- DE_ARMODE = mode d'arrondi éventuel du résultat.
  Il appartient au type énuméré ARRONTYP qui contient les modes suivants :
  AUCUN,ARRONDI_COMPL_A_2,ARRONDI_SIGNE_AMPL,ARRONDI_MATH,
  TRONCATURE_COMPL_A_2,TRONC_NS_SIGNE_AMPL,TRONC_S_SIGNE_AMPL,ALEATOIRE.

FIN DE LA LISTE DES PARAMETRES.

ANNEXE 3

PROGRAMME CREATESOU.PAS-----

Le fichier décrivant le filtre est nommé MONFILTRE.DAT, et il a été produit par le programme DATAQUEST.PAS à partir du fichier MONFILTRE.DAT issu de TRADUCASS.

Ce fichier est lu par CREATESOU.PAS qui va ,pour chaque opération, demander à LOF les opérateurs contenus dans la bibliothèque et susceptibles de réaliser cette opération.
La bibliothèque est nommée FILTRE.CLF et se trouve dans le directory

DISQUE:[REYSS.FILTRE.LANG]

ainsi que le source et l'exécutable de CREATESOU. Ce programme procède donc à une consultation interactive de FILTRE.CLF, et fourni à l'usager des renseignements qui lui permettent de juger de l'adéquation de chaque opérateur à chaque opération.
Au vu des opérations de tel type présentes dans le filtre, l'usager formule une question à LOF en termes d'une liste de paramètres fonctionnels dont nous dirons que, si un opérateur les réalisait, il serait apte à faire toutes les opérations regroupées sous ces paramètres.
LOF analyse cette question, et rend à l'usager la liste de tous les opérateurs de la bibliothèque répondant aux conditions de la question. Cette réponse se fait sous la forme d'une liste de paramètres matériels que nous détaillons plus loin, et d'une liste de paramètres fonctionnels images de la question.
Le format de la question est indiqué à l'usager de manière interactive, et ne comprend que des paramètres fonctionnels identiques, pour chaque opérateur, à ceux des opérations, et dont la liste est donnée à l'annexe 2.
La liste des paramètres fonctionnels retournée par LOF peut être différente de la liste question si LOF retourne l'opérateur surabondant le plus proche de la question.

Exemple :
Un multiplieur 12x12 rendu à la demande d'un multiplieur 11x11.

(Nous donnons plus loin la liste des paramètres matériels fournis par LOF)

COMMANDE DE RUN : = CREATESOU

CREATESOU:==@DISQUE:[REYSS.FILTRE.LANG]CREATESOU.COM

LISTE DES PARAMETRES FOURNIS PAR CREATESOU.PAS :

Opérateurs de type "ENTREE" :

- LM_DUREE = durée totale de fonctionnement de l'opérateur.
  (réel exprimé en ns)
- LM_DUR_SETUP = durée de set-up.
  (réel exprimé en ns)
- LM_SURFACE = surface de l'opérateur.
  (réel exprimé en microns carrés)

Opérateur de type "SORTIE" :

- LM_DUREE = durée totale de fonctionnement de l'opérateur.
  (réel exprimé en ns)
- LM_DUR_SETUP = durée de set-up.
  (réel exprimé en ns)
- LM_SURFACE = surface de l'opérateur.
  (réel exprimé en microns carrés)

Opérateur de type "MULTIPLICATION" :

- LM_DUREE = durée totale de fonctionnement de l'opérateur (latch compris).
  (réel exprimé en ns)
- LM_DUR_LATCH = durée de latchage.
  (réel exprimé en ns)
- LM_DUR_SETUP = durée de set-up du latch.
  (réel exprimé en ns)
- LM_SURFACE = surface de l'opérateur.
  (réel exprimé en microns carrés)

Opérateur de type "ADDITION" :

- LM_DUREE = durée totale de fonctionnement de l'opérateur (latch compris).
  (réel exprimé en ns)
- LM_DUR_LATCH = durée de latchage.
  (réel exprimé en ns)
- LM_DUR_SETUP = durée de set-up du latch.
  (réel exprimé en ns)
- LM_SURFACE = surface de l'opérateur.
  (réel exprimé en microns carrés)

Opérateur de type "REGISTRE" :

- LM_DUREE = durée totale de fonctionnement de l'opérateur.
  (réel exprimé en ns)
- LM_DUR_SETUP = durée de set-up du registre.
  (réel exprimé en ns)
- LM_SURFACE = surface de l'opérateur.
  (réel exprimé en microns carrés)

Opérateur de type "ARRONDI" :

- LM_DUREE = durée totale de fonctionnement de l'opérateur (latch compris).
  (réel exprimé en ns)
- LM_DUR_LATCH = durée de latchage.
  (réel exprimé en ns)
- LM_DUR_SETUP = durée de set-up du latch.
  (réel exprimé en ns)
- LM_SURFACE = surface de l'opérateur.
  (réel exprimé en microns carrés)

Opérateur de type "CHANGEMENT DE SIGNE" :

- LM_DUREE = durée totale de fonctionnement de l'opérateur (latch compris).
  (réel exprimé en ns)
- LM_DUR_LATCH = durée de latchage.
  (réel exprimé en ns)
- LM_DUR_SETUP = durée de set-up du latch.
  (réel exprimé en ns)
- LM_SURFACE = surface de l'opérateur.
  (réel exprimé en microns carrés)

Opérateur de type "DECALAGE" :

- LM_DUREE = durée totale de fonctionnement de l'opérateur (latch compris).
  (réel exprimé en ns)
- LM_DUR_LATCH = durée de latchage.
  (réel exprimé en ns)
- LM_DUR_SETUP = durée de set-up du latch.
  (réel exprimé en ns)
- LM_SURFACE = surface de l'opérateur.
  (réel exprimé en microns carrés)

Opérateur de type "BUS" :

- LM_DUREE = durée maximale garantie de transit sur le bus.
  (réel exprimé en ns)

LOF fournit à l'usager une fourchette de temps pour la durée d'un transit sur le bus. L'usager choisit une valeur qui sera tenus comme impérative car IMHOTEP s'en servira comme durée pour l'opérateur bus en question.

FIN DE LA LISTE DES PARAMETRES.

CREATESOU.PAS demande un renseignement supplémentaire concernant le matériel du circuit, il s'agit de la période de l'horloge qui le séquencera.
Cette periode est exprimée en nanosecodes. (entier)

FORMAT DU FICHIER FOURNI PAR CREATESOU.PAS :

Ce fichier nommé NOMFILTRE.SOU est un fichier de type TEXT.
Il est donc éditable et contient ligne par ligne les renseignements suivants :

```
(début du fichier)

NOM DU FILTRE (sur 8 lettres),
DATE DE CREATION (sur 11 lettres),
HEURE DE CREATION (sur 11 lettres),
(CR)
DUREE ENTRE CHAQUE ARRIVEE D'UN ECHANTILLON (réel),
(CR)
NOMBRE DE TACHES DU CIRCUIT (entier),
(CR)
(puis, pour chaque tache répéter : )
  *NUMERO DE LA TACHE (entier),
  *NOM DE LA TACHE (sur 8 lettres),
  *ESPECE DE LA TACHE (sur 8 lettres),
  *NUMERO D'ESPECE DE LA TACHE (entier),
  *(CR)
  *RANG DE LA TACHE DANS LE GRAPHE (entier),
  *NOMBRE D'ANTECEDENTS DE LA TACHE (entier),
  *NOMBRE DE SUCCESSEURS DE LA TACHE (entier),
  *(CR)
  *(paramètres de la tache : (par espèce))
  * *(entrée : )
  * *  MODE DE TRANSMISSION (type énuméré ECHANGETYP ...cf déclarations),
  * *  NOMBRE DE BITS EN ENTREE (entier),
  * *  (CR)
  * *(sortie : )
  * *  MODE DE TRANSMISSION (type énuméré ECHANGETYP ...cf déclarations),
  * *  NOMBRE DE BITS EN SORTIE (entier),
  * *  (CR)
  * *(multiplication : )
  * *  NOMBRE DE BITS EN ENTREE (entier),
  * *  NOMBRE DE BITS DE COEFFICIENT (entier),
  * *  NOMBRE DE BITS EN SORTIE (entier),
  * *  VALEUR DU COEFFICIENT (réel),
  * *  MODE D'ARRONDI EN SORTIE (type énuméré ARRONTYP ...cf déclarations),
  * *  (CR)
  * *(additions : )
  * *  NOMBRE DE BITS EN ENTREE 1 (entier),
  * *  NOMBRE DE BITS EN ENTREE 2 (entier),
  * *  NOMBRE DE BITS EN SORTIE   (entier),
  * *  MODE DE CONTROLE DU DEBORDEMENT (type énuméré OVFTYP ...cf déclarations),
  * *  (CR)
  * *(registre : )
  * *  NOMBRE DE BITS A STOCKER (entier),
  * *  GENRE DE L'EXTREMITE DE REGISTRE M-E (1=esclave , 2=maitre),
  * *  (CR)
  * *(opérateur d'arrondi : )
  * *  NOMBRE DE BITS EN ENTREE (entier),
  * *  NOMBRE DE BITS EN SORTIE (entier),
  * *  MODE D'ARRONDI (type énuméré ARRONTYP ...cf déclarations),
  * *  (CR)
```

```
   * *(opérateur de changement de signe : )
   * *  NOMBRE DE BITS EN ENTREE (entier),
   * *  (CR)
   * *(décaleur : )
   * *  NOMBRE DE BITS EN ENTREE (entier),
   * *  NOMBRE DE RANGS DE DECALAGE (entier),
   * *  NOMBRE DE BITS EN SORTIE (entier),
   * *  MODE D'ARRONDI EN SORTIE (type énuméré ARRONTYP ...cf déclarations),
   * *  SENS DU DECALAGE         (type énuméré SENSTYP ...cf déclarations),
   * *  (CR)
   * *(bus : )
   * *  NOMBRE DE BITS (entier),
   * *  NOMBRE DE JETONS (entier),
   * *  (CR)
   *(fin des paramètres)
   *(pour chaque tache antécédante : )
   * *NUMERO DE LA TACHE ANTECEDANTE (entier),
   * *NUMERO DE L'ENTREE ATTEINTE (entier),
   * *(CR)
   *(pour chaque tache succédante : )
   * *NUMERO DE LA TACHE SUCCEDANTE (entier),
   * *NUMERO DE L'ENTREE ATTEINTE SUR CETTE TACHE (entier),
   * *(CR)
(fin de la tache courante)
PERIODE DE L'HORLOGE (entier exprimé en ns)
(CR)
(pour chaque type d'opérateur : )
  *(si il n'y a aucun opérateur dans cette catégorie : )
  * *-1
  * *(CR)
  *(si il y a un opérateur : )
  * *NUMERO DE L'OPERATEUR (entier),
  * *(CR)
  * *NOM DE L'OPERATEUR (sur 8 lettres),
  * *NOM DE L'ESPECE DE L'OPERATEUR (sur 8 lettres),
  * *NUMERO DE L'ESPECE DE L'OPERATEUR (entier),
  * *(CR)
  * *(paramètres fonctionnels de l'opérateur : (par espèce))
  * * *(entrée : )
  * * *  MODE DE TRANSMISSION (type énuméré ECHANGETYP),
  * * *  NOMBRE DE BITS EN ENTREE (entier),
  * * *  (CR)
  * * *(sortie : )
  * * *  MODE DE TRANSMISSION (type énuméré ECHANGETYP),
  * * *  NOMBRE DE BITS EN SORTIE (entier),
  * * *  (CR)
  * * *(multiplication : )
  * * *  NOMBRE DE BITS EN ENTREE (entier),
  * * *  NOMBRE DE BITS DE COEFFICIENT (entier),
  * * *  NOMBRE DE BITS EN SORTIE (entier),
  * * *  VALEUR DU COEFFICIENT (réel),
  * * *  MODE D'ARRONDI EN SORTIE (type énuméré ARRONTYP),
  * * *  (CR)
  * * *(additions : )
  * * *  NOMBRE DE BITS EN ENTREE 1 (entier),
  * * *  NOMBRE DE BITS EN ENTREE 2 (entier),
  * * *  NOMBRE DE BITS EN SORTIE   (entier),
  * * *  MODE DE CONTROLE DU DEBORDEMENT (type énuméré OVFTYP),
  * * *  (CR)
```

```
  * * *(registre : )
  * * *  NOMBRE DE BITS A STOCKER (entier),
  * * *  GENRE DE L'EXTREMITE DE REGISTRE M-E (1=esclave , 2=maitre),
  * * *  (CR)
  * * *(opérateur d'arrondi : )
  * * *  NOMBRE DE BITS EN ENTREE (entier),
  * * *  NOMBRE DE BITS EN SORTIE (entier),
  * * *  MODE D'ARRONDI (type énuméré ARRONTYP),
  * * *  (CR)
  * * *(opérateur de changement de signe : )
  * * *  NOMBRE DE BITS EN ENTREE (entier),
  * * *  (CR)
  * * *(décaleur : )
  * * *  NOMBRE DE BITS EN ENTREE (entier),
  * * *  NOMBRE DE RANGS DE DECALAGE (entier),
  * * *  NOMBRE DE BITS EN SORTIE (entier),
  * * *  MODE D'ARRONDI EN SORTIE (type énuméré ARRONTYP),
  * * *  SENS DU DECALAGE         (type énuméré SENSTYP),
  * * *  (CR)
  * * *(bus : )
  * * *  NOMBRE DE BITS (entier),
  * * *  NOMBRE DE JETONS (entier),
  * * *  (CR)
  * *(fin des paramètres fonctionnels)
  * *(liste des paramètres matériels)
  * *DUREE DE FONCTIONNEMENT (réel exprimé en ns),
  * *DUREE DE LATCHAGE (réel éventuellement nul exprimé en ns),
  * *DUREE DE SET-UP (réel exprimé en ns),
  * *SURFACE TOTALE (réel exprimé en ns),
  * *(CR)
  * *(fin des parametres matériels)
  * *0
  * *(CR)
  *(fin de cet opérateur)
(fin de la liste des types d'opérateurs)
(fin du fichier).
```

ANNEXE 4

PROGRAMME IMHOTEP.PAS-----

IMHOTEP.PAS est le programme principal du générateur d'architectures. Nous donnons son listing simplifié en expliquant chaque étape afin de donner à l'usager une vue globale de sa démarche.

COMMANDE DE RUN EN INTERACTIF : = IMHOTEPI

```
IMHOTEPI:==@DISQUE:[REYSS.IMHOTEP]IMHOTEPI.COM
```

Cette commande fait les assignations nécessaires, les paramètres du travail sont demandés de manière interactive.

COMMANDE DE RUN EN BATCH : = IMHOTEPB

```
IMHOTEPB:==@DISQUE:[REYSS.IMHOTEP]IMHOTEPB.COM
```

Cette commande demande les paramètres nécessaires au lancement du travail sur une queue de batch ,ainsi que ceux du travail.

LISTING DU PROGRAMME :

```
PROGRAM IMHOTEP(INPUT,OUTPUT);

(* fichier de déclaration des types utilisés *)

%INCLUDE 'DISQUE:[REYSS.IMHOTEP.PROCED]DECLAR.PAS'

(* Variables locales *)

VAR     NUMTABLE,MAX_NIVEAU,I,NBRdeTACHES,HORLOGE,NUMTAB_OV,LIM_TACH_FONC,
     LIM_REG_FONC,NBRdeOPER,NUMTAB_TM,
     MAX_ESSAI_OP,MAX_ESSAI_BUS,NUMTAB_RL                    : INTEGER;
     CTR_TEMPS,VARTRAI,TEMPSCPU,CPU_MAX,SURF_CUM,SURF_BEST: REAL;
     TA1                                : TABLE_ACCES;
     TETES_1,MOYENS_S                        : TETE_de_TABLE;
     SOURCE                                  : TEXT;
     FLAG_CONTINUER,TEST,BOOL1,BOOL2                 : BOOLEAN;
     LETTRE                                  : CHAR;
     ENS_SOLUTIONS                           : SOLUSET;
     VARDATE,VARTIME                                 : TIMETYP;
     NOMCIR                                          : MOT8CAR;
     LOF_MODULE                              : STRING;

(* Variables globales *)

     COMMENT_LISTE_OP,COMMENT_LISTE_BUS,COMMENT_LISTE_H,COMMENTAIRE,
     COMMENT_SOL,COMMENT_LOF                        :[GLOBAL] BOOLEAN;
     DUMP_OV,DUMP_SOL,FICH_P_OPER,FICHIER_GIL,GRAFDUMP,FICH_DUMP_CONT,
     IMAGE_ARCHI,DUMP_TM,DUMP_GNT,DUMP_ARCHI        :[GLOBAL] TEXT;
     CRIT_SEUIL_OP,CRIT_SEUIL_BUS,SEUIL_BAS_OP,SEUIL_BAS_BUS,
     PLAGE_TEMPS_MIN,PLAGE_TEMPS_MAX                :[GLOBAL] REAL;
     DATASIZE                          :[GLOBAL] INTEGER;

(* fichier des procédures externes *)

%INCLUDE 'DISQUE:[REYSS.IMHOTEP.PROCED]LISTNIV.PAS'

PROCEDURE IMH$THESEE(VAR TETES_de_TABLE:TETE_de_TABLE;
          VAR NUMTABLE,NUMTAB_OV,NUMTAB_TM,NUMTAB_RL:INTEGER;
                VAR LIM_TACH_FONC,LIM_REG_FONC:INTEGER;
                VAR CTR_TEMPS,CPU_MAX:REAL;
                VAR MAX_ESSAI_OP,MAX_ESSAI_BUS,T_HORLO:INTEGER;
                VAR NBRdeTACHES,NBRdeOPER:INTEGER;
                PROFONDEUR:INTEGER;VAR MAX_NIVEAU:INTEGER;
                VAR SURF_CUM,SURF_BEST,CPU_DEPART:REAL;
                VAR FLAG_CONTINUER:BOOLEAN;VAR SOLUTIONS :SOLUSET);EXTERNAL;
[EXTERNAL] PROCEDURE Reset_lof; EXTERN;
[EXTERNAL] PROCEDURE Charger_lof(Bib:STRING) ;EXTERN;
```

```
(* début du programme *)

BEGIN

(* Chargement de la bibliothèque des latches, 3_états, *)
(* et multiplexeurs nomée AUXIFI.CLF                   *)

RESET_LOF;
LOF_MODULE:='AUXIFI';
CHARGER_LOF(LOF_MODULE);

(* Initialisation des variables *)

DATASIZE:=0; NUMTABLE:=1; NUMTAB_OV:=2; NUMTAB_TM:=3; NUMTAB_RL:=4;
MAX_NIVEAU:=0; FLAG_CONTINUER:=TRUE; MAX_ESSAI_OP:=2;
MAX_ESSAI_BUS:=2; SURF_CUM:=0; SURF_BEST:=0;
FOR I:=1 TO MAX_SOL DO ENS_SOLUTIONS[I].SO_SURF_G:=0;
FOR I:=1 TO MAX_CASES DO TETES_1[I]:=NIL;

(* Ouverture des fichiers assignés à IMHOTEP *)

OPEN(DUMP_SOL,'DUMP_SOL',RECORD_LENGTH:=200,HISTORY:=NEW,
     ACCESS_METHOD:=SEQUENTIAL,RECORD_TYPE:=VARIABLE); REWRITE(DUMP_SOL);
OPEN(DUMP_ARCHI,'DUMP_ARCHI',RECORD_LENGTH:=200,HISTORY:=NEW,
     ACCESS_METHOD:=SEQUENTIAL,RECORD_TYPE:=VARIABLE); REWRITE(DUMP_ARCHI);
OPEN(IMAGE_ARCHI,'IMAGE_ARCHI',RECORD_LENGTH:=200,HISTORY:=NEW,
     ACCESS_METHOD:=SEQUENTIAL,RECORD_TYPE:=VARIABLE); REWRITE(IMAGE_ARCHI);
OPEN(FICHIER_GIL,'FICHIER_GIL',RECORD_LENGTH:=500,HISTORY:=NEW,
     ACCESS_METHOD:=SEQUENTIAL,RECORD_TYPE:=VARIABLE); REWRITE(FICHIER_GIL);
OPEN(DUMP_GNT,'DUMP_GNT',RECORD_LENGTH:=140,HISTORY:=NEW,
     ACCESS_METHOD:=SEQUENTIAL,RECORD_TYPE:=VARIABLE); REWRITE(DUMP_GNT);

(* Création de la structure de données pour ce circuit *)

IMH$LIRE_SOURCE(TETES_1,MOYENS_S,NUMTABLE,NBRdeTACHES,NOMCIR,
                VARDATE,VARTIME,CTR_TEMPS,HORLOGE,SOURCE);
IMH$CRE_TABLE_TM(TETES_1,NUMTAB_TM,MOYENS_S,NBRdeOPER);
IMH$CRE_TABLE_OV(TETES_1,MOYENS_S,NUMTABLE,NUMTAB_OV,NBRdeTACHES);

(*  Caractéristiques du travail : *)

WRITE('Temps CPU maximum alloue pour trouver l''architecture (en minutes)? ');
READLN(CPU_MAX);
WRITE('Voulez-vous negliger les durees de T_HOLD des latches et registres ? ');
READLN(LETTRE);
IF LETTRE = 'O' THEN BOOL1:=TRUE ELSE BOOL1:=FALSE;
WRITE('Voulez-vous rendre toutes les durees multiples de l''horloge (O/N) ? ');
READLN(LETTRE);
IF LETTRE = 'O' THEN BOOL2:=TRUE ELSE BOOL2:=FALSE;

IMH$CALIBRAGE_DUREES(TETES_1,NUMTAB_TM,NBRdeOPER,HORLOGE,BOOL1,BOOL2);
FOR I:=1 TO TYPFINOP DO IMH$DISPOS_TABLE(MOYENS_S[I]);
IMH$CRITERE_DYN(TETES_1,NUMTAB_OV,NUMTAB_TM,NBRdeTACHES,-1);
IMH$CLASSE_SPER_OV(TETES_1,NUMTAB_OV,NBRdeTACHES,-1);
IMH$VALUE_TACHE_REA(TETES_1,NUMTAB_OV,NBRdeTACHES);
IMH$CHEMIN_CRITIC(TETES_1,NUMTAB_OV,NBRdeTACHES,FALSE,VARTRA1);
IMH$GANTT_P(TETES_1,NUMTAB_OV,NUMTAB_TM,NBRdeTACHES,NBRdeOPER);
LIM_TACH_FONC:=NBRdeTACHES; LIM_REG_FONC:=NBRdeOPER;
```

```
(* Ajustage des paramètres du travail *)

CRIT_SEUIL_OP:=0.75; CRIT_SEUIL_BUS:=0.7;  (* valeurs par défaut *)
SEUIL_BAS_OP:=0.4; SEUIL_BAS_BUS:=0.4;
PLAGE_TEMPS_MIN:=0.05; PLAGE_TEMPS_MAX:=0.1;
PUIS DEMANDE DES VALEURS REELLES;

(* Sélection des commentaires désirés *)

COMMENT_LISTE_OP:=FALSE; COMMENT_LISTE_BUS:=FALSE; COMMENT_LISTE_H:=FALSE;
COMMENTAIRE:=FALSE; COMMENT_SOL:=FALSE; COMMENT_LOF:=FALSE;
PUIS DEMANDE DES VALEURS REELLES;

(* Lancement de la procédure récursive *)

IMH$THESEE(TETES_1,NUMTABLE,NUMTAB_OV,NUMTAB_TM,NUMTAB_RL,
       LIM_TACH_FONC,LIM_REG_FONC,
       CTR_TEMPS,CPU_MAX,
       MAX_ESSAI_OP,MAX_ESSAI_BUS,HORLOGE,
       NBRdeTACHES,NBRdeOPER,
       1,MAX_NIVEAU,
       SURF_CUM,SURF_BEST,TEMPSCPU,
       FLAG_CONTINUER,ENS_SOLUTIONS);

(* Exploitation des res'ultats, et stockage dans les fichiers *)

IMH$DUMP_ARCHITECTURE(ENS_SOLUTIONS,1,DUMP_ARCHI);
IMH$DUMP_CONTROLE(ENS_SOLUTIONS[1].SO_P_CONTROLE,DUMP_ARCHI);
IMH$CREE_IMAGE_ARCHI(ENS_SOLUTIONS,1,LIM_REG_FONC,IMAGE_ARCHI);
IMH$DUMP_SOLUSET(ENS_SOLUTIONS,DUMP_SOL);
IMH$FICHIER_PROTO(LIM_REG_FONC,1,ENS_SOLUTIONS,FICHIER_GIL);
IMH$SHOW_PARALLELE(ENS_SOLUTIONS,HORLOGE,1,LIM_REG_FONC,DUMP_GNT);
CLOSE(DUMP_ARCHI); CLOSE(IMAGE_ARCHI); CLOSE(DUMP_SOL); CLOSE(FICHIER_GIL);
END.
```

ANNEXE 5

LISTE DES PROCEDURES UTILISEES PAR IMHOTEP-----

Nous donnons ici la liste des procédures utilisées par IMHOTEP ainsi que leur organisation en MODULES. La répartition en modules est hiérarchique en ce sens que chaque procédure d'un module de niveau i ne peut appeler que des procédures situées dans des modules de niveau j avec j <= i.
Nous avons ici 6 niveaux de procédure numérotés de 0 à 5avec la syntaxe suivante :

MNIVXY.PAS = Y ième module de niveau X.

Pour chaque procédure, nous donnons quand cela est nécessaire quelques informations sur son rôle ou ses paramètres. Une descrition plus complète se trouve sous forme de commentaires dans le fichier source lui-même.

Ces procédures sont stochées dans le directory

DISQUE:[REYSS.IMHOTEP.PROCED]

LISTE DES PROCEDURES PAR NIVEAUX

MODULE MNIV01 :

```
PROCEDURE IMH$TABLE_L(P_TABLE:P_TABLE_REC;INDICE:INTEGER; VAR OBJET:TABLE_ACCES);
PROCEDURE IMH$TABLE_E(VAR P_TABLE:P_TABLE_REC;INDICE:INTEGER; OBJET:TABLE_ACCES);
```

Ces procédures lisent et écrivent dans les tables de longueur variable.

```
PROCEDURE IMH$DISPOS_TABLE(VAR P_TABLE:P_TABLE_REC);
PROCEDURE IMH$DISPOS_TABLE_TOUT(VAR P_TABLE:P_TABLE_REC; GENRE:MOTCAR);
PROCEDURE IMH$DISPOS_SEQ(VAR P_TABLE:P_TABLE_REC);
PROCEDURE IMH$DISPOS_GANTT(VAR P_GANTT :P_TRANCHE_T);
```

Ces procédures libèrent la place utilisée par les tables et leur contenu de manière adaptée aux différents contenus de ces tables.

```
PROCEDURE IMH$ECRIS(ASC:INTEGER);
PROCEDURE IMH$WRC(CR:CHAR);
PROCEDURE IMH$WRIT(CH:TEXTE);
PROCEDURE IMH$WRITLN(CH:TEXTE);
PROCEDURE IMH$UP;
PROCEDURE IMH$BACK(NOMBRE:INTEGER);
PROCEDURE IMH$BLANC(NOMBRE:INTEGER);
PROCEDURE IMH$LIS(VAR ASC:INTEGER);
PROCEDURE IMH$LISMENU(MEN:INTEGER;VAR INDEX:INTEGER);
PROCEDURE IMH$INITIALISATION;
PROCEDURE IMH$LisM(MEN:INTEGER;VAR INDEX:INTEGER);
PROCEDURE IMH$RETARD(RETARD:INTEGER);

FUNCTION IMH$MAXINT(A,B:INTEGER):INTEGER;
FUNCTION IMH$MAXREEL(A,B:REAL):REAL;
FUNCTION IMH$MINREEL(A,B:REAL):REAL;

FUNCTION IMH$RECTIF_CS(CRITERE_S:REAL;TYPOPERATEUR:INTEGER):REAL;
FUNCTION IMH$RECTIF_CD(CRITERE_D:REAL;TYPOPERATEUR:INTEGER):REAL;
FUNCTION IMH$GEN_CRITERE(CRITERE_S,CRITERE_D:REAL;TYPOPERATEUR:INTEGER):REAL;
```

Ces fonctions sont ajustables par recompilation et modifient le calcul des différents criteres selon la nature de l'opérateur concerné.

MODULE MNIV11 :

```
PROCEDURE IMH$CALIBRAGE_DUREES(VAR TETES_de_TABLE:TETE_de_TABLE;
      VAR NUMTAB_TM,NBRdeOPER,T_HORLO:INTEGER; HOLD0,MULTIPLE:BOOLEAN);
```

Cette procédure permet de négliger la durée de HOLD des latch et registres et aussi de rendre toutes les durées des opérateurs du circuit multiples de la période de l'horloge.

```
PROCEDURE IMH$CRE_TABLE_TM(VAR TETES_de_TABLE:TETE_de_TABLE;NUMTAB_TM:INTEGER;
      MOYENS:TETE_de_TABLE;VAR NBRdeOPER :INTEGER);
```

Cette procédure crée la table des TEMOINS (numéro NUMTAB_TM), à partir des MOYENS en opérateurs lus dans le fichier SOUrce. Elle rend le nombre d'opérateurs disponibles NBRdeOPER.

```
PROCEDURE IMH$TUE_TACHE(VAR TETES_de_TABLE:TETE_de_TABLE;
        NUMTABLE,NBRdeTACHAV,I:INTEGER;VAR NBRdeTACHAP:INTEGER);
PROCEDURE IMH$LECTUR_GRAPH(VAR TETES_de_TABLE:TETE_de_TABLE;NUMTABLE:INTEGER;
        VAR NBRdeTACHES :INTEGER; VAR NOMTC:MOT8CAR; VAR VARDATE,VARTIME:TIMETYP;
        VAR CTR_TEMPS:REAL; VAR DATAGRAF:TEXT);
PROCEDURE IMH$CREE_SOURCE(VAR TETES_de_TABLE,MOYENS:TETE_de_TABLE;NUMTABLE,
        NBRdeTACHES:INTEGER; VAR NOM_CIRCUIT:MOT8CAR;
        VAR DATE_CREATION,HEURE_CREATION:TIMETYP; VAR CTR_TEMPS:REAL;
        VAR T_HORLOGE:INTEGER;VAR SOURCE :TEXT);
PROCEDURE IMH$DISPOS_CASES(VAR TETES_de_TABLE:TETE_de_TABLE;
        NUMTABLE,NUMTAB_OV,PREMIER,DERNIER:INTEGER);
```

MODULE MNIV12 :

```
PROCEDURE IMH$CALCUL_RANG(VAR TETES_de_TABLE:TETE_de_TABLE;
                  NUMTABLE,NBRdeTACHES:INTEGER;VAR RETOURANG:INTEGER);
PROCEDURE IMH$CHEMIN_CRITIC(VAR TETES_de_TABLE:TETE_de_TABLE;
                   NUMTAB_OV,NBRdeTACHES:INTEGER;TRAIT_LATCH:BOOLEAN;
                   VAR TEMPS_CIR:REAL);
```

Ces procédures calculent le rang ,le chemin critique ainsi que les dates au plus tôt et plus tard pour chaque tâche du graphe. Le graphe est décrit soit par la table des tâches NUMTABLE, soit par la table des TRAVAUX NUMTAV_OV.

```
PROCEDURE IMH$CRITERE_DYN(VAR TETES_de_TABLE:TETE_de_TABLE;NUMTAB_OV,
      NUMTAB_TM,NBRdeTACHES,TYPOPERATEUR:INTEGER);
```

Cette procédure calcule le critère dynamique de chaque opérateur pour les CANDIDATS de chaque tâche.

```
PROCEDURE IMH$CLASSE_SPER_OV(VAR TETES_de_TABLE:TETE_de_TABLE;NUMTAB_OV,
      NBRdeTACHES,ESPECEOP:INTEGER);
```

Cette procédure classe les candidats de chaque tâche par ordre des critère décroissant.

```
PROCEDURE IMH$VALUE_TACHE_REA(VAR TETES_de_TABLE:TETE_de_TABLE;
      NUMTAB_OV,NBRdeTACHES:INTEGER);
PROCEDURE IMH$VALUE_TACHE_OPT(VAR TETES_de_TABLE:TETE_de_TABLE;
      NUMTAB_OV,NBRdeTACHES:INTEGER);
PROCEDURE IMH$VALUE_TACHE_NUL(VAR TETES_de_TABLE:TETE_de_TABLE;
      NUMTAB_OV,NBRdeTACHES:INTEGER);
```

Ces procédures calculent la durée de chaque tâche de manière REAliste ou OPTimiste, ou anNULent la durée des tâches déjà faites.

MODULE MNIV13 :

Ce module contient des procédures délivrant des traces de la structure de données dans des fichiers. Elles sont utilisées en phase de mise au point des algorithmes et pour les recherches d'erreurs.

```
PROCEDURE IMH$DUMP_TABLE_OV(VAR TETES_de_TABLE:TETE_de_TABLE;
      NUMTAB_OV,NUMTAB_TM,NBRdeTACHES:INTEGER;VAR DUMP_TABLE_OV:TEXT);
PROCEDURE IMH$DUMP_TABLE_TM(VAR TETES_de_TABLE:TETE_de_TABLE;
      NUMTAB_OV,NUMTAB_TM,NBRdeOPER:INTEGER;VAR DUMP_TABLE_TM:TEXT);
PROCEDURE IMH$DUMP_BESOIN(VAR BESOIN:TETE_de_TABLE;VAR FICHDUMP:TEXT);
PROCEDURE IMH$DUMP_GRAPHE(VAR TETES_de_TABLE:TETE_de_TABLE;
      NUMTABLE,NBRdeTACHES:INTEGER;VAR DATADUMP:TEXT);
PROCEDURE IMH$DUMP_GANTT(VAR TETES_de_TABLE:TETE_de_TABLE;
      NUMTAB_TM,NBRdeOPER:INTEGER;VAR DUMP_GNT :TEXT);
PROCEDURE IMH$DUMP_LISTE_H(VAR P_LISTE_H :P_TABLE_REC; VAR FICH_DUMP:TEXT);
PROCEDURE IMH$DUMP_LISTE_OP(VAR P_LISTE_OP :P_TABLE_REC; VAR FICH_DUMP:TEXT);
PROCEDURE IMH$DUMP_LISTE_BUS(VAR P_LISTE_BUS :P_TABLE_REC; VAR FICH_DUMP:TEXT);
```

MODULE MNIV14 :

```
PROCEDURE IMH$LISTE_ESSAI_OP(VAR TETES_de_TABLE:TETE_de_TABLE;
      NUMTAB_OV,NUMTAB_TM,LIM_REG_FONC:INTEGER;
      CLIENT:LIST_H; VAR P_LISTE_OP :P_TABLE_REC );
```

Cette procédure construit la liste des opérateurs à essayer pour la tâche mentionnée dans le record CLIENT, et la rend dans la liste P_LISTE_OP.

```
PROCEDURE IMH$LISTE_ESSAI_BUS(VAR TETES_de_TABLE:TETE_de_TABLE;
      NUMTAB_OV,NUMTAB_TM:INTEGER;
      CLIENT :LIST_H;VAR P_LISTE_BUS :P_TABLE_REC );
```

Cette procédure construit la liste des bus à essayer pour la tâche mentionnée dans le record CLIENT, et la rend dans la liste P_LISTE_OP.

```
FUNCTION IMH$NBR_ESSAI(VAR P_LISTE_MOY:P_TABLE_REC;
     MAX_ESSAI_OP,MAX_ESSAI_BUS:INTEGER):INTEGER;
```

Cette fonction détermine dans la liste des opérateurs à essayer P_LISTE_MOY le nombre MAX_ESSAI d'opérateurs ou de bus qu'il faudra essayer d'affecter.

MODULE MNIV15 :

```
PROCEDURE IMH$CHERCHE_MEMOIR(VAR TETES_de_TABLE:TETE_de_TABLE;
      NUMTAB_OV,TACHE_SOURCE:INTEGER; VAR P_LIBER:P_TACHSEUR;
      VAR DATE :REAL; VAR LIBRE :BOOLEAN);
```

Cette procédure recherche dans le graphe, les éléments de mémorisation qui libèrent l'opérateur occupé sur TACHE_SOURCE. La liste en est rendue par le pointeur P_LIBER, et la date de libération est DATE si LIBRE est rendu vrai.

```
PROCEDURE IMH$LIRE_SOURCE(VAR TETES_de_TABLE,MOYENS:TETE_de_TABLE;
      VAR NUMTABLE,NBRdeTACHES:INTEGER; VAR NOM_CIRCUIT:MOT8CAR;
      VAR DATE_CREATION,HEURE_CREATION:TIMETYP; VAR CTR_TEMPS:REAL;
      VAR T_HORLOGE:INTEGER; VAR SOURCE :TEXT);
```

Cette procédure lit le fichier SOUrce issu de CREATESOU, et stocke les données dans NUMTABLE et MOYENS. - 216 -

```
PROCEDURE IMH$CORRESPOND_EN(VAR TETES_de_TABLE:TETE_de_TABLE;
          NUMTAB_OV,NUMTAB_TM,NUMOP:INTEGER;
          VAR P_PREU :P_PREUVE; FLAG_OBLIG:BOOLEAN );
PROCEDURE IMH$RESET_PREUVES(VAR TETES_de_TABLE:TETE_de_TABLE;
          NUMTAB_TM :INTEGER; VAR P_LISTE_I:P_TABLE_REC);
```

Ces procédures établissent et annulent la correspondance entre les entrées de l'opérateur NUMOP et celles de la tâche pointée par P_PREU.

```
PROCEDURE IMH$CRE_NOUV_REG(VAR TETES_de_TABLE:TETE_de_TABLE;
      NUMTAB_TM,NUMTAB_RL:INTEGER;VAR NBRdeOPER:INTEGER;
      NBRdeBITS:INTEGER;DUREE_LIMIT:REAL; VAR NUM_REG:INTEGER);
```

Cette procédure demande à LOF un nouveau registre simple-latch de NBRdeBITS bits et de temps de fonctionnement inférieur à DUREE_LIMIT. Ce registre NUM_REG est rangé dans la table NUMTAB_RL pour éviter les appels répétitifs.

```
PROCEDURE IMH$ENLEVE_NOUV_REG(VAR TETES_de_TABLE:TETE_de_TABLE;
      NUMTAB_OV,NUMTAB_TM, NBRdeTACHES,LIM_TACH_FONC,NUM_REG :INTEGER);
```

Cette procédure défait le travail de IMH$ATTRIBUE_NOUV_REG;

```
PROCEDURE IMH$MET_LIEN(VAR TETES_de_TABLE: TETE_de_TABLE;
      NUMTAB_OV,TACHE_SOURCE,TACHE_DEST:INTEGER; VAR DEJAFAIT:BOOLEAN);
PROCEDURE IMH$OTE_LIEN(VAR TETES_de_TABLE: TETE_de_TABLE;
      NUMTAB_OV,TACHE_SOURCE,TACHE_DEST :INTEGER;DEJAFAIT:BOOLEAN);
```

Ces procédures mettent et enlèvent un lien entre les taches TACHE_SOURCE et TACHE_DEST et mettent à jour les listes des voisins.

MODULE MNIV16 :

```
PROCEDURE IMH$P_OPERATIVE(VAR TETES_de_TABLE:TETE_de_TABLE;
      NUMTAB_OV,NUMTAB_TM,NBRdeOPER:INTEGER);
```

Cette procédure extrait la partie opérative de l'architecture trouvée et la stocke en complétant les champs de commande de chaque opérateur.

```
PROCEDURE IMH$P_GEN_CONTR(VAR TETES_de_TABLE:TETE_de_TABLE;
      NUMTAB_OV,NUMTAB_TM,NBRdeTACHES,LIM_TACH_FONC:INTEGER;
      TEMPS_MAX:REAL; VAR RACINE_SEQ:P_TABLE_REC; VAR DERNIER:INTEGER);
PROCEDURE IMH$CLASSE_CONTROLE(VAR RACINE_SEQ:P_TABLE_REC; VAR DERNIER:INTEGER);
PROCEDURE IMH$COMPACTE_CONTROLE(VAR RACINE_SEQ:P_TABLE_REC;VAR DERNIER:INTEGER);
PROCEDURE IMH$NORMALISE_CONTROLE(VAR RACINE_SEQ :P_TABLE_REC;
      VAR NBRdeETAPES :INTEGER);
```

Ces procédures génèrent la partie contrôle de l'architecture et la traitent pour l'optimiser et la normaliser.

```
PROCEDURE IMH$SURFACE_FINE(VAR TETES_de_TABLE:TETE_de_TABLE; NUMTAB_OV,
      NUMTAB_TM,NUMTAB_RL,NBRdeOPER:INTEGER;VAR SURF_FINE:REAL);
```

Cette procédure procède à une estimation fine de la surface du circuit. Elle fait appel à LOF et à la bibliothèque DISQUE:[REYSS.IMHOTEP]AUXIFI.CLF pour estimer la surface des interconnexions.

MODULE MNIV17 :

Ce module contient des procédures délivrant des traces des résultats dans des fichiers.

```
PROCEDURE IMH$DUMP_SOLUSET(VAR SOLU:SOLUSET; VAR DUMP_SOL:TEXT);
PROCEDURE IMH$DUMP_OPERATIVE(VAR TETES_de_TABLE:TETE_de_TABLE;
      NUMTAB_OV,NUMTAB_TM,NBRdeOPER :INTEGER; VAR DUMP_OPERATIVE :TEXT);
PROCEDURE IMH$DUMP_CONTROLE(VAR RACINE_SEQ:P_TABLE_REC;
      VAR DUMP_CONTROLE :TEXT);
PROCEDURE IMH$DUMP_ARCHITECTURE(VAR SOLUTION:SOLUSET; NUM_SOL:INTEGER;
      VAR DUMP_ARCHI :TEXT);
PROCEDURE IMH$CREE_IMAGE_ARCHI(VAR SOLUTION:SOLUSET;
      NUM_SOL,LIM_REG_FONC:INTEGER; VAR IMAGE_ARCHI :TEXT);
PROCEDURE IMH$FICHIER_GILLES(LIM_REG_FONC,NUM_SOL :INTEGER;
      VAR SOLUTION:SOLUSET;VAR FICHIER_GILLES :TEXT);
PROCEDURE IMH$SHOW_PARALLELE(VAR SOLUTION:SOLUSET;
      T_HORLO,NUMSOL,LIM_REG_FONC:INTEGER; VAR FICHIER_GANTT :TEXT);
```

MODULE MNIV21 :

```
PROCEDURE IMH$GANTT_P(VAR TETES_de_TABLE:TETE_de_TABLE;
      NUMTAB_OV,NUMTAB_TM,NBRdeTACHES,NBRdeOPER:INTEGER);
PROCEDURE IMH$GANTT_R(VAR TETES_de_TABLE:TETE_de_TABLE;
      NUMTAB_OV,NUMTAB_TM,NBRdeTACHES,NBRdeOPER:INTEGER);
PROCEDURE IMH$GANTT_T(VAR TETES_de_TABLE:TETE_de_TABLE;
      NUMTAB_OV,NUMTAB_TM,NBRdeTACHES,NBRdeOPER:INTEGER);
```

Ces procédures génèrent le tableau de GANTT de l'architecture dans différentes situations.

```
PROCEDURE IMH$ATTRIBUE_NOUV_REG(VAR TETES_de_TABLE:TETE_de_TABLE;
      NUMTAB_OV,NUMTAB_TM,NBRdeTACHES,LIM_TACH_FONC,NUM_REG :INTEGER);
```

Cette procédure rajoute un opérateur registre nouvellement crée dans la liste des candidats des tâches de type registre-simple-latch non encore satisfaites.

```
PROCEDURE IMH$MET_LIENS(VAR TETES_de_TABLE:TETE_de_TABLE;
      NUMTABLE,NUMTAB_OV,NUMTAB_TM,NBRdeTACHES,NBRdeOPER,OPERATEUR,
      TACHE_DEST :INTEGER;VAR P_LISTE_L :P_TABLE_REC;VAR CIRCUIT :BOOLEAN);
PROCEDURE IMH$OTE_LIENS(VAR TETES_de_TABLE:TETE_de_TABLE; NUMTABLE,
      NUMTAB_OV,NUMTAB_TM,NBRdeTACHES,TACHE_DEST :INTEGER;
      VAR P_LISTE_L :P_TABLE_REC);
```

Ces procédures mettent des liens de succession entre toutes les taches qui libèrent l'opérateur OPERATEUR et la tâche TACHE_DEST qui va le réutiliser.
La liste des liens mis est rendue dans P_LISTE_L.

MODULE MNIV22 :

```
PROCEDURE IMH$PASSIV_LATCH(VAR TETES_de_TABLE:TETE_de_TABLE;
      NUMTABLE,NUMTAB_OV,NBRdeTACHES,NUMLATCH:INTEGER);
PROCEDURE IMH$DEPASSIV_LATCH(VAR TETES_de_TABLE:TETE_de_TABLE;
      NUMTABLE,NUMTAB_OV,NBRdeTACHES,NUMLATCH:INTEGER);
```

Ces procédures passivent et dépassivent le latch NUMLATCH dans le graphe.

```
PROCEDURE IMH$CREE_LATCH(VAR TETES_de_TABLE:TETE_de_TABLE;
          NUMTABLE,NUMTAB_OV:INTEGER;VAR NBRdeTACHES:INTEGER;
          TACHE_SUPPORT,ENTREE_VISEE:INTEGER;VAR NUMLATCH:INTEGER);
```

Cette procédure rajoute une tâche de type latch sur l'entrée ENTREE_VISEE de la tâche TACHE_SUPPORT.

```
PROCEDURE IMH$CREE_REGISTRE_NF(VAR TETES_de_TABLE:TETE_de_TABLE;
      NUMTABLE,NUMTAB_OV,NUMTAB_TM:INTEGER;
      VAR NBRdeTACHES:INTEGER;NBRdeOPER,LIM_REG_FONC,TACHE_SUPPORT,
      ENTREE_VISEE:INTEGER;DUREE_LIMIT:REAL;VAR NUMREG:INTEGER);
```

Cette procédure rajoute une tâche de type registre-simple-latch numérotée NUM_REG sur l'entrée ENTREE_VISEE de la tâche TACHE_SUPPORT.

```
PROCEDURE IMH$CREE_BUS_F(VAR TETES_de_TABLE:TETE_de_TABLE;
      NUMTABLE,NUMTAB_OV,NUMTAB_TM:INTEGER;VAR NBRdeTACHES:INTEGER;
      NBRdeOPER,TACHE_SUPPORT,ENTREE_VISEE:INTEGER;VAR NUMBUS:INTEGER);
```

Cette procédure rajoute une tâche de type bus numérotée NUM_REG sur l'entrée ENTREE_VISEE de la tâche TACHE_SUPPORT.

```
PROCEDURE IMH$PASSIV_TACHE(VAR TETES_de_TABLE:TETE_de_TABLE;
      NUMTABLE,NUMTAB_OV,NBRdeTACHES,NUMTACHE:INTEGER);
PROCEDURE IMH$DEPASSIV_TACHE(VAR TETES_de_TABLE:TETE_de_TABLE;
      NUMTABLE,NUMTAB_OV,NBRdeTACHES,NUMTACHE:INTEGER);
```

Ces procédures passivent et dépassivent la tâche NUMTACHE dans le graphe.

MODULE MNIV23 :

```
PROCEDURE IMH$LISTE_HIERARCH(VAR TETES_de_TABLE:TETE_de_TABLE;
      NUMTABLE,NUMTAB_OV,NUMTAB_TM,NBRdeTACHES,NBRdeOPER,
      LIM_TACH_FONC:INTEGER;VAR P_LISTE_H :P_TABLE_REC;
      VAR NBR_ELEM:INTEGER);
```

Cette procédure crée la liste hiérarchisée des tâches à accomplir et la rend dans la table pointée par P_LISTE_H contenant NBR_ELEM éléments.

MODULE MNIV24 :

```
PROCEDURE IMH$INCLURE_BUS(VAR TETES_de_TABLE:TETE_de_TABLE;
      NUMTABLE:INTEGER;VAR NBRdeTACHES:INTEGER);
PROCEDURE IMH$OPTIMISATION(VAR TETES_de_TABLE:TETE_de_TABLE;NUMTABLE,
      NBRdeTACHAV :INTEGER;SIMPLIFICATION :BOOLEAN;
      VAR NBRdeTACHAP :INTEGER);
```

Cette procédure effectue les optimisations possibles dans CREATESOU.PAS, au niveau des multiplications et troncatures.

```
PROCEDURE IMH$EVALUE_TEMPS(VAR TETES_de_TABLE:TETE_de_TABLE;
      NUMTAB_OV,NBRdeTACHES:INTEGER; VAR EVALU_T:REAL);
```

Cette procédure évalue le temps de fonctionnement d'une architecture non encore completement terminée.

MODULE MNIV25 :

```
PROCEDURE IMH$CRE_TABLE_OV(VAR TETES_de_TABLE,MOYENS:TETE_de_TABLE;
      NUMTABLE,NUMTAB_OV,NBRdeTACHES:INTEGER);
```

Cette procédure crée la table des TRAVAUX du circuit décrit dans NUMTABLE et la range dans NUMTAB_OV.

```
PROCEDURE IMH$SYNCHRO_CONTROLE(VAR RACINE_SEQ:P_TABLE_REC;
      VAR NBRdeETAPES,T_HORLO:INTEGER; VAR TEMPS_MAX:REAL);
PROCEDURE IMH$ECHANTILLONNE_ETAPES(VAR RACINE_SEQ:P_TABLE_REC;
      VAR NBRdeETAPES :INTEGER);
```

Ces procédures traitent la partie contrôle de l'architecture pour l'optimiser et la normaliser.

MODULE MNIV31 :

```
PROCEDURE IMH$LIBERE_OPERATEUR(VAR TETES_de_TABLE:TETE_de_TABLE;
      NUMTABLE,NUMTAB_OV,NUMTAB_TM:INTEGER; VAR NBRdeTACHES:INTEGER;
      NBRdeOPER:INTEGER;T_OCCUPANTE,OPERATEUR:INTEGER;
      VAR P_LISTE_I,P_LISTE_LATCH :P_TABLE_REC);
```

Cette procédure libère l'opérateur OPERATEUR occupé par la tâche T_OCCUPANTE. Il crée donc des latchs dont la liste est rendue dans la table P_LISTE_LATCH et il est amené à préciser certaines correspondances entre entrées et sorties de cet opérateur et des tâches qu'il a déjà réalisées. La liste des tâches concernées est rendue dans P_LISTE_I.

```
PROCEDURE IMH$RESET_LIBERE(VAR TETES_de_TABLE:TETE_de_TABLE;NUMTABLE,
      NUMTAB_OV,NUMTAB_TM,NBRdeTACHES,NBRdeOPER :INTEGER;
      VAR P_LISTE_I,P_LISTE_LATCH :P_TABLE_REC);
```

Cette procédure défait le travail de IMH$LIBERE_OPERATEUR.

```
PROCEDURE IMH$UNIFORMIS_LATCH(VAR TETES_de_TABLE:TETE_de_TABLE;NUMTABLE,
      NUMTAB_OV,NUMTAB_TM:INTEGER; VAR NBRdeTACHES:INTEGER;
      NBRdeOPER,TACHE,OPERATEUR:INTEGER;
      VAR P_LISTE_I,P_LISTE_LATCH :P_TABLE_REC);
```

Cette procédure uniformise la distribution des latchs entre l'OPERATEUR et toutes les tâches déjà faites par cet opérateur ,y compris la dernière qui est TACHE. La liste des latchs rajoutés est rendue dans P_LISTE_LATCH.
Elle est amenée à préciser certaines correspondances entre entrées et sorties de cet opérateur et des tâches qu'il a déjà réalisées.
La liste des tâches concernées est rendue dans P_LISTE_I.

```
PROCEDURE IMH$RESET_UNIFORMIS(VAR TETES_de_TABLE:TETE_de_TABLE;NUMTABLE,
      NUMTAB_OV,NUMTAB_TM,NBRdeTACHES,NBRdeOPER,TACHE,OPERATEUR:INTEGER;
      VAR P_LISTE_I,P_LISTE_LATCH :P_TABLE_REC);
```

Cette procédure défait le travail de IMH$UNIFORMIS_LATCH;

MODULE MNIV32 :

```
PROCEDURE IMH$STOCK_SOLUTION(VAR TETES_de_TABLE:TETE_de_TABLE;NUMTAB_OV,
      NUMTAB_TM,NBRdeTACHES,NBRdeOPER,T_HORLO:INTEGER;
      CPU_DEPART,Treponse,SURF_FINE :REAL;VAR RACINE_SEQ:P_TABLE_REC;
      VAR SURF_BEST:REAL);
```

Cette procédure travaille aux noeuds terminaux de l'arbre des solutions. Elle est chargée d'élaborer la solution trouvée sous la forme d'une partie opérative et d'une partie contrôle, et elle rend des données comme le temps de réponse et la surface réelle.

MODULE MNIV41 :

```
PROCEDURE IMH$GENERE_SOLUTION(VAR TETES_de_TABLE:TETE_de_TABLE; NUMTABLE,
      NUMTAB_OV,NUMTAB_TM,NUMTAB_RL,NBRdeTACHES,NBRdeOPER,LIM_REG_FONC,
      T_HORLO:INTEGER;VAR FICH_P_OPER,FICH_P_CONT,GRAFDUMP,DUMP_OV,
      DUMP_TM,DUMP_GNT,DUMP_SOL:TEXT; VAR SOLUTIONS:SOLUSET;
      CPU_DEPART,TEMPS_MAX:REAL; VAR SURF_BEST:REAL);
```

Cette procédure travaille aussi aux noeuds terminaux de l'arbre des solutions. Elle génère la solution élaborée par IMH$STOCK_SOLUTION et la range dans le tableau SOLUTIONS en la classant dans ce tableau et en éliminant éventuellement les solutions devenues sous-optimales.

MODULE MNIV51 :

```
PROCEDURE IMH$THESEE(VAR TETES_de_TABLE:TETE_de_TABLE;VAR NUMTABLE,
      NUMTAB_OV,NUMTAB_TM,NUMTAB_RL:INTEGER;
      VAR LIM_TACH_FONC,LIM_REG_FONC:INTEGER;
      VAR TEMPS_MAX,CPU_MAX:REAL;
      VAR MAX_ESSAI_OP,MAX_ESSAI_BUS,T_HORLO:INTEGER;
      VAR NBRdeTACHES,NBRdeOPER:INTEGER;
      PROFONDEUR:INTEGER;VAR MAX_NIVEAU:INTEGER;
      VAR SURF_CUM,SURF_BEST,CPU_DEPART:REAL;
      VAR FLAG_CONTINUER:BOOLEAN;VAR SOLUTIONS :SOLUSET);
```

Cette procédure est celle qui explore de manière récursive l'arbre des solutions architecturales pour le filtre.

ANNEXE 6

CLASSIFICATION DES PROBLEMES D'ORDONNANCEMENT-----

Au début du travail sur le générateur d'architectures, nous nous sommes demandés si il existait des algorithmes résolvant le genre de problème auquel nous étions confrontés. Cette recherche de renseignements nous a fait contacter le Pr FONLUPT à l'IMAG (Grenoble), qui nous a conseillé d'exposer notre problème au Dr J.K. Lenstra du Centre De Mathématiques Appliquées d'Amsterdam.

Un échange de correspondance eut lieu qui nous conforta dans notre impression qu'aucune étude similaire n'avait encore été menée sur cette catégorie de problémes d'ordonnancement. Notre intention d'essayer une méthode heuristique était donc justifiée de facto.

Nous reproduisons dans cette annexe 6, une lettre de J.K. LENSTRA qui nous parait situer la complexité du problème.

Stichting Mathematisch Centrum

Centrum voor Wiskunde en Informatica

Kruislaan 413 1098 SJ Amsterdam
Telefoon (020) 592 9333 *Telex 12571*

Uw referentie
Datum
Onze referentie JKL/AKB
Datum May 15, 1984

Dr. J.F. Reyss-Brion
CNET Grenoble
B.P. 42
Chemin du Vieux Chêne
38240 Meylan
FRANCE

Telefoon (020) 592 4087

Dear Dr. Reiss-Brion,

My colleague Ben Lageweg and I have taken a look at the material you sent us in January.

It seems to us that, in terms of resource constrained scheduling, the problem of assigning operators to operations could be denoted by $P|m \geq n, prec, res..1|C_{max}$. This will, no doubt, look a bit frightening to you. In the enclosed paper by Blazewicz et al., this kind of short-hand notation to classify scheduling problems is defined. In the present case, it boils down to the following: n jobs (the operations) have to be scheduled on m parallel identical machines, whose capacity restrictions are unimportant since $m \geq n$; each feasible schedule has to satisfy given precedence constraints; there is a certain number of resources, each corresponding to a type of operator; the size of resource h is given by the number of available operators of type h; the amount r_{hj} of resource h required by job j at all times during its execution is equal to 0 or 1; and the objective function to be minimized is the maximum completion time C_{max}.

Now, on one hand, you are dealing with a special case of this problem, since, if $r_{hj} = 1$, then $r_{h'j} = 0$ for all $h' \neq h$. On the other hand, your problem is more general, since the "quality parameter" has not been taken into account.

The above exercise in modelling does not contribute very much towards a solution method. However, it tells us two things. First, your problem is extremely complicated, also in the formal sense of computational complexity theory, and this fully justifies a heuristic solution approach. Secondly, this specific problem type has not been studied before in the scheduling literature; see the enclosed annotated bibliography, in particular §7.1, for a survey of recent work in this direction.

I apologize that it took so long before we found time to arrive at these, not very deep, conclusions. It would require a substantial research effort to develop a reasonably efficient algorithm for your problem. In this respect, I would advise you to get in touch with Jacques Carlier, who is an internationally reputed expert in the area of sequencing and scheduling. His address is:

Université Pierre et Marie Curie - Paris VI
U.E.R. 50
Institut de Programmation
Tour 55-65
4, Place Jussieu
75230 Paris Cédex 05

.../...

Postbus 4079 1009 AB Amsterdam

JKL/AKB
May 15, 1984

I hope that our comments will be of some use to you.

Yours sincerely,

Jan Karel Lenstra

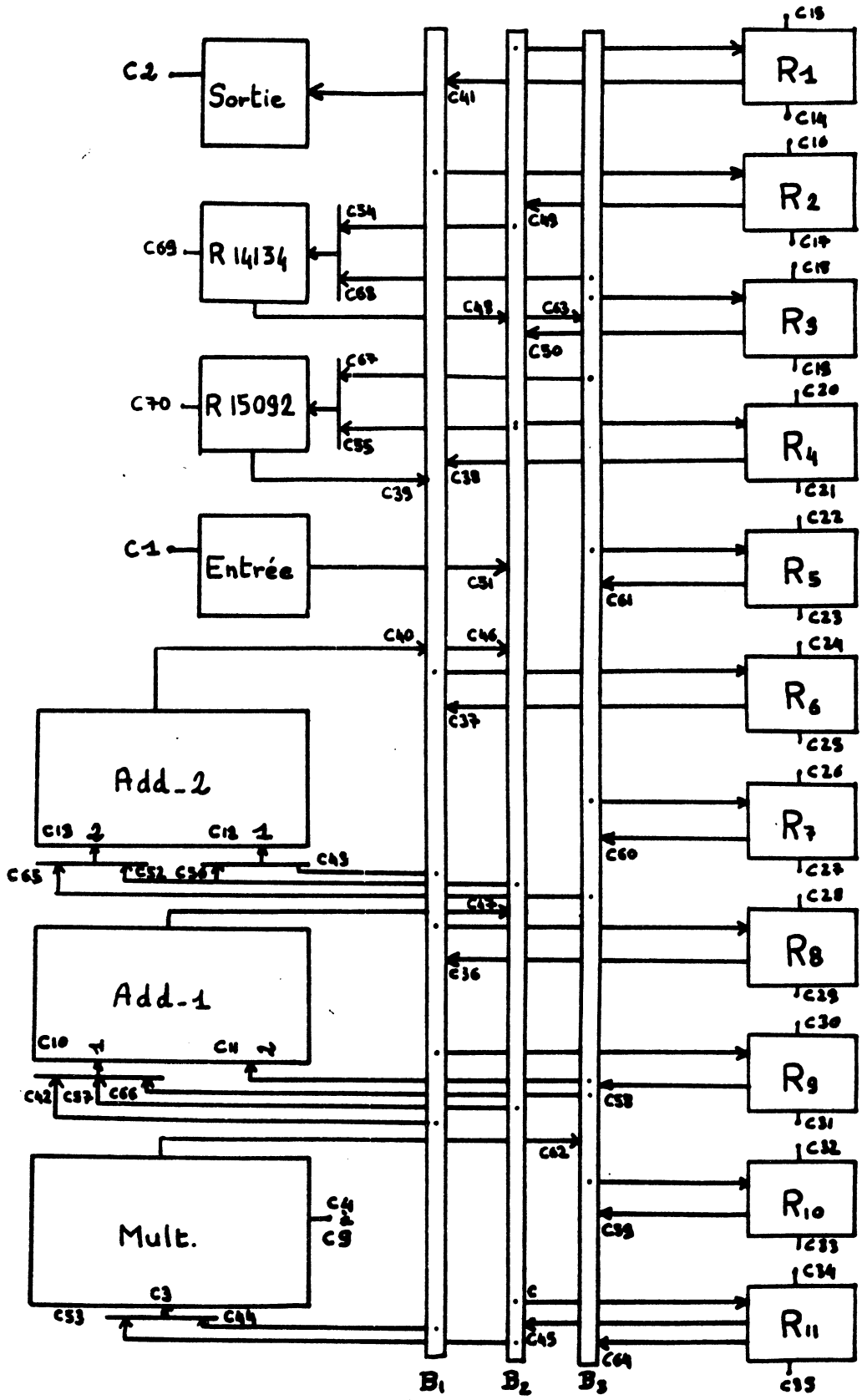
Enclosures

ANNEXE 7

COMPLEMENTS SUR LES ARCHITECTURES OBTENUES : -----

Nous donnons ici les descriptions complètes des parties opératives et des parties contrôles pour chacun des exemples traités au chapitre 8.

SHEMA-BLOC DE L'ARCHITECTURE TROUVEE POUR " FIR-11 "



PARTIE OPERATIVE DU CIRCUIT FIR-11

Opérateur numéro : 1 nommé ENTREE F d'espèce ENTREE

Commande propre numéro : 1
Commande de RAZ : 72

Opérateur ayant 0 antécédents :
et 1 successeurs :
- successeur numéro : 29

Opérateur numéro : 2 nommé SORTIE F d'espèce SORTIE

Commande propre numéro : 2
Commande de RAZ : 72

Opérateur ayant 1 antécédents :
- antécédent numero : 28
et 0 successeurs :

Opérateur numéro : 3 nommé MULT1 d'espèce MULTIPLI

l'entrée : 1 est latchée avec la commande numéro : 3

Commande de RAZ : 72

Ce multiplieur général fait 6 multiplications
la multiplication numéro 6 avec la commande numéro : 4
la multiplication numéro 19 avec la commande numéro : 5
la multiplication numéro 25 avec la commande numéro : 6
la multiplication numéro 31 avec la commande numéro : 7
la multiplication numéro 37 avec la commande numéro : 8
la multiplication numéro 42 avec la commande numéro : 9

Opérateur ayant : 2 antécédents sur l'entrée : 1
- antécédent numéro : 29
- antécédent numéro : 28
et : 1 successeurs
- successeur numéro : 30

Opérateur numéro : 4 nommé ADD_1 d'espèce ADDITION

l'entrée : 1 est latchée avec la commande numéro : 10
l'entrée : 2 est latchée avec la commande numéro : 11

Commande de RAZ : 72

Opérateur ayant : 3 antécédents sur l'entrée : 1
- antécédent numéro : 28
- antécédent numéro : 30
- antécédent numéro : 29
Opérateur ayant : 1 antécédents sur l'entrée : 2
- antécédent numéro : 30
et : 1 successeurs
- successeur numéro : 29

Opérateur numéro : 5 nommé ADD_2 d'espèce ADDITION

l'entrée : 1 est latchée avec la commande numéro : 12
l'entrée : 2 est latchée avec la commande numéro : 13

Commande de RAZ : 72

Opérateur ayant : 2 antécédents sur l'entrée : 1
- antécédent numéro : 30
- antécédent numéro : 29
Opérateur ayant : 2 antécédents sur l'entrée : 2
- antécédent numéro : 29
- antécédent numéro : 28
et : 2 successeurs
- successeur numéro : 29
- successeur numéro : 28

Opérateur numéro : 7 nommé REG1 d'espèce REGISTRE

Commandes propres numéro : 14,15
Commande de RAZ : 72

Opérateur ayant 1 antécédents :
- antécédent numero : 29
et 1 successeurs :
- successeur numéro : 28

Opérateur numéro : 9 nommé REG2 d'espèce REGISTRE

Commandes propres numéro : 16,17
Commande de RAZ : 72

Opérateur ayant 1 antécédents :
- antécédent numero : 28
et 1 successeurs :
- successeur numéro : 29

Opérateur numéro : 11 nommé REG3 d'espèce REGISTRE

Commandes propre numéro : 18,19
Commande de RAZ : 72

Opérateur ayant 1 antécédents :
- antécédent numero : 30
et 1 successeurs :
- successeur numéro : 29

Opérateur numéro : 13 nommé REG4 d'espèce REGISTRE

Commandes propres numéro : 20,21
Commande de RAZ : 72

Opérateur ayant 1 antécédents :
- antécédent numero : 29
et 1 successeurs :
- successeur numéro : 28

Opérateur numéro : 15 nommé REG5 d'espèce REGISTRE

Commandes propres numéro : 22,23
Commande de RAZ : 72

Opérateur ayant 1 antécédents :
- antécédent numero : 30
et 1 successeurs :
- successeur numéro : 30

Opérateur numéro : 17 nommé REG6 d'espèce REGISTRE

Commandes propres numéro : 24,25
Commande de RAZ : 72

Opérateur ayant 1 antécédents :
- antécédent numero : 28
et 1 successeurs :
- successeur numéro : 28

Opérateur numéro : 19 nommé REG7 d'espèce REGISTRE

Commandes propres numéro : 26,27
Commande de RAZ : 72

Opérateur ayant 1 antécédents :
- antécédent numero : 30
et 1 successeurs :
- successeur numéro : 30

Opérateur numéro : 21 nommé REG8 d'espèce REGISTRE

Commandes propres numéro : 28,29
Commande de RAZ : 72

Opérateur ayant 1 antécédents :
- antécédent numero : 28
et 1 successeurs :
- successeur numéro : 28

Opérateur numéro : 23 nommé REG9 d'espèce REGISTRE

Commandes propres numéro : 30,31
Commande de RAZ : 72

Opérateur ayant 1 antécédents :
- antécédent numero : 28
et 1 successeurs :
- successeur numéro : 30

Opérateur numéro : 25 nommé REG10 d'espèce REGISTRE

Commandes propres numéro : 32,33
Commande de RAZ : 72

Opérateur ayant 1 antécédents :
- antécédent numero : 30
et 1 successeurs :
- successeur numéro : 30

Opérateur numéro : 27 nommé REG11 d'espèce REGISTRE

Commandes propres numéro : 34,35
Commande de RAZ : 72

Opérateur ayant 1 antécédents :
- antécédent numero : 29
et 2 successeurs :
- successeur numéro : 30
- successeur numéro : 29

Opérateur numéro : 28 nommé BUS1 d'espèce BUS

Bus ayant 6 antécedents :
Opérateur numéro : 20 avec la commande numéro : 36
Opérateur numéro : 16 avec la commande numéro : 37
Opérateur numéro : 12 avec la commande numéro : 38
Opérateur numéro : 32 avec la commande numéro : 39
Opérateur numéro : 5 avec la commande numéro : 40
Opérateur numéro : 6 avec la commande numéro : 41
et 8 successeurs :
Opérateur numéro : 4 atteint sur l'entrée : 1 commande numéro : 42
Opérateur numéro : 5 atteint sur l'entrée : 2 commande numéro : 43
Opérateur numéro : 3 atteint sur l'entrée : 1 commande numéro : 44
Opérateur numéro : 2 atteint sur l'entrée : 1 pas de commande
Opérateur numéro : 23 atteint sur l'entrée : 1 pas de commande
Opérateur numéro : 21 atteint sur l'entrée : 1 pas de commande
Opérateur numéro : 17 atteint sur l'entrée : 1 pas de commande
Opérateur numéro : 9 atteint sur l'entrée : 1 pas de commande

Opérateur numéro : 29 nommé BUS2 d'espèce BUS

Bus ayant 7 antécedents :
Opérateur numéro : 26 avec la commande numéro : 45
Opérateur numéro : 5 avec la commande numéro : 46
Opérateur numéro : 4 avec la commande numéro : 47
Opérateur numéro : 31 avec la commande numéro : 48
Opérateur numéro : 8 avec la commande numéro : 49
Opérateur numéro : 10 avec la commande numéro : 50
Opérateur numéro : 1 avec la commande numéro : 51
et 9 successeurs :
Opérateur numéro : 5 atteint sur l'entrée : 2 commande numéro : 52
Opérateur numéro : 3 atteint sur l'entrée : 1 commande numéro : 53
Opérateur numéro : 31 atteint sur l'entrée : 1 commande numéro : 54
Opérateur numéro : 32 atteint sur l'entrée : 1 commande numéro : 55
Opérateur numéro : 5 atteint sur l'entrée : 1 commande numéro : 56
Opérateur numéro : 4 atteint sur l'entrée : 1 commande numéro : 57
Opérateur numéro : 27 atteint sur l'entrée : 1 pas de commande
Opérateur numéro : 13 atteint sur l'entrée : 1 pas de commande
Opérateur numéro : 7 atteint sur l'entrée : 1 pas de commande

Opérateur numéro : 30 nommé BUS3 d'espèce BUS

Bus ayant 7 antécedents :
Opérateur numéro : 22 avec la commande numéro : 58
Opérateur numéro : 24 avec la commande numéro : 59
Opérateur numéro : 18 avec la commande numéro : 60
Opérateur numéro : 14 avec la commande numéro : 61
Opérateur numéro : 3 avec la commande numéro : 62
Opérateur numéro : 31 avec la commande numéro : 63
Opérateur numéro : 26 avec la commande numéro : 64
et 9 successeurs :
Opérateur numéro : 5 atteint sur l'entrée : 1 commande numéro : 65
Opérateur numéro : 4 atteint sur l'entrée : 2 pas de commande
Opérateur numéro : 4 atteint sur l'entrée : 1 commande numéro : 66
Opérateur numéro : 32 atteint sur l'entrée : 1 commande numéro : 67
Opérateur numéro : 31 atteint sur l'entrée : 1 commande numéro : 68
Opérateur numéro : 25 atteint sur l'entrée : 1 pas de commande
Opérateur numéro : 19 atteint sur l'entrée : 1 pas de commande
Opérateur numéro : 15 atteint sur l'entrée : 1 pas de commande
Opérateur numéro : 11 atteint sur l'entrée : 1 pas de commande

Opérateur numéro : 31 nommé 00014134 d'espèce REGISTRE

Commande propre numéro : 69
Commande de RAZ : 72

Opérateur ayant 2 antécédents :
- antécédent numero : 29
- antécédent numero : 30
et 2 successeurs :
- successeur numéro : 29
- successeur numéro : 30

Opérateur numéro : 32 nommé 00015092 d'espèce REGISTRE

Commande propre numéro : 70 ,Commande de RAZ : 72

Opérateur ayant 2 antécédents :
- antécédent numero : 29
- antécédent numero : 30
et 1 successeurs :
- successeur numéro : 28

PARTIE CONTROLE DU CIRCUIT FIR-11

Etape numéro : 1 commencant au : 0.00000E+00 ième top
et ayant 14 commandes actives :

4,23,25,27,29,31,33,35,36,42,45,52,58,65.

Etape numéro : 2 commencant au : 1.00000E+00 ième top
et ayant 19 commandes actives :

1,4,14,16,18,20,22,24,26,28,30,32,34,36,42,45,52,58,65.

Etape numéro : 3 commencant au : 4.00000E+00 ième top
et ayant 21 commandes actives :

1,4,10,12,13,14,16,18,20,22,24,26,28,30,32,34,37,43,46,53,59.

Etape numéro : 4 commencant au : 7.00000E+00 ième top
et ayant 23 commandes actives :

1,4,10,11,12,13,14,16,18,20,22,24,26,28,30,32,34,37,43,46,53,60,65.

Etape numéro : 5 commencant au : 8.00000E+00 ième top
et ayant 21 commandes actives :

1,4,10,11,14,16,18,20,22,24,26,28,30,32,34,37,43,47,54,60,65.

Etape numéro : 6 commencant au : 9.00000E+00 ième top
et ayant 22 commandes actives :

1,3,4,10,11,14,16,18,20,22,24,26,28,30,32,34,37,43,47,54,60,65.

Etape numéro : 7 commencant au : 1.00000E+01 ième top
et ayant 22 commandes actives :

1,3,4,10,11,12,14,16,18,20,22,24,26,28,30,32,34,38,47,54,60,65.

Etape numéro : 8 commencant au : 1.10000E+01 ième top
et ayant 24 commandes actives :

1,3,4,10,11,12,13,14,16,18,20,22,24,26,28,30,32,34,38,43,47,54,61,65.

Etape numéro : 9 commencant au : 1.20000E+01 ième top
et ayant 25 commandes actives :

1,3,4,10,11,12,13,14,16,18,20,22,24,26,28,30,32,34,38,43,47,54,61,65,69,

Etape numéro : 10 commencant au : 1.30000E+01 ième top
et ayant 23 commandes actives :

1,3,4,12,13,14,16,18,20,22,24,26,28,30,32,34,38,43,46,55,61,65,69

Etape numéro : 11 commencant au : 1.60000E+01 ième top
et ayant 24 commandes actives :

1,3,4,12,13,14,16,18,20,22,24,26,28,30,32,34,38,43,46,55,61,65,69,70

Etape numéro : 12 commencant au : 1.70000E+01 ième top
et ayant 22 commandes actives :

1,3,4,14,16,18,20,22,24,26,28,30,32,34,38,43,48,53,61,65,69,70

Etape numéro : 13 commencant au : 1.80000E+01 ième top
et ayant 22 commandes actives :

1,3,4,12,13,14,16,18,20,22,24,26,28,30,32,34,39,48,53,62,69,70,

Etape numéro : 14 commencant au : 2.10000E+01 ième top
et ayant 23 commandes actives :

1,5,11,12,13,14,16,18,20,22,24,26,28,30,32,34,39,48,53,62,66,69,70

Etape numéro : 15 commencant au : 2.20000E+01 ième top
et ayant 24 commandes actives :

1,3,5,11,12,13,14,16,18,2022,24,26,28,30,32,34,39,44,49,56,62,66,70

Etape numéro : 16 commencant au : 3.10000E+01 ième top
et ayant 24 commandes actives :

1,6,10,11,12,13,14,16,18,20,22,24,26,28,30,32,34,39,44,49,56,62,67,70

Etape numéro : 17 commencant au : 3.20000E+01 ième top
et ayant 24 commandes actives :

1,3,6,10,11,12,13,14,16,18 20,22,24,26,28,30,32,34,40,44,49,56,62,67

Etape numéro : 18 commencant au : 4.10000E+01 ième top
et ayant 25 commandes actives :

1,3,6,10,11,12,13,14,16,18,20,22,24,26,28,30,32,34,40,44,49,56,62,67,70

Etape numéro : 19 commencant au : 4.20000E+01 ième top
et ayant 23 commandes actives :

1,7,10,11,12,13,14,16,18,20,22,24,26,28,30,32,34,40,44,49,56,63,70

Etape numéro : 20 commencant au : 4.30000E+01 ième top
et ayant 22 commandes actives :

1,3,7,10,11,14,16,18,20,22,24,26,28,30,32,34,39,42,49,56,63,70

Etape numéro : 21 commencant au : 4.40000E+01 ième top
et ayant 23 commandes actives :

1,3,7,10,11,12,14,16,18,20,22,24,26,28,30,32,34,39,42,50,52,63,70

Etape numéro : 22 commencant au : 4.70000E+01 ième top
et ayant 24 commandes actives :

1,3,7,10,11,12,13,14,16,18,20,22,24,26,28,30,32,34,39,42,47,54,63,70

Etape numéro : 23 commencant au : 5.00000E+01 ième top
et ayant 25 commandes actives :

1,3,7,10,11,12,13,14,16,18,20,22,24,26,28,30,32,34,39,42,47,54,63,69,70

Etape numéro : 24 commencant au : 5.10000E+01 ième top
et ayant 23 commandes actives :

1,3,7,12,13,14,16,18,20,22,24,26,28,30,32,34,39,42,46,53,63,69,70

Etape numéro : 25 commencant au : 5.20000E+01 ième top
et ayant 23 commandes actives :

1,3,7,10,12,13,14,16,18,20,22,24,26,28,30,32,34,41,46,53,63,69,70

Etape numéro : 26 commencant au : 5.40000E+01 ième top
et ayant 24 commandes actives :

1,3,7,10,11,12,13,14,16,18,20,22,24,26,28,30,32,34,41,46,53,62,68,70

Etape numéro : 27 commencant au : 5.70000E+01 ième top
et ayant 25 commandes actives :

1,3,7,10,11,12,13,14,16,18,20,22,24,26,28,30,32,34,41,46,53,62,68,69,70

Etape numéro : 28 commencant au : 5.80000E+01 ième top
et ayant 24 commandes actives :

1,8,10,11,12,13,14,16,18,20,22,24,26,28,30,32,34,41,46,53,63,65,69,70

Etape numéro : 29 commencant au : 5.90000E+01 ième top
et ayant 24 commandes actives :

1,3,8,10,11,14,16,18,20,22,24,26,28,30,32,34,41,44,47,52,63,65,69,70

Etape numéro : 30 commencant au : 6.10000E+01 ième top
et ayant 24 commandes actives :

1,3,8,10,11,12,14,16,18,20,22,24,26,28,30,32,34,41,44,47,52,62,69,70

Etape numéro : 31 commencant au : 6.20000E+01 ième top
et ayant 23 commandes actives :

1,3,8,12,13,14,16,18,20,22,24,26,28,30,32,34,41,44,46,57,62,69,70

Etape numéro : 32 commencant au : 6.60000E+01 ième top
et ayant 22 commandes actives :

1,3,8,11,14,16,18,20,22,24,26,28,30,32,34,41,44,47,52,62,69,70

Etape numéro : 33 commencant au : 6.80000E+01 ième top
et ayant 23 commandes actives :

1,9,10,11,14,16,18,20,22,24,26,28,30,32,34,41,44,47,52,62,65,69,70

Etape numéro : 34 commencant au : 6.90000E+01 ième top
et ayant 23 commandes actives

1,3,9,1011,14,16,18,20,22,24,26,28,30,32,34,40,47,52,62,65,69,70

Etape numéro : 35 commencant au : 7.20000E+01 ième top
et ayant 23 commandes actives :

1,3,9,10,11,13,14,16,18,20,22,24,26,28,30,32,34,40,51,62,65,69,70

Etape numéro : 36 commencant au : 7.50000E+01 ième top
et ayant 23 commandes actives :

1,3,9,10,11,13,14,16,18,20,22,24,26,28,30,32,34,40,49,62,65,69,70

Etape numéro : 37 commencant au : 7.80000E+01 ième top
et ayant 24 commandes actives :

1,3,9,10,11,12,13,14,16,18,20,21,22,24,26,28,30,32,34,40,50,64,69,70

Etape numéro : 38 commencant au : 8.10000E+01 ième top
et ayant 24 commandes actives :

1,3,9,10,11,12,13,14,15,16,18,20,21,22,24,26,28,30,32,34,40,59,69,70

Etape numéro : 39 commencant au : 8.20000E+01 ième top
et ayant 25 commandes actives :

1,2,3,9,10,11,12,13,14,15,16,18,20,21,22,24,26,28,30,32,34,36,59,69,70

Etape numéro : 40 commencant au : 8.40000E+01 ième top
et ayant 25 commandes actives :

1,2,3,9,10,11,12,13,14,15,16,18,20,21,22,24,26,28,30,32,34,36,60,69,70

Etape numéro : 41 commencant au : 8.50000E+01 ième top
et ayant 25 commandes actives :

1,23,9,10,11,12,13,14,15,16,18,20,21,22,24,26,28,30,32,34,37,60,69,70

Etape numéro : 42 commencant au : 8.70000E+01 ième top
et ayant 25 commandes actives :

1,2,3,9,10,11,12,13,14,15,16,18,20,21,22,24,26,28,30,32,34,37,61,69,70

Etape numéro : 43 commencant au : 8.80000E+01 ième top
et ayant 25 commandes actives :

1,2,3,9,10,11,12,13,14,15,16,18,20,21,22,24,26,28,30,32,34,38,61,69,70

Etape numéro : 44 commencant au : 9.00000E+01 ième top
et ayant 25 commandes actives :

1,2,3,9,10,11,12,13,14,15,16,18,19,20,21,22,24,26,28,30,32,34,38,69,70

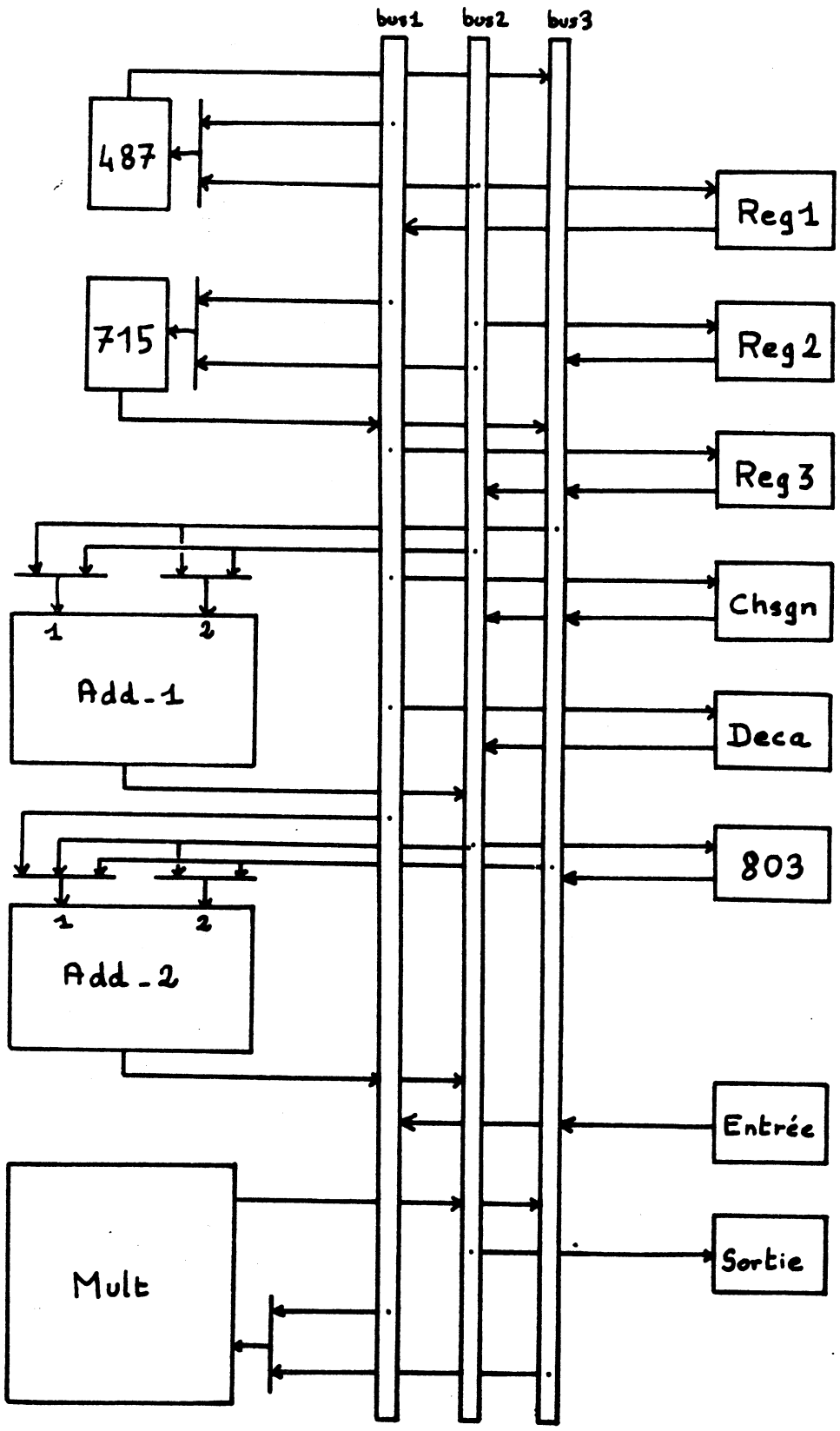
Etape numéro : 45 commencant au : 9.10000E+01 ième top
et ayant 25 commandes actives :

1,2,3,9,10,11,12,13,14,15,16,18,19,20,21,22,24,26,28,30,32,34,41,69,70

Etape numéro : 46 commencant au : 9.40000E+01 ième top
et ayant 26 commandes actives :

0,1,2,3,9,10,11,12,13,14,15,16,17,18,19,20,21,22,24,26,28,30,32,34,69,70

SHEMA-BLOC DE L'ARCHITECTURE TROUVEE POUR " LOUVAIN "



PARTIE OPERATIVE DU CIRCUIT LOUVAIN

Opérateur numéro : 1 nommé ENTREE d'espèce ENTREE

Commande propre numéro : 1
Commande de RAZ : 56

Opérateur ayant 0 antécédents :
et 2 successeurs :
- successeur numéro : 14
- successeur numéro : 16

Opérateur numéro : 2 nommé SORTIE d'espèce SORTIE

Commande propre numéro : 2
Commande de RAZ : 56

Opérateur ayant 1 antécédents :
- antécédent numero : 15
et 0 successeurs :

Opérateur numéro : 3 nommé MULT d'espèce MULTIPLI

l'entrée : 1 est latchée avec la commande numéro : 3

Commande de RAZ : 56

Ce multiplieur général fait 3 multiplications
la multiplication numéro 22 avec la commande numéro : 4
la multiplication numéro 14 avec la commande numéro : 5
la multiplication numéro 5 avec la commande numéro : 6

Opérateur ayant : 2 antécédents sur l'entrée : 1
- antécédent numéro : 14
- antécédent numéro : 16
et : 2 successeurs
- successeur numéro : 16
- successeur numéro : 15

Opérateur numéro : 4 nommé ADDIT1 d'espèce ADDITION

l'entrée : 1 est latchée avec la commande numéro : 7
l'entrée : 2 est latchée avec la commande numéro : 8

Commande de RAZ : 56

Opérateur ayant : 2 antécédents sur l'entrée : 1
- antécédent numéro : 16
- antécédent numéro : 15
Opérateur ayant : 2 antécédents sur l'entrée : 2
- antécédent numéro : 16
- antécédent numéro : 15
et : 1 successeurs
- successeur numéro : 15

Opérateur numéro : 5 nommé ADDIT2 d'espèce ADDITION

l'entrée : 1 est latchée avec la commande numéro : 9
l'entrée : 2 est latchée avec la commande numéro : 10

Commande de RAZ : 56

Opérateur ayant : 3 antécédents sur l'entrée : 1
- antécédent numéro : 16
- antécédent numéro : 14
- antécédent numéro : 15
Opérateur ayant : 2 antécédents sur l'entrée : 2
- antécédent numéro : 15
- antécédent numéro : 16
et : 2 successeurs
- successeur numéro : 14
- successeur numero : 15

Opérateur numéro : 7 nommé REG1 d'espèce RETARD Z

Commandes propres numéro : 11,12
Commande de RAZ : 56

Opérateur ayant 1 antécédents :
- antécédent numero : 15
et 1 successeurs :
- successeur numéro : 14

Opérateur numéro : 9 nommé REG2 d'espèce RETARD Z

Commandes propres numéro : 13,14
Commande de RAZ : 56

Opérateur ayant 1 antécédents :
- antécédent numero : 15
et 1 successeurs :
- successeur numéro : 16

Opérateur numéro : 11 nommé REG3 d'espèce RETARD Z

Commandes propres numéro : 15,16
Commande de RAZ : 56

Opérateur ayant 1 antécédents :
- antécédent numero : 14
et 2 successeurs :
- successeur numéro : 16
- successeur numéro : 15

Opérateur numéro : 12 nommé CHSGN d'espèce CHANGSGN

l'entrée : 1 est latchée avec la commande numéro : 17

Commande de RAZ : 56

Opérateur ayant : 1 antécédents sur l'entrée : 1
- antécédent numéro : 14
et : 2 successeurs
- successeur numéro : 15
- successeur numéro : 16

Opérateur numéro : 13 nommé DECA d'espèce DECALAGE

l'entrée : 1 est latchée avec la commande numéro : 18

Commande de RAZ : 56

Ce décaleur est spécialisé (pas de commandes).

Opérateur ayant : 1 antécédents sur l'entrée : 1
- antécédent numéro : 14
et : 1 successeurs
- successeur numéro : 15

Opérateur numéro : 14 nommé BUS_1 d'espèce BUS

Bus ayant 4 antécedents :
Opérateur numéro : 5 avec la commande numéro : 19
Opérateur numéro : 1 avec la commande numéro : 20
Opérateur numéro : 6 avec la commande numéro : 21
Opérateur numéro : 17 avec la commande numéro : 22
et 7 successeurs :
Opérateur numéro : 3 atteint sur l'entrée : 1 commande numéro : 23
Opérateur numéro : 5 atteint sur l'entrée : 1 commande numéro : 24
Opérateur numéro : 12 atteint sur l'entrée : 1 pas de commande
Opérateur numéro : 17 atteint sur l'entrée : 1 commande numéro : 25
Opérateur numéro : 18 atteint sur l'entrée : 1 commande numéro : 26
Opérateur numéro : 13 atteint sur l'entrée : 1 pas de commande
Opérateur numéro : 11 atteint sur l'entrée : 1 pas de commande

Opérateur numéro : 15 nommé BUS_2 d'espèce BUS

Bus ayant 6 antécedents :
Opérateur numéro : 10 avec la commande numéro : 27
Opérateur numéro : 4 avec la commande numéro : 28
Opérateur numéro : 5 avec la commande numéro : 29
Opérateur numéro : 12 avec la commande numéro : 30
Opérateur numéro : 3 avec la commande numéro : 31
Opérateur numéro : 13 avec la commande numéro : 32
et 10 successeurs :
Opérateur numéro : 5 atteint sur l'entrée : 2 commande numéro : 33
Opérateur numéro : 18 atteint sur l'entrée : 1 commande numéro : 34
Opérateur numéro : 5 atteint sur l'entrée : 1 commande numéro : 35
Opérateur numéro : 4 atteint sur l'entrée : 1 commande numéro : 36
Opérateur numéro : 19 atteint sur l'entrée : 1 pas de commande
Opérateur numéro : 17 atteint sur l'entrée : 1 commande numéro : 37
Opérateur numéro : 4 atteint sur l'entrée : 2 commande numéro : 38
Opérateur numéro : 2 atteint sur l'entrée : 1 pas de commande
Opérateur numéro : 9 atteint sur l'entrée : 1 pas de commande
Opérateur numéro : 7 atteint sur l'entrée : 1 pas de commande

Opérateur numéro : 16 nommé BUS_3 d'espèce BUS

Bus ayant 8 antécedents :
Opérateur numéro : 8 avec la commande numéro : 39
Opérateur numéro : 3 avec la commande numéro : 40
Opérateur numéro : 12 avec la commande numéro : 41
Opérateur numéro : 1 avec la commande numéro : 42
Opérateur numéro : 18 avec la commande numéro : 43
Opérateur numéro : 10 avec la commande numéro : 44
Opérateur numéro : 19 avec la commande numéro : 45
Opérateur numéro : 17 avec la commande numéro : 46
et 5 successeurs :
Opérateur numéro : 5 atteint sur l'entrée : 1 commande numéro : 47
Opérateur numéro : 4 atteint sur l'entrée : 1 commande numéro : 48
Opérateur numéro : 4 atteint sur l'entrée : 2 commande numéro : 49
Opérateur numéro : 5 atteint sur l'entrée : 2 commande numéro : 50
Opérateur numéro : 3 atteint sur l'entrée : 1 commande numéro : 51

Opérateur numéro : 17 nommé 11043081 d'espèce REGISTRE

Commande propre numéro : 52
Commande de RAZ : 56

Opérateur ayant 2 antécédents :
- antécédent numero : 14
- antécédent numero : 15
et 2 successeurs :
- successeur numéro : 14
- successeur numéro : 16

Opérateur numéro : 18 nommé 11045484 d'espèce REGISTRE

Commande propre numéro : 53
Commande de RAZ : 56

Opérateur ayant 2 antécédents :
- antécédent numero : 15
- antécédent numero : 14
et 1 successeurs :
- successeur numéro : 16

Opérateur numéro : 19 nommé 11063109 d'espèce REGISTRE

Commande propre numéro : 54
Commande de RAZ : 56

Opérateur ayant 1 antécédents :
- antécédent numero : 15
et 1 successeurs :
- successeur numéro : 16

PARTIE CONTROLE DU CIRCUIT LOUVAIN

Etape nume'ro : 1 commencant au : 0.00000E+00 ie'me top
et ayant 9 commandes actives :

4,12,19,23,27,33,39,47,48

Etape nume'ro : 2 commencant au : 1.00000E+00 ie'me top
et ayant 12 commandes actives :

1,4,11,13,15,19,23,27,33,39,47,48

Etape nume'ro : 3 commencant au : 4.00000E+00 ie'me top
et ayant 14 commandes actives :

1,4,7,9,10,11,13,15,19,23,28,33,40,49

Etape nume'ro : 4 commencant au : 9.00000E+00 ie'me top
et ayant 13 commandes actives :

1,3,4,7,11,13,15,20,24,28,33,40,49

Etape nume'ro : 5 commencant au : 1.20000E+01 ie'me top
et ayant 13 commandes actives :

1,3,4,7,10,11,13,15,21,28,33,40,49

Etape nume'ro : 6 commencant au : 1.50000E+01 ie'me top
et ayant 15 commandes actives :

1,3,4,7,10,11,13,15,17,19,25,28,33,40,49

Etape nume'ro : 7 commencant au : 2.10000E+01 ie'me top
et ayant 15 commandes actives :

1,3,4,7,8,10,11,13,15,17,19,25,28,33,41

Etape nume'ro : 8 commencant au : 2.30000E+01 ie'me top
et ayant 16 commandes actives :

1,3,4,7,8,10,11,13,15,17,19,25,28,33,41,47

Etape nume'ro : 9 commencant au : 2.70000E+01 ie'me top
et ayant 15 commandes actives :

1,3,4,7,8,9,10,11,13,15,17,19,25,41,47

Etape nume'ro : 10 commencant au : 3.20000E+01 ie'me top
et ayant 16 commandes actives :

1,3,4,7,8,9,10,11,13,15,17,19,25,41,47,52

Etape nume'ro : 11 commencant au : 3.30000E+01 ie'me top
et ayant 17 commandes actives :

1,3,4,7,8,10,11,13,15,17,22,23,29,34,41,47,52

Etape nume'ro : 12 commencant au : 3.40000E+01 ie'me top
et ayant 18 commandes actives :

1,3,4,7,8,9,10,11,13,15,17,22,23,29,34,42,50,52

Etape nume'ro : 13 commencant au : 4.20000E+01 ie'me top
et ayant 19 commandes actives :

1,3,4,7,8,9,10,11,13,15,17,22,23,29,34,42,50,52,53

Etape nume'ro : 14 commencant au : 4.30000E+01 ie'me top
et ayant 16 commandes actives :

1,3,4,7,8,11,13,15,17,22,23,28,35,43,52,53

Etape nume'ro : 15 commencant au : 4.60000E+01 ie'me top
et ayant 15 commandes actives :

1,3,4,9,11,13,15,17,22,23,30,36,43,52,53

Etape nume'ro : 16 commencant au : 4.90000E+01 ie'me top
et ayant 14 commandes actives :

1,3,4,7,9,11,13,15,22,23,31,43,52,53

Etape nume'ro : 17 commencant au : 5.20000E+01 ie'me top
et ayant 15 commandes actives :

1,3,4,7,9,11,13,15,22,23,31,43,52,53,54

Etape nume'ro : 18 commencant au : 5.30000E+01 ie'me top
et ayant 15 commandes actives :

1,5,7,9,11,13,15,22,23,31,33,43,52,53,54

Etape nume'ro : 19 commencant au : 5.40000E+01 ie'me top
et ayant 15 commandes actives :

1,3,5,7,9,11,13,15,19,31,33,43,51,53,54

Etape nume'ro : 20 commencant au : 6.60000E+01 ie'me top
et ayant 16 commandes actives :

1,3,5,7,9,10,11,13,15,19,30,35,43,51,53,54

Etape nume'ro : 21 commencant au : 7.10000E+01 ie'me top
et ayant 16 commandes actives :

1,3,5,7,11,13,15,17,20,26,30,35,43,51,53,54

Etape nume'ro : 22 commencant au : 7.70000E+01 ie'me top
et ayant 17 commandes actives :

1,3,5,7,9,11,13,15,17,20,26,31,37,43,51,53,54

Etape nume'ro : 23 commencant au : 8.00000E+01 ie'me top
et ayant 18 commandes actives :

1,3,5,7,9,11,13,15,17,20,26,31,37,43,51,52,53,54

Etape nume'ro : 24 commencant au : 8.10000E+01 ie'me top
et ayant 17 commandes actives :

1,6,7,9,11,13,15,17,20,26,31,38,43,51,52,53,54

Etape nume'ro : 25 commencant au : 8.20000E+01 ie'me top
et ayant 17 commandes actives :

1,3,6,7,9,11,13,15,17,20,26,31,38,44,49,52,54

Etape nume'ro : 26 commencant au : 8.30000E+01 ie'me top
et ayant 18 commandes actives :

1,3,6,7,9,11,13,15,17,20,26,31,38,44,49,52,53,54

Etape nume'ro : 27 commencant au : 9.50000E+01 ie'me top
et ayant 18 commandes actives :

1,3,6,7,8,9,11,13,15,17,19,28,33,44,49,52,53,54

Etape nume'ro : 28 commencant au : 1.00000E+02 ie'me top
et ayant 16 commandes actives :

1,3,6,9,10,11,13,15,17,19,31,44,49,52,53,54

Etape nume'ro : 29 commencant au : 1.01000E+02 ie'me top
et ayant 17 commandes actives :

1,3,6,8,9,10,11,13 15,17,19,31,45,48,52,53,54

Etape nume'ro : 30 commencant au : 1.04000E+02 ie'me top
et ayant 17 commandes actives :

1,3,6,7,8,9,10,11,13,15,17,19,31,46,47,52,53

Etape nume'ro : 31 commencant au : 1.05000E+02 ie'me top
et ayant 18 commandes actives :

1,3,6,7,8,9,10,11,13,15,17,19,31,46,47,52,53,54

Etape nume'ro : 32 commencant au : 1.06000E+02 ie'me top
et ayant 17 commandes actives :

1,3,6,7,8,11,13,15,17,18,19,32,46,47,52,53,54

Etape nume'ro : 33 commencant au : 1.08000E+02 ie'me top
et ayant 19 commandes actives :

1,3,6,7,8,9,11,13,15,17,18,19,32,43,49,50,52,53,54

Etape nume'ro : 34 commencant au : 1.09000E+02 ie'me top
et ayant 20 commandes actives :

1,2,3,6,7,8,9,11,13,15,17,18,19,28,43,49,50,52,53,54

Etape nume'ro : 35 commencant au : 1.11000E+02 ie'me top
et ayant 20 commandes actives :

1,2,3,6,7,8,9,10,11,13,15,17,18,19,28,43,49,52,53,54

Etape nume'ro : 36 commencant au : 1.12000E+02 ie'me top
et ayant 20 commandes actives :

1,2,3,6,7,9,10,11,13 14,15,17,18,19,28,43,49,52,53,54

Etape nume'ro : 37 commencant au : 1.13000E+02 ie'me top
et ayant 21 commandes actives :

1,2,3,6,7,8,9,10,11,13,14,15,17,18,19,28,45,49,52,53,54

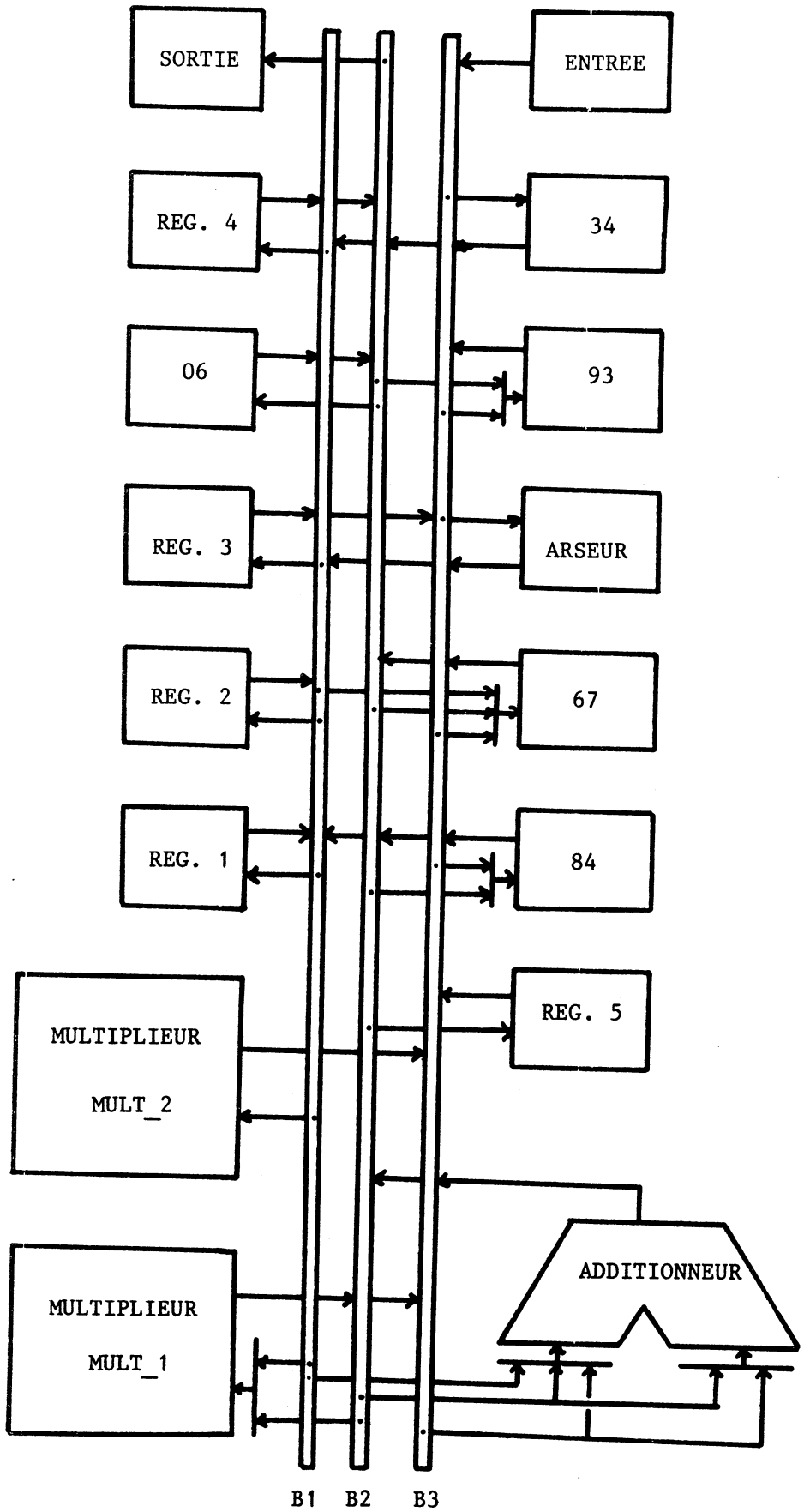
Etape nume'ro : 38 commencant au : 1.16000E+02 ie'me top
et ayant 21 commandes actives :

1,2,3,6,7,8,9,10,11,13,14,15,16,17,18,28,45,49,52,53,54

Etape nume'ro : 39 commencant au : 1.21000E+02 ie'me top
et ayant 20 commandes actives :

0,1,2,3,6,7,8,9,10,11,13,14,15,16,17,18,49,52,53,54

SHEMA BLOC DE L'ARCHITECTURE DU FILTRE BIQUAD 2



PARTIE OPERATIVE DU CIRCUIT "BIQUAD 2"

Opérateur numéro : 1 nommé ENTREE_F d'espèce ENTREE

Commande propre numéro : 1
Commande de RAZ : 71

Opérateur ayant 0 antécédents :
et 1 successeurs :
- successeur numéro : 19

Opérateur numéro : 2 nommé SORTIE_F d'espèce SORTIE

Commande propre numéro : 2
Commande de RAZ : 71

Opérateur ayant 1 antécédents :
- antécédent numero : 18
et 0 successeurs :

Opérateur numéro : 3 nommé MULT_1 d'espèce MULTIPLI

l'entrée : 1 est latchée avec la commande numéro : 3

Commande de RAZ : 71

Ce multiplieur général fait 5 multiplications
la multiplication numéro 32 avec la commande numéro : 4
la multiplication numéro 18 avec la commande numéro : 5
la multiplication numéro 29 avec la commande numéro : 6
la multiplication numéro 8 avec la commande numéro : 7
la multiplication numéro 16 avec la commande numéro : 8

Opérateur ayant : 2 antécédents sur l'entrée : 1
- antécédent numéro : 17
- antécédent numéro : 18
et : 2 successeurs
- successeur numéro : 19
- successeur numéro : 18

Opérateur numéro : 4 nommé MULT_2 d'espèce MULTIPLI

l'entrée : 1 est latchée avec la commande numéro : 9

Commande de RAZ : 71

Ce multiplieur général fait 3 multiplications
la multiplication numéro 14 avec la commande numéro : 10
la multiplication numéro 23 avec la commande numéro : 11
la multiplication numéro 31 avec la commande numéro : 12

Opérateur ayant : 1 antécédents sur l'entrée : 1
- antécédent numéro : 17
et : 1 successeurs
- successeur numéro : 19

Opérateur numéro : 5 nommé ADDNEUR d'espèce ADDITION

l'entrée : 1 est latchée avec la commande numéro : 13
l'entrée : 2 est latchée avec la commande numéro : 14

Commande de RAZ : 71

Opérateur ayant : 2 antécédents sur l'entrée : 1
- antécédent numéro : 19
- antécédent numéro : 18
Opérateur ayant : 3 antécédents sur l'entrée : 2
- antécédent numéro : 17
- antécédent numéro : 19
- antécédent numéro : 18
et : 2 successeurs
- successeur numéro : 19
- successeur numéro : 18

Opérateur numéro : 7 nommé R1 d'espèce RETARD Z

Commandes propres numéro : 15,16
Commande de RAZ : 71

Opérateur ayant 1 antécédents :
- antécédent numero : 17
et 1 successeurs :
- successeur numéro : 17

Opérateur numéro : 9 nommé R2 d'espèce RETARD Z

Commandes propres numéro : 17,18
Commande de RAZ : 71

Opérateur ayant 1 antécédents :
- antécédent numero : 17
et 1 successeurs :
- successeur numéro : 17

Opérateur numéro : 11 nommé R3 d'espèce RETARD Z

Commandes propres numéro : 19,20
Commande de RAZ : 71

Opérateur ayant 1 antécédents :
- antécédent numero : 17
et 2 successeurs :
- successeur numéro : 19
- successeur numéro : 17

Opérateur numéro : 13 nommé R4 d'espèce RETARD Z

Commandes propres numéro : 21,22
Commande de RAZ : 71

Opérateur ayant 1 antécédents :
- antécédent numero : 17
et 2 successeurs :
- successeur numéro : 18
- successeur numéro : 17

Opérateur numéro : 15 nommé R5 d'espèce RETARD Z

Commandes propres numéro : 23,24
Commande de RAZ : 71

Opérateur ayant 1 antécédents :
- antécédent numero : 18
et 1 successeurs :
- successeur numéro : 19

Opérateur numéro : 16 nommé ARRSEUR d'espèce ARRONDI

l'entrée : 1 est latchée avec la commande numéro : 25

Commande de RAZ : 71

Opérateur ayant : 1 antécédents sur l'entrée : 1
- antécédent numéro : 19
et : 2 successeurs
- successeur numéro : 19
- successeur numéro : 17

Opérateur numéro : 17 nommé BUS_1 d'espèce BUS

Bus ayant 8 antécedents :
Tache numéro : 12 avec la commande numéro : 26
Tache numéro : 10 avec la commande numéro : 27
Tache numéro : 6 avec la commande numéro : 28
Tache numéro : 6 avec la commande numéro : 29
Tache numéro : 21 avec la commande numéro : 30
Tache numéro : 22 avec la commande numéro : 31
Tache numéro : 24 avec la commande numéro : 32
Tache numéro : 16 avec la commande numéro : 33
et 8 successeurs :
Tache numéro : 4 atteinte sur l'entrée : 1 pas de commande
Tache numéro : 5 atteinte sur l'entrée : 2 commande numéro : 34
Tache numéro : 3 atteinte sur l'entrée : 1 commande numéro : 35
Tache numéro : 20 atteinte sur l'entrée : 1 commande numéro : 36
Tache numéro : 13 atteinte sur l'entrée : 1 pas de commande
Tache numéro : 9 atteinte sur l'entrée : 1 pas de commande
Tache numéro : 11 atteinte sur l'entrée : 1 pas de commande
Tache numéro : 7 atteinte sur l'entrée : 1 pas de commande

Opérateur numéro : 18 nommé BUS_2 d'espèce BUS

Bus ayant 7 antécedents :
Tache numéro : 12 avec la commande numéro : 37
Tache numéro : 5 avec la commande numéro : 38
Tache numéro : 20 avec la commande numéro : 39
Tache numéro : 3 avec la commande numéro : 40
Tache numéro : 21 avec la commande numéro : 41
Tache numéro : 22 avec la commande numéro : 42
Tache numéro : 24 avec la commande numéro : 43
et 9 successeurs :
Tache numéro : 24 atteinte sur l'entrée : 1 pas de commande
Tache numéro : 23 atteinte sur l'entrée : 1 commande numéro : 44
Tache numéro : 3 atteinte sur l'entrée : 1 commande numéro : 45
Tache numéro : 5 atteinte sur l'entrée : 2 commande numéro : 46
Tache numéro : 20 atteinte sur l'entrée : 1 commande numéro : 47
Tache numéro : 22 atteinte sur l'entrée : 1 commande numéro : 48
Tache numéro : 5 atteinte sur l'entrée : 1 commande numéro : 49
Tache numéro : 15 atteinte sur l'entrée : 1 pas de commande
Tache numéro : 2 atteinte sur l'entrée : 1 pas de commande

```
**********************************************************
Opérateur numéro :  19 nommé BUS_3    d'espèce BUS

Bus ayant    10 antécedents :
Tache numéro :    4 avec la commande numéro :   50
Tache numéro :    5 avec la commande numéro :   51
Tache numéro :    1 avec la commande numéro :   52
Tache numéro :   10 avec la commande numéro :   53
Tache numéro :   20 avec la commande numéro :   54
Tache numéro :   22 avec la commande numéro :   55
Tache numéro :   14 avec la commande numéro :   56
Tache numéro :   16 avec la commande numéro :   57
Tache numéro :   23 avec la commande numéro :   58
Tache numéro :    3 avec la commande numéro :   59
      et          7 successeurs :
Tache numéro :    5 atteinte sur l'entrée :   1 commande numéro :   60
Tache numéro :   20 atteinte sur l'entrée :   1 commande numéro :   61
Tache numéro :   22 atteinte sur l'entrée :   1 commande numéro :   62
Tache numéro :    5 atteinte sur l'entrée :   2 commande numéro :   63
Tache numéro :   21 atteinte sur l'entrée :   1 pas de commande
Tache numéro :   16 atteinte sur l'entrée :   1 pas de commande
Tache numéro :   23 atteinte sur l'entrée :   1 commande numéro :   64
**********************************************************
Opérateur numéro :  20 nommé 23010567 d'espèce REGISTRE

Commande propre numéro :   65
Commande de RAZ        :         71

Opérateur ayant      3 antécédents :
- antécédent numero :  19
- antécédent numero :  17
- antécédent numero :  18
          et          2 successeurs :
- successeur numéro :  19
- successeur numéro :  18
**********************************************************
Opérateur numéro :  21 nommé 23011034 d'espèce REGISTRE

Commande propre numéro :   66
Commande de RAZ        :         71

Opérateur ayant      1 antécédents :
- antécédent numero :  19
          et          2 successeurs :
- successeur numéro :  17
- successeur numéro :  18
**********************************************************
Opérateur numéro :  22 nommé 23131484 d'espèce REGISTRE

Commande propre numéro :   67
Commande de RAZ        :         71

Opérateur ayant      2 antécédents :
- antécédent numero :  19
- antécédent numero :  18
          et          3 successeurs :
- successeur numéro :  19
- successeur numéro :  17
- successeur numéro :  18
**********************************************************
Opérateur numéro :  23 nommé 23524893 d'espèce REGISTRE

Commande propre numéro :   68
Commande de RAZ        :         71

Opérateur ayant      2 antécédents :
- antécédent numero :  18
- antécédent numero :  19
          et          1 successeurs :
- successeur numéro :  19
**********************************************************
Opérateur numéro :  24 nommé 00103506 d'espèce REGISTRE

Commande propre numéro :   69
Commande de RAZ        :         71

Opérateur ayant      1 antécédents :
- antécédent numero :  18
          et          2 successeurs :
- successeur numéro :  17
- successeur numéro :  18
```

PARTIE CONTROLE DU CIRCUIT BIQUAD 2

Etape numéro : 1 commencant au : 0.00000E+00 ième top
et ayant 9 commandes actives :

4,10,20,22,24,26,37,50,60

Etape numéro : 2 commencant au : 1.00000E+00 ième top
et ayant 12 commandes actives :

1,4,10,15,17,19,21,23,26,37,50,60

Etape numéro : 3 commencant au : 3.00000E+00 ième top
et ayant 13 commandes actives :

1,4,10,15,17,19,21,23,26,37,50,60,69

Etape numéro : 4 commencant au : 4.00000E+00 ième top
et ayant 14 commandes actives :

1,4,9,10,15,17,19,21,23,27,34,50,60,69

Etape numéro : 5 commencant au : 6.00000E+00 ième top
et ayant 14 commandes actives :

1,4,9,10,14,15,17,19,21,23,26,50,60,69

Etape numéro : 6 commencant au : 1.50000E+01 ième top
et ayant 14 commandes actives :

1,4,11,13,14,15,17,19,21,23,26,51,61,69

Etape numéro : 7 commencant au : 1.60000E+01 ième top
et ayant 16 commandes actives :

1,4,9,11,13,14,15,17,19,21,23,29,35,51,61,69

Etape numéro : 8 commencant au : 1.90000E+01 ième top
et ayant 16 commandes actives :

1,3,4,9,11,13,14,15,17,19,21,23,29,51,61,69

Etape numéro : 9 commencant au : 2.00000E+01 ième top
et ayant 17 commandes actives :

1,3,4,9,11,13,14,15,17,19,21,23,29,51,61,65,69

Etape numéro : 10 commencant au : 2.10000E+01 ième top
et ayant 17 commandes actives :

1,3,4,9,11,15,17,19,21,23,29,36,44,52,62,65,69

Etape numéro : 11 commencant au : 2.30000E+01 ième top
et ayant 18 commandes actives :

1,3,4,9,11,15,17,19,21,23,29,36,44,52,62,65,67,69

Etape numéro : 12 commencant au : 2.40000E+01 ième top
et ayant 18 commandes actives :

1,3,4,9,11,15,17,19,21,23,29,36,44,50,63,65,67,69

Etape numéro : 13 commencant au : 2.90000E+01 ième top
et ayant 18 commandes actives :

1,3,4,12,14,15,17,19,21,23,29,36,44,50,60,65,67,69

Etape numéro : 14 commencant au : 3.00000E+01 ième top
et ayant 20 commandes actives :

1,3,4,9,12,14,15,17,19,21,23,26,36,36,44,50,60,65,67,69

Etape numéro : 15 commencant au : 4.10000E+01 ième top
et ayant 20 commandes actives :

1,3,4,9,12,13,14,15,17,19,21,23,26,36,38,44,53,65,67,69

Etape numéro : 16 commencant au : 4.30000E+01 ième top
et ayant 21 commandes actives :

1,3,4,9,12,13,14,15,17,19,21,23,26,36,38,44,53,65,66,67,69

Etape numéro : 17 commencant au : 4.40000E+01 ième top
et ayant 22 commandes actives :

1,3,4,9,12,13,14,15,17,19,21,23,26,36,38,44,54,60,65,66,67,69

Etape numéro : 18 commencant au : 4.70000E+01 ième top
et ayant 23 commandes actives :

1,3,4,9,12,13,14,15,17,19,21,23,26,36,38,44,54,60,65,66,67,68,69

Etape numéro : 19 commencant au : 4.80000E+01 ième top
et ayant 19 commandes actives :

1,3,4,9,12,15,17,19,21,23,26,36,54,60,65,66,67,68,69

Etape numéro : 20 commencant au : 4.90000E+01 ième top
et ayant 20 commandes actives :

1,3,4,9,12,13,15,17,19,21,23,26,36,39,55,63,66,67,68,69

Etape numéro : 21 commencant au : 5.00000E+01 ième top
et ayant 21 commandes actives :

1,3,4,9,12,13,15,17,19,21,23,26,36,39,55,63,65,66,67,68,69

Etape numéro : 22 commencant au : 5.10000E+01 ième top
et ayant 21 commandes actives :

1,3,4,9,12,13,15,17,19,21,23,30,35,39,55,63,65,66,67,68,69

Etape numéro : 23 commencant au : 5.30000E+01 ième top
et ayant 20 commandes actives :

1,3,4,9,12,13,14,15,17,19,21,23,30,35,39,51,65,66,68,69

Etape numéro : 24 commencant au : 5.60000E+01 ième top
et ayant 20 commandes actives :

1,3,4,9,12,15,17,19,21,23,25,30,35,39,56,63,65,66,68,69

Etape numéro : 25 commencant au : 6.00000E+01 ième top
et ayant 21 commandes actives :

1,3,4,9,12,14,15,17,19,21,23,25,30,35,39,57,62,65,66,68,69

Etape numéro : 26 commencant au : 6.50000E+01 ième top
et ayant 22 commandes actives :

1,3,4,9,12,14,15,17,19,21,23,25,30,35,39,57,62,65,66,67,68,69

Etape numéro : 27 commencant au : 6.60000E+01 ième top
et ayant 22 commandes actives :

1,3,4,9,12,14,15,17,19,21,23,25,30,35,39,58,60,65,66,67,68,69

Etape numéro : 28 commencant au : 6.80000E+01 ième top
et ayant 22 commandes actives :

1,3,4,9,12,13,14,15,17,19,21,23,25,30,35,39,51,64,65,66,67,69

Etape numéro : 29 commencant au : 7.30000E+01 ième top
et ayant 23 commandes actives :

1,3,4,9,12,13,14,15,17,19,21,23,25,30,35,39,51,64,65,66,67,68,69

Etape numéro : 30 commencant au : 7.40000E+01 ième top
et ayant 21 commandes actives :

1,3,4,9,12,15,17,19,21,23,25,30,35,39,59,60,65,66,67,68,69

Etape numéro : 31 commencant au : 7.60000E+01 ième top
et ayant 20 commandes actives :

1,5,9,12,13,15,17,19,21,23,25,30,35,39,59,65,66,67,68,69

Etape numéro : 32 commencant au : 7.70000E+01 ième top
et ayant 20 commandes actives :

1,3,5,9,12,13,15,17,19,21,23,25,31,39,45,59,65,67,68,69

Etape numéro : 33 commencant au : 8.80000E+01 ième top
et ayant 21 commandes actives :

1,3,5,9,12,13,15,17,19,21,23,25,31,39,45,59,65,66,67,68,69

Etape numéro : 34 commencant au : 8.90000E+01 ième top
et ayant 20 commandes actives :

1,6,9,12,13,15,17,19,21,23,25,31,39,45,58,65,66,67,68,69

Etape numéro : 35 commencant au : 9.00000E+01 ième top
et ayant 21 commandes actives :

1,3,6,9,12,13,15,17,19,21,23,25,31,36,40,46,58,66,67,68,69

Etape numéro : 36 commencant au : 1.01000E+02 ième top
et ayant 21 commandes actives :

1,7,9,12,13,14,15,17,19,21,23,25,31,35,36,47,58,66,67,68,69

Etape numéro : 37 commencant au : 1.02000E+02 ième top
et ayant 21 commandes actives :

1,3,7,9,12,13,14,15,17,19,21,23,25,32,35,36,47,58,66,68,69

Etape numéro : 38 commencant au : 1.06000E+02 ième top
et ayant 22 commandes actives :

1,3,7,9,12,13,14,15,17,19,21,23,25,32,35,36,47,58,65,66,68,69

Etape numéro : 39 commencant au : 1.17000E+02 ième top
et ayant 22 commandes actives :

1,3,7,9,12,13,14,15,17,19,21,23,25,32,35,40,48,58,65,66,68,69

Etape numéro : 40 commencant au : 1.13000E+02 ième top
et ayant 21 commandes actives :

1,3,7,9,12,15,17,19,21,23,25,32,35,40,48,58,65,66,67,68,69

Etape numéro : 41 commencant au : 1.14000E+02 ième top
et ayant 20 commandes actives :

1,8,9,12,15,17,19,21,23,25,32,35,40,46,58,65,66,67,68,69

Etape numéro : 42 commencant au : 1.15000E+02 ième top
et ayant 19 commandes actives :

1,3,8,9,12,15,17,19,21,23,25,33,40,46,58,65,66,67,68

Etape numéro : 43 commencant au : 1.17000E+02 ième top
et ayant 18 commandes actives :

1,3,8,9,12,15,17,19,21,23,33,40,46,58,65,66,67,68

Etape numéro : 44 commencant au : 1.18000E+02 ième top
et ayant 19 commandes actives :

1,3,8,9,12,15,17,19,21,23,25,33,40,46,54,65,66,67,68

Etape numéro : 45 commencant au : 1.25000E+02 ième top
et ayant 20 commandes actives :

1,3,8,9,12,15,17,18,19,21,23,25,33,40,46,54,65,66,67,68

Etape numéro : 46 commencant au : 1.26000E+02 ième top
et ayant 21 commandes actives :

1,3,8,9,12,14,15,17,18,19,21,23,25,33,41,49,54,65,66,67,68

Etape numéro : 47 commencant au : 1.28000E+02 ième top
et ayant 21 commandes actives :

1,3,8,9,12,13,14,15,17,18,19,21,23,25,33,38,54,65,66,67,68

Etape numéro : 48 commencant au : 1.33000E+02 ième top
et ayant 22 commandes actives :

1,3,8,9,12,13,14,15,17,18,19,21,23,25,33,38,54,65,66,67,68,69

Etape numéro : 49 commencant au : 1.34000E+02 ième top
et ayant 21 commandes actives :

1,3,8,9,12,15,17,18,19,21,23,25,33,42,46,54,65,66,67,68,69

Etape numéro : 50 commencant au : 1.36000E+02 ième top
et ayant 24 commandes actives :

1,3,8,9,12,14,15,17,18,19,21,23,25,33,34,43,49,54,60,65,66,67,68,69

Etape numéro : 51 commencant au : 1.38000E+02 ième top
et ayant 24 commandes actives :

1,3,8,9,12,13,14,15,17,18,19,21,23,25,33,34,38,54,60,65,66,67,68,69

Etape numéro : 52 commencant au : 1.43000E+02 ième top
et ayant 22 commandes actives :

1,3,8,9,12,15,17,18,19,21,23,25,33,34,38,54,60,65,66,67,68,69

Etape numéro : 53 commencant au : 1.44000E+02 ième top
et ayant 21 commandes actives :

1,3,8,9,12,13,14,15,17,18,19,21,23,25,26,38,65,66,67,68,69

Etape numéro : 54 commencant au : 1.46000E+02 ième top
et ayant 21 commandes actives :

1,3,8,9,12,13,14,15,17,18,19,21,23,25,28,38,65,66,67,68,69

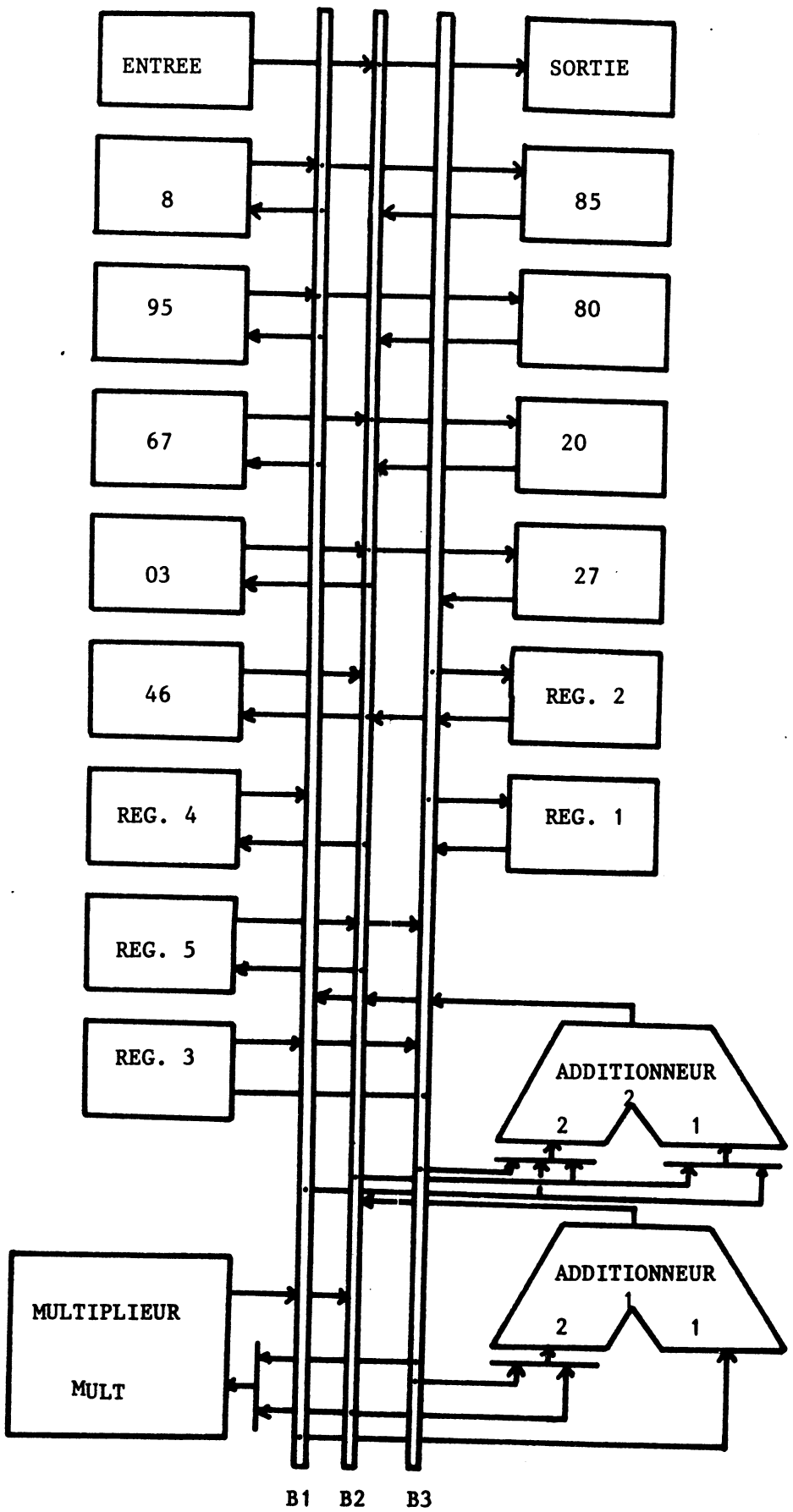
Etape numéro : 55 commencant au : 1.48000E+02 ième top
et ayant 21 commandes actives :

1,3,8,9,12,13,14,15,16,17,18,19,21,23,25,38,65,66,67,68,69

Etape numéro : 56 commencant au : 1.49000E+02 ième top
et ayant 22 commandes actives :

0,1,2,3,8,9,12,13,14,15,16,17,18,19,21,23,25,65,66,67,68,69

SHEMA BLOC DE L'ARCHITECTURE DU FILTRE " BIQUAD 2 (bis)"



PARTIE OPERATIVE DU CIRCUIT "BIQUAD 2(v2)"

```
.....
Opérateur numéro : 1 nommé ENTREE_F d'espèce ENTREE

Commande propre numéro : 1
Commande de RAZ : 70

Opérateur ayant 0 antécédents :
et 1 successeurs :
- successeur numéro : 17
.....
Opérateur numéro : 2 nommé SORTIE_F d'espèce SORTIE

Commande propre numéro : 2
Commande de RAZ : 70

Opérateur ayant 1 antécédents :
- antécédent numero : 17
et 0 successeurs :
.....
Opérateur numéro : 3 nommé MULT d'espèce MULTIPLI

l'entrée : 1 est latchée avec la commande numéro : 3

Commande de RAZ : 70

Ce multiplieur général fait 8 multiplications
la multiplication numéro 13 avec la commande numéro : 4
la multiplication numéro 21 avec la commande numéro : 5
la multiplication numéro 29 avec la commande numéro : 6
la multiplication numéro 30 avec la commande numéro : 7
la multiplication numéro 15 avec la commande numéro : 8
la multiplication numéro 17 avec la commande numéro : 9
la multiplication numéro 27 avec la commande numéro : 10
la multiplication numéro 7 avec la commande numéro : 11

Opérateur ayant : 2 antécédents sur l'entrée : 1
- antécédent numéro : 18
- antécédent numéro : 17
et : 2 successeurs
- successeur numéro : 17
- successeur numéro : 16
.....
Opérateur numéro : 4 nommé ADD_1 d'espèce ADDITION

l'entrée : 1 est latchée avec la commande numéro : 12
l'entrée : 2 est latchée avec la commande numéro : 13

Commande de RAZ : 70


Opérateur ayant : 1 antécédents sur l'entrée : 1
- antécédent numéro : 16
Opérateur ayant : 2 antécédents sur l'entrée : 2
- antécédent numéro : 17
- antécédent numéro : 18
et : 1 successeurs
- successeur numéro : 17
.....
Opérateur numéro : 5 nommé ADD_2 d'espèce ADDITION

l'entrée : 1 est latchée avec la commande numéro : 14
l'entrée : 2 est latchée avec la commande numéro : 15

Commande de RAZ : 70


Opérateur ayant : 2 antécédents sur l'entrée : 1
- antécédent numéro : 17
- antécédent numéro : 16
Opérateur ayant : 3 antécédents sur l'entrée : 2
- antécédent numéro : 18
- antécédent numéro : 16
- antécédent numéro : 17
et : 3 successeurs
- successeur numéro : 16
- successeur numéro : 17
- successeur numéro : 18
.....
Opérateur numéro : 7 nommé REG1 d'espèce RETARD Z

Commandes propres numéro : 16,17
Commande de RAZ : 70

Opérateur ayant 1 antécédents :
- antécédent numero : 18
et 1 successeurs :
- successeur numéro : 18
.....
Opérateur numéro : 9 nommé REG2 d'espèce RETARD Z

Commandes propres numéro : 18,19
Commande de RAZ : 70

Opérateur ayant 1 antécédents :
- antécédent numero : 18
et 2 successeurs :
- successeur numéro : 17
- successeur numéro : 18
.....
Opérateur numéro : 11 nommé REG3 d'espèce RETARD Z

Commandes propres numéro : 20,21
Commande de RAZ : 70

Opérateur ayant 1 antécédents :
- antécédent numero : 18
et 2 successeurs :
- successeur numéro : 16
- successeur numéro : 18
.....
Opérateur numéro : 13 nommé REG4 d'espèce RETARD Z

Commandes propres numéro : 22,23
Commande de RAZ : 70

Opérateur ayant 1 antécédents :
- antécédent numero : 17
et 1 successeurs :
- successeur numéro : 16
.....
Opérateur numéro : 15 nommé REG5 d'espèce RETARD Z

Commandes propres numéro : 24,25
Commande de RAZ : 70

Opérateur ayant 1 antécédents :
- antécédent numero : 17
et 2 successeurs :
- successeur numéro : 17
- successeur numéro : 18
.....
Opérateur numéro : 16 nommé BUS1 d'espèce BUS

Bus ayant 6 antécedents :
Tache numéro : 5 avec la commande numéro : 26
Tache numéro : 10 avec la commande numéro : 27
Tache numéro : 3 avec la commande numéro : 28
Tache numéro : 12 avec la commande numéro : 29
Tache numéro : 22 avec la commande numéro : 30
Tache numéro : 24 avec la commande numéro : 31
et 8 successeurs :
Tache numéro : 4 atteinte sur l'entrée : 1 pas de commande
Tache numéro : 21 atteinte sur l'entrée : 1 pas de commande
Tache numéro : 5 atteinte sur l'entrée : 2 commande numéro : 32
Tache numéro : 22 atteinte sur l'entrée : 1 pas de commande
Tache numéro : 24 atteinte sur l'entrée : 1 pas de commande
Tache numéro : 25 atteinte sur l'entrée : 1 pas de commande
Tache numéro : 5 atteinte sur l'entrée : 1 commande numéro : 33
Tache numéro : 26 atteinte sur l'entrée : 1 pas de commande
.....
Opérateur numéro : 17 nommé BUS2 d'espèce BUS

Bus ayant 12 antécedents :
Tache numéro : 3 avec la commande numéro : 34
Tache numéro : 1 avec la commande numéro : 35
Tache numéro : 14 avec la commande numéro : 36
Tache numéro : 8 avec la commande numéro : 37
Tache numéro : 4 avec la commande numéro : 38
Tache numéro : 19 avec la commande numéro : 39
Tache numéro : 21 avec la commande numéro : 40
Tache numéro : 23 avec la commande numéro : 41
Tache numéro : 5 avec la commande numéro : 42
Tache numéro : 25 avec la commande numéro : 43
Tache numéro : 26 avec la commande numéro : 44
Tache numéro : 27 avec la commande numéro : 45
et 11 successeurs :
Tache numéro : 5 atteinte sur l'entrée : 1 commande numéro : 46
Tache numéro : 4 atteinte sur l'entrée : 2 commande numéro : 47
Tache numéro : 19 atteinte sur l'entrée : 1 pas de commande
Tache numéro : 20 atteinte sur l'entrée : 1 pas de commande
Tache numéro : 23 atteinte sur l'entrée : 1 pas de commande
Tache numéro : 3 atteinte sur l'entrée : 1 commande numéro : 48
Tache numéro : 27 atteinte sur l'entrée : 1 pas de commande
Tache numéro : 5 atteinte sur l'entrée : 2 commande numéro : 49
Tache numéro : 15 atteinte sur l'entrée : 1 pas de commande
Tache numéro : 13 atteinte sur l'entrée : 1 pas de commande
Tache numéro : 2 atteinte sur l'entrée : 1 pas de commande
```

```
.....
Opérateur numéro :      18 nommé BUS3     d'espèce BUS

Bus ayant  7 antécedents :
Tache numéro : 14 avec la commande numéro : 50
Tache numéro : 10 avec la commande numéro : 51
Tache numéro :  8 avec la commande numéro : 52
Tache numéro :  6 avec la commande numéro : 53
Tache numéro : 20 avec la commande numéro : 54
Tache numéro :  5 avec la commande numéro : 55
Tache numéro :  9 avec la commande numéro : 56
     et        6 successeurs :
Tache numéro :  3 atteinte sur l'entrée :  1 commande numéro : 57
Tache numéro :  5 atteinte sur l'entrée :  2 commande numéro : 58
Tache numéro :  9 atteinte sur l'entrée :  1 pas de commande
Tache numéro :  4 atteinte sur l'entrée :  2 commande numéro : 59
Tache numéro : 11 atteinte sur l'entrée :  1 pas de commande
Tache numéro :  7 atteinte sur l'entrée :  1 pas de commande
.....
Opérateur numéro :      19 nommé 03011903 d'espèce REGISTRE

Commande propre numéro : 60
Commande de RAZ       :      70

Opérateur ayant  1 antécédents :
- antécédent numero :      17
         et          1 successeurs :
- successeur numéro :      17
.....
Opérateur numéro :      20 nommé 03074627 d'espèce REGISTRE

Commande propre numéro : 61
Commande de RAZ       :      70

Opérateur ayant  1 antécédents :
- antécédent numero :      17
         et          1 successeurs :
- successeur numéro :      18
.....
Opérateur numéro :      21 nommé 03190520 d'espèce REGISTRE

Commande propre numéro : 62
Commande de RAZ       :      70

Opérateur ayant  1 antécédents :
- antécédent numero :      16
         et          1 successeurs :
- successeur numéro :      17
.....
Opérateur numéro :      22 nommé 03195395 d'espèce REGISTRE

Commande propre numéro : 63
Commande de RAZ       :      70

Opérateur ayant  1 antécédents :
- antécédent numero :      16
         et          1 successeurs :
- successeur numéro :      16
.....
Opérateur numéro :      23 nommé 03203446 d'espèce REGISTRE

Commande propre numéro : 64
Commande de RAZ       :      70

Opérateur ayant  1 antécédents :
- antécédent numero :      17
         et          1 successeurs :
- successeur numéro :      17
.....
Opérateur numéro :      24 nommé 03321208 d'espèce REGISTRE

Commande propre numéro : 65
Commande de RAZ       :      70

Opérateur ayant  1 antécédents :
- antécédent numero :      16
         et          1 successeurs :
- successeur numéro :      16
.....
Opérateur numéro :      25 nommé 03321680 d'espèce REGISTRE

Commande propre numéro : 66
Commande de RAZ       :      70

Opérateur ayant  1 antécédents :
- antécédent numero :      16
         et          1 successeurs :
- successeur numéro :      17
.....
Opérateur numéro :      26 nommé 03393485 d'espèce REGISTRE

Commande propre numéro : 67
Commande de RAZ       :      70

Opérateur ayant  1 antécédents :
- antécédent numero :      16
         et          1 successeurs :
- successeur numéro :      17
.....
Opérateur numéro :      27 nommé 03393967 d'espèce REGISTRE

Commande propre numéro : 68
Commande de RAZ       :      70

Opérateur ayant  1 antécédents :
- antécédent numero :      17
         et          1 successeurs :
- successeur numéro :      17
```

PARTIE CONTROLE DU CIRCUIT "BIQUAD 2(v2)"

Etape numéro : 1 commencant au : 0.00000E+00 ième top
et ayant 9 commandes actives :

4,21,23,25,26,34,46,50,57

Etape numéro : 2 commencant au : 1.00000E+00 ième top
et ayant 12 commandes actives :

1,4,16,18,20,22,24,26,34,46,50,57

Etape numéro : 3 commencant au : 3.00000E+00 ième top
et ayant 13 commandes actives :

1,3,4,16,18,20,22,24,26,34,46,51,56

Etape numéro : 4 commencant au : 5.00000E+00 ième top
et ayant 14 commandes actives :

1,3,4,15,16,18,20,22,24,26,34,46,52,57

Etape numéro : 5 commencant au : 1.40000E+01 ième top
et ayant 14 commandes actives :

1,5,14,15,16,18,20,22,24,26,35,47,52,57

Etape numéro : 6 commencant au : 1.50000E+01 ième top
et ayant 15 commandes actives :

1,3,5,14,15,16,18,20,22,24,26,35,47,53,57

Etape numéro : 7 commencant au : 1.60000E+01 ième top
et ayant 15 commandes actives :

1,3,5,13,14,15,16,18,20,22,24,26,36,53,57

Etape numéro : 8 commencant au : 1.80000E+01 ième top
et ayant 16 commandes actives :

1,3,5,13,14,15,16,18,20,22,24,26,36,53,57,60

Etape numéro : 9 commencant au : 1.90000E+01 ième top
et ayant 16 commandes actives :

1,3,5,13,14,15,16,18,20,22,24,26,37,53,57,60

Etape numéro : 10 commencant au : 2.00000E+01 ième top
et ayant 15 commandes actives :

1,3,5,12,13,16,18,20,22,24,27,37,53,57,60

Etape numéro : 11 commencant au : 2.10000E+01 ième top
et ayant 16 commandes actives :

1,3,5,12,13,16,18,20,22,24,27,37,53,57,60,61

Etape numéro : 12 commencant au : 2.20000E+01 ième top
et ayant 17 commandes actives :

1,3,5,12,13,16,18,20,22,24,27,38,53,57,60,61,62

Etape numéro : 13 commencant au : 2.40000E+01 ième top
et ayant 17 commandes actives :

1,3,5,12,13,16,18,20,22,24,38,53,57,60,61,62,64

Etape numéro : 14 commencant au : 2.50000E+01 ième top
et ayant 18 commandes actives :

1,3,5,12,13,16,18,20,22,24,34,46,53,57,60,61,62,64

Etape numéro : 15 commencant au : 2.90000E+01 ième top
et ayant 19 commandes actives :

1,6,12,13,15,16,18,20,22,24,28,32,39,53,57,60,61,62,64

Etape numéro : 16 commencant au : 3.00000E+01 ième top
et ayant 20 commandes actives :

1,3,6,12,13,15,16,18,20,22,24,28,32,39,53,57,60,61,62,64

Etape numéro : 17 commencant au : 4.10000E+01 ième top
et ayant 20 commandes actives :

1,3,7,12,13,14,15,16,18,20,22,24,26,39,53,57,60,61,62,64

Etape numéro : 18 commencant au : 4.20000E+01 ième top
et ayant 20 commandes actives :

1,3,7,12,13,14,15,16,18,20,22,24,26,39,48,54,60,61,62,64

Etape numéro : 19 commencant au : 4.50000E+01 ième top
et ayant 21 commandes actives :

1,3,7,12,13,14,15,16,18,20,22,24,26,39,48,54,60,61,62,63,64

Etape numéro : 20 commencant au : 4.60000E+01 ième top
et ayant 19 commandes actives :

1,3,7,12,13,16,18,20,22,24,29,39,48,54,60,61,62,63,64

Etape numéro : 21 commencant au : 4.80000E+01 ième top
et ayant 20 commandes actives :

1,3,7,12,13,16,18,20,22,24,29,39,48,54,60,61,62,63,64,65

Etape numéro : 22 commencant au : 5.90000E+01 ième top
et ayant 20 commandes actives :

1,3,7,12,13,16,18,20,22,24,28,39,48,54,60,61,62,63,64,65

Etape numéro : 23 commencant au : 5.50000E+01 ième top
et ayant 21 commandes actives :

1,3,7,12,13,16,18,20,22,24,28,39,48,54,60,61,62,63,64,65,66

Etape numéro : 24 commencant au : 5.60000E+01 ième top
et ayant 21 commandes actives :

1,8,12,13,16,18,20,22,24,28,32,39,48,54,60,61,62,63,64,65,66

Etape numéro : 25 commencant au : 5.70000E+01 ième top
et ayant 22 commandes actives :

1,3,8,12,13,16,18,20,22,24,28,32,40,48,54,60,61,62,63,64,65,66

Etape numéro : 26 commencant au : 6.80000E+01 ième top
et ayant 22 commandes actives :

1,9,12,13,15,16,18,20,22,24,28,33,40,48,54,60,61,62,63,64,65,66

Etape numéro : 27 commencant au : 6.90000E+01 ième top
et ayant 23 commandes actives :

1,3,9,12,13,15,16,18,20,22,24,28,33,41,54,57,60,61,62,63,64,65,66

Etape numéro : 28 commencant au : 8.00000E+01 ième top
et ayant 22 commandes actives :

1,10,12,13,14,15,16,18,20,22,24,28,41,54,57,60,61,62,63,64,65,66

Etape numéro : 29 commencant au : 8.10000E+01 ième top
et ayant 22 commandes actives :

1,3,10,12,13,14,15,16,18,20,22,24,28,41,48,60,61,62,63,64,65,66

Etape numéro : 30 commencant au : 9.20000E+01 ième top
et ayant 23 commandes actives :

1,3,10,12,13,14,15,16,18,20,22,24,28,41,48,60,61,62,63,64,65,66,67

Etape numéro : 31 commencant au : 9.30000E+01 ième top
et ayant 23 commandes actives :

1,11,12,13,14,15,16,18,20,22,24,30,33,41,48,60,61,62,63,64,65,66,67

Etape numéro : 32 commencant au : 9.40000E+01 ième top
et ayant 23 commandes actives :

1,3,11,12,13,14,15,16,18,20,22,24,30,33,42,60,61,62,63,64,65,66,67

Etape numéro : 33 commencant au : 9.60000E+01 ième top
et ayant 24 commandes actives :

1,3,11,12,13,14,15,16,18,20,22,24,30,33,42,60,61,62,63,64,65,66,67,68

Etape numéro : 34 commencant au : 9.70000E+01 ième top
et ayant 23 commandes actives :

1,3,11,12,13,16,18,20,22,24,30,33,43,55,60,61,62,63,64,65,66,67,68

Etape numéro : 35 commencant au : 9.80000E+01 ième top
et ayant 25 commandes actives :

1,3,11,12,13,14,16,18,20,22,24,31,32,43,46,55,60,61,62,63,64,65,66,67,68

Etape numéro : 36 commencant au : 1.00000E+02 ième top
et ayant 24 commandes actives :

1,3,11,12,13,14,15,16,18,20,22,24,26,43,46,55,60,61,62,63,64,66,67,68

Etape numéro : 37 commencant au : 1.04000E+02 ième top
et ayant 24 commandes actives :

1,3,11,12,13,16,18,19,20,22,24,26,43,46,56,59,60,61,62,63,64,66,67,68

Etape numéro : 38 commencant au : 1.05000E+02 ième top
et ayant 25 commandes actives :

1,3,11,12,13,14,16,18,19,20,22,24,26,44,49,56,59,60,61,62,63,64,66,67,68

Etape numéro : 39 commencant au : 1.07000E+02 ième top
et ayant 25 commandes actives :

1,3,11,12,13,14,15,16,18,19,20,22,24,26,45,46,56,59,60,61,62,63,64,66,68

Etape numéro : 40 commencant au : 1.08000E+02 ième top
et ayant 26 commandes actives :

1,3,11,12,13,14,15,16,18,19,20,22,24,26,45,46,56,59,60,61,62,63,64,66,67,68

Etape numéro : 41 commencant au : 1.09000E+02 ième top
et ayant 26 commandes actives :

1,3,11,12,13,14,15,16,18,19,20,22,24,26,45,46,56,59,60,61,62,63,64,66,67,68

Etape numéro : 42 commencant au : 1.11000E+02 ième top
et ayant 27 commandes actives :

1,3,11,12,13,14,15,16,18,19,20,22,24,26,45,46,56,59,60,61,62,63,64,65,66,67,68

Etape numéro : 43 commencant au : 1.12000E+02 ième top
et ayant 25 commandes actives :

1,3,11,12,13,16,18,19,20,22,24,31,45,46,56,59,60,61,62,63,64,65,66,67,68

Etape numéro : 44 commencant au : 1.13000E+02 ième top
et ayant 26 commandes actives :

1,3,11,12,13,14,16,18,19,20,22,24,31,44,49,56,59,60,61,62,63,64,65,66,67,68

Etape numéro : 45 commencant au : 1.15000E+02 ième top
et ayant 26 commandes actives :

1,3,11,12,13,14,15,16,18,19,20,22,24,31,36,56,59,60,61,62,63,64,65,66,67,68

Etape numéro : 46 commencant au : 1.17000E+02 ième top
et ayant 24 commandes actives :

1,3,11,14,15,16,18,19,20,22,24,31,42,56,59,60,61,62,63,64,65,66,67,68

Etape numéro : 47 commencant au : 1.18000E+02 ième top
et ayant 24 commandes actives :

1,3,11,12,13,14,15,16,18,19,20,22,24,42,50,60,61,62,63,64,65,66,67,68

Etape numéro : 48 commencant au : 1.20000E+02 ième top
et ayant 24 commandes actives :

1,3,11,12,13,14,15,16,18,19,20,22,24,36,52,60,61,62,63,64,65,66,67,68

Etape numéro : 49 commencant au : 1.22000E+02 ième top
et ayant 25 commandes actives :

0,1,2,3,11,12,13,14,15,16,17,18,19,20,22,24,60,61,62,63,64,65,66,67,68

INPG **département des études doctorales**

AUTORISATION de SOUTENANCE

VU les dispositions de l'article 15 Titre III de l'arrêté du 5 juillet 1984 relatif aux études doctorales

VU les rapports de présentation de Messieurs

. J. GASTINEL, Chargé de recherche
. R. GERBER, Professeur

Monsieur REYSS-BRION Jean-Frédéric

est autorisé à présenter une thèse en soutenance en vue de l'obtention du diplôme de DOCTEUR de l'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE, spécialité "Informatique".

Fait à Grenoble, le 09 mai 1985

Le Président de l'I.N.P.-G

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

P.O. le Vice-Président,