Etude et réalisation d'un calculateur spécialisé pour la synthèse sonore en temps réel par simulation de mécanismes instrumentaux

Talin Dars-Berberyan

THESE

PRESENTEE A

L'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

pour obtenir

LE TITRE DE DOCTEUR INGENIEUR

par

MADAME TALIN DARS - BERBERYAN

*

ÉTUDE ET RÉALISATION D'UN CALCULATEUR SPÉCIALISÉ
POUR LA SYNTHÈSE SONORE EN TEMPS RÉEL
PAR SIMULATION DE MÉCANISMES INSTRUMENTAUX

*

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Année universitaire 1981 - 1982

Président : Daniel BLOCH ; Vice-Présidents : René CARRÉ
Hervé CHÉRADAME
Marcel IVANÈS

PROFESSEURS DES UNIVERSITES		ENSEEG	ENSERG	ENSIEG	ENSHG	ENSIMAG	MCPP
ANCEAU	F.					●	
BARRAUD	A.			●			
BESSON	J.	●					
BLIMAN	S.		●				
BLOCH	D.			●			
BOIS	P.				●		
BONNETAIN	L.	●					
BONNIER	E.	●					
BOUVARD	M.				●		
BRISSONNEAU	P.			●			
BUYLE-BODIN	M.		●				
CAVAIGNAC	J.F.			●			
CHARTIER	G.			●			
CHENEVIER	P.		●				
CHERADAME	H.						●
CHERUY	Arlette			●			
CHIAVERINA	J.						●
COHEN	J.		●				
COUMES	A.		●				
DURAND	F.	●					
DURAND	J.L.			●			
FELICI	N.			●			
FOULARD	C.			●			
GENTIL	P.		●				
GUERIN	B.		●				
GUYOT	P.	●					
IVANES	M.			●			
JAUSSAUD	P.			●			
JOUBERT	J.C.			●			
JOURDAIN	Geneviève			●			
LACOUME	J.L.			●			
LATOMBE	J.C.					●	
LEROY	P.				●		
LESIEUR	M.				●		
LESPINARD	G.						
LONGEQUEUE	J.P.						
MAZARÉ	G.						
MOREAU	R.						
MORET	R.						
MOSSIERE	J.						
PARIAUD	J.C.	●					
PAUTHENET	R.						
PERRET	René						
PERRET	Robert						
PIAU	J.M.						
POLOUJADOFF	M.						
POUPOT	C.						
RAMEAU	J.J.	●					
RENAUD	M.						
ROBERT	A.						
ROBERT	F.						
SABONNADIERE	J.C.						
SAUCIER	Gabrielle						
SCHLENKER	Claire						
SCHLENKER	M.						
SERMET	P.						
SOUQUET	J.L.	●					
SILVY	J.						
SOHM	J.C.	●					
VEILLON	G.						
ZADWORNY	F.						
PROFESSEURS ASSOCIES							
GANDINI	A.						
MAXWORTHY	T.						
MROVEC	S.	●					
PARRIAUX	O.						
PEISNER	J.						

PROFESSEURS E.N.S. MINES SAINT-ETIENNE : RIEU J.
SOUSTELLE M.

CHERCHEURS DU C.N.R.S.

(Directeur et Maîtres de Recherche) :	FRUCHART	R.	Directeur de Recherche
	ALLIBERT	M.	Maître de Recherche
	ANSARA	I.	Maître de Recherche
	CARRÉ	R.	Maître de Recherche
	DAVID	R.	Maître de Recherche
	DRIOLE	J.	Maître de Recherche
	KAMARINOS	G.	Maître de Recherche
	KLEITZ	M.	Maître de Recherche
	LANDAU	I.D.	Maître de Recherche
	MERMET	J.	Maître de Recherche
	MUNIER	J.	Maître de Recherche
	VERDILLON	A.	Maître de Recherche

CHERCHEURS DU MINISTERE DE L'INDUSTRIE

(Directeur et Maîtres de Recherche - E.N.S. Mines St Etienne) :	LESBATS	P.	Directeur de Recherche
	BISCONDI	M.	Maître de Recherche
	KOBYLANSKI	A.	Maître de Recherche
	LE COZE	J.	Maître de Recherche
	THEVENOT	F.	Maître de Recherche
	TRAN MINH	C.	Maître de Recherche
	LALAUZE	R.	Maître de Recherche
	LANCELOT	F.	Maître de Recherche

PERSONNALITES HABILITEES A DIRIGER DES TRAVAUX DE RECHERCHE

(Décision du Conseil scientifique) :

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

BERNARD	C.	
BONNET	R.	
CAILLET	M.	
CHATILLON	Catherine	
COULON	M.	
EUSTATHOPOULOS	N.	
HAMMOU	A.	
JOUD	J.C.	
MALMEJAC	Y.	(C.E.N.G.)
RAVAINE	D.	
SAINFORT		(C.E.N.G.)
SARRAZIN	P.	
TOUZAIN	P.	
URBAIN	G.	(Laboratoire des Ultra-réfractaires , ODEILLO)

E.N.S.M. St Etienne

GUILHOT	B.
THOMAS	G.
DRIVER	J.

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

BOREL	J.
CHEHIKIAN	A.

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

BORNARD	G.
DESCHIZEAUX	P.
GLANGEAUD	F.
LEJEUNE	G.
PERARD	J.

E.N.S.H.G.

DELHAYE	J.M.

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

COURTIN	J.

Le travail présenté dans ce mémoire a été effectué à l'Ecole Nationale Supérieure d'ELECTRONIQUE et de RADIOELECTRICITE de Grenoble. Je remercie vivement Monsieur le Professeur M. BUYLE-BODIN, Directeur de l'ENSERG, pour son accueil bienveillant.

Je remercie Monsieur R. CARRÉ, Directeur de Recherche du C.N.R.S., Directeur du Laboratoire de la Communication Parlée et de l'Instrumentation de Mesure, d'avoir bien voulu m'accepter au sein de ce laboratoire et d'avoir permis l'accomplissement de ce travail.

Je remercie le Ministère de la Culture et la Direction de la Musique pour le soutien moral, matériel et financier qu'ils ont accordé à notre Equipe.

Je remercie Monsieur Cl. CADOZ, Directeur de l'ACROE, et mes Collègues J.-L. FLORENS, A. LUCIANI, P. LACORNERIE, qui ont largement contribué à l'aboutissement de ces travaux.

Je remercie Mademoiselle N. BIOUD, Secrétaire du Laboratoire, pour ses conseils utiles à la présentation de ce document et son extrême gentillesse.

Je remercie enfin toutes les personnes des Services généraux de l'ENSERG qui m'ont aidée, à un moment ou à un autre, pour l'accomplissement des différentes démarches que j'ai effectuées.

☆ ☆
★

INTRODUCTION

MUSIQUE ET INFORMATIQUE

CRITIQUE DE LA SYNTHESE ACOUSTIQUE COMME MOYEN DE CREATION MUSICALE
PRIMAUTE DU RAPPORT INSTRUMENTAL
L'INSTRUMENT : OBJET D'UNE PERCEPTION GLOBALE, MULTISENSORIELLE

PARTICULARITE DU CANAL GESTUEL
TRANSDUCTEURS GESTUELS

Première partie : LE SYSTEME DE SIMULATION CORDIS

A/ PRINCIPES GENERAUX DU **SYSTEME CORDIS**

1. Généralités
2. Analyse mécanique des instruments
   - 2.1. La séparation en excitateur - structure vibrante
   - 2.2. Etude de la structure vibrante
     - 2.2.1. Modèle de la ligne vibrante
     - 2.2.2. Modèles de la surface vibrante et du volume vibrant
     - 2.2.3. La cellule : Module vibratoire élémentaire
   - 2.3. Etude de l'excitateur et de sa liaison avec la structure vibrante
3. Algorithmes du système CORDIS
   - 3.1. Algorithme de la masse
   - 3.2. Algorithmes du ressort et du frottement
   - 3.3. Algorithme de la cellule
   - 3.4. Algorithmes de la liaison
4. Le langage CORDIS
   - 4.1. Fonctions et modes opératoires généraux du système CORDIS
   - 4.2. Le langage CORDIS

B/ DETERMINATION DES FONCTIONS DU PROCESSEUR SPECIALISE **CORDIS TEMPS REEL**

Deuxième partie : LE CALCULATEUR SPECIALISE CORDIS TEMPS REEL (CTR)

A/ PRELIMINAIRES

1. Généralités
2. Liste des instructions. Algorithmes
   - 2.1. Les instructions correspondant à des modules internes
     - 2.1.1. Modules matériels
     - 2.1.2. Modules de liaison
   - 2.2. Les instructions correspondant à des modules d'entrée-sortie lente
     - 2.2.1. Modules correspondant à un geste d'excitation
     - 2.2.2. Modules correspondant à un geste de modification

INTRODUCTION

MUSIQUE ET INFORMATIQUE

Les appareils électroniques ont été utilisés pour la première fois dans un but de création musicale, dans la démarche appelée LA MUSIQUE CONCRETE fondée par Pierre SCHAEFFER en 1948. Cette musique "a été appelée concrète parce qu'elle est constituée à partir d'éléments préexistants empruntés à n'importe quel matériau sonore, qu'il soit bruit ou son musical, puis composée expérimentalement par une construction directe, aboutissant à réaliser une volonté de composition, sans le secours devenu impossible d'une notation musicale ordinaire." (P. SCHAEFFER, La musique concrète, Que sais-je, P.U.F., 1973). Le matériau sonore réel est enregistré sur bande magnétique et traité dans un but de création. Les appareils qui interviennent le plus couramment sont les magnétophones, filtres, tables de mixage.... Partant du constat de l'insuffisance de la notation traditionnelle et du bouleversement des règles du solfège, les efforts de SCHAEFFER tendent à qualifier le matériau sonore selon de nouveaux critères, à définir un nouveau solfège à travers des expérimentations concrètes avec le GRM (Groupe de Recherche Musicale) et à mener une analyse et une réflexion approfondies sur les nouvelles données de l'univers musical (P. SCHAEFFER 1966, Le Traité des Objets Musicaux, Seuil, Paris).

Parallèlement à cette démarche se développa LA MUSIQUE ELECTRONIQUE (1950 HEIMERT à Cologne) avec l'utilisation des synthétiseurs analogiques. Le son n'existe pas en amont du haut-parleur. Au départ il existe un signal électrique auquel peuvent s'appliquer tous les traitements possibles sur un signal de ce type : addition de plusieurs signaux, addition de bruit, filtrage d'un signal, modulation d'amplitude, modulation de fréquence etc.. On se trouve en présence d'une catégorie de sons dits sons électroniques par opposition aux sons issus d'objets concrets. Ainsi pour la première fois le son peut être fabriqué directement selon ses caractéristiques acoustiques (amplitude, fréquence, spectre etc.), sans référence à une cause instrumentale. Le son ainsi généré est la plupart du temps enregistré et traité comme en musique concrète. MUSIQUE ELECTRONIQUE et MUSIQUE CONCRETE sont réunies dans le terme de MUSIQUE ELECTROACOUSTIQUE.

L'introduction de l'informatique dans le domaine musical se fait pour la première fois en 1956 avec l'expérience de COMPOSITION AUTOMATIQUE de HILLER et ISAACSON à l'Université d'Illinois aux Etats-Unis. Cette démarche qui considère l'ordinateur comme un outil de composition est intéressante en ce

qu'elle suppose une analyse préalable des règles de composition et une formulation rigoureuse de ces règles. En cela elle permet de mieux comprendre les mécanismes de la composition. Les attitudes extrêmes dans ce domaine vont de l'utilisation de fonctions aléatoires ayant une formulation purement mathématique (XENAKIS), jusqu'à l'utilisation de règles très particulières. L'oeuvre réalisée est souvent exécutée par des instruments traditionnels (c'est le cas de plusieurs oeuvres de XENAKIS). Elle peut être aussi exécutée directement par un procédé de synthèse sonore connu, comme la synthèse additive.

La synthèse numérique prend progressivement le relais de la synthèse analogique avec les travaux de MATHEWS en 1960. Le premier système cohérent, qui permet de contrôler de manière globale et efficace la microstructure du son est le programme MUSIC V. MUSIC V est initialement un procédé de synthèse additive. Il suppose une analyse préalable du son selon ses caractéristiques acoustiques. Un son complexe est obtenu par combinaison de plusieurs signaux périodiques élémentaires, issus de modules appelés **oscillateurs,** ayant chacun une forme d'onde définie, dont les paramètres (fréquence, amplitude) peuvent être contrôlés par une forme. Celle-ci est soit un signal périodique issu d'un autre oscillateur, soit un signal non périodique, issu de modules appelés **générateurs d'enveloppe,** pour lequel on détermine une amplitude et trois durées correspondant respectivement à un temps d'établissement, de maintien, et d'extinction du signal sonore. Le formalisme et la combinatoire propre à MUSIC V permettent de dépasser le cadre des utilisations prévues au départ, comme c'est le cas avec le procédé de modulation de fréquence.

La modulation de fréquence (J. CHOWNING, 1973, The synthesis of complex audio spectra by means of frequency modulation, JAES *21*, N° 7) est un procédé de synthèse globale, obtenu en contrôlant la fréquence d'un oscillateur de MUSIC V par un autre module oscillateur ou générateur de signaux aléatoires. C'est un moyen de contrôle simple et économique du point de vue du nombre de paramètres à préciser, qui permet d'obtenir un événement sonore soumis à des lois de variations en fréquence. Pour certains instruments, en particulier les cuivres, ces lois de variations sont étroitement corrélées avec le timbre de l'instrument.

Dans MUSIC V, la détermination des événements sonores se fait en deux phases : la définition d'une **structure - instrument** et la définition de notes jouées sur cet instrument. Pour une même structure, les sons obtenus ont en général certaines caractéristiques perceptives reconnaissables, liées à cette structure. Dans une situation simple, nous aurons par exemple pour un instrument donné, des sons ayant un même timbre lié à la structure, (à condition d'inclure dans la structure la définition de formes d'onde), mais des hauteurs, des intensités, des durées différentes. Ce cas est assez proche de la définition traditionnelle de l'instrument et de la note. Dans d'autres cas, cette séparation ne correspond pas aussi distinctement à des critères perceptifs. Nous pourrons par exemple entendre comme une seule note, un son calculé selon plusieurs notes simultanées dans la partition de MUSIC V. Inversement par le procédé de modulation de fréquence, nous pourrons entendre une suite de notes différentes correspondant à une seule note dans la partition de MUSIC V. Enfin MUSIC V apparaît comme un système de composition qui s'étend jusqu'au niveau de la microstructure même du son.

Dans les vingt années qui ont suivi son apparition, le programme MUSIC V a été modifié et implanté sur des ordinateurs très différents à travers le monde. Le langage MUSIC V est initialement un langage proche de l'in-

formatique. Les paramètres sont abstraits et ne font pas directement référence à des critères perceptifs. Par la suite, des langages plus accessibles aux musiciens ont été développés.

Trente ans après la première utilisation de l'ordinateur à des fins musicales, les travaux dans ce domaine se sont diversifiés. Nous allons tenter de présenter brièvement les tendances principales :

a/ Les travaux sur le traitement numérique de matériaux sonores préexistants, dans un but de création d'oeuvres musicales (Groupe de recherche musicale - I.N.A., Paris).

b/ Les travaux sur l'ordinateur considéré comme un outil de composition. Nous pouvons citer dans ce groupe les travaux sur les langages de composition par ordinateur. La composition automatique en est une branche bien particulière.

c/ Les travaux sur les systèmes temps réel. Il s'agit quelquefois de petits systèmes de synthèse sonore en temps réel. Ces systèmes construits à base de microprocesseurs, avec une puissance de calcul et une taille mémoire limitées, mettent en oeuvre des procédés de synthèse simples. L'ensemble est destiné souvent à l'usage de petits studios ou de particuliers, ou encore à l'usage de la pédagogie musicale. Le langage est souvent un langage intermédiaire entre un langage musical évolué et un langage machine. Il peut s'agir aussi de gros systèmes de synthèse sonore et de traitement comme le processeur 4X de l'IRCAM ou le processeur de LUCASFILM (Etats-Unis).

d/ Les travaux sur l'ordinateur utilisé en tant qu'organe de simulation de modèles physiques. Les simulations de cordes frottées, grattées ou percutées entrent dans cette catégorie, de même que les simulations du milieu de diffusion et de propagation sonore. Les travaux de l'ACROE en ce qui concerne la simulation de mécanismes instrumentaux peuvent aussi être classés dans ce groupe.

e/ Les travaux sur l'intelligence artificielle. Ils impliquent une analyse des mécanismes de l'intelligence humaine et de l'activité de création.

f/ Les travaux sur les codes musicaux et sur la partition musicale.

g/ Les travaux dans le domaine de la psycho-acoustique.

h/ Les travaux de recherche en musicologie et dans l'enseignement musical par ordinateur.

A travers la description de quelques systèmes, nous avons constaté jusque-là que des bouleversements importants se sont produits en trente ans dans les données de la création musicale. Les idées qui s'en dégagent sont les suivantes :

- La notation traditionnelle et la notion même de note de musique ne suffisent plus à rendre compte de l'événement sonore en tant qu'objet musical.
- Dans l'oeuvre musicale, les participations respectives du luthier, du compositeur ou de l'instrumentiste ne peuvent plus être distinguées facilement.
- Le son est considéré comme un objet d'étude en soi et non en fonction de sa causalité.

- Le dialogue et la collaboration entre musiciens et scientifiques sont nécessaires, mais la relation s'avère difficile à établir.

- Enfin le problème essentiel réside à notre avis dans la difficulté de corréler un effet musical désiré et les caractéristiques du signal à générer.

CRITIQUE DE LA SYNTHESE ACOUSTIQUE COMME MOYEN DE CREATION MUSICALE

PRIMAUTE DU RAPPORT INSTRUMENTAL

L'INSTRUMENT : OBJET D'UNE PERCEPTION GLOBALE, MULTISENSORIELLE

Nous désignons par le terme de **synthèse acoustique**, la démarche qui considère comme objet central le son en soi. Or, quand il s'agit d'événements sonores inouïs, tels que ceux obtenus par ordinateur, nous pensons que la référence à la causalité de l'événement sonore est une condition nécessaire à la constitution de sa charge signifiante. Dans un contexte nouveau, tel que celui dans lequel nous nous trouvons actuellement, pour utiliser l'ordinateur comme outil de création musicale, nous considérons qu'une première phase d'expérimentation de type instrumental est indispensable.

Dans ce qui suit nous utilisons le mot instrument dans un sens large, pour désigner l'objet élémentaire ou complexe qui produit le son. Nous pensons que le préalable nécessaire à toute création musicale est une expérimentation sensorielle de l'instrument. Une expérimentation est une somme d'expériences sensorielles primitives qui permettent d'avoir une connaissance de l'instrument. Le son qui résulte de l'expérience n'est qu'une des manifestations sensibles de l'instrument, mais elle n'est pas la seule. L'instrument, soumis à une activité physique gestuelle, se manifeste globalement à nos sens, parmi lesquels les plus concernés sont le toucher, la vue, l'ouïe. Dans la situation traditionnelle, quand il s'agit d'instruments mécaniques, le compositeur, qui semble être la personne la moins concernée par cette expérimentation, parmi les trois catégories - luthier, instrumentiste, compositeur - utilise un matériau sonore issu d'instruments réels. Sans être instrumentiste, il a néanmoins une connaissance instrumentale, par la simple observation d'instrumentistes en action, et plus fondamentalement il a une connaissance naturelle de l'univers concret des objets les plus divers, ayant un comportement sonore.

La démarche de l'ACROE consiste à utiliser l'ordinateur pour représenter non pas le son, mais l'instrument. Il s'en dégage deux orientations de recherche :

- La première est celle de la **simulation** de l'instrument. Nous signalons tout de suite que le but n'est pas tant de faire une simulation réaliste de l'instrument, c'est-à-dire de s'approcher le plus possible de ce qu'est l'instrument réellement, que de faire une simulation réduite mais suffisante de ses comportements sensibles. Le premier système de simulation CORDIS est fondé sur des approximations assez importantes, mais ce n'est qu'à l'issue de la première phase d'expérimentation, ultérieure à la constitution de notre outil de travail (système de simulation et transducteurs), que nous pourrons dégager les données pertinentes du point de vue perceptif et modifier notre modèle.

- La deuxième direction est celle de la création des conditions concrètes de l'expérimentation. En effet l'instrument n'existe pas réellement, mais il est représenté par un **modèle** dans l'ordinateur. Nous avons dit que l'instrument est perçu de manière globale par trois canaux perceptifs. Par la suite nous allons parler du canal gestuel, du canal visuel et du canal sonore. Le canal gestuel fait référence à une notion plus large que le simple toucher. Les informations perçues concernent les caractéristiques mécaniques statiques et dynamiques de l'objet. Par un premier recul analytique, nous pouvons dissocier le rôle de chaque canal, ce qui va nous conduire à réaliser trois types de coupleurs, dont le rôle est de permettre une communication entre l'ordinateur et chacun des canaux perceptifs concernés. Ces dispositifs connectés à l'ordinateur sont appelés **transducteurs.**

PARTICULARITE DU CANAL GESTUEL

TRANSDUCTEURS GESTUELS

Le canal gestuel a la particularité d'être à la fois émetteur et récepteur d'informations. Nous appelons **transducteurs gestuels rétroactifs** les dispositifs qui rendent compte de la bilatéralité du canal gestuel. Ce type de dispositifs n'a pas été étudié à notre connaissance pour des applications musicales. Dans les travaux de l'ACROE, la recherche sur les transducteurs gestuels rétroactifs prend une part très importante.

Il est matériellement impossible de construire un transducteur qui tienne compte du geste instrumental dans toute sa généralité : six degrés de liberté, une énergie importante concentrée dans un court laps de temps, des déplacements très réduits ou très vastes dans une direction d'action .... Nous nous sommes orientés donc vers la construction de dispositifs comportant des réductions en ce qui concerne la généralité du geste, mais essayant de tenir compte, dans une direction d'action privilégiée, des caractères signifiants du geste instrumental.

Il apparaît alors qu'une analyse préalable du geste instrumental est nécessaire pour la détermination des caractéristiques des transducteurs gestuels. Cette analyse a été faite selon plusieurs axes. Nous présentons ici brièvement quelques résultats.

Nous faisons la distinction entre le **geste d'excitation** et le **geste de modification de structure,** selon que l'énergie contenue dans le geste participe ou non à l'énergie sonore rayonnée. Nous distinguons aussi le geste à caractère corporel du geste à caractère digital, selon son amplitude. Nous analysons encore le geste selon ses directions d'action privilégiées : action frontale, action verticale, action latérale, ou selon les modes de manipulation de l'objet : avec saisie ou sans saisie, contact permanent ou momentané.

Dans le cas d'un geste de modification, la bilatéralité ne joue pas un rôle aussi important dans le contrôle du son que dans le cas d'un geste d'excitation. Le terme de modification recouvre deux notions différentes : la notion de **désignation** ou **sélection,** et la notion de **modulation.** La première intervient surtout quand l'instrument est un poly-instrument, c'est à dire formé de plusieurs instruments élémentaires appelés mono-instruments. Comme exemple d'un geste de sélection ou désignation, nous pouvons donner celui qui

consiste à choisir une touche sur un clavier de piano, une corde sur un instrument à cordes, une cloche dans une série de cloches etc.. Dans ce cas, la touche (à laquelle est associée une corde), la corde ou la cloche, sont considérées comme des mono-instruments. Dans le cas d'une touche ou d'une cloche, à partir du moment où elle est choisie, nous ne pouvons plus lui appliquer qu'un geste d'excitation. Dans le cas de la corde cependant, nous avons la possibilité de modifier sa tension ou sa longueur. Nous appelons ce type de modification de structure, une modulation. Il apparaît alors, que dans le cas d'un geste de sélection, la bilatéralité du canal gestuel ne joue aucun rôle. Par contre dans le cas d'un geste de modulation, nous pensons qu'elle joue un rôle dont la nature et l'importance restent à définir. Ainsi, dans un premier temps nous avons négligé ce rôle, en utilisant uniquement des transducteurs gestuels non rétroactifs tels que manches à balai, curseurs, jauges de contrainte, qui sont par ailleurs suffisamment développés.

Deux prototypes de transducteurs gestuels rétroactifs ont été réalisés à l'ACROE. Le geste est ramené à une seule direction d'action. La trajectoire est rectiligne, horizontale dans le premier cas, verticale dans le second. Pour le premier prototype, le geste est à caractère corporel (course de 50 cm dans la direction d'action) ; pour le second, il est à caractère digital (course de 5 cm). Le premier prototype nous a servi à expérimenter des cas de simulation simples comme la simulation d'une percussion entre un objet de masse donnée tenu par l'opérateur et un obstacle fixe détérminé par sa raideur. Le second prototype qui est plus réduit et plus performant sur le plan mécanique, nous servira d'abord à une expérimentation systématique, sur la perception gestuelle seule au niveau élémentaire : détermination des seuils de sensibilité aux déplacements, aux variations d'effort, détermination d'une corrélation entre la perception des déplacements et la perception des efforts etc. et sur la corrélation entre perceptions gestuelles et perceptions sonores élémentaires d'une part et perceptions gestuelles et perceptions visuelles élémentaires d'autre part.

PREMIÈRE PARTIE : LE SYSTEME DE SIMULATION CORDIS

A/PRINCIPES GENERAUX DU **SYSTEME CORDIS**

Nous avons annoncé dans l'introduction que la démarche de l'ACROE consistait à envisager l'ordinateur comme moyen de **représentation de l'instrument.**

La représentation **idéale** de l'instrument est une représentation physique réelle c'est-à-dire sans aucune réduction. C'est dans ce cas que le champ d'expérimentation est le plus large. Mais ceci est utopique. Concrètement le moyen de représentation impose des contraintes et nous oblige à faire des réductions quant au degré de réalité de l'objet. Nous sommes alors amenés à faire certaines hypothèses a priori, pour ne garder de l'instrument que les caractéristiques qui nous semblent les plus pertinentes sur le plan perceptif. Nous faisons ainsi une représentation fonctionnelle approchée de l'instrument. Deux considérations cependant nous confortent dans l'idée de la nécessité d'une telle procédure :

- La configuration mécanique réelle d'un instrument n'est pas forcément l'expression la plus parfaite de sa fonction. Les contraintes mécaniques introduisent des lourdeurs qui ne contribuent pas forcément à une richesse expressive. L'analyse mécanique, orientée vers l'obtention de modèles fonctionnels simples peut contribuer à une meilleure connaissance du domaine de la mécanique instrumentale, dans sa relation avec les phénomènes perceptifs.
- La recherche d'une finesse ou d'une exactitude au sens de la mesure physique, peut s'avérer inutile au-delà d'une certaine faculté de perception. Ainsi par exemple, des sons obtenus par une simulation de cordes dans CORDIS (ayant un spectre inharmonique) sont cependant perçus comme étant des sons de cordes. Avant de chercher des modèles de plus en plus fins, une approche perceptive plus globale doit permettre de déterminer les contributions respectives, de la structure de l'instrument (forme, matière, articulations...), des modalités de communication d'énergie à l'instrument, des phénomènes de diffusion etc..

Enfin nous ne pouvons que procéder par étapes en nous attachant tout d'abord à la constitution d'un premier outil de travail et d'expérimentation, aussi imparfait soit-il.

Dans ce chapitre, nous allons présenter le système CORDIS (système de simulation des mécanismes instrumentaux) d'abord du point de vue des principes généraux. Ensuite nous présenterons brièvement sa deuxième réalisation logicielle par C. CADOZ en 1981. Mais c'est la première version logicielle datant de 1979 (C.CADOZ) qui est le point de départ de notre travail en ce qui concerne la détermination des fonctions prises en charge par le processeur temps réel. Nous rappellerons quelques particularités de cette première version.

1. Généralités

Le système CORDIS a comme point de départ une analyse mécanique des instruments existants (instrument au sens large comme nous l'avons déjà défini). Le problème de l'analyse mécanique pour la définition du système CORDIS ne se réduit pas au fait de trouver pour chaque cas particulier un modèle mécanique correspondant, mais de construire ce modèle à partir de modèles de base que nous aurons définis auparavant. Nous introduisons dès ce niveau deux notions importantes et complémentaires qui sont : **la modularité** et **la combinatoire.**

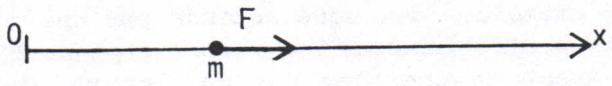
Les modules sont les éléments de base du système CORDIS. La simulation d'un instrument par le système CORDIS, se traduit par un assemblage particulier d'un certain nombre de modules. L'ensemble des règles d'assemblage constitue la combinatoire. Le choix des modules doit être tel, que ceux-ci apparaissent comme des "objets" du système de représentation des mécanismes instrumentaux. Leurs combinaisons, donc la combinatoire, doit elle aussi se comprendre à l'intérieur de ce système. Les modules élémentaires que nous avons choisis sont les plus petits éléments correspondant à cette définition.

Les modules du système CORDIS ont une triple détermination :

- une détermination mécanique,
- une détermination algorithmique,
- une détermination linguistique.

Un module correspond d'abord à un modèle mécanique élémentaire : par exemple une masse ponctuelle de valeur m, pouvant se déplacer dans une seule direction Ox, sans aucun frottement, et soumise à une force extérieure F s'exerçant dans la direction de déplacement (**Figure 1**).

FIGURE 1.



Le fonctionnement d'un tel modèle obéit à l'équation :

$$F(t) = m.x''(t) \quad ;$$

F(t) est la force extérieure, x"(t) est l'accélération de la masse.
Ceci correspond à la détermination mécanique du module.

Sa détermination algorithmique correspond à la mise en oeuvre informatique de cette équation : partage en éléments discrets, échantillonnage temporel, introduction d'hypotèses réductrices pour le séquencement des calculs etc..

Pour pouvoir représenter des mécanismes plus complexes avec les modules de base, nous devons avant tout les désigner et décrire la manière de les ordonner. D'une manière générale nous devons adopter un langage. Celui-ci peut avoir comme support le graphique, l'écriture, la voie etc.. Le choix du type de support pour le langage fait l'objet d'une étude en soi, dans le cadre d'une recherche sur l'interactivité. Le choix du type de terminal d'ordinateur (console alphanumérique, console graphique, terminal spécialisé à définir etc.) est une application de cette étude. Le système CORDIS utilise actuellement l'écriture comme support du langage, mais ceci n'est en aucun cas figé.

2. Analyse mécanique des instruments

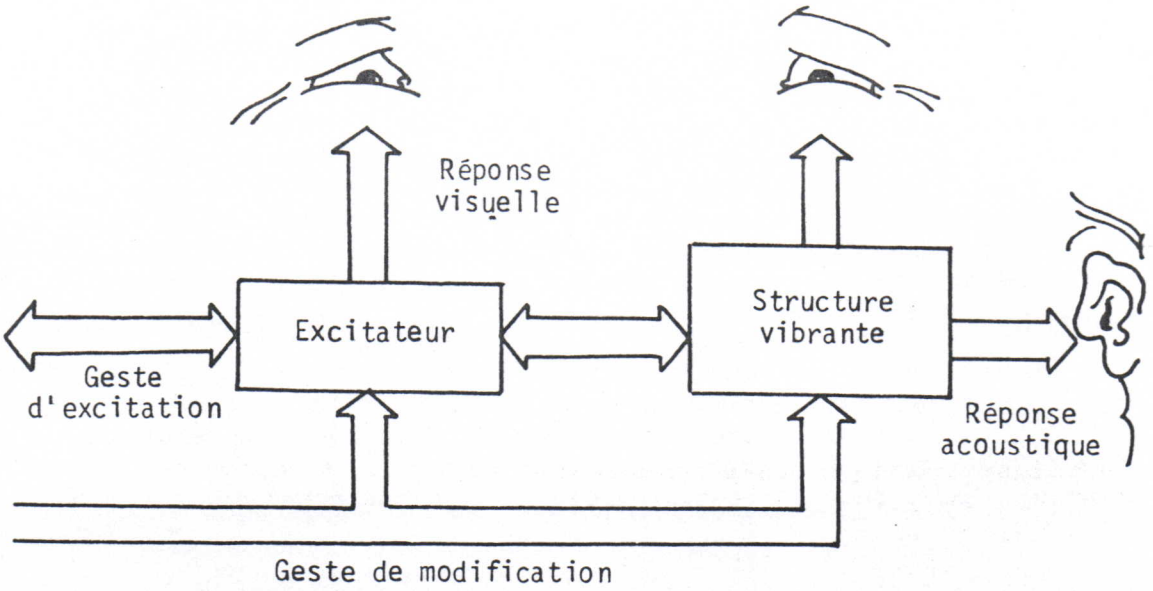
2.1. La séparation en excitateur - structure vibrante

Nous allons commencer par une analyse macroscopique sur le plan fonctionnel (**Figure 2**).

Tout instrument possède au moins une **structure vibrante.** Celle-ci est par définition la structure mécanique qui produit l'onde acoustique. Ses mouvements sont rapides et vibratoires. Par exemple une corde tendue entre deux points est une structure vibrante. Celle-ci peut être excitée directement par le doigt, la paume, les lèvres etc.. Elle peut aussi être excitée par l'intermédiaire d'un mécanisme que nous appelons **excitateur** auquel s'applique le geste. Dans le cas de la corde tendue, elle peut être pincée par le doigt, grattée par l'ongle ou par une lame etc..

Ainsi dans le cas le plus général, un instrument est formé de deux ensembles mécaniques dont le premier (l'excitateur) a un mouvement lent et non nécessairement vibratoire et le second (la structure vibrante) a un mouvement rapide et vibratoire. Dans certains cas particuliers, l'excitateur peut se réduire à un mécanisme très simple ou même disparaître.

FIGURE 2.



2.2. Etude de la structure vibrante

L'observation des organes qui vibrent dans les instruments traditionnels ainsi que parmi les objets de la vie courante, montre que les structures vibrantes peuvent se réduire aux catégories suivantes :

- les cordes,
- les lames,
- les colonnes d'air,
- les membranes,
- les volumes solides ou fluides.

Cette dernière catégorie englobe en fait toutes les précédentes dans la mesure où toutes ces structures vibrantes sont des objets réels tridimensionnels. Nous attachons toutefois une importance particulière à l'analyse de la corde et de la membrane car, dans la mesure où une ou deux dimensions deviennent prépondérantes devant l'autre, les modèles unidimensionnels que nous allons développer conviendront mieux pour l'étude de ces cas particuliers.

En général la vibration sonore n'est pas directement diffusée par la partie qui la produit mais elle est transmise par des liaisons mécaniques à des résonateurs. Ceux-ci font une adaptation d'énergie et modifient la vibration sonore selon leur propre morphologie. Nous allons considérer les résonateurs comme faisant partie de la structure vibrante et nous les analyserons dans les mêmes termes.

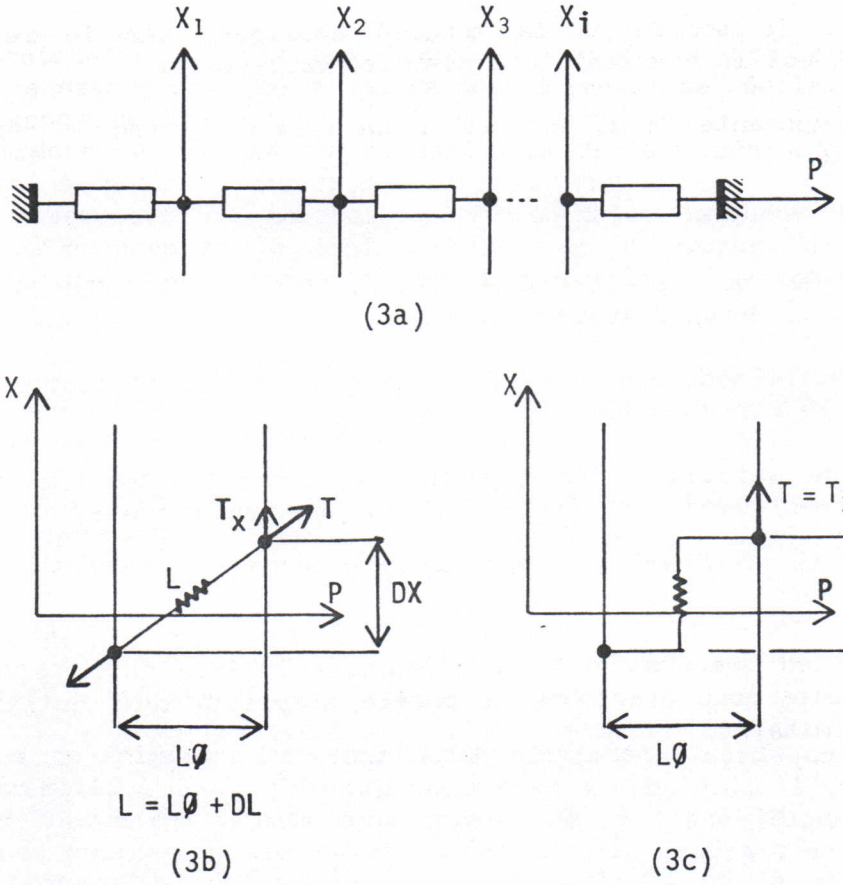
Nous pensons pouvoir modéliser toutes les catégories de structures vibrantes énoncées ci-dessus à l'aide d'assemblages des trois éléments suivants :

- la ligne vibrante,
- la surface vibrante,
- le volume vibrant.

2.2.1 Modèle de la ligne vibrante

C'est par le modèle de la ligne vibrante formée d'éléments discrets que nous essaierons d'approcher le comportement réel d'une corde vibrante.

FIGURE 3.



La vibration d'une corde réelle fait intervenir dans le cas général une vibration transversale, une vibration longitudinale, et une vibration en rotation. Une première approximation consiste à considérer qu'il n'y a pas de couplage mécanique entre chacune de ces vibrations. Dans ce cas il est possible de les étudier séparément. On introduit ainsi le modèle théorique de

la ligne vibrante continue, possédant une seule direction de vibration et une direction de propagation. Les caractéristiques mécaniques de la ligne continue sont sa masse, sa longueur, sa tension, les coefficients de perte par frottement interne et par frottement dans l'air. Le partage en éléments discrets intervient quand nous considérons que cette ligne vibrante théorique est formée de points matériels distincts possédant une direction unique de déplacement (**Figure 3a**). Par convention, cette direction de déplacement, nous la représentons perpendiculaire à la direction de propagation. Si la ligne est homogène, à chaque point matériel nous attribuons une masse dont la valeur est égale à la masse globale divisée par le nombre de points. Il faut maintenant définir la manière dont les éléments discrets interagissent.

Pour tenir compte de l'élasticité de la ligne, il faudrait faire intervenir entre deux points matériels un élément de rappel agissant dans la direction réelle définie par les deux points consécutifs (**Figure 3b**). Ceci implique le calcul de la longueur DL : l'élongation du ressort dans le plan. On aurait alors la relation :

$$T = K.DL \quad ;$$

T est le module de la tension appliquée par le ressort au point matériel, K est la constante d'élasticité du ressort.

La composante Tx de la tension dans la direction de déplacement Ox a comme module :

$$Tx = K' . DX^3$$

si on considère que DL est faible (CADOZ, LUCIANI, FLORENS, 1981, Synthèse Musicale ....., Revue d'Acoustique N° 59).

DX est le module de la différence des positions relatives des points X1 et X2 dans la direction Ox.

Prenons maintenant comme modèle, un ressort entre les deux points, qui agit uniquement dans la direction Ox (**Figure 3c**). Alors :

$$Tx = K.DX \quad .$$

La nature de la simplification faite, ainsi que son intérêt apparaissent directement en comparant ces deux équations. Pour notre modélisation de la ligne vibrante nous prendrons ce modèle simplifié tant qu'il ne se révélera pas trop limitatif.

Pour tenir compte enfin des pertes par frottement nous introduirons deux éléments de frottement :

- Le frottement interne sera représenté par un élément placé en parallèle avec le ressort et agissant dans la direction Ox. Le module de la force de frottement est donné par l'équation :

$$Ff = Z.VR \quad ;$$

Z est la constante de frottement,
VR est la vitesse relative des deux points dans la direction Ox.

- Le frottement dans l'air sera représenté par un élément de même nature, s'exerçant entre chaque point matériel et une origine fixe sur les axes Ox. Enfin la liaison des points extrêmes à un support fixe sera aussi modélisée par des éléments ressort - frottement en parallèle.

Dans le cas de modélisation d'une ligne homogène, les masses, les constantes de raideur, les constantes de frottement auront des valeurs égales pour chacun des éléments distincts.

Le comportement de telles lignes vibrantes a été étudié et nous connaissons la nature des écarts par rapport à une ligne réelle. Le recours à un modèle discontinu contient deux sources d'écart :

- le nombre de composantes sinusoïdales de la vibration est limité et égal au nombre de points discrets qui interviennent ;
- les fréquences des composantes sinusoïdales marquent des écarts par rapport aux fréquences de la corde réelle, et le rapport ne tend vers un rapport harmonique que lorsque le nombre de points distincts devient très grand.

Sur un plan qualitatif général, on retrouve les aspects fondamentaux : la vibration est complexe, les phénomènes de propagation existent, de même que les phénomènes de réflexion aux extrémités.

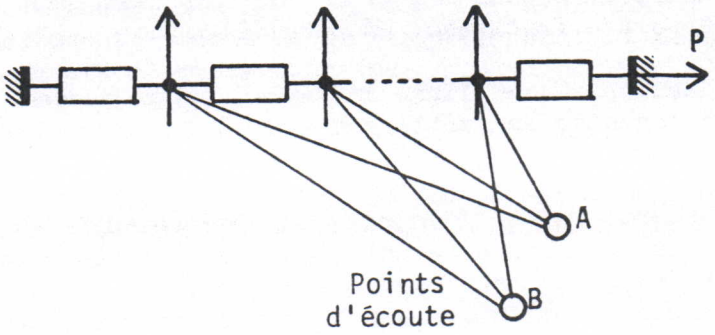
Nous avons considéré jusqu'à maintenant la modélisation d'une ligne vibrante unidimensionnelle (avec une direction de vibration), sur laquelle nous avons fait intervenir ensuite le partage en éléments discrets. Nous avons alors introduit une approximation dans le calcul des distances pour l'élément ressort. Avant de passer à une approche de la corde réelle nous soulevons dès ce niveau le problème de la constitution de l'onde sonore résultante qui arrive au point d'écoute.

Si nous ne faisons intervenir aucun résonateur (corde uniquement fixée aux deux extrémités à des objets rigoureusement immobiles), l'onde sonore est diffusée par chacun des points constituants de la ligne (**Figure 4a**). Entre chacune de ces sources et les points d'écoute, il faudrait en toute rigueur étudier la propagation de chacune de ces vibrations et la manière dont le milieu de propagation agit sur ces vibrations. Il faudrait enfin faire la sommation de toutes les vibrations en bout de chaîne c'est-à-dire au point d'écoute.

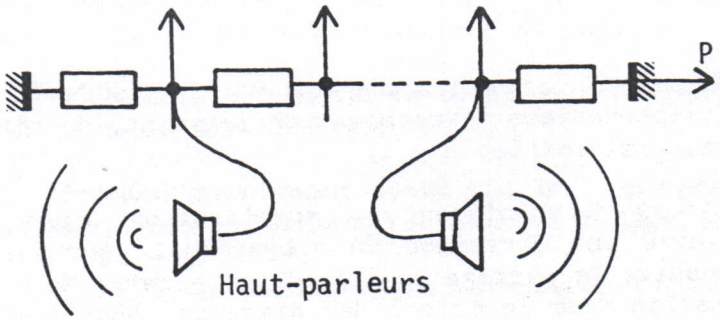
Il est dans un premier temps hors de question pour nous de modéliser tout cela. Dans le cas d'une corde par exemple, nous avons une approche pragmatique de ce problème qui consiste à considérer les vibrations en deux points assez éloignés de la corde et de les envoyer chacune vers un haut-parleur en stéréophonie (**Figure 4b**). Nous obtenons de cette manière-là une

impression de l'existence de l'objet dans l'espace. Dans le cas où la corde est reliée à un résonateur, la vibration de la corde se transmet par une liaison particulière au résonateur ; celui-ci vibre alors à son tour et c'est essentiellement cette vibration qui est diffusée. Il faut alors d'une part étudier la liaison source de vibration - résonateur, pour simuler la manière dont celui-ci est excité, d'autre part il faut modéliser le résonateur comme un volume vibrant et considérer quelques points éloignés les uns des autres comme sources effectives de vibrations. On peut alors faire une somme pondérée de ces vibrations pour les envoyer vers les haut-parleurs.

FIGURE 4.



4a : Constitution de l'onde acoustique directe.

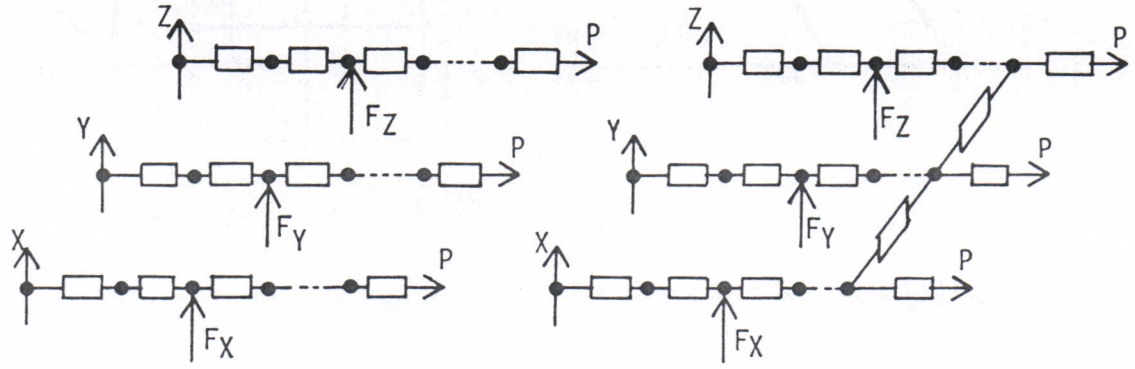


4b : Exemple de constitution de l'onde acoustique dans une expérience avec CORDIS.

Nous allons voir maintenant comment nous pouvons approcher le comportement d'une corde réelle (avec trois directions de vibration) à l'aide du modèle de la ligne vibrante unidimensionnelle (**Figure 5a**). Considérons trois lignes vibrantes indépendantes. Ce sont trois représentations d'une même corde dans les trois directions de vibration. Nous les exciterons par trois composantes de force à l'aide d'un capteur à trois degrés de liberté. Les paramètres de chacune des lignes peuvent être différents et choisis de manière à créer les conditions quantitatives pour chaque type de vibration de la corde réelle. En l'absence de résonateur, nous considérons comme sources de vibrations deux ou plusieurs points éloignés les uns des autres, pour faire une sommation pondérée des vibrations, mais chacune de ces vibrations est elle-même une combinaison, linéaire par exemple de trois vibrations : transversale, longitudinale et en rotation. Dans le cas où la vibration est transmise à un résonateur, il faut toujours considérer la liaison source de

vibration - résonateur, et voir quel rôle joue chacune des trois vibrations, dans l'excitation du résonateur. On peut enfin introduire un couplage en un ou plusieurs points des trois lignes unidimensionnelles par des liaisons élastiques ou par frottement (**Figure 5b**).

FIGURE 5.



Cordes non couplées (5a) Cordes couplées (5b)

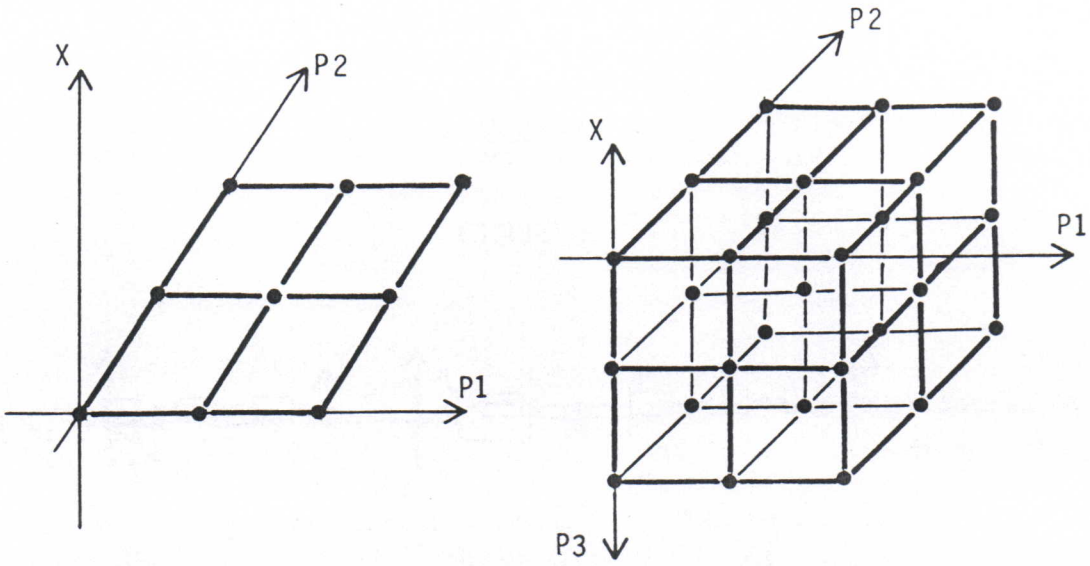
2.2.2 Modèles de la surface vibrante et du volume vibrant

Une surface réelle discrète (nous parlons de surface quand une des dimensions, "l'épaisseur", est de très faible valeur par rapport aux deux autres dimensions) serait formée par des points distincts possédant chacun six degrés de liberté et relié chacun à quatre autres points identiques. Il en résulterait des calculs de distances dans l'espace pour les éléments de liaison et six variables de position pour chacun des points.

Nous procédons de la même manière que pour le modèle de la ligne vibrante en adoptant le modèle d'une surface vibrante discrète, possédant une direction de vibration que nous représenterons par convention perpendiculaire au plan de la surface (**Figure 6a**). Les liaisons entre les points différents sont des liaisons, élastiques et par frottement, qui ne tiennent compte que de la différence des positions et des vitesses dans la direction de vibration.

Nous construirons de la même manière un modèle de volume vibrant, comme un réseau à trois dimensions où chaque point est relié à six autres points, mais ne se déplace que dans une seule direction (**Figure 6b**). Les éléments de liaison élastique et par frottement tiennent compte des distances et des vitesses relatives selon la direction de vibration.

FIGURE 6.



(6a) Modèle de la surface vibrante.

(6b) Modèle du volume vibrant.

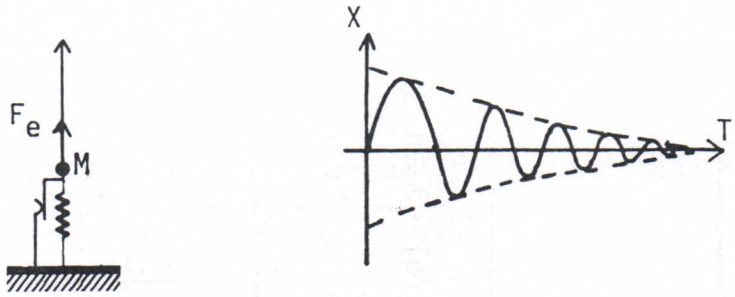
2.2.3 La cellule : Module vibratoire élémentaire

Le plus petit élément vibratoire, dans les modèles que nous venons de décrire, apparaît comme étant une masse reliée à un point fixe par une liaison élastique et par frottement (**Figure 7**). La liaison élastique est représentée par un ressort. Le ressort et le frottement agissent dans la direction de déplacement. Le mouvement de la masse sera un mouvement oscillatoire sinusoïdal amorti, si nous lui appliquons initialement une impulsion de force extérieure ou si, en l'éloignant de sa position de repos nous la "lâchons" avec une vitesse initiale nulle. Si K (la constante de raideur du ressort), Z (la constante de frottement) sont données, l'amplitude et la fréquence de l'oscillation sont entièrement déterminées. L'ensemble ainsi constitué est la cellule de base du système vibratoire; nous l'appelons **la cellule.**

La cellule est le **module vibratoire élémentaire** du système CORDIS. L'événement sonore qu'elle produit est entièrement déterminé par la donnée, d'une part de la structure mécanique (définition de la cellule, attribution de valeurs aux paramètres K et Z), d'autre part du mode d'excitation.

Quand on veut, à partir de plusieurs cellules, former une structure vibrante, il est nécessaire d'avoir des éléments de liaison. L'élément de liaison est formé d'un ressort et d'un frottement en parallèle. Contrairement à la cellule il n'a pas de détermination sonore en soi.

FIGURE 7.



$\forall kc, z < 2\sqrt{kc} \Rightarrow$

Réponse impulsionnelle de la cellule.

$$x(t) = \frac{f_0}{\omega} e^{-z/2\,t} \sin\omega t$$

$$\omega = \frac{\sqrt{4kc - z^2}}{2} = 2\pi f \text{ (pulsation)}$$

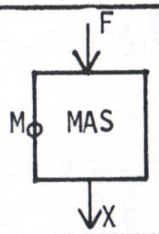
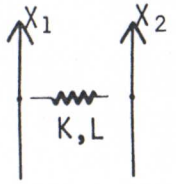
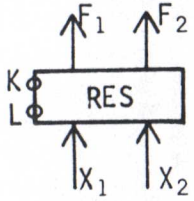
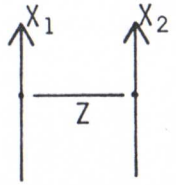
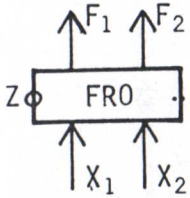
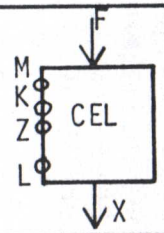
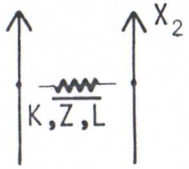
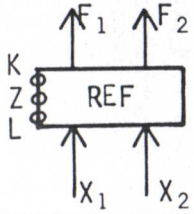
$$\frac{f_0}{\omega} e^{-z/2\,t} \text{ est l'enveloppe}$$

2.3 Etude de l'excitateur et de la liaison excitateur - structure vibrante

L'excitateur est un ensemble mécanique au mouvement lent (la rapidité est du même ordre que pour le geste) et non nécessairement vibratoire. C'est un relais entre le geste et la structure vibrante. C'est lui qui reçoit les paramètres (forces, déplacements) du geste d'excitation, à travers les transducteurs gestuels rétroactifs, et contrôle les transducteurs en simulant la réponse du système à un geste. L'analyse mécanique de l'excitateur se fait dans les mêmes termes que pour la structure vibrante, mais alors que pour celle-ci le module le plus élémentaire est la cellule, pour l'excitateur les éléments de base vont être la masse, le ressort, le frottement. Pour éviter les calculs complexes de distances réelles dans l'espace, nous adoptons des modèles unidimensionnels comme pour la structure vibrante.

L'excitateur se caractérise par ailleurs par des comportements discontinus soit dans l'articulation de ses différents constituants, soit dans sa liaison avec la structure vibrante. Nous allons donner ici la formulation la plus générale, qui permet de représenter les liaisons discontinues. Elle est déduite de l'analyse des cas particuliers de jonction excitateur - structure vibrante que sont la percussion, le pincement, l'oscillation entretenue, le frottement (CADOZ, LUCIANI, FLORENS, 1981, Synthèse Musicale .... , Revue d'Acoustique N° 59).

FIGURE 8.

Mnémonique	Pictogramme	Diagramme	Algorithme
MAS masse			$X(n) = \frac{F(n)}{M} + 2X(n-1) - X(n-2)$
RES ressort			$F_2(n) = K.(X_1(n) - X_2(n)) + K.L$ $F_1 = -F_2$
FRO frottement			$F_2(n) = Z.((X_1(n) - X_1(n-1)) - (X_2(n) - X_2(n-1)))$ $F_1 = -F_2$
CEL cellule			$X(n) = \frac{F(n)}{M} + \left(2 - \frac{K-Z}{M}\right).X(n-1) + \left(\frac{Z}{M} - 1\right).X(n-2) + \frac{K.L}{M}$
REF ressort frottement			$F_2(n) = (K+Z).(X_1(n) - X_2(n)) - Z.(X_1(n-1) - X_2(n-1)) + K.L$ $F_1(n) = -F_2$

Soient M1 et M2 deux points appartenant respectivement à l'excitateur et à la structure vibrante, entre lesquels on veut établir une liaison discontinue. D'une manière générale la liaison sera formée d'un élément de liaison normale (ressort, frottement) qui prendra plusieurs états conditionnels, caractérisés par des valeurs différentes des constantes K et Z. L'état de "non liaison" qui est un état parmi d'autres, aura comme valeurs pour K et Z des valeurs nulles. Le passage d'un état à l'autre est déterminé par un certain nombre de conditions sur les variables X1 et X2 entre autres (variables de déplacement des points M1 et M2). Ces conditions sont différentes selon le couple état initial - état final considéré. Elles s'expriment à partir des variables X1 et X2 et certains paramètres qui sont en général des constantes pour la condition.

3. Algorithmes du système CORDIS (**Figure 8** p. 20)

A chaque module du système CORDIS correspond un algorithme qui est un processus de calcul numérique pour résoudre les équations correspondant au fonctionnement du modèle simulé. Ces équations lient entre elles deux types de variables : les forces et les déplacements. Ces variables sont dans la réalité des fonctions continues du temps t. Le principe du calcul numérique par l'ordinateur fait remplacer la variable continue t par la variable discrète n.

3.1. Algorithme de la masse

Le modèle mécanique correspondant est celui d'une masse ponctuelle se déplaçant dans une seule direction sous l'action d'une force externe F. Son évolution est décrite à chaque instant par l'équation :

$$F(t) = M.X''(t) \quad ;$$

F(t) est la force appliquée à la masse,
M est la valeur de la masse,
X"(t) est l'accélération de la masse.

L'équation discontinue qui lui correspond est :

$$F(n) = M.X''(n) \quad ;$$

F(n) est la valeur de la force appliquée à la masse à l'instant n,
X"(n) est l'accélération de la masse à l'instant n.

Comme valeur de la vitesse à l'instant n nous prenons l'expression approchée :

$$X'(n) = X(n) - X(n-1) \quad .$$

L'accélération est alors :

$$X''(n) = X(n) - 2X(n-1) + X(n-2) \quad .$$

Alors :

$$F(n) = M.(X(n) - 2X(n-1) + X(n-2)) \quad ,$$

expression qui peut être aussi écrite sous la forme :

$$X(n) = (F(n)/M) + 2X(n-1) - X(n-2) \quad .$$

Dans le premier cas, nous considérons que la réponse du système est la force à l'instant n opposée à l'extérieur, quand on lui impose un déplacement à partir d'une position de repos prise comme origine. Dans le second cas c'est le déplacement qui est une réponse du système à une force imposée. Sur le plan de l'exécution de l'algorithme les deux expressions ne sont pas équivalentes. Par convention, nous retenons comme algorithme de la masse cette deuxième expression.

Le module masse fait partie d'un groupe de modules que nous appelons **modules matériels.** Pour ceux-ci nous retenons par convention comme entrée de l'algorithme la variable de **force** et comme sortie la variable de **déplacement.** M est un paramètre de l'algorithme.

3.2. Algorithmes du ressort et du frottement

Soient deux éléments matériels M1 et M2 liés par un ressort. Sous l'action de forces externes au système ainsi considéré, M1 et M2 prennent des positions X1(n) et X2(n) à l'instant n. Soient F1(n) et F2(n) les forces égales en module et de sens opposé exercées par le ressort sur M1 et M2 :

$$F1(n) = K.(X2(n) - X1(n)) - K.L$$
$$F2(n) = - F1(n) \quad ;$$

L est la longueur du ressort au repos.
Ces deux expressions constituent l'algorithme du module ressort.

Si maintenant les deux points matériels M1 et M2 sont liés par un frottement, la force exercée sur M1 et M2 dépend de la vitesse relative des deux masses :

$$F1(n) = Z.(X2(n) - X1(n) - X2(n-1) + X1(n-1))$$
$$F2(n) = - F2(n) \quad .$$

Ces deux expressions définissent l'algorithme du module frottement.

Les modules ressort et frottement font partie d'un groupe de modules que nous appelons modules de liaison. Pour ces modules nous retenons par convention comme entrée de l'algorithme, la distance relative des deux éléments liés entre eux, et comme sortie, les variables de force s'exerçant sur ces mêmes éléments matériels.

3.3. Algorithme de la cellule

Comme module de base du système CORDIS, nous ne retenons que les modules masse, ressort, frottement. Ce sont les modules élémentaires. Or, nous avons vu déjà dans le cas de la ligne vibrante que le module **cellule** possède un degré d'élémentarité suffisant pour représenter les structures vibrantes. Il est donc inutile de la dissocier à chaque fois en masse, ressort, frottement. D'une manière générale, quand un assemblage particulier de modules élémentaires semble suffisamment intéressant pour le type de structures que nous considérons, nous constituons des **modules primaires** et nous définissons pour ces modules un système de variables d'entrée, un système de variables de sortie, des paramètres de contrôle, un algorithme ayant une forme compacte et nous leur attribuons un nom. Ils deviennent ainsi des éléments de langage à leur tour.

Par l'utilisation de modules primaires, nous nous dispensons d'une analyse mécanique trop fine quand elle n'est pas nécessaire, nous économisons un temps de calcul précieux, de la place dans la mémoire de l'ordinateur et du temps pendant la phase de description de la structure.

Dès lors que modules élémentaires et modules primaires coexistent, pour garder une homogénéité sur le plan algorithmique, nous considérons ces derniers comme une **combinaison** de modules élémentaires. Ainsi la cellule est la combinaison d'une masse, d'un ressort et d'un frottement. Le ressort et le frottement lient la masse à un point fixe.

L'équation de la masse est la suivante :

$$X(n) = (F(n)/M) + 2X(n-1) - X(n-2) \quad , \qquad (1)$$

celle du ressort :

$$Fr(n) = - K.X(n) + K.L \quad , \qquad (2)$$

celle du frottement :

$$Ff(n) = - Z.(X(n)- X(n-1)) \qquad (3)$$

et la combinaison se traduit par :

$$F(n) = Fe(n) + Fr(n) + Ff(n) \quad ; \qquad (4)$$

en toute rigueur nous obtenons :

$$X(n) = (1+K'+Z')^{-1}.((Fe(n)/M) + (2+Z').X(n-1) - X(n-2) + K'.L) \quad . \qquad (5)$$

Or pour une description éclatée de la cellule, les calculs (1), (2) et (3) se font de manière séquentielle dans un ordre précis. Si (1) se fait avant (2) et (3), il utilise comme valeurs Fr(n-1) et Ff(n-1). Si (2) et (3) se font avant (1) alors ils utilisent comme valeurs X(n-1) et X(n-2). Dans tous les cas le calcul réel fait de manière séquentielle correspond à l'expression :

$$X(n) = (Fe(n)/M) + (2-K'-Z').X(n-1) - (1-Z').X(n-2) + K'.L \quad . \quad (6)$$

K' et Z' sont égales aux valeurs de K et Z divisées par M.

C'est l'expression (6) que nous retenons pour l'algorithme de la cellule.

3.4. Algorithmes de la liaison

Nous considérons un autre module primaire que nous appelons **liaison**, formé d'un ressort et d'un frottement en parallèle. Sa nécessité nous est apparue aussi lors de l'étude de la ligne vibrante. L'algorithme de ce module est formé des deux expressions :

$$F2(n) = (K+Z).(X1(n)-X2(n)) - Z.(X1(n-1)-X2(n-1) + K.L)$$
$$F1(n) = - F2(n) \quad .$$

4. Le langage CORDIS (**Figure 8** p. 20)

Le langage CORDIS est le moyen pour l'opérateur d'utiliser le système CORDIS. Actuellement son support est l'écriture, néanmoins elle est doublée d'un système graphique (qui reste en amont des moyens informatiques) par l'attribution d'un **pictogramme** à chaque élément. Le pictogramme est un dessin qui évoque d'une manière simple le modèle simulé. Parallèlement aux pictogrammes, nous attribuons à chaque module un **diagramme fonctionnel** qui est un dessin où figurent les entrées, les sorties, les paramètres et le nom du module.

Quand l'utilisateur du système CORDIS construit une structure sur le papier, il peut dans un premier temps employer indifféremment le système de pictogrammes, de diagrammes ou l'écriture. Dans un deuxième temps, pour entrer ces informations dans l'ordinateur au moyen du clavier de télétype, il est obligé d'employer l'écriture codée avec les mots et les règles d'écriture du langage CORDIS.

4.1. Fonctions et modes opératoires généraux du système CORDIS

L'activité musicale traditionnelle s'organise autour de trois pôles qui sont :

- la lutherie,
- le jeu instrumental,
- la composition.

La lutherie (activité dont le but est de construire l'instrument) se situe en général bien avant les deux autres. Toujours d'une manière générale, le luthier se distingue de l'instrumentiste et du compositeur, mais il est en même temps facteur et expérimentateur. En effet il expérimente l'instrument et en modifie la facture jusqu'à ce qu'il le juge satisfaisant comme produit fini.

Le jeu instrumental peut avoir lieu avant, après ou pendant la phase de composition.

L'instrument étant construit, la personne qui en dispose l'expérimente par le jeu. La phase d'apprentissage du jeu instrumental se fait d'abord indépendamment de toute idée musicale élaborée ou activité de composition. C'est une activité sensori-motrice qui sollicite le geste, la vue et l'ouïe, dont le but est la connaissance de cet objet-instrument.

Le jeu instrumental peut avoir lieu après l'activité de composition, quand l'instrumentiste interprète une oeuvre musicale. Enfin en situation d'improvisation, jeu instrumental et composition ont lieu simultanément.

Le problème de la conservation de l'instrument ne se pose pas explicitement quand il s'agit d'instruments mécaniques réels. Dans le contexte informatique, pour garder une trace de l'instrument, il faut mémoriser l'ensemble des données et des programmes qui correspondent à sa structure sur un plan qualitatif et quantitatif.

Dans la pratique traditionnelle, la partition était le seul moyen de mémorisation objective de l'expérience. Elle donnait une représentation écrite relative au son, et également (implicitement) relative au geste. Dans le contexte informatique nous pouvons enregistrer le geste et le son séparément ou ensemble.

L'activité musicale ayant comme support l'ordinateur et le système CORDIS se découpe elle aussi en trois phases :

- La phase de prestructuration correspond à la lutherie. Elle se divise elle-même en deux phases : la prestructuration qualitative et la prestructuration quantitative. Le choix des modules constituants de l'instrument et de leur assemblage particulier; le choix des conduites que nous pouvons avoir à l'égard de l'instrument, en particulier par la détermination des points d'accès à la structure; le choix des points d'observation visuelle; le choix des points de diffusion sonore, constituent la prestructuration qualitative. Le choix des valeurs des paramètres mécaniques constitue la prestructuration quantitative.
- La phase de jeu correspond à l'expérience instrumentale en amont de la phase de composition et à un véritable jeu instrumental en aval.
- La phase de composition s'appuie sur la mémorisation objective du geste et du son et la définition de traitements divers sur les "objets" ainsi constitués.

La spécificité de l'outil informatique apparaît d'une part dans le fait que ces trois phases coexistent potentiellement à tout instant, d'autre part dans le fait qu'il est possible de mémoriser de manière objective l'expérience instrumentale dans sa globalité, comme complément à la mémoire (subjective) de l'instrumentiste.

Ce que nous venons de décrire de manière informelle se concrétise dans le système CORDIS par différents modes d'utilisation, auxquels nous associons la présence de l'enregistrement du geste ou du son ou des deux à la fois. Nous allons présenter ici brièvement les fonctions et modes opératoires généraux du programme CORDIS-LOGICIEL Version 2 (CLAUDE CADOZ. 1981) qui permet la simulation de structures mécaniques décrites dans le langage CORDIS (**Figure 9**).

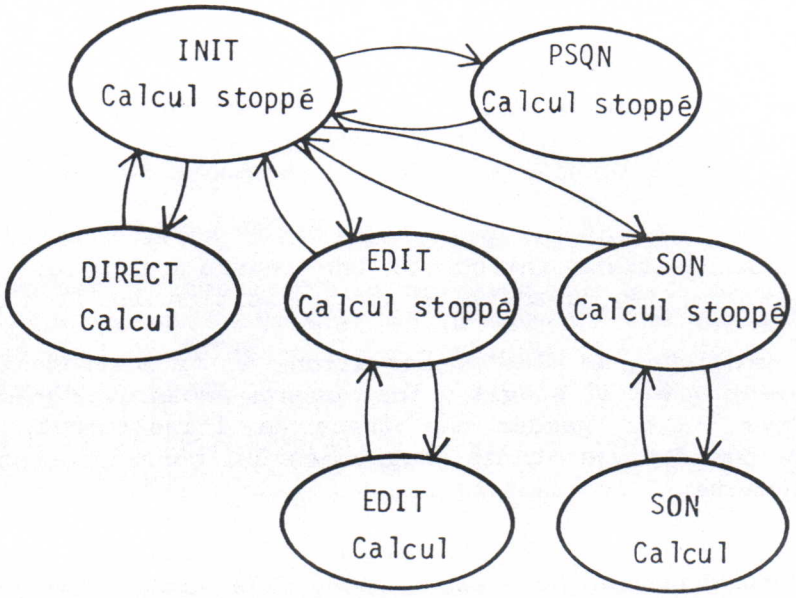


FIGURE 9.

Le programme CORDIS-LOGICIEL Version 2 est un programme de simulation en temps différé de mécanismes instrumentaux.

C'est un programme complet qui peut prendre en charge la simulation de l'excitateur et de la structure vibrante. Mais si la simulation de l'excitateur peut se faire en temps réel (Fe = 100 Hz), il n'est pas envisageable d'obtenir des sons en temps réel (Fe = 25,6 kHz) même pour une seule cellule.

La détermination des caractéristiques qualitatives de la structure se fait pendant la phase de prestructuration qualitative **PSQL**. Le principe de la structure à simuler étant défini, traduit en schémas à l'aide de pictogrammes ou de diagrammes, celle-ci est décrite par une série d'instructions du langage CORDIS, qui sont stockées dans un fichier structure auquel on donne un nom. Avant toute utilisation effective de ce fichier, intervient une phase d'assemblage, pendant laquelle certaines erreurs de langage pourront être détectées et signalées. Si tel est le cas, il faut modifier le fichier structure et assembler de nouveau jusqu'à ce que la phase d'assemblage se fasse sans messages d'erreur. Nous disposons alors d'un fichier structure exécutable. Par l'activation de ce fichier, nous entrons dans un processus de calcul en boucle. Nous disposons cependant d'un certain nombre de modes pour lancer, interrompre, modifier les conditions, relancer le processus de calcul.

Après l'assemblage du fichier structure, il reste à déterminer les valeurs des paramètres mécaniques de la structure. Il faut donc passer par la phase de prestructuration quantitative avant le calcul effectif.

Le calcul intervient entre les variables d'entrée, qui sont les variables du jeu gestuel et les variables de sortie, qui sont des signaux à destination sonore ou à destination visuelle (observation sur écran). Pour les deux catégories de variables se pose le problème de la mémorisation. La **figure 9** montre les modes fondamentaux de la **phase active.**

Le mode **INIT** est celui dans lequel nous nous trouvons, dès que nous appelons un fichier structure exécutable. Dans le mode INIT, il n'y a pas de calcul. Nous pouvons demander l'affichage de toutes les commandes possibles, en particulier celles qui permettent de passer dans les autres modes.

Le mode **PSQN** permet de préciser les valeurs des paramètres mécaniques de la structure. L'assemblage du fichier structure permet d'avoir, quand nous passons en mode PSQN, un affichage ordonné des paramètres nécessaires pour cette structure. Dans le mode PSQN il est possible de modifier les paramètres un par un ou de manière globale. Il est possible d'enregistrer l'ensemble des paramètres définis à un moment donné sous forme de fichier de paramètres. Il est possible d'appeler un fichier précédemment défini pour l'utiliser ou pour le modifier.

Le mode **DIRECT** est celui où il n'y a pas d'enregistrement, ni pour les variables d'entrée, ni pour les variables de sortie. Le passage en mode direct démarre automatiquement le calcul. Le mode DIRECT permet de vérifier si une structure déterminée de manière qualitative et quantitative fonctionne correctement. Cette vérification peut se faire par l'observation visuelle des mécanismes simulés, ou par l'observation des informations fournies par le programme quant aux dépassements de capacité dans les calculs etc.. Dans la mesure où il n'y a pas d'enregistrement des échantillons sonores, il est évident qu'il n'est pas possible de restituer le son en temps différé.

En mode **SON** les variables de sortie sont enregistrées. En mode **EDIT** (éditeur) les variables d'entrée sont enregistrées.

Chacun de ces modes fait appel à des sous-modes . Mais notre but étant ici de donner simplement un bref aperçu de la manière dont le programme CORDIS-LOGICIEL opère, nous n'allons pas entamer une description plus détaillée.

4.2. Le langage CORDIS

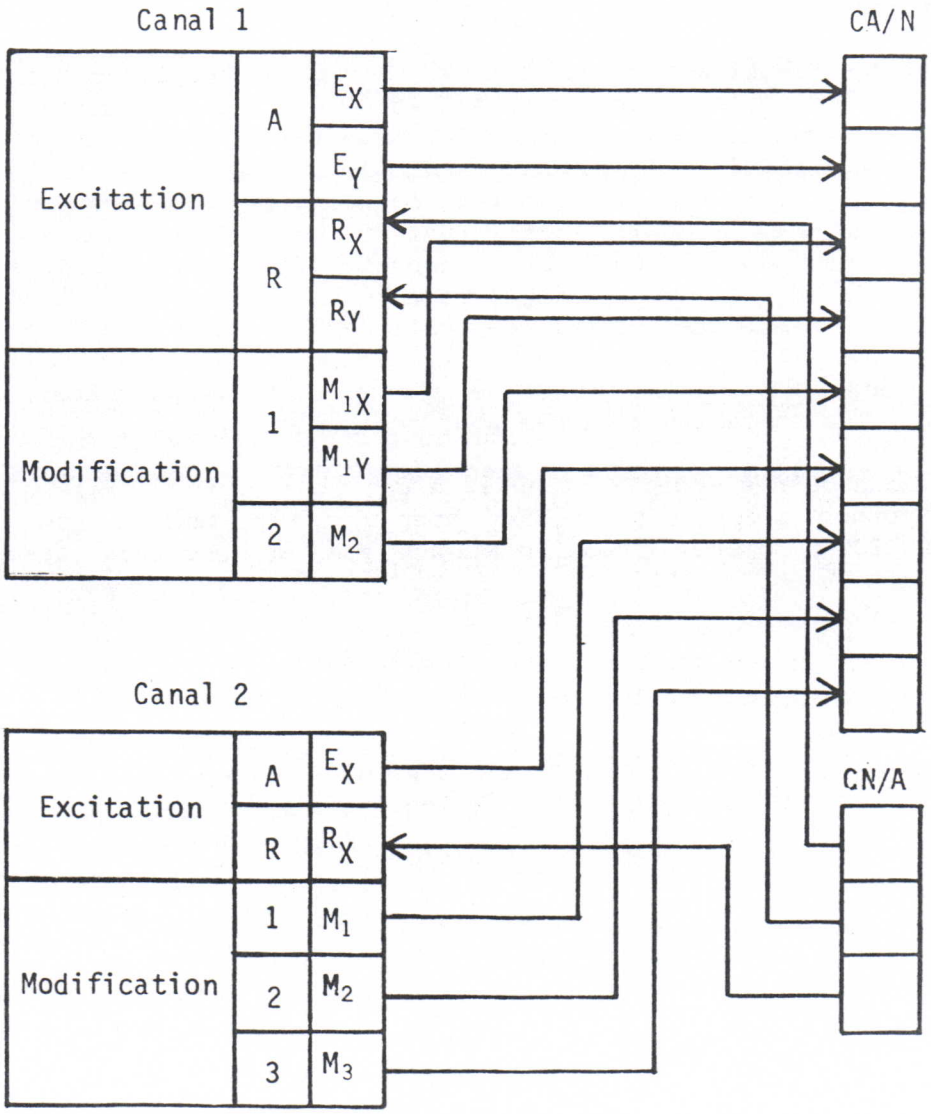
Le langage CORDIS comporte trois grandes catégories d'instructions :

- pour la description des points d'accès,
- pour la description de la structure mécanique proprement dite,
- pour la description des points de sortie.

La première catégorie d'instructions s'appuie sur la notion de **CANAL**, et de nombre de voies, rétroactives ou non, par canal. Un canal correspond à un geste élémentaire. Un geste élémentaire est un geste complet (excitation et modification), permettant d'obtenir un événement sonore élémentaire. A ce titre, il comporte un geste d'excitation unique qui peut être unidimensionnel ou multidimensionnel. Il comporte aussi zéro, un ou plusieurs gestes de modification qui peuvent être unidimensionnels ou multidimensionnels. Un geste élémentaire peut être complexe pour celui qui l'effectue et inversement un geste simple à effectuer peut ne pas être élémentaire.

Un canal comporte plusieurs voies (**Figure 10**). Il comporte obligatoirement une voie unique d'excitation. Cette voie est rétroactive. Si le geste a un degré de liberté on lui associe un convertisseur analogique/numérique et un convertisseur numérique/analogique ; s'il en a plusieurs, on lui associe plusieurs convertisseurs de chaque type. Une voie de modification n'est pas rétroactive. De même que pour un geste d'excitation, on lui associe plusieurs convertisseurs analogique/numérique selon le nombre de degrés de liberté.

FIGURE 10.



La première catégorie d'instructions du langage CORDIS prend en charge la description des canaux, la description des voies, la liaison entre chaque signal physique en provenance ou à destination d'un convertisseur et une grandeur de la simulation, après cadrage et application d'un facteur d'échelle.

La deuxième catégorie d'instructions correspond à deux types de modules de la structure mécanique proprement dite : les modules matériels et les modules de liaison. Ils sont complémentaires car le type de variables qui constitue les entrées pour l'un constitue les sorties pour l'autre et inversement. Les variables étant les forces et les déplacements nous définissons la convention suivante :

Les modules matériels ont comme variables d'entrée des forces et comme variables de sortie des déplacements. Les modules de liaison sont définis par la règle inverse.

La troisième catégorie d'instructions s'appuie sur deux notions : l'observation sur un écran et l'enregistrement pour la sortie sonore en monophonie ou en stéréophonie. Les deux types de modules permettent d'affecter les variables de position des éléments matériels à des convertisseurs numérique/analogique lents ou rapides.

La structure mécanique peut être définie dès le départ avec un certain nombre d'options, correspondant à des parties de structure que nous aurons la possibilité de supprimer ou de réintroduire sans arrêter le processus de calcul. Cela est obtenu par action sur une touche du clavier de télétype qui définit deux états. Cette action correspond à une modification qualitative de la structure en cours de jeu. Dans la description globale, cette partie sera clairement indiquée entre deux lignes de référence à cette touche.

A ces catégories d'instructions, il nous faut ajouter une instruction qui permet d'indiquer la fin de la description de la structure.

Nous ne donnons pas ici des informations plus détaillées sur le langage, car ces informations figurent sur le rapport interne intitulé "Description de CORDIS-LOGICIEL Version 2" par C. CADOZ, 1982.

B/ DETERMINATION DES FONCTIONS DU PROCESSEUR SPECIALISE **CORDIS TEMPS REEL**

Les travaux qui ont permis la détermination des fonctions du processeur spécialisé ont démarré au début de l'année 1980, à l'issue d'une première version complète du programme CORDIS-LOGICIEL (1979). La réalisation matérielle du CTR a commencé au début de l'année 1981. Nous avons donc consacré environ un an à l'élaboration du cahier des charges du CTR, alors même que les principes du système CORDIS et du programme CORDIS-LOGICIEL étaient partiellement remis en cause. Cette remise en cause a abouti à un aspect théorique plus solide et à une formulation plus générale et fondamentale des principes de CORDIS. C'est la raison pour laquelle le CTR possède des lacunes vis-à-vis du système CORDIS-LOGICIEL Version 2 (1981).

Tout au long de cette étude, nous avons confronté un ensemble de choix de fonctions à des moyens techniques s'articulant autour des pôles : technologie, représentation des données, représentation des instructions, rapidité, encombrement, consommation etc.. Nous avons procédé par étapes, en allégeant une fonction à chaque fois que sa réalisation paraissait introduire une complexité non souhaitable, mais uniquement dans le cas où cette fonction réduite restait compatible avec les objectifs de base du CTR. Nous allons maintenant expliciter ces objectifs (texte marqué en gras dans les paragraphes suivants) et commenter les fonctions qui s'en déduisent. Nous ne présentons pas ici l'étude technique qui a été faite pour déterminer les moyens à mettre en oeuvre pour la réalisation de ces fonctions. Cette étude sera présentée dans la deuxième partie de ce document à propos de la technologie adoptée, du format de calcul choisi etc..

Objectif 1 :

Le CTR doit être un processeur rapide pour la simulation en temps réel de mécanismes instrumentaux représentés dans le système CORDIS.

Il doit être conçu comme un périphérique du mini-ordinateur LSI-11, piloté par celui-ci qui se décharge d'une partie de ses fonctions de simulation sur le CTR. Cette partie des fonctions, précédemment attribuées au LSI dans le cadre du programme CORDIS-LOGICIEL, concerne surtout la simulation de mécanismes vibratoires rapides (appelés structures vibrantes) dont les déplacements engendrent le signal acoustique. Le partage des fonctions de simulation entre les deux calculateurs ne se fait pas de manière dynamique. Il est déterminé pour tirer la meilleure partie des capacités de chacun des calculateurs. L'architecture du CTR et son premier jeu d'instructions (celui-ci peut être étendu ultérieurement) sont conçus de manière à pouvoir simuler avant tout des structures vibrantes. Les mécanismes d'excitation sont simulés par le LSI. La raison en est simple : le programme existant CORDIS-LOGICIEL permet de simuler déjà en temps réel des mécanismes d'excitation qui sont des mécanismes aux mouvements lents. D'une manière générale les mécanismes d'excitation peuvent être simulés en temps réel par un calculateur relativement lent jusqu'à un certain degré de complexité de structure. La liaison excitateur - structure vibrante est représentée dans le CTR par un ou

plusieurs points particuliers de l'excitateur qui entrent (de manière instantanée ou permanente) en liaison avec la structure vibrante au moyen de liaisons conditionnelles.

Objectif 2

Le langage de programmation du CTR et son jeu d'instructions doivent être entièrement compatibles avec le langage CORDIS.

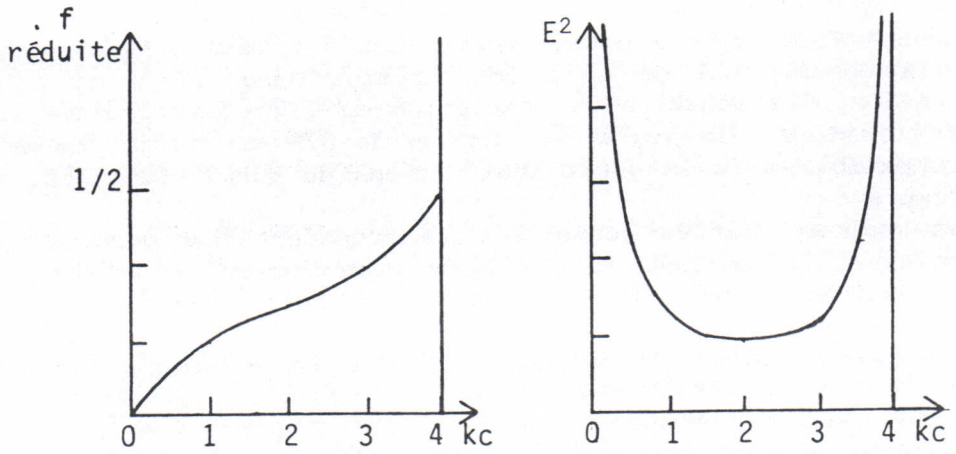
A terme, il s'agit de construire un seul et même programme CORDIS qui permettra la description de tout l'instrument dans une même séquence de prestructuration. La fonction de simulation du CTR étant limitée aux seules structures vibrantes, le jeu d'instruction du CTR est défini comme un sous-ensemble compatible avec le jeu d'instructions du CORDIS-LOGICIEL. Ce sous-ensemble comprend :

- des modules d'accès permettant le transfert des données gestuelles (excitation et modification). Ces modules se comportent comme les images des points de l'excitateur qui entrent en contact avec la structure vibrante.
- des modules de structure servant d'une part à la simulation de la structure vibrante seule et d'autre part à la simulation des liaisons conditionnelles entre l'excitateur et la structure vibrante. Nous avons vu, en présentant les modules de base du système CORDIS que les modules cellule et liaison integrée étaient plus adaptés à la simulation des seules structures vibrantes. Dans le CTR, ce sont ces deux modules qui seront utilisés comme éléments de base. Les modules de liaison conditionnelle sont quant à eux de trois types : unilatéral, échappement, condition extérieure. Ces trois types de modules proviennent directement du CORDIS-LOGICIEL version 1. Leur définition est antérieure à la formulation générale des modules de liaison conditionnelle comme elle apparaît dans CORDIS-LOGICIEL version 2.
- des modules de sortie sonore qui permettent le transfert des vibrations acoustiques vers les convertisseurs numérique/analogique de sortie.

Nous remarquons quelques simplifications par rapport au programme CORDIS-LOGICIEL :

- Les modules d'observation visuelle n'existent pas car ce qui nous intéresse de la structure vibrante dans un premier temps, c'est sa vibration acoustique.
- Les modules de contrôle rapide de paramètres qui permettent de contrôler, à la cadence du calcul rapide, l'évolution des paramètres mécaniques de la structure par des variables internes, n'existent pas car ils ne nous semblent pas nécessaires pour une première phase d'expérimentation des structures vibrantes. Par ailleurs leur présence augmente le nombre de calculs à faire dans une boucle rapide de calcul.
- Les possibilités de changement de structure en cours de jeu n'ont pas été retenues pour le CTR car elles nécessiteraient des branchements conditionnels au niveau du programme selon l'état d'un ou de plusieurs indicateurs provenant du LSI. De même que précédemment, nous avons écarté cette fonction qui alourdit l'architecture du séquenceur de programme.

FIGURE 11.



$$x(n) = f_e(n) + (2 - kc)\, x(n-1) - x(n-2)$$

$0 < kc < 4$ la réponse impulsionnelle est :

$$x_0(n) = \frac{f_0}{\sin\alpha} \sin(n+1)\alpha$$

$$\alpha = 2\pi f_r = \text{Arc cos } \frac{2 - kc}{2}, \quad f = \frac{F_{ech}}{2\pi} \text{ Arc cos } \frac{2 - kc}{2}$$

$$E^2 = \frac{1}{\sin^2\alpha} = \frac{4}{kc(4 - kc)}$$

Objectif 3

La fréquence d'échantillonnage du signal acoustique doit être au moins égale à 10 kHz, pour avoir une bande passante de 5 kHz.

En fait l'étude du modèle de la cellule montre que la cohérence avec le modèle physique continu demeure jusqu'à une certaine valeur du paramètre mécanique K (**Figure 11**). Au-delà de cette valeur l'amplitude de l'oscillation augmente avec sa fréquence. A cette valeur de K correspond la fréquence d'oscillation Fe/4, si Fe est la fréquence d'échantillonnage. Donc avec le

modèle d'une cellule simple, il faut avoir une fréquence d'échantillonnage égale à 20 kHz pour avoir un comportement cohérent jusqu'à 5 kHz de fréquence d'oscillation. Dans le cas où nous voudrions élargir cette limite jusqu'à 10 kHz, il nous faudrait échantillonner à 40 kHz, ce qui serait prohibitif pour la simulation en temps réel de nos modèles. Nous avons choisi une fréquence d'échantillonnage de 25,6 kHz pour le CTR; en effet c'est la première valeur qui dépasse la limite inférieure de 20 kHz en donnant un rapport (de 256) qui est une puissance entière de deux par rapport à la fréquence du calcul lent (100 Hz). En effet pour la synchronisation des deux boucles de calcul, ce rapport est particulièrement facile à mettre en oeuvre. Dans ce cas, le domaine de cohérence avec le modèle physique continu se situe entre 0 et 6,4 kHz. Mais tout en sachant qu'au-delà de cette limite l'amplitude de la vibration augmente avec sa fréquence, nous pouvons quand même utiliser le modèle pour la synthèse sonore.

La fréquence d'échantillonnage standard étant choisie égale à 25,6 kHz, l'étude de nos moyens techniques montre que nous serons assez vite limités dans la simulation de modèles complexes. Dans ce cas, il est intéressant d'abaisser la fréquence d'échantillonnage. Le CTR prend en compte cet aspect puisque nous disposons au total de sept fréquences d'échantillonnage s'échelonnant entre 6,4 kHz et 25,6 kHz. Ces fréquences sont : 6,4 kHz ; 8,5 kHz ; 12,8 kHz ; 14,62 kHz ; 17,07 kHz ; 20,48 kHz ; 25,6 kHz.

Le CTR reste uniquement un organe de calcul en temps réel car il ne dispose pas de moyens de mémorisation des échantillons calculés. Nous avons fait ce choix dans un premier temps, car le programme CORDIS-LOGICIEL peut prendre en charge la simulation en temps différé de structures vibrantes. Mais un organe de calcul rapide pourrait être très intéressant pour faire du calcul en temps différé beaucoup plus rapidement que ne peut faire le LSI. Pour un deuxième prototype, nous pensons qu'il serait intéressant d'introduire cette fonction.

Objectif 4

Le système d'algorithmes du CTR doit rester de même nature que le système d'algorithmes du CORDIS-LOGICIEL.

Ces algorithmes forment un sous-ensemble du système d'algorithmes de CORDIS-LOGICIEL. L'architecture du CTR est avant tout adaptée à ces algorithmes. La principale conséquence est la rapidité du calcul. Par rapport au système d'algorithmes du programme CORDIS-LOGICIEL nous opérons certaines modifications . Elles concernent essentiellement le nombre des algorithmes et la prise en charge du calcul concernant les fonctions gestuelles par le LSI. Comme nous l'avons déjà signalé à propos du langage, les modules de base du CTR sont la cellule et la liaison. Ces modules sont suffisants pour la simulation de structures vibrantes.

A titre d'exemple, les modules matériels dans le programme CORDIS-LOGICIEL SONT :

- la masse. L'algorithme correspondant est :

$$X(n) = (F(n)/M) + 2X(n-1) - X(n-2) \quad ; \qquad (1)$$

- la cellule unitaire. L'algorithme correspondant est :

$$X(n) = F(n) + (2-K-Z).X(n-1) + (Z-1).X(n-2) + K.L \quad ; \quad (2)$$

- la cellule. L'algorithme correspondant est :

$$X(n) = (F(n)/M) + (2-((K+Z)/M)).X(n-1) + ((Z/M)-1).X(n-2) + K.L/M \ .(3)$$

Dans tous ces algorithmes X(n), X(n-1), X(n-2) sont les trois dernières valeurs de la variable de déplacement du point matériel, F(n) est la dernière valeur de la variable de force externe au système, M est la masse du point matériel qui est nulle dans le cas de la cellule unitaire, K est la constante de raideur du ressort, Z est le coefficient de frottement, L est la longueur du ressort au repos.

Dans le CTR, les modules matériels sont réduits à un seul qui est la cellule unitaire. L'algorithme correspondant est :

$$X(n) = F(n) + A.X(n-1) + B.X(n-2) + C \ .$$

Comme nous le constatons il présente moins de calculs que l'équation (2). Les valeurs des paramètres A, B, C sont calculées dans le LSI. Les algorithmes correspondant sont :

$$A = 2 - K - Z$$

$$B = Z - 1$$

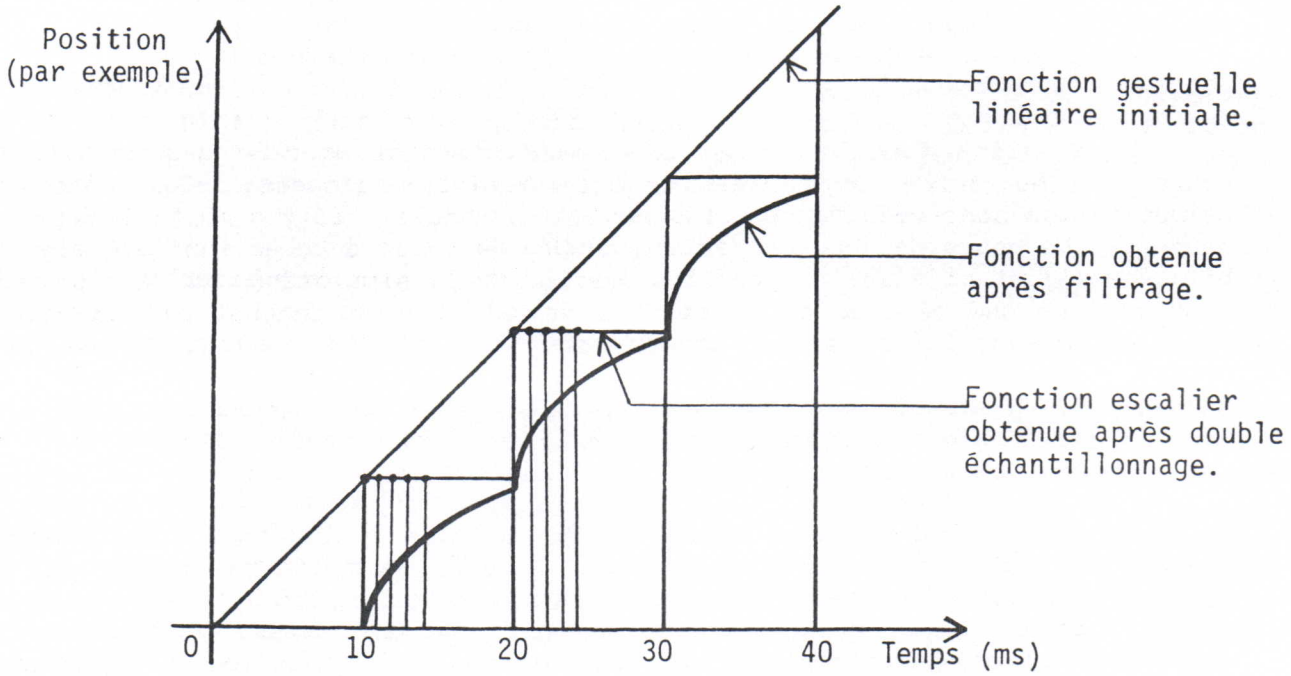
$$C = K.L \ .$$

Le fait de disposer uniquement de cellules unitaires réduit la généralité de la simulation. Néanmoins la pratique avec le programme CORDIS-LOGICIEL montre que des masses non unitaires (selon le système interne de référence) sont rarement utilisées pour la simulation de structures vibrantes.

En ce qui concerne les modules de liaison, nous faisons aussi le même type de modification : réduction en nombre, prise en charge du calcul des paramètres par le LSI.

Comme conséquence de ces modifications, nous obtenons pour les modules internes de structure, des algorithmes simples et homogènes. Il apparaît notamment qu'aucun des algorithmes du CTR ne fait intervenir de division rapide, ce qui évite la construction d'une unité de division rapide.

FIGURE 12.



Objectif 5

La liaison entre les deux calculateurs doit pouvoir assumer plusieurs fonctions :

- le chargement de toutes les données structurelles concernant une simulation,

- la communication bilatérale en cours de jeu,

- le contrôle du fonctionnement du CTR,

- la possibilité d'accéder à toutes les données internes au CTR pour la mise en route et le dépannage de celui-ci.

La première fonction se fait en temps différé, entre la phase de prestructuration et la phase de jeu. Le temps de transfert est un temps mort pour l'utilisateur, aussi nous avons intérêt à concevoir un protocole de transfert qui soit le plus rapide possible. Néanmoins cette vitesse est limitée par le LSI. Le transfert se fait par un accès direct mémoire du LSI dans toutes les mémoires du CTR. Les données peuvent être transférées en bloc. A cette fin, un système d'incrémentation automatique pour les adresses internes du CTR est prévu sur l'interface.

Dans la pratique, pour des structures d'une complexité moyenne, le transfert se fait suffisamment rapidement pour que l'utilisateur ne se rende pas compte du temps mort. Ainsi après la phase de prestructuration, la commande qui permet de basculer en mode de jeu peut demander le chargement des données structurelles et le démarrage du calcul. Nous avons déjà signalé que le CTR ne prévoyait pas les changements de structure qualitative en cours de jeu. Cette carence peut être comblée en partie par la préparation préalable de plusieurs structures qui pourront être chargées l'une après l'autre mais très rapidement, par une commande simple. Cette notion appelée "registration" a été élaborée par P. LACORNERIE qui a travaillé sur le premier logiciel pour le CTR.

La deuxième fonction exige la communication bilatérale des données en cours de jeu selon un certain nombre de voies d'accès. Ces voies de communication sont réduites physiquement à une seule. Il y a multiplexage des données. En cours de jeu, un certain nombre de voies d'accès (unilatérales ou bilatérales) étant établies pour les besoins de la simulation, il faut pouvoir assurer dans une période de 10 ms, un transfert pour chacune de ces voies. Nous avons souhaité avoir au départ une multiplicité de huit voies dans chaque sens. Notons cependant que la représentation des données sur 16 bits dans le LSI et sur 32 bits dans le CTR, nécessite deux transferts physiques pour chaque donnée significative pour le CTR. Malgré cela la période de 10 ms est largement suffisante pour huit voies dans chaque sens.

Une autre limitation intervient cependant : une partie des données transférées en cours de jeu du LSI vers le CTR, est filtrée par un filtre passe-bas du second ordre (**Figure 12**). Ces données correspondent aux fonctions d'excitation. En effet, si nous considérons une fonction d'excitation linéaire, comme celle montrée sur la figure, cette fonction devient après échantillonnage à 100 Hz d'abord, et à 25600 Hz ensuite, une fonction en escalier pour le calcul rapide. Elle est formée d'une suite d'échelons, qui produit une excitation périodique de la structure vibrante, qui ne correspond pas à la fonction gestuelle linéaire initiale. Comme nous pouvons le voir sur la figure, nous obtenons après filtrage, une fonction lissée. La nécessité d'un tel filtrage a été mis en évidence, par les simulations en temps différé, faites avec le LSI. Les coefficients du filtre ont été ajustés de manière pratique, par l'observation de la fonction gestuelle initiale et l'observation de la fonction échantillonnée et filtrée. Nous donnerons les caractéristiques du filtre, dans la deuxième partie du document, à propos du processeur d'entrée-sortie.

Matériellement il y a un seul filtre travaillant de manière séquentielle sur toutes les voies. Le filtre que nous avons réalisé ne permet pas de traiter plus de six voies à la fréquence de 25,6 kHz. Par ailleurs un tel filtre du second ordre, fonctionnant sur des données de 32 bits, et son système de séquencement regroupent une quarantaine de circuits integrés. Nous avons préféré abaisser le nombre de voies d'entrée plutôt que de réaliser deux filtres en parallèle. L'utilisation des voies d'accès dans le sens CTR - LSI correspond uniquement à des fonctions d'excitation. Or les structures vibrantes que nous pouvons simuler en temps réel dans le CTR sont des structures relativement simples. Nous avons pensé que deux fonctions d'excitation sont suffisantes. Ainsi les voies de communication entre le LSI et le CTR en cours de jeu, se réduisent au total à six voies dans le sens LSI - CTR, et à deux voies dans le sens CTR - LSI.

Le contrôle du fonctionnement du CTR par le LSI implique que certaines commandes de type démarrage, arrêt, redémarrage etc. soient prévues au niveau du logiciel. Le CTR ne possède en effet aucune commande manuelle de ce type. Dans la mesure où l'utilisation du CTR implique un dialogue préalable entre les deux calculateurs, il est normal que ces commandes soient gérées au niveau logiciel. Nous avons prévu à cet effet un registre de commande et d'état sur l'interface. Ainsi la gestion de ces commandes équivaut à des transferts de données particulières dans ce registre.

Enfin nous avons prévu un certain nombre de facilités telles que l'exécution d'un cycle de calcul unique, pour la mise en route et le démarrage de la machine ; mais ces dispositions sont difficiles à définir dès le départ quand il s'agit de la réalisation d'un premier prototype. L'expérience a montré qu'elles sont insuffisantes.

Objectif 6

Le produit sonore, résultat de la synthèse, doit répondre à des exigences de qualité.

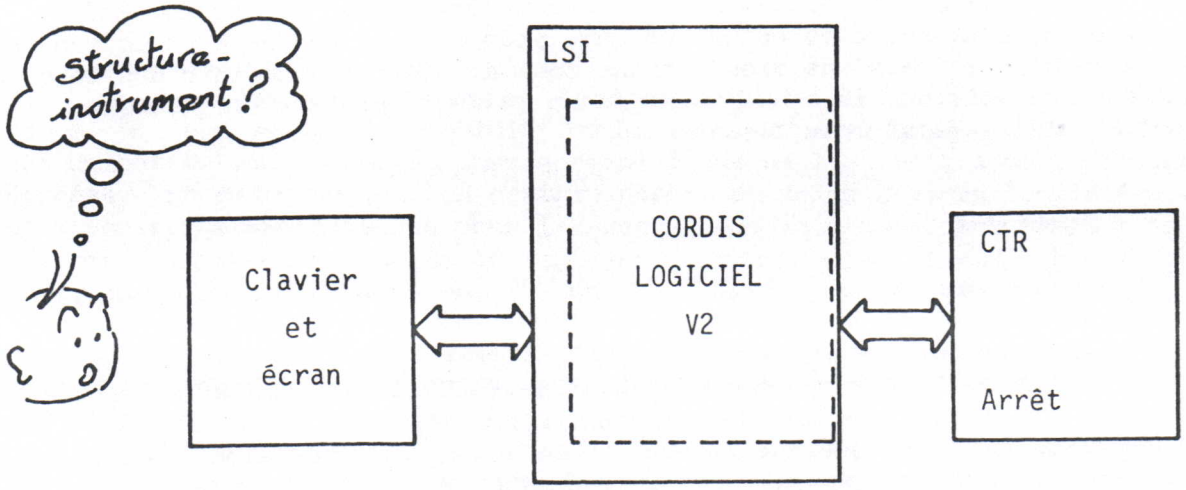
Que ce soit avec le programme CORDIS-LOGICIEL en temps différé, ou avec le CTR en temps réel, les échantillons sonores sont envoyés à des convertisseurs numérique/analogique de 14 bits, soigneusement réalisés pour avoir un niveau signal/bruit véritablement égal à 84 DB. La vibration acoustique (correspondant aux déplacements rapides des points matériels distincts de la structure vibrante) doit être calculé avec suffisamment de précision, compte tenu du cumul des erreurs dans un calcul de type récursif, pour que le niveau signal/bruit soit au moins égal à 84 DB.

Une étude sur la représentation des données dans le CTR, nous a permis de satisfaire cette exigence et de répondre à deux autres critères. Le premier est une contrainte dynamique : c'est celle de pouvoir représenter des distances de l'ordre de dix à vingt centimètres, dans le même format que la vibration acoustique, qui a une amplitude maximale de l'ordre du millimètre. Elles pourraient correspondre à des distances réalistes entre l'excitateur et la structure vibrante, ou entre plusieurs structures vibrantes séparées, représentées simultanément. Le deuxième critère est celui de la représentation précise des coefficients mécaniques de la structure, dans toute leur gamme de variation.

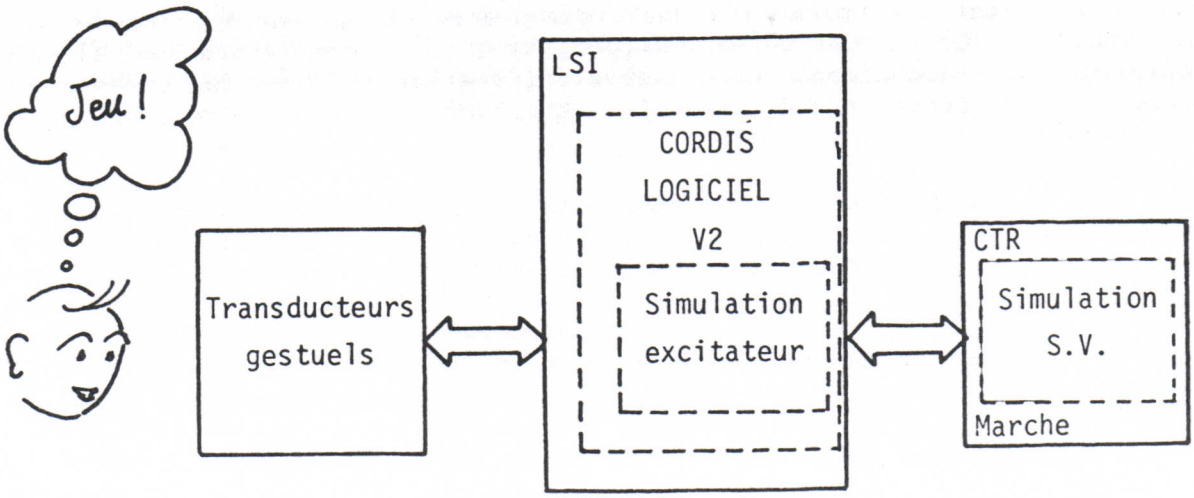
Nous avons pris en considération ces trois critères dans notre étude sur la représentation des données dans le CTR. Cette étude sera présentée dans la deuxième partie de ce document.

Les sons, résultats de la synthèse en temps différé ou en temps réel, sont dans tous les cas envoyés sur deux voies de sortie sonore. Dans une optique expérimentale et non de production d'oeuvres musicales, deux voies nous paraissent suffisantes.

FIGURE 13.



Phase PSQL + PSQN



Phase de Jeu

DEUXIÈME PARTIE : LE CALCULATEUR SPECIALISE CORDIS TEMPS REEL

A/ PRELIMINAIRES

1. Généralités

Le calculateur spécialisé CTR est conçu pour travailler comme un périphérique du mini-ordinateur LSI 11. Dans la simulation de l'instrument, il prend en charge uniquement la structure vibrante et une partie de la liaison excitateur - structure vibrante. A cette fin, parmi l'ensemble des modules du système CORDIS nous avons choisi un nombre restreint de modules pour le CTR.

Nous insistons sur le fait que l'utilisateur du système CORDIS ne doit pas être gêné par l'existence de deux calculateurs (le LSI et le CTR). Il peut même l'ignorer. Mais il doit pouvoir analyser "l'instrument" qu'il veut simuler en termes mécaniques, définir l'excitateur et la structure vibrante ainsi que la liaison entre les deux, définir les points de la structure où s'effectuent les gestes d'excitation et les gestes de modification, les points d'écoute, les points d'observation visuelle etc..

Après l'analyse vient la description dans le langage CORDIS (**Figure 13**). Pendant la phase de prestructuration qualitative (PSQL), l'utilisateur doit décrire son instrument qualitativement. Le programme CORDIS-LOGICIEL V2 (C. CADOZ) est un programme interactif qui le guidera dans cette description. Il lui donnera des indications sur la démarche à suivre, il lui indiquera des fautes de langage etc.. Après la description qualitative, l'utilisateur doit attribuer des valeurs à tous les paramètres et définir l'état initial de la structure. C'est la phase de prestructuration quantitative (PSQN). Il pourra s'attarder aussi longtemps qu'il veut dans ces deux phases, passer d'un mode à l'autre, modifier des paramètres ; il pourra préparer plusieurs structures quantitatives pour une même structure qualitative. Les critères dans cette première phase sont la cohérence du langage, la facilité d'apprentissage, la souplesse d'utilisation. A la fin de la phase de prestructuration le programme CORDIS-LOGICIEL se charge de préparer deux programmes exécutables l'un par le mini-ordinateur, l'autre par le CTR. Le programme exécutable par le CTR peut

être chargé dans la machine à l'issue de cette phase. Mais la vitesse de transfert étant très grande, il pourra être chargé aussi au moment du passage en mode de jeu, sans que ce soit perceptible par l'utilisateur.

En cours de jeu, celui-ci manipule les différents transducteurs gestuels, rétroactifs ou non. Les données d'origine gestuelle sont traitées par le programme CORDIS-LOGICIEL en temps réel et une partie des informations est envoyée dans le CTR au moyen des voies d'entrée-sortie lente, pour l'excitation et le contrôle des paramètres de la structure vibrante.

Comme nous l'avons déjà signalé, la version actuelle du logiciel d'exploitation du système CORDIS, c'est-à-dire le programme CORDIS-LOGICIEL V2, ne prend pas en charge la description de la structure vibrante pour piloter le CTR. Aussi il a été préparé un programme spécifique (par P.LACORNERIE) pour les premiers essais de fonctionnement du CTR. Ce programme garde un aspect interactif, mais la description de la structure se fait uniquement avec certains éléments parmi l'ensemble des éléments de langage prévus pour le CTR. Ceci a été fait pour utiliser le CTR tout de suite pour une expérimentation sur des structures simples et des gestes d'excitation très simples (percussion, pincement), sans mécanisme excitateur intermédiaire. La réalisation du logiciel complet pour le CTR est en cours.

2. Liste des instructions. Algorithmes

La liste des instructions du CTR contient 22 macroinstructions. Elles appartiennent à différentes familles.

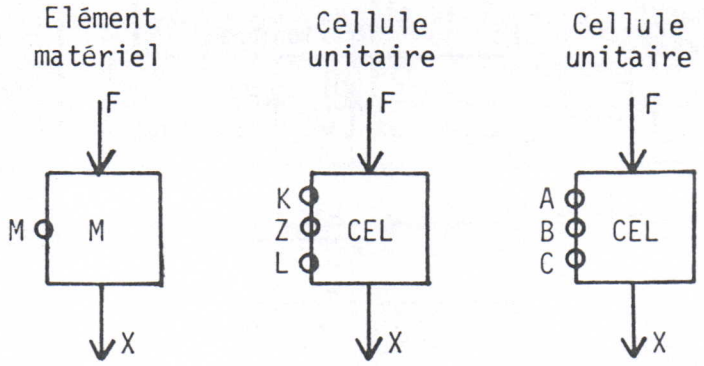
2.1. Les instructions correspondant à des modules internes

Nous appelons **module interne** tout module servant à définir la structure, excluant les points d'entrée-sortie gestuelle et les points de sortie sonore. Les modules internes sont des **modules matériels** ou des **modules de liaison.** Pour un module matériel, le champ de paramètres de l'instruction ne comporte que le numéro propre du module matériel. En effet pour éviter toute redondance dans la description d'une structure nous avons adopté la convention suivante : un module matériel comporte un numéro et à ce numéro correspondent autant de cases mémoires que le module possède de variables et de paramètres. Toutes ces cases mémoires "sont attribuées" au module matériel. Un module de liaison comporte aussi un numéro. A ce numéro correspondent autant de cases mémoires que le module possède de variables intermédiaires et de paramètres. Les variables intermédiaires ne sont ni des variables d'entrée, ni des variables de sortie. Pour un module de liaison, le champ de paramètres de l'instruction contient deux autres numéros, qui sont les numéros des deux modules matériels reliés au module de liaison. Le module de liaison utilise pour ses variables d'entrée et de sortie les cases mémoires attribuées aux modules matériels.

2.1.1. Modules matériels

La variable d'entrée est une force, la variable de sortie est une position (**Figure 14**). Un même module matériel peut être lié à plusieurs modules de liaison. Dans ce cas toutes les forces d'entrée exercées sur le module matériel sont additionnées. La position de sortie constitue une entrée pour chacun de ces modules de liaison.

FIGURE 14.



O Paramètres

Dans le CTR, les modules matériels ont un seul représentant : **la cellule unitaire.** Elle est formée par un élément de masse unitaire, un élément de raideur et un élément de frottement. L'instruction correspondante est **CEL i** (i est le numéro de la cellule). L'équation correspondante est :

$$Xi(n) = Fi(n) + Ai.Xi(n-1) + Bi.Xi(n-2) + Ci \quad . \qquad (1)$$

Ai, Bi, Ci sont les paramètres de calcul. Leurs variations, qui correspondent à un geste de modification, sont lentes. Ils sont calculés à partir de Ki, Zi, Li qui sont les paramètres utilisateur dans le LSI.

$$Ai = 2 - Ki - Zi$$
$$Bi = Zi - 1$$
$$Ci = Ki.Li \quad .$$

Le calcul correspondant à l'équation (1) doit être suivi immédiatement par :

$$Fi(n)=0 \quad .$$

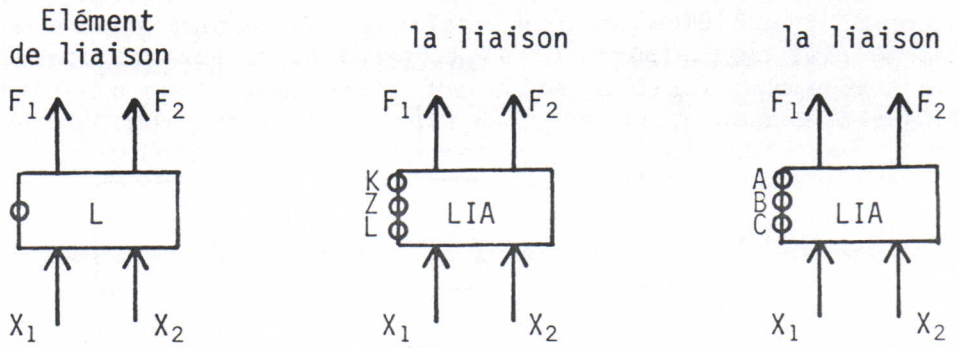
En effet la case mémoire où est rangée l'information de force pour chacune des cellules, est utilisée comme accumulateur par les modules de liaison. Chaque module de liaison, lié à la cellule N° i, rajoute à la fin de son calcul la force de sortie Fi, à la valeur qui se trouve dans la case mémoire d'entrée de la cellule N° i. Ceci a comme corollaire que les calculs des différents modules matériels et des différents modules de liaison d'une structure ne peuvent pas être intercalés.

2.1.2. Modules de liaison

Ils sont **simples** ou **conditionnels** (**Figure 15**).

Un module de liaison est toujours lié à deux modules matériels qui se trouvent de part et d'autre de la liaison. Les forces exercées par le module de liaison sur chacun d'eux sont égales et opposées. L'égalité F1 = - F2 est toujours vérifiée.

FIGURE 15.



O Paramètres

LIAISON SIMPLE

Les modules de liaison simple ont un seul représentant qui est **la liaison.** Il est formé par un élément de raideur et un élément de frottement.

L'instruction correspondante est : **LIA i,j,k ;**

i est le numéro de la liaison,
j est le numéro de la première cellule,
k est le numéro de la deuxième cellule.

Les équations correspondantes sont :

$$Fk(n) = Ai.((Xj-Xk)(n)) + Bi.((Xj-Xk)(n-1)) + Ci \qquad (2)$$
$$Fj(n) = - Fk(n) \quad .$$

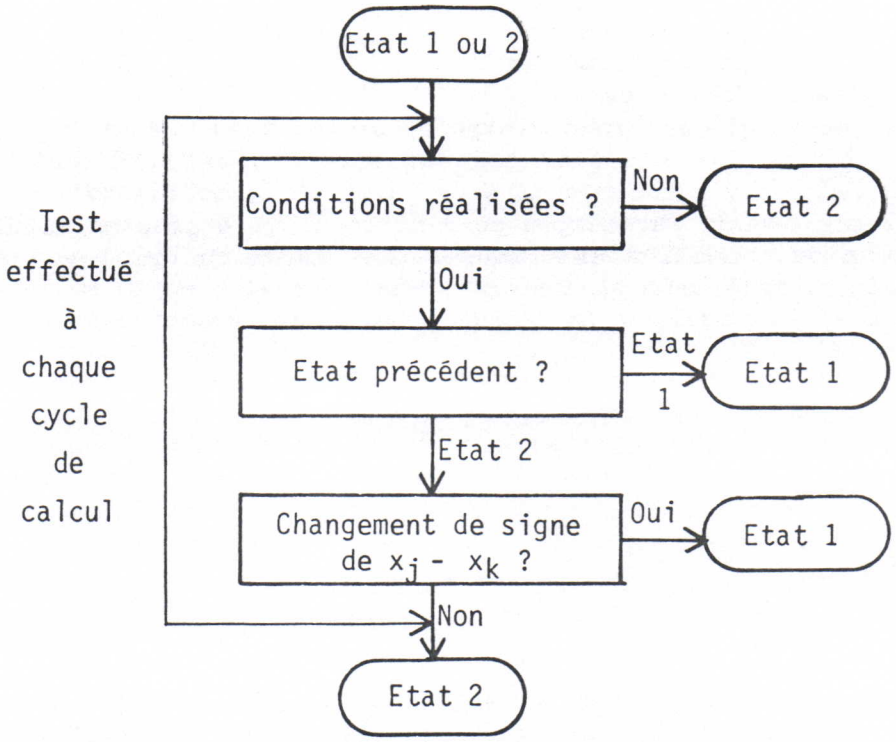
Les paramètres utilisateur sont Ki, Zi, Li. Les paramètres de calcul sont: Ai, Bi, Ci.

$$Ai = Ki + Zi$$
$$Bi = - Zi$$
$$Ci = Ki.Li \quad .$$

La valeur Fk(n) calculée par l'instruction LIA est rajoutée à la valeur contenue dans la case d'entrée de la cellule N° k. La valeur Fj(n) est rajoutée à celle de la cellule N° j.

LIAISON CONDITIONNELLE

FIGURE 16.



La liaison conditionnelle est une liaison simple ressort - frottement pouvant prendre deux états différents (caractérisés par des valeurs différentes des paramètres K et Z), selon qu'une condition est réalisée ou non. L'instruction correspondante est **LIAC i,j,k.** Le jeu d'équations de la liaison simple se dédouble dans ce cas :

dans l'état 1

$$Fk(n) = A1i.((Xj-Xk)(n)) + B1i.((Xj-Xk)(n-1)) + C1i \quad (3)$$
$$Fj(n) = - Fk(n) ;$$

dans l'état 2

$$Fk(n) = A2i.((Xj-Xk)(n)) + B2i.((Xj-Xk)(n-1)) + C2i \quad (4)$$
$$Fj(n) = - Fk(n) \quad .$$

L'état 1 est caractérisé par une condition vraie, l'état 2 par une condition fausse. A un instant n, l'état de la liaison dépend aussi de l'état précédent et de l'évolution des entrées Xj et Xk l'une par rapport à l'autre aux instants n et n-1. L'organigramme qui permet de déterminer l'état d'une liaison conditionnelle est tracé sur la **figure 16.**

Les modules de liaison conditionnelle sont de trois types : liaison unilatérale **UNL** ; liaison à échappement **ECH** ; liaison dont l'état dépend d'une condition extérieure non reliée aux positions des entrées Xj et Xk, **CDEX**.

UNL a deux paramètres de condition : L et L'.
La condition est :

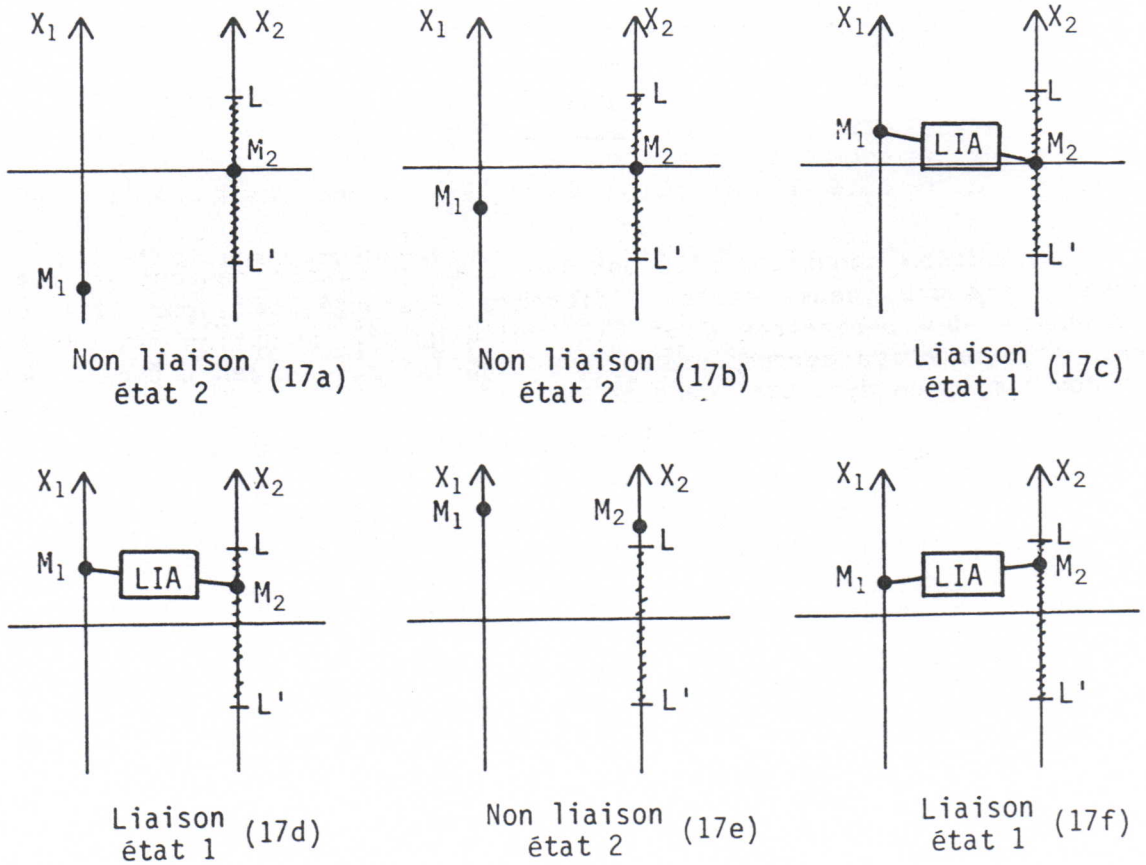
$$Xk - L' \leqslant Xj \leqslant Xk + L \quad .$$

ECH a aussi deux paramètres : L et L'.
La condition est :

$$L' \leqslant Xj \leqslant L \quad .$$

CDEX n'a pas de paramètres de condition. L'état d'un module CDEX dépend d'une touche de condition extérieure. Le nombre de touches différentes peut atteindre 8.

FIGURE 17.



Non liaison état 2 (17a)

Non liaison état 2 (17b)

Liaison état 1 (17c)

Liaison état 1 (17d)

Non liaison état 2 (17e)

Liaison état 1 (17f)

Nous allons maintenant donner un exemple d'utilisation du module ECH (échappement), dans le cas de la simulation "d'un pincement de cordes". Le pincement est simulé :

a/ en saisissant la corde "discrète" en un de ses points matériels,
b/ en l'éloignant de sa position de repos jusqu'à un seuil déterminé,
c/ en lâchant la corde au-delà de ce seuil,
d/ en la laissant vibrer librement.

L'état 1 sera "l'état de liaison", avec des valeurs déterminées pour les paramètres A, B, C. L'état 2 sera l'état de non liaison avec A = B = C = 0

Soient M1 et M2, deux points matériels dont les déplacements au cours du temps sur l'axe OX, sont donnés par les variables X1 et X2. Le point M1 appartient à l'excitateur, le point M2 à la structure vibrante. La **figure 17a** montre l'état initial du système. Sur la **figure 17b** nous voyons que le point M1 s'est rapproché du point M2 toujours au repos. Tant que la variable X2-X1 ne change pas de signe, le système reste dans le même état (non liaison). Sur la **figure 17c** nous voyons que le point M1 vient de dépasser le point M2; par ailleurs la variable X2 remplit la condition $L' \leq X2 \leq L$. Alors le système passe dans l'état 1 (liaison). Le point M2 sera entraîné par le point M1. Sur la **figure 17d** les deux points se sont déplacés, mais ils sont toujours en liaison tant que l'inégalité sur X2 est satisfaite. Sur la **figure 17e** la condition $L' \leq X2 \leq L$ n'est plus vérifiée, la liaison est alors rompue. Les points M1 et M2 continuent leurs évolutions indépendamment. Enfin sur la **figure 17f** nous voyons le réaccrochage, quand la variable X2-X1 change de signe.

2.2. Les instructions correspondant aux modules d'entrée-sortie lente

Ces modules se répartissent en deux groupes : les modules correspondant à un geste d'excitation, ceux correspondant à un geste de modification.

2.2.1. Modules correspondant à un geste d'excitation

Ils servent à simuler une partie de la liaison excitateur - structure vibrante. Ils sont au nombre de quatre **PL, PLF, FL, FLP.** La **figure 18** montre ces modules dans leur utilisation comme charnière entre le LSI et le CTR. Les modules PR, FR, PRF, FRP sont des modules duals faisant partie du système de simulation résidant dans le LSI.

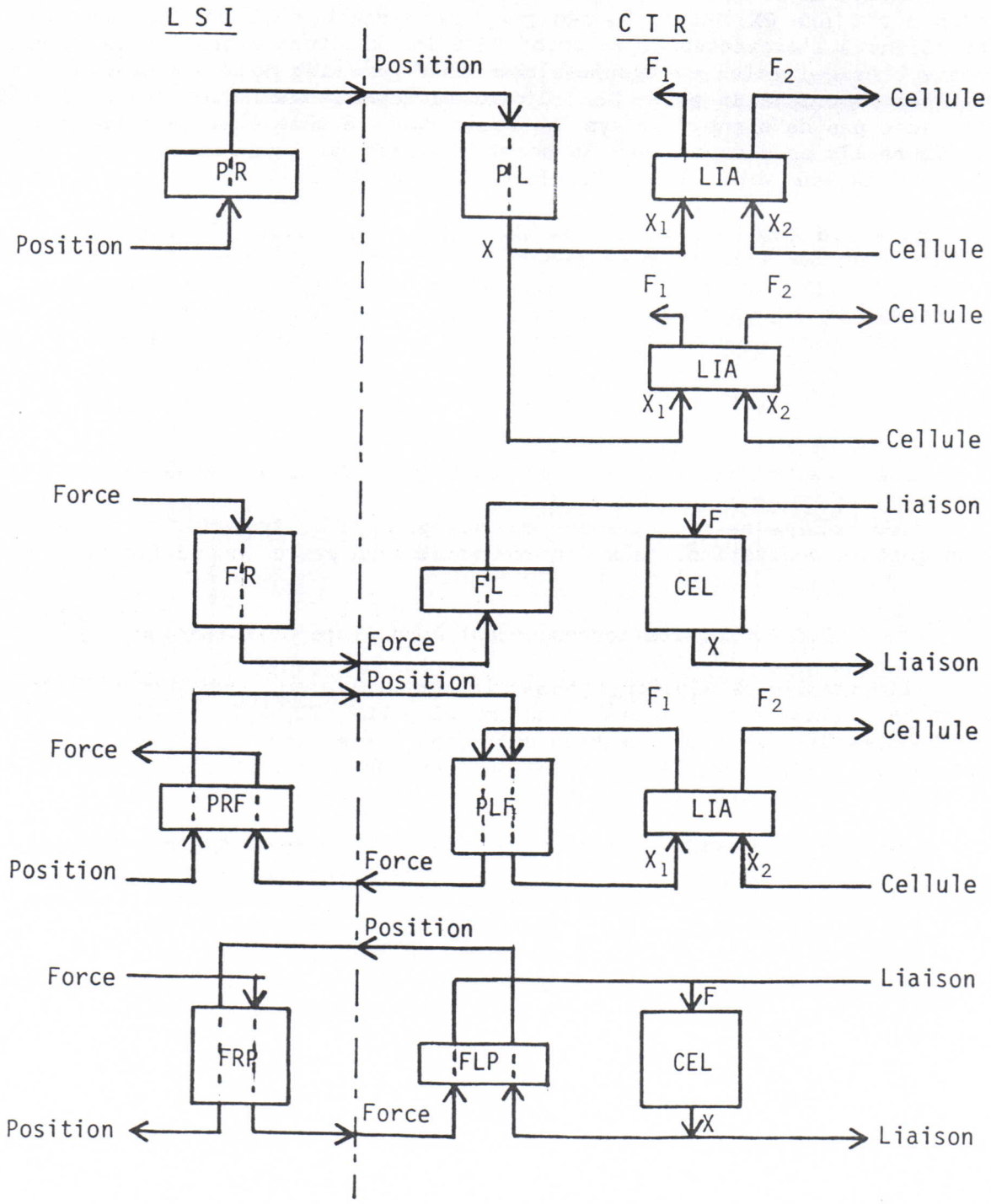
Les modules **PL** (Position Lente), **FL** (Force Lente) correspondent uniquement à des voies d'entrée, c'est-à-dire à des voies non rétroactives. Ce sont des modules qui transfèrent une information (correspondant à une position ou à une force) venant du LSI, vers les modules internes du CTR.

PL est un module de **type matériel**, car sa variable de sortie est une position. Il peut être lié à un ou plusieurs modules de liaison. Le calcul correspondant à :

PL i,p i : le numéro du module PL
        p : le numéro de la voie d'entrée

est un simple transfert de la donnée de position se trouvant sur la voie d'entrée p, dans une case mémoire correspondant au module PL N° i. Tout module de liaison, lié au module PL i,p , utilisera cette donnée de position exactement de la même manière que la sortie d'une cellule.

FIGURE 18.



FL est un module de **type liaison** car il a comme variable de sortie une force. Il peut être lié à une seule cellule. Le calcul correspondant à :

FL i,p i : numéro de la cellule à laquelle est lié FL
p : numéro de la voie d'entrée

est un transfert de la donnée de force se trouvant sur la voie d'entrée p, dans la case mémoire correspondant à l'entrée de la cellule N° i. Or cette case est utilisée comme accumulateur par les modules internes de liaison. Pour FL il en est de même. La donnée est ajoutée au contenu de cette case d'entrée, car la cellule N° i peut être liée par ailleurs à d'autres modules de liaison.

Les modules **PLF** (Position - Force Lente), et **FLP** (Force - Position Lente) correspondent à des voies d'entrée - sortie, c'est-à-dire à des voies rétroactives. **PLF** est un module de **type matériel** car la sortie est une position, l'entrée est une force. L'instruction correspondante est **PLF i,p,q** où i est le numéro du module PLF, p est le numéro de la voie d'entrée, q celui de la voie de sortie. Le calcul se déduit de celui des modules PL et FL.

FLP est un module de **type liaison** car la sortie est une force et l'entrée est une position. L'instruction correspondante est **FLP i,p,q** où i est le numéro de la cellule en liaison avec FLP et p et q ont la même signification que précédemment. Le calcul correspondant se déduit des modules PL et FL.

En résumé les modules PL, PLF, FL, FLP sont considérés par les modules internes du CTR, comme s'ils étaient des cellules ou des liaisons, mais leurs variables de sortie ne résultent pas d'un calcul interne. Ce sont des données externes au CTR, qui sont en général l'image d'un calcul ayant eu lieu dans le LSI.

2.2.2. Modules correspondant à un geste de modification

Ce sont des modules qui servent au contrôle lent de paramètres. L'instruction générique correspondant est **CLPX i,p.** X désigne le type de paramètre à contrôler, i est le numéro du module auquel appartient le paramètre et p le numéro de la voie d'entrée. Le calcul correspondant est un transfert de l'information se trouvant sur la voie d'entrée p dans la case mémoire correspondant au paramètre X du module N° i.

Il y a autant d'instructions CLP que de paramètres à contrôler :

CLPAC, CLPBC, CLPCC pour les paramètres de la cellule ;

CLPAL, CLPBL, CLPCL pour les paramètres de la liaison simple ;

CLPA1, CLPB1, CLPA2, CLPB2 pour les paramètres de la liaison conditionnelle dans les états 1 et 2.

Les paramètres C1 et C2 de la liaison conditionnelle sont nuls dans ce cas.

2.3. Les instructions correspondant aux modules de sortie sonore

La variable de sortie est soit une position, soit la somme de plusieurs positions. L'instruction générique est **SORX i.** X = 0 ou 1 indique la voie de sortie (deux voies possibles) . I est le numéro de la cellule dont on utilise la position comme variable de sortie. Pour chaque voie il y a un registre accumulateur. Quand les instructions de sortie ont toutes été activées, les contenus des registres accumulateurs sont envoyés aux convertisseurs numérique/analogique et les registres sont remis à zéro. Le calcul pour l'instruction SORX i, consiste à ajouter au contenu du registre de la voie X, le contenu de la case mémoire de sortie de la cellule N° i.

2.4. Les instructions utilitaires

Il y a une seule instruction utilitaire c'est l'instruction **FIN.** Elle doit figurer à la fin de chaque programme exécutable par le CTR. ELLe permet d'arrêter le calcul à chaque cycle d'échantillonnage quand toutes les instructions ont été décodées et exécutées.

3. La technologie

3.1. Considérations générales

Pour l'élaboration d'un calculateur, la technologie est un facteur très important :

- Le temps de propagation à travers les circuits impose la durée du cycle machine.
- La disponibilité de fonctions complexes réalisées en un seul boîtier (circuits de type LSI et VLSI) conditionne le détail de l'architecture.
- Les précautions spéciales de câblage pour garantir une meilleure immunité au bruit, augmentent le temps de réalisation et de mise en route de la machine.

Le choix est très délicat en raison de l'évolution rapide de la technologie. Ce choix peut paraître dépassé si un temps important s'écoule entre le moment du choix et le moment où la machine est disponible. En plus dans une technologie donnée, le degré d'intégration des circuits va en croissant. Des nouvelles fonctions apparaissent tous les ans (quelquefois même plus rapidement), mais quelquefois elles sont simplement annoncées par les fabricants et leur apparition sur le marché peut être retardée ou annulée.

La famille logique la plus rapide est la famille ECL (logique non saturée, à émetteurs couplés). La sous-famille ECL 100K a les caractéristiques suivantes :

Temps de propagation à travers une porte = 1 ns.
Puissance moyenne divisée par porte = 40 mW.

En contrepartie de la rapidité, nous avons une consommation très élevée. Des circuits imprimés spéciaux et des règles strictes de câblage doivent être utilisés. Actuellement des fabricants proposent des circuits très rapides qui utilisent intérieurement la technologie ECL, mais qui sont compatibles avec des circuits TTL sur leurs entrées-sorties.

En technologie TTL, des nouvelles familles sont apparues qui réalisent un compromis rapidité - consommation assez bon. Ce sont les familles TTL S (Schottky) , TTL LS (Low power Schottky). Les sous-familles TTL 25S et TTL 25LS et leur version rapide A permettent de réaliser des temps de cycles de l'ordre de 150 à 200 ns.

La technologie MOS est utilisée pour des circuits de forte intégration mais pas très rapides. Beaucoup de circuits MOS sont compatibles TTL pour leurs entrées-sorties.

3.2. Technologie adoptée

Nous avons fait notre choix en tenant compte des points suivants :

- la rapidité des calculs,
- l'importance d'une faible consommation,
- l'existence de boîtiers LSI et VLSI,
- l'absence de mesures spéciales (en dehors des précautions d'usage telles que la réalisation d'un plan de masse soigné, l'adaptation des lignes de communication au delà d'une certaine longueur etc.) dans la réalisation des circuits imprimés et du câblage,
- la possibilité de faire les tests et la mise en route avec les moyens dont nous disposons au laboratoire,
- l'existence d'autres réalisations au laboratoire dans la même technologie,
- l'existence de circuits standards pour minimiser le coût.

Toutes ces considérations nous ont fait opter pour la technologie TTL pour la grande majorité des circuits. Nous avons utilisé leurs versions les plus performantes 25S, 25LS, 25LSA.

En ce qui concerne les mémoires de programme et de données nous avons utilisé des boîtiers réalisés en technologie HMOS et compatibles TTL. Elles sont suffisamment rapides (temps d'accès maximal de 70ns). Les mémoires mortes de microprogramme sont des mémoires bipolaires très rapides, de même que les mémoires de faible taille utilisées dans la structure du filtre (temps d'accès maximal de 35ns).

Les quatre multiplieurs, qui multiplient chacun deux mots de 16 bits en parallèle, sont des circuits bipolaires. Au moment de notre choix ils représentaient les seuls circuits rapides disponibles pour cette fonction. Actuellement il existe des multiplieurs entièrement compatibles avec ceux-ci et deux fois plus rapides.

Nous avons obtenu un temps de cycle de 150 ns pour le calcul et 75 ns pour le filtre d'entrée.

FIGURE 19.

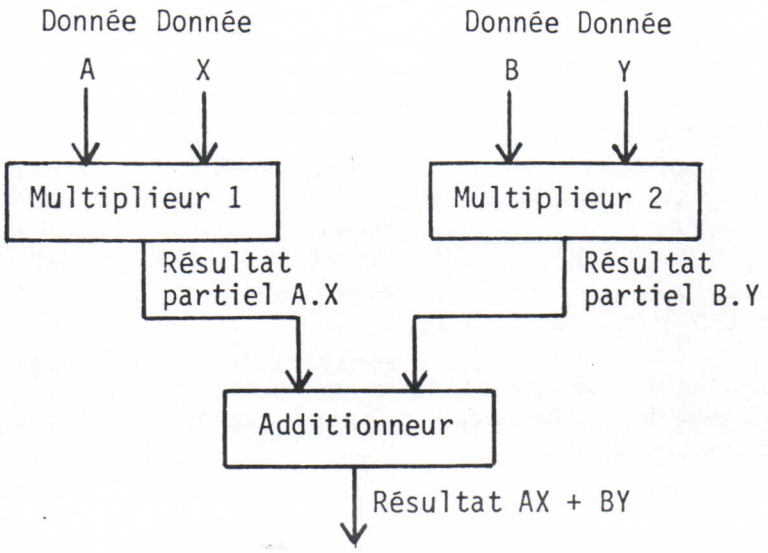
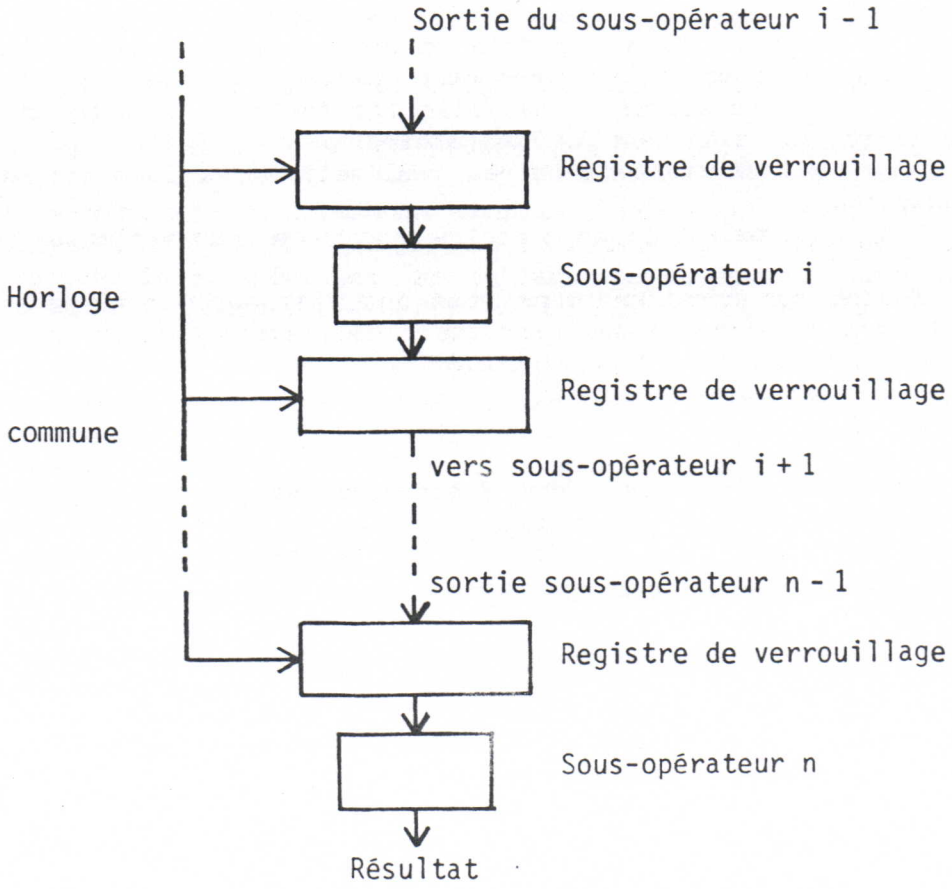


FIGURE 20.



4. Architecture des calculateurs

4.1. Considérations générales

Les petits calculateurs spécialisés (traitement d'images, traitement de la parole, transformée de Fourier discrète etc.) ont chacun une architecture qui est très fortement dépendante du type de calcul qui doit être effectué. Plus le calculateur est spécialisé, c'est-à-dire destiné au calcul d'un nombre restreint d'algorithmes, plus il est facile d'introduire des techniques qui vont accélérer le calcul. En contrepartie si avec ce même calculateur nous voulons calculer d'autres algorithmes, en général c'est possible. Mais cela demande un temps de calcul plus important qu'avec un calculateur moins spécialisé. Donc au moment où nous décidons de l'architecture, il est très important de savoir ce que nous voulons en faire, pour décider du degré de spécialisation du calculateur. Nous allons maintenant voir quelques techniques qui permettent d'améliorer les performances d'un petit calculateur.

4.1.1. Le parallélisme

Le parallélisme consiste à utiliser plusieurs opérateurs, fonctionnant en même temps et effectuant chacun une partie du calcul. Par exemple, dans un calcul du type AX + BY où tous les nombres sont des réels, nous pourrions utiliser deux multiplieurs identiques pour effectuer simultanément les calculs des produits AX et BY (**Figure 19**). Ainsi le résultat est obtenu deux fois plus rapidement. Mais cette technique est désavantageuse en coût et en place, car le nombre de circuits utilisés augmente rapidement.

4.1.2. La séparation des mémoires

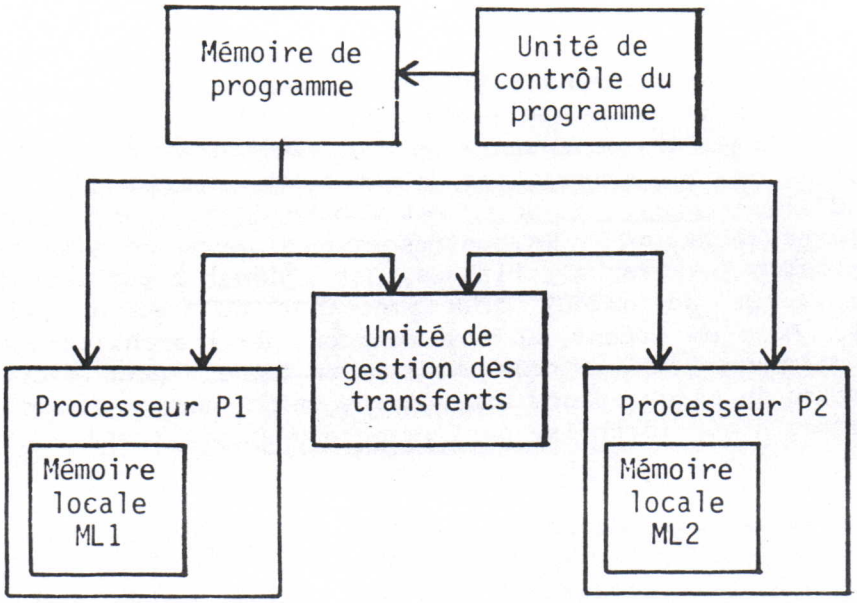
La mémoire de programme peut être séparée de la mémoire de données pour autoriser un accès simultané aux instructions et aux opérandes. Nous pouvons dans ce cas accéder à l'instruction de rang n+1, la décoder, faire le calcul des adresses des opérandes de rang n+1, alors que l'instruction de rang n est en cours d'exécution et nous préparons les opérandes de rang n. Les mémoires de données peuvent être aussi séparées en plusieurs unités, destinées à un type bien déterminé de données. L'utilisation du parallélisme implique très souvent la séparation des mémoires, car plusieurs données doivent être lues simultanément pour charger les opérateurs en parallèle.

4.1.3. Le pipe-line

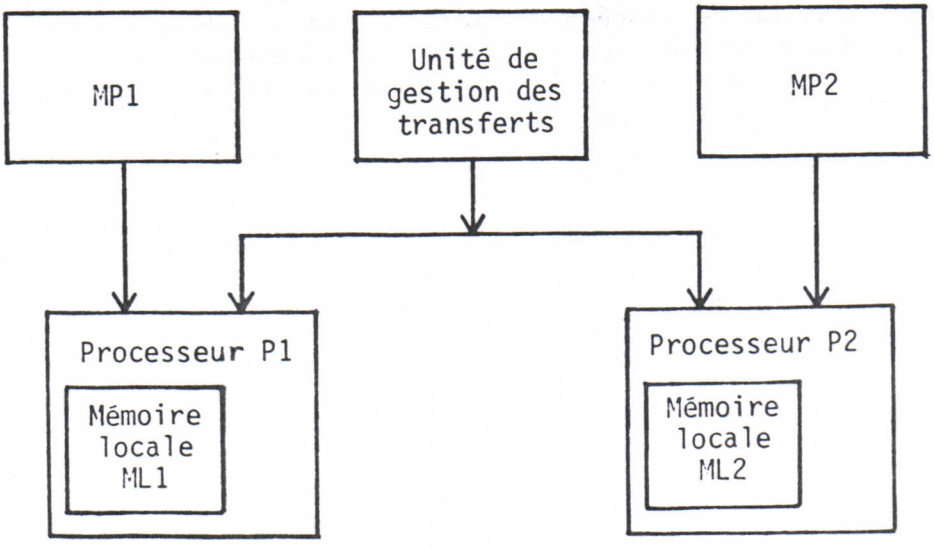
La **figure 20** montre le principe général d'un opérateur pipe-line.

Un opérateur pipe-line est constitué de n sous-opérateurs en général non identiques. Ceux-ci sont séparés par des registres de verrouillage. Chaque sous-opérateur effectue une tâche particulière qui est une partie de la tâche globale. A chaque cycle, le sous-operateur i utilise les résultats fournis par le sous-opérateur i-1, et fournit ses résultats au sous-opérateur i+1. Ainsi, si une opération doit prendre un temps T au total, l'opérateur pipe-line permet d'accélérer le processus en fournissant un résultat à des intervalles réguliers t=T/n, au bout d'un temps d'initialisation qui est égal à T. La

FIGURE 21.



(21a) Structure de type simple instruction - multiples données.



(21b) Structure de type multiples instructions - multiples données.

technique du pipe-line peut être utilisée dans une unité de calcul (exemple : multiplieur pipe-line), ou pour l'ensemble de la structure du calculateur (exemple : lecture et décodage d'une instruction pendant l'exécution de l'instruction précédente).

4.1.4. Les opérateurs spécialisés

Si un groupe d'opérations revient très souvent dans le calcul, nous pouvons utiliser un opérateur spécialisé pour le groupe. Par exemple un opérateur de type XY + Z conçu indépendamment de l'additionneur peut être utile en traitement du signal.

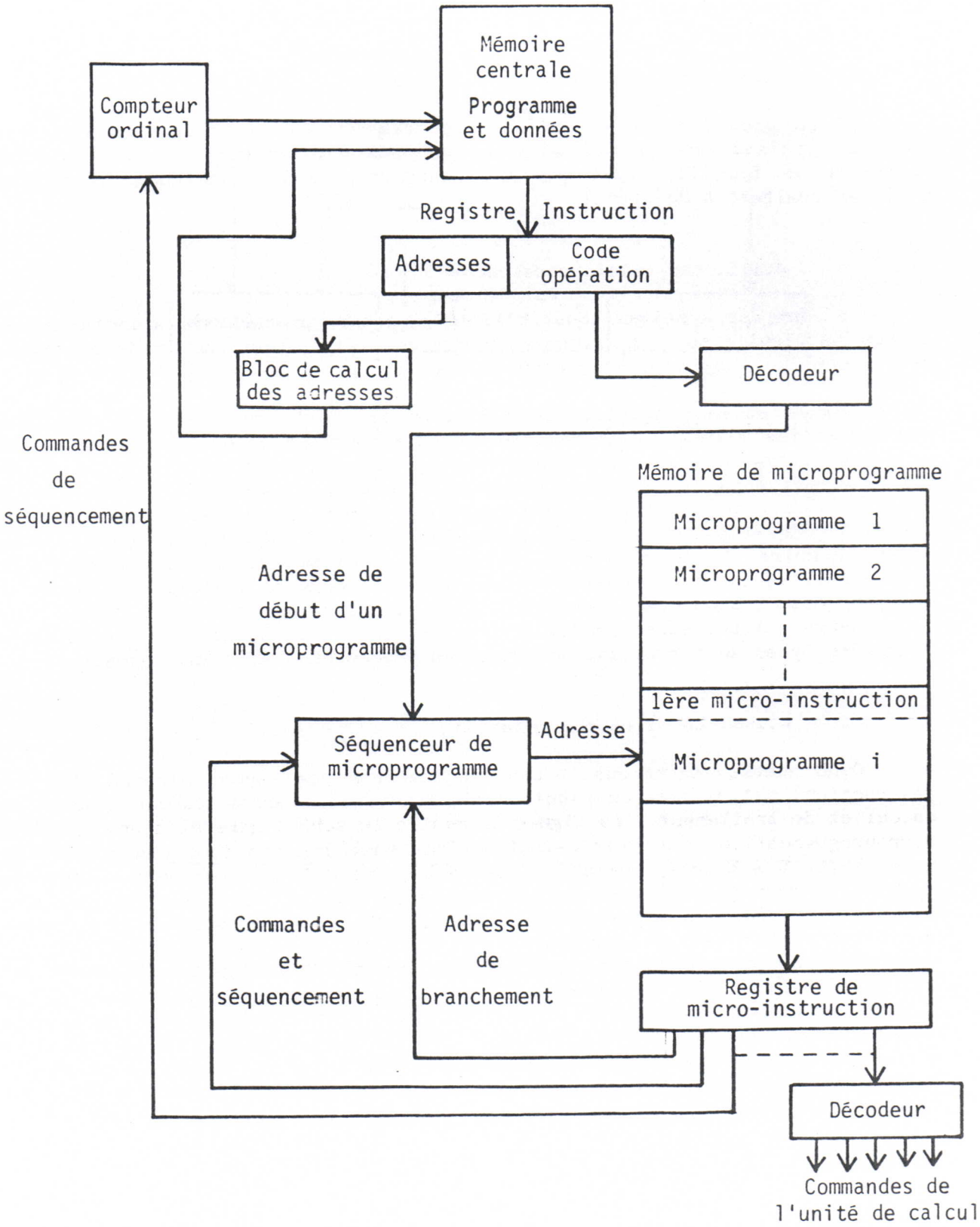
4.1.5. Les multiprocesseurs

Ce sont des systèmes constitués de plusieurs processeurs affectés à des tâches particulières. Un processeur est constitué d'une unité de contrôle, d'un ensemble de registres internes ou de mémoires locales (d'importance très variable) et d'une unité de calcul travaillant sur des opérandes fournis par les mémoires et les registres sous l'effet de l'unité de contrôle. Il faut distinguer les systèmes de type SIMD : simple instruction multiple donnée, des systèmes de type MIMD : multiple instruction multiple donnée. Le principe de ces systèmes apparaît sur la **figure 21.** Dans les premiers, la mémoire de programme est commune à plusieurs processeurs travaillant en parallèle avec leurs propres mémoires locales, dans les seconds, chaque processeur a sa propre mémoire de programme et sa propre mémoire locale de données. La présence de plusieurs processeurs pose des problèmes de synchronisation et de gestion des ressources communes. Les vitesses de transfert entre les processeurs doivent être grandes et la synchronisation minutieusement étudiée. Plusieurs types de communication entre les processeurs sont envisageables.

4.1.6. La microprogrammation

Dans cette technique, une instruction de type élaboré (macro-instruction) est divisée en plusieurs instructions exécutables par l'unité de calcul et de traitement. La **figure 22** montre le schéma général d'une unité de microprogrammation. Les micro-instructions sont rangées dans une mémoire de type RAM ou ROM d'accès rapide, appelée mémoire de microprogramme. Dans une machine microprogrammée les compteurs de programme et de microprogramme sont séparés. Le décodage d'une macro-instruction fournit l'adresse de la première micro-instruction du microprogramme correspondant. Pendant que celui-ci est exécuté, le compteur de programme est incrémenté ; l'instruction suivante est lue et décodée et les adresses pour l'accès aux opérandes sont préparées. La micro-instruction fournit les commandes pour l'unité de calcul de manière directe (sans codage) ou de manière indirecte (avec codage). Elle peut être divisée en plusieurs champs de commandes dont la signification peut varier selon le type de macro-instruction exécuté. Le codage et la séparation par champs réduisent la longueur du mot mais ralentissent l'exécution. Une micro-instruction peut être séparée en plusieurs groupes de commandes intervenant à des moments différents dans le cycle de calcul. Dans ce cas ils sont associés à des phases d'horloge différentes et la longueur de la micro-instruction est réduite par une écriture séquentielle. La microprogrammation est dite "verticale" . Dans le cas où la micro-instruction est longue mais fournit des commandes directes, elle est dite "horizontale".

FIGURE 22.


Mémoire centrale
Programme et données
Compteur ordinal
Registre Instruction
Adresses
Code opération
Bloc de calcul des adresses
Décodeur
Commandes de séquencement
Mémoire de microprogramme
Microprogramme 1
Microprogramme 2
Adresse de début d'un microprogramme
1ère micro-instruction
Séquenceur de microprogramme
Adresse
Microprogramme i
Commandes et séquencement
Adresse de branchement
Registre de micro-instruction
Décodeur
Commandes de l'unité de calcul

4.2. Techniques utilisées pour notre calculateur

Bien que n'ayant pas encore abordé la structure de notre calculateur, nous allons signaler quelques unes des techniques utilisées.

Le CTR est un multiprocesseur de type "multiple instruction multiple donnée". Il est formé de trois processeurs - processeur interne, processeur d'entrée-sortie, processeur de sortie sonore - et d'un automate pour la gestion des transferts entre les processeurs. Les unités de contrôle sont microprogrammées pour chacun d'entre eux. La microprogrammation est horizontale.

Le parallélisme est exploité en différents endroits :

- dans le processeur d'entrée-sortie au niveau du filtre (les deux multiplications se font simultanément par deux multiplieurs identiques),
- dans le processeur de sortie sonore (le traitement des deux voies de sortie se fait par des unités identiques fonctionnant en même temps),
- dans le processeur interne (le multiplieur, l'additionneur et les mémoires peuvent être exploités simultanément). Cela est possible grâce à la séparation des mémoires, à la présence de deux bus de données et aux connexions multiples entre les opérateurs, les mémoires et les deux bus (les bus A et B, formé de 32 lignes chacun).

Les mémoires sont séparées par tâches et structurées. Dans le processeur interne, nous distinguons la mémoire de programme des deux mémoires de données, dont l'une est destinée à stocker des variables et l'autre des coefficients. Le processeur d'entrée-sortie possède une mémoire pour le stockage des variables d'entrée du filtre, une pour le stockage des variables de sortie et une autre pour les coefficients du filtre.

La technique du **pipe-line** est exploitée au niveau de la mémoire de programme, de la mémoire de microprogramme, des mémoires de données et dans la structure interne du multiplieur.

L'opérateur test est un opérateur spécialisé câblé, pour le type de tests introduit par les éléments de liaison conditionnelle. Sa structure est directement déduite de l'étude des algorithmes pour les modules "unilatéral, échappement, condition extérieure".

5. Représentation des données

5.1. Considérations générales

Trois types de représentation des nombres sont couramment utilisés dans les systèmes numériques.

5.1.1. Représentation des nombres en virgule fixe

Les nombres sont représentés sous une forme où la virgule binaire a une position fixe. Les bits qui se trouvent à la droite de la virgule représentent la partie fractionnaire du nombre et les bits qui se trouvent à sa gauche représentent la partie entière. Deux types de nombres sont couramment utilisés dans les calculateurs :

LES NOMBRES ENTIERS

Ils sont employés principalement dans le calcul des adresses des opérandes et le contrôle des boucles de programme. Un nombre entier positif représenté par n bits, a_0, a_1 ... a_{n-1} a pour valeur :

$$a_{n-1}\ 2^{n-1} + a_{n-2}\ 2^{n-2} + \ldots + 2\ a_1 + a_0 \quad ;$$

a_0 est le bit de poids le plus faible, a_{n-1} le bit de poids le plus fort.

LES NOMBRES FRACTIONNAIRES

Un nombre fractionnaire positif représenté par n bits, a_{-1}, a_{-2} ... a_{-n} a pour valeur :

$$a_{-1}\ 2^{-1} + a_{-2}\ 2^{-2} + \ldots + a_{-n}\ 2^{-n} \quad .$$

La représentation fractionnaire des nombres a pour avantage par rapport à une représentation entière d'éviter tout débordement de capacité lors d'une opération de multiplication.

REPRESENTATION DU SIGNE

Les nombres signés peuvent être représentés par la méthode signe-amplitude, par celle du complément à 1 ou par celle du complément à 2. Dans la méthode du complément à 2 un bit supplémentaire occupant la position de bit de poids le plus fort est le bit de signe. Il est égal à 0 pour un nombre positif et à 1 pour un nombre négatif. En arithmétique virgule fixe, complément à 2, un nombre formé de b bits, avec une partie entière de i bits et une partie fractionnaire de b-i bits, est représenté par :

$$-\ a_{i-1}\ 2^{i-1} + a_{i-2}\ 2^{i-2} + \ldots + a_0\ 2^0 + a_{-1}\ 2^{-1} + \ldots + a_{i-b}\ 2^{i-b} \quad .$$

La plupart des opérateurs existant sous forme intégrée utilisent l'arithmétique complément à 2. En virgule fixe la précision absolue est fonction du nombre de bits.

5.1.2. Représentation des nombres en virgule flottante

Un nombre représenté en virgule flottante est donné par :

$$m \cdot b^a$$

m est la mantisse, b la base (en général b=2) et a est l'exposant. M et a sont deux nombres signés, représentés en virgule fixe avec respectivement b1 et b2 bits. Si n est le nombre total de bits, en général b1 est environ égal à (3/4)n. Les valeurs maximale et minimale d'un nombre représenté en virgule flottante, sont déterminées par b et le nombre de bits de l'exposant. La précision relative est fonction du nombre de bits de la mantisse. Cette représentation permet d'obtenir le maximum de dynamique lors d'un calcul, mais la gestion des exposants et la normalisation des mantisses ralentissent les calculs.

5.1.3. Représentation des nombres en bloc flottant

Au lieu d'associer, comme dans la représentation en virgule flottante un exposant à chaque nombre, on associe un exposant à un groupe de nombres représentés en virgule fixe. Lors d'un débordement dans une opération, l'ensemble des nombres est divisé par deux, l'exposant est corrigé et on recommence l'opération. On obtient ainsi un maximum de précision, une bonne dynamique et un calcul relativement rapide.

5.2. Représentation des données dans le CTR

Le format des données est le paramètre principal déterminant la rapidité et la complexité des différents opérateurs. Il faut qu'il soit adapté pour représenter avec une précision suffisante, les variables et les coefficients tout en permettant une bonne dynamique.

Nous allons maintenant décrire les différentes étapes, qui nous ont permis de déterminer le format des données dans le CTR.

5.2.1. Etude théorique des erreurs de troncature et d'arrondi

L'étude a été menée pour les deux types de représentation des nombres : en virgule fixe et en virgule flottante.

a/ Erreurs de troncature et d'arrondi dans le cas de la représentation en virgule fixe

Nous avons exploité les résultats obtenus par S. SANKAR, et nous avons appliqué ces résultats à l'algorithme de la cellule.

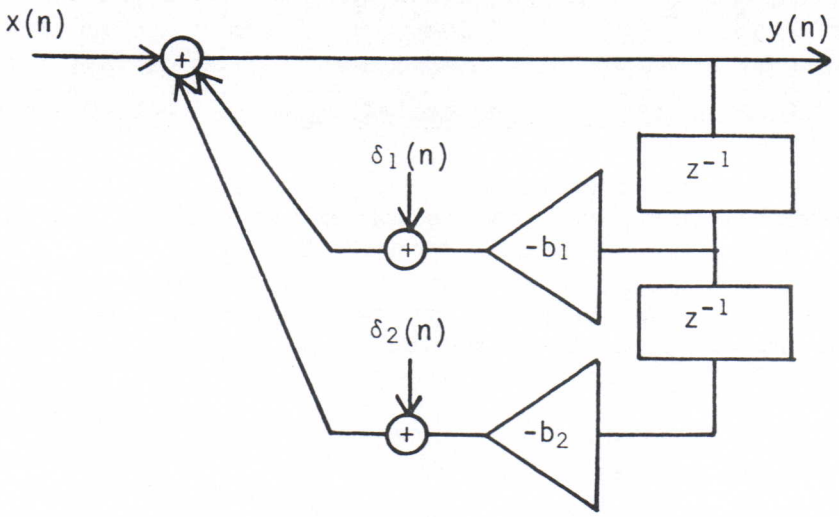
Dans le cas d'un filtre numérique du second ordre, purement récursif ou autorégressif (**figure 23**), correspondant à l'équation :

$$y(n) = x(n) - b1.y(n-1) - b2.y(n-2)$$

la fonction de transfert s'écrit :

$$H(z^{-1}) = \frac{1}{1 + b1.z^{-1} + b2.z^{-2}} \quad .$$

FIGURE 23.



Nous formulons les hypothèses suivantes :
- La séquence d'entrée du filtre est une séquence de bruit blanc stationnaire, de valeur moyenne nulle, de variance σ_x^2 .
- Les erreurs (arrondi ou troncature) sont des sources mutuellement non corrélées et non corrélées avec le signal d'entrée.
- Les sources d'erreur à chaque noeud de calcul, sont représentées par le modèle statistique suivant :

soit q le pas de quantification ;

pour l'arrondi, l'erreur est un bruit blanc stationnaire, uniformément distribué entre -q/2 et q/2, dont la valeur moyenne m est nulle, la variance est égale à $q^2/12$;

pour la troncature le modèle est le même, mais la valeur moyenne m est égale à q/2, et la variance est égale à $q^2/12$.

Nous donnons les résultats obtenus par SANKAR :

La valeur moyenne de l'erreur de sortie est égale à :

$$\bar{e} = \frac{2\,m}{1 + b1 + b2} \quad .$$

Or, m=0 pour l'arrondi, m=q/2 pour la troncature. Alors :

$\bar{e} = 0$ pour l'arrondi,

$\bar{e} = \frac{q}{1 + b1 + b2}$ pour la troncature.

La variance de l'erreur en sortie est égale à :

$$\sigma_e^2 = \frac{2q^2}{12} \cdot \frac{1 + b2}{(1 - b2)\,((1 + b2)^2 - b1^2)}$$ pour l'arrondi et pour la troncature.

Application au cas d'une cellule ayant pour équation :

$$X(n) = F(n)/M + (2-K-Z).X(n-1) - (1-Z).X(n-2) \ ;$$

b1 = - (2-K-Z) peut être positif ou négatif
b2 = 1-Z est positif car

Nous ne tenons pas compte de l'erreur commise sur l'entrée. Alors :

$\bar{e} = \frac{q}{1 + b1 + b2} = \frac{q}{K}$ pour la troncature,

et $\sigma_e^2 = \frac{2q^2}{12} \cdot \frac{2 - Z}{Z.K.(2(2 - Z) - K)}$.

b/ Erreurs de troncature et d'arrondi dans le cas de la représentation en virgule flottante

Nous avons exploité les résultats obtenus par LIU et KANEKO, par SANDBERG, et par OPPENHEIM. LIU et KANEKO considèrent un modèle statistique pour l'erreur d'arrondi et de troncature, alors que SANDBERG s'oriente vers un procédé de majoration de l'erreur.

SANDBERG prouve que, pour une très grande classe de filtres récursifs, il existe une fonction f(K), telle que :

$$f(K) \rightarrow 0 \ , \ \text{quand} \ K \rightarrow \infty \quad ;$$

et une constante c dépendant a/ des coefficients de l'équation de récurrence, b/ de l'ordre dans lequel les produits individuels sont sommés dans la machine, c/ du nombre de bits de la mantisse, telle que :

$$\langle e \rangle_K \leqslant C.\langle y \rangle_K + f(K) \quad .$$

Pour :

$$\omega_n = \sum_{\ell=0}^{M} b_\ell x_{n-\ell} - \sum_{=1}^{N} a_\ell \omega_{n-\ell} \qquad n \geq N ,$$

K étant le nombre d'échantillons servant à calculer la valeur moyenne ou la variance,

$\langle e \rangle_K^2$ est la valeur moyenne quadratique de l'erreur sur K échantillons,

$\langle y \rangle_K^2$ est la valeur moyenne quadratique du signal de sortie sur K échantillons.

$$\langle e \rangle_K = \left(\frac{1}{K+1} \sum_{n=0}^{K} |\omega_n - y_n|^2 \right)^{1/2}$$

$$\langle y \rangle_K = \left(\frac{1}{K+1} \sum_{n=0}^{K} |y_n|^2 \right)^{1/2}$$

Alors, pour $K \to \infty$ $f(K) \to 0$, et :

$$\frac{\langle y \rangle_K}{\langle e \rangle_K} \geqslant \frac{1}{C}$$

20 log(1/C) va donner une limite inférieure asymptotique du rapport S/B en sortie.

Nous avons appliqué les résultats théoriques obtenus par SANDBERG, au cas de l'équation de la cellule définie précédemment.

5.2.2. Etude pratique des erreurs de calcul

Considérons de nouveau l'équation de la cellule :

$$X(n) = F(n)/M + (2-K-Z).X(n-1) - (1-Z).X(n-2)$$

Ce calcul a été effectué dans le LSI, avec un format de référence pour lequel le signal de sortie est considéré comme étant sans erreur, et d'autres formats virgule fixe ou virgule flottante avec un nombre de bits variable. Le format de référence est celui du mini-ordinateur. C'est un format de calcul en virgule flottante avec 8 bits d'exposant, 24 bits de mantisse et un bit de signe. La représentation du signe se fait par la méthode du complément à deux. Nous avons fait deux séries de programmes de calcul.

a/ Calcul en format virgule fixe

La représentation des nombres (variables et coefficients), se fait avec m bits, m variant de 16 à 32. Le signe est représenté par la méthode du complément à deux. A chaque opération intervient une troncature qui réduit le nombre de bits du résultat à m bits. Chacun des programmes calcule un signal de sortie y(n) qui est la réponse impulsionnelle de la cellule dont le

comportement est décrit par l'équation ci-dessus. Le calcul se fait pour un nombre d'échantillons assez important. Chacun des programmes calcule en outre un signal de sortie selon le format de référence, et un signal d'erreur. Celui-ci est la différence entre le signal de référence et le signal y(n). Le programme calcule aussi la valeur absolue maximale du signal d'erreur sur un nombre d'échantillons important, et le niveau S/B défini par l'expression :

$$\frac{\text{Valeur absolue maximale du signal}}{\text{Valeur absolue maximale de l'erreur}}$$

L'expression réelle du rapport signal sur bruit est :

$$S/B = 10 \cdot \log \frac{\text{valeur moyenne quadratique du signal}}{\text{valeur moyenne quadratique de l'erreur}} \quad .$$

Mais le calcul de la valeur moyenne quadratique du signal cause des dépassements de capacité dans les calculs intermédiaires pour un nombre d'échantillons important. Nous utilisons pour cette raison l'expression approchée ci-dessus.

Pour tous les essais, pour m variant de 16 à 32, nous obtenons un signal d'erreur oscillant, de même fréquence que le signal de sortie, déphasé par rapport à celui-ci, ayant une valeur moyenne non nulle. Les essais ont été faits avec des valeurs différentes pour les paramètres K et Z. Nous avons remarqué que pour un même signal de référence, quand le nombre de bits m passe de 24 à 25, il y a une diminution assez importante du signal d'erreur.

b/ Calcul en format virgule flottante

Notre signal de référence est toujours le même. La représentation des nombres (variables et coefficients) se fait maintenant avec 8 bits d'exposant et m bits de mantisse plus un bit de signe. M varie de 16 à 23. Pour m = 24 nous retrouvons le signal de référence, le signal d'erreur étant nul dans ce cas. Nous calculons le signal d'erreur et le rapport S/B de la même manière que précédemment. Le signal d'erreur ne présente aucune caractéristique remarquable.

5.2.3. Exploitation des résultats

Au vu des résultats théoriques et expérimentaux, nous avons rapidement laissé de côté le cas du calcul en virgule fixe avec 16 bits, de même que le cas du calcul en flottant avec 16 bits de mantisse. Ayant remarqué une nette amélioration du niveau S/B, en passant de m = 24 bits à m = 25 bits en virgule fixe, considérant par ailleurs les contraintes de la réalisation matérielle à l'aide de circuits intégrés existants, nous avons surtout comparé les trois cas suivants :

- calcul en virgule fixe avec 24 bits,
- calcul en virgule fixe avec 32 bits,
- calcul en virgule flottante avec 23 bits de mantisse plus un bit de signe.

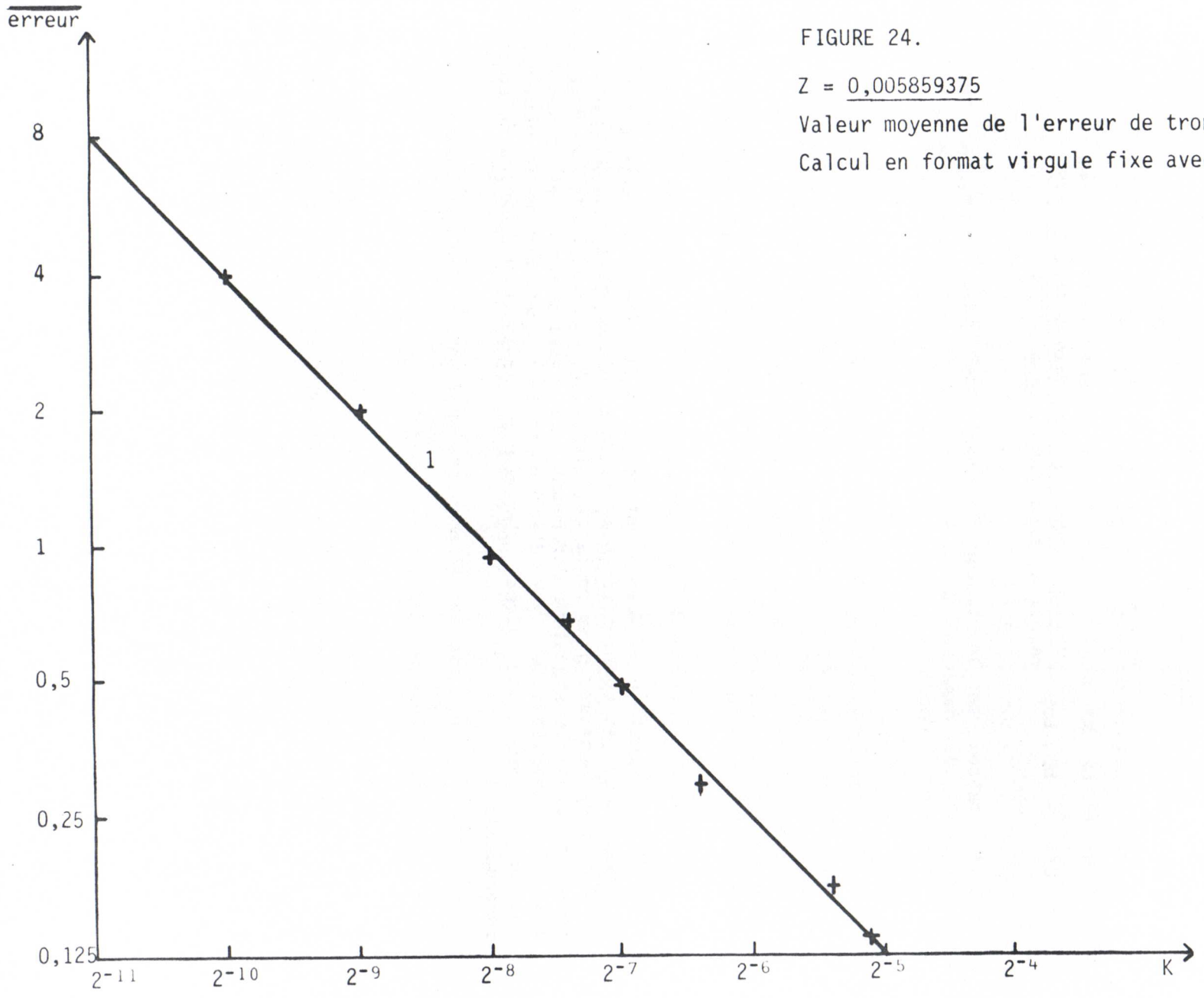


FIGURE 24.

Z = 0,005859375

Valeur moyenne de l'erreur de troncature en sortie.

Calcul en format virgule fixe avec 24 bits.

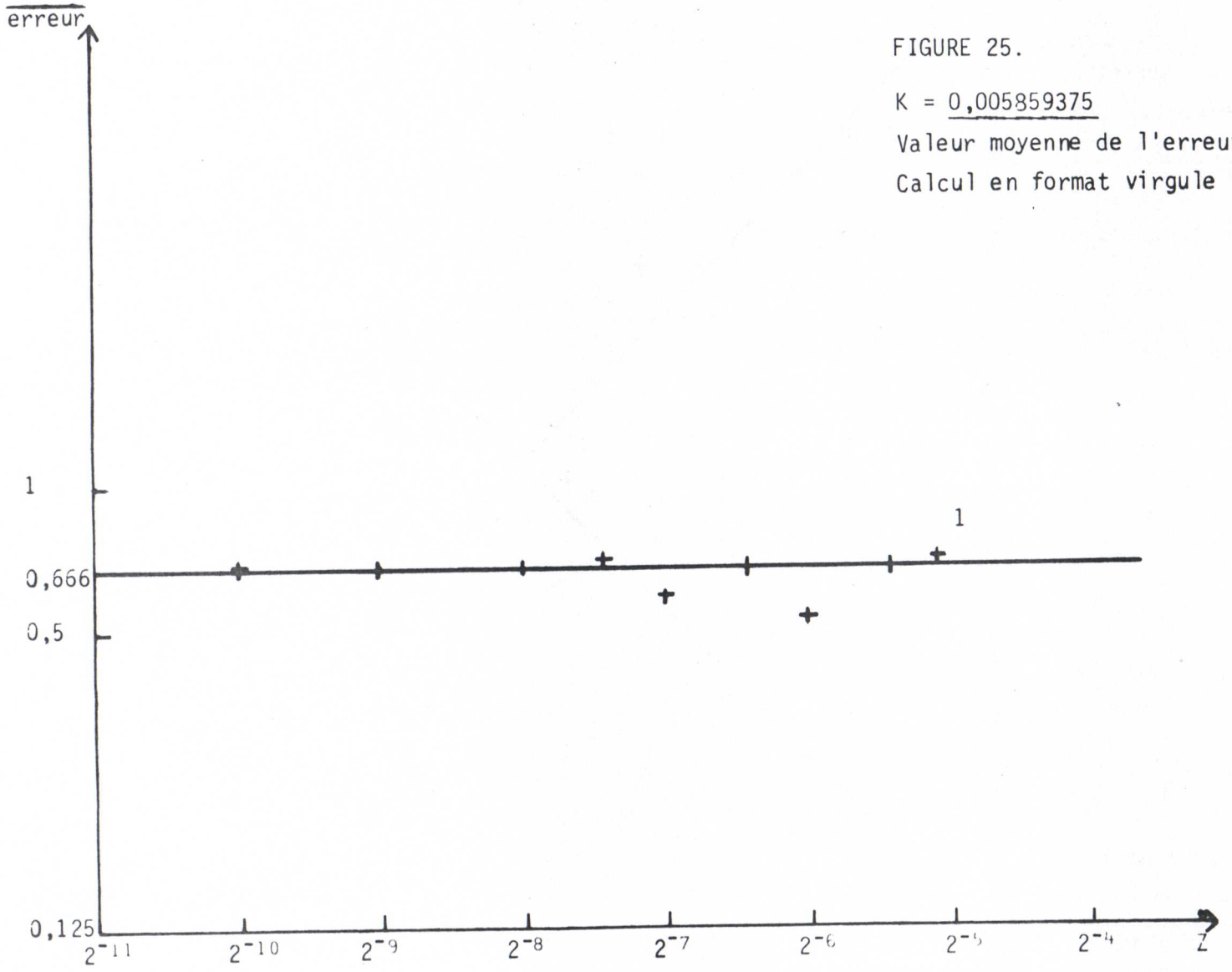


FIGURE 25.

K = 0,005859375

Valeur moyenne de l'erreur de troncature en sortie.

Calcul en format virgule fixe avec 24 bits.

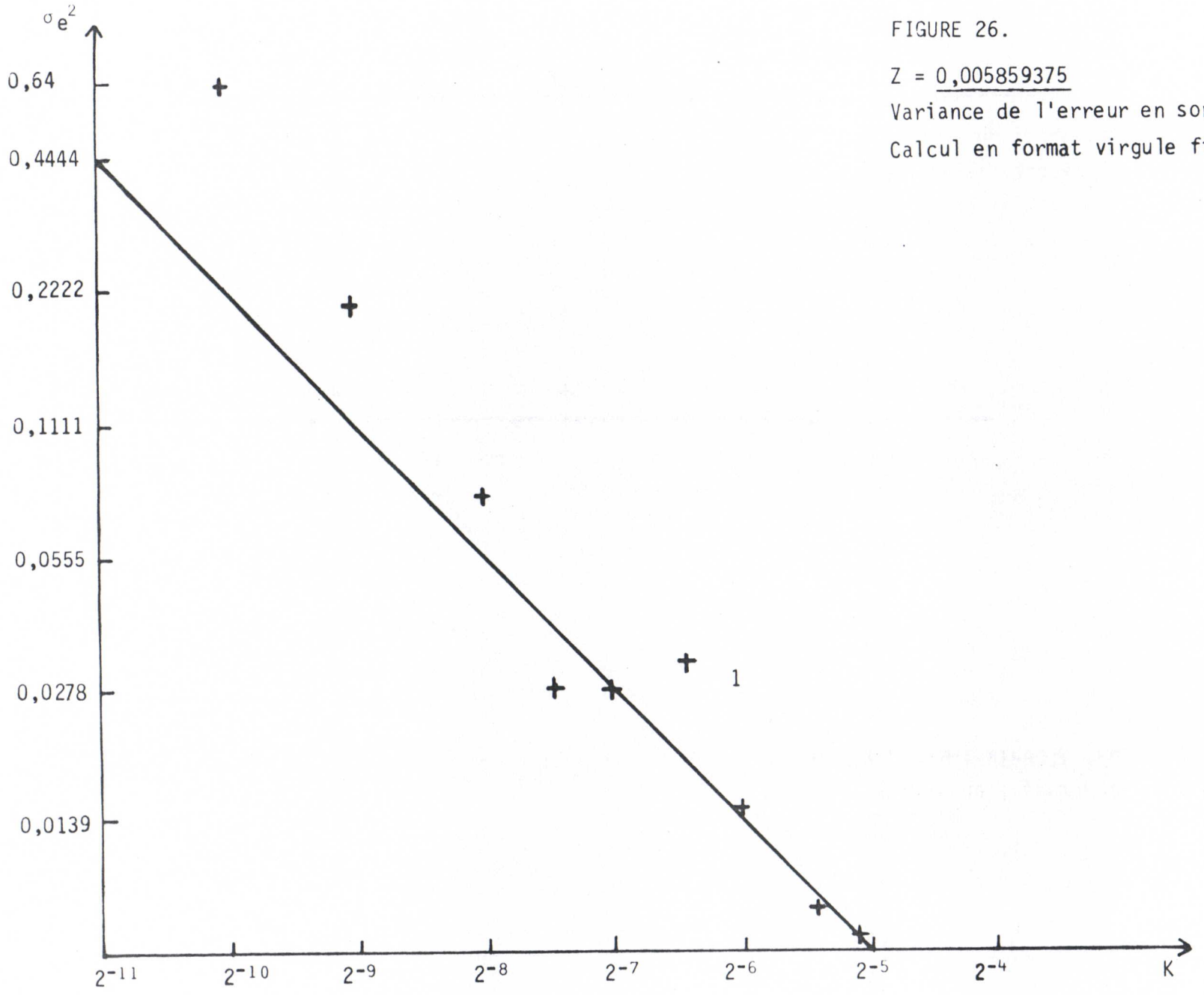


FIGURE 26.

Z = 0,005859375

Variance de l'erreur en sortie.

Calcul en format virgule fixe avec 24 bits.

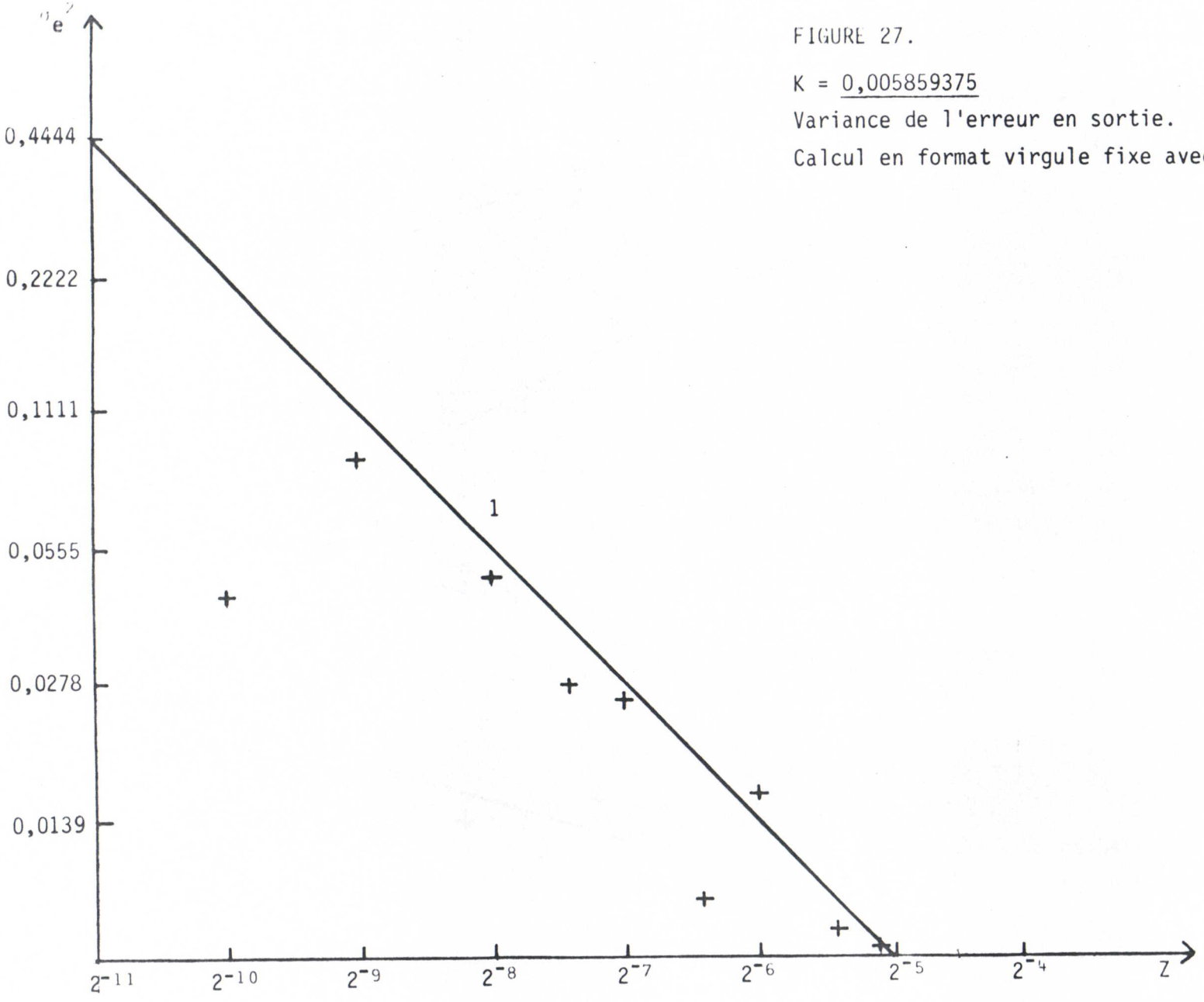


FIGURE 27.

K = 0,005859375

Variance de l'erreur en sortie.

Calcul en format virgule fixe avec 24 bits.

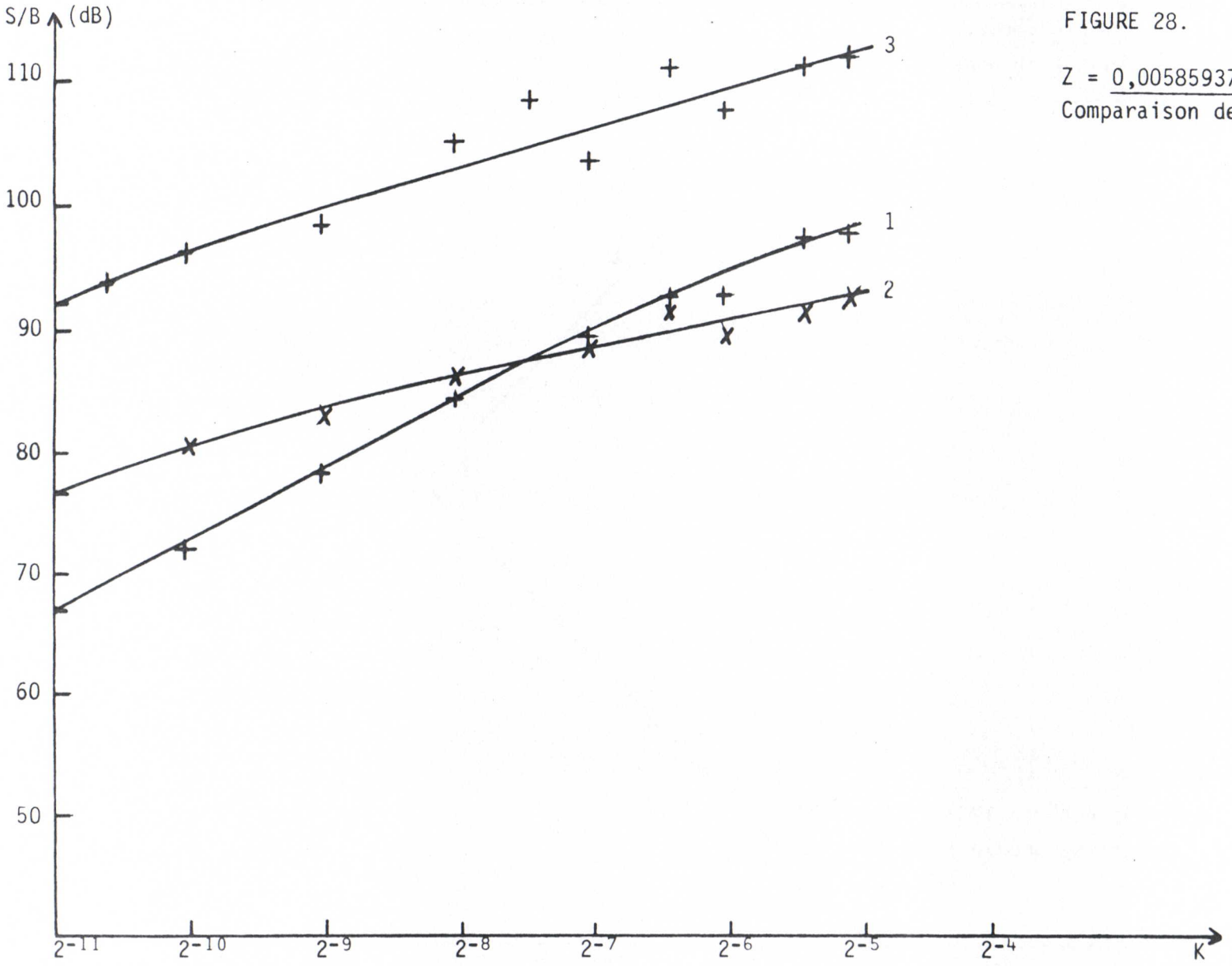


FIGURE 28.

Z = 0,005859375

Comparaison des rapports S/B.

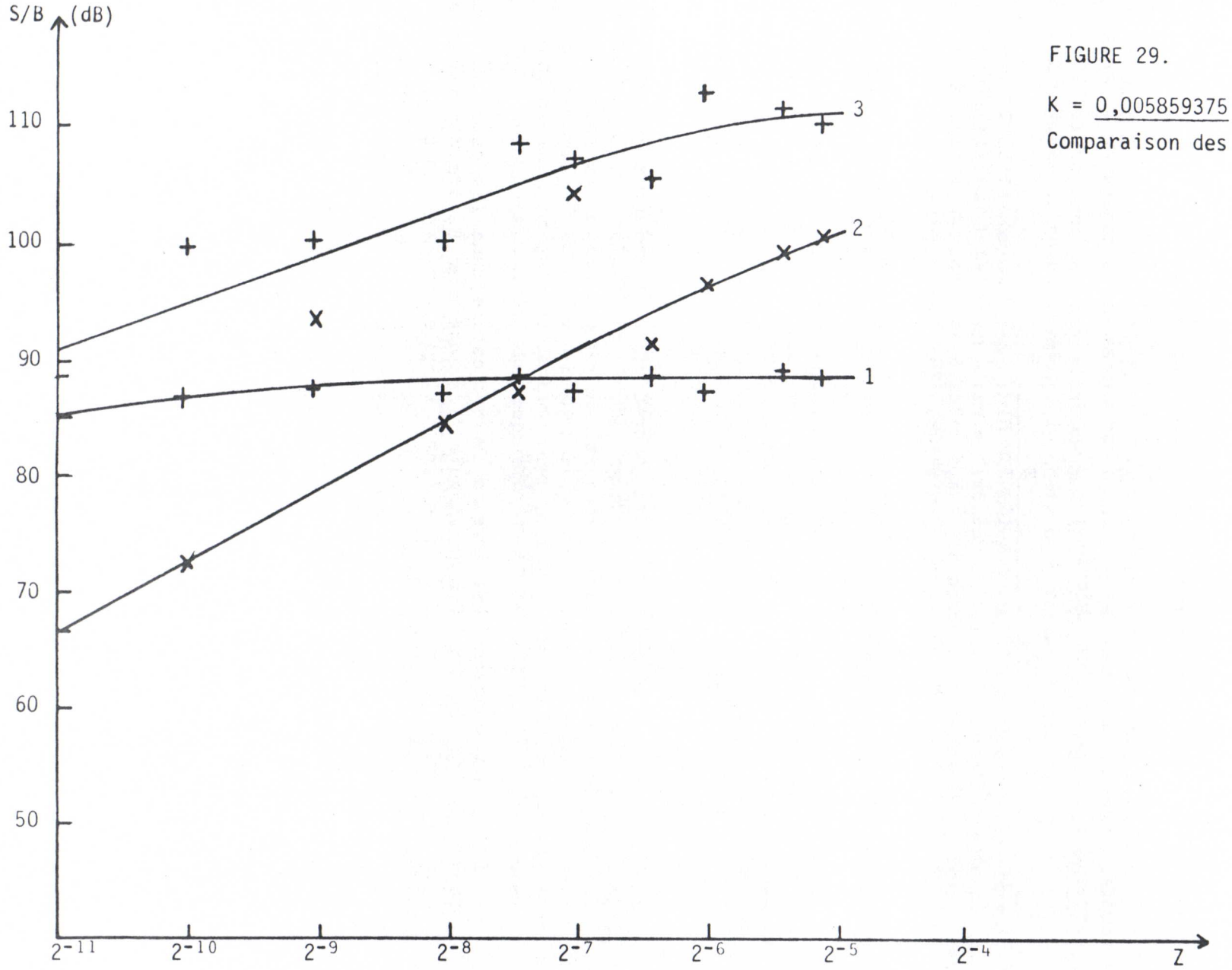


FIGURE 29.

K = 0,005859375

Comparaison des rapports S/B.

Sur la **figure 24** nous voyons les courbes qui donnent la valeur moyenne de l'erreur de troncature en sortie, en fonction du paramètre K (Z est constant) dans le cas du calcul en virgule fixe sur 24 bits. La courbe 1 correspond à l'expression théorique : valeur moyenne = q/K. Les valeurs expérimentales, sont représentées par des croix. Nous observons une bonne concordance entre valeurs théoriques et valeurs pratiques.

Sur la **figure 25** nous voyons la même chose, mais pour K constant, Z variable. La aussi nous remarquons une bonne concordance.

Sur la **figure 26** nous voyons la variance de l'erreur de troncature en sortie, pour le même cas de calcul. K est variable, Z constant. La courbe 1 correspond à l'expression théorique :

$$\sigma_e^2 = \frac{2q^2}{12} \cdot \frac{2 - Z}{Z.K.(4 - 2Z - K)} \quad .$$

Les résultats pratiques sont représentés par des croix.

Sur la **figure 27** nous voyons la même chose pour K constant et Z variable.

Sur la **figure 28** nous voyons une comparaison des résultats pratiques pour le niveau S/B, pour trois types de calcul. La courbe 1 présente les résultats pour le calcul en virgule fixe avec 24 bits, pour K variable et Z constant. La courbe 2 présente les résultats pour le calcul en virgule flottante avec 23 bits de mantisse et un bit de signe. La courbe 3 présente les résultats pour le calcul en virgule fixe avec 32 bits.

Enfin sur la **figure 29** nous voyons une comparaison des résultats pratiques pour les niveaux S/B, pour les mêmes cas de calcul, avec K constant et Z variable.

5.2.4. Conclusion de l'étude

En ce qui concerne les erreurs de troncature dans les calculs, nous avons vu que les cas d'un format virgule flottante avec 8 bits d'exposant et 24 bits de mantisse au total et d'un format virgule fixe avec 24 bits, donnent des résultats pratiques comparables. Or le premier cas est beaucoup plus difficile à mettre en oeuvre. La mise en oeuvre implique une multiplication de deux nombres de 24 bits dans les deux cas. Avant de faire notre choix définitif, nous avons attendu la commercialisation des multiplieurs 24 bits annoncés chez le fabricant TRW. Mais ceux-ci n'ont jamais été vraiment commercialisés.

Le calcul en format virgule fixe sur 32 bits donne un niveau S/B supérieur à 90 DB, même pour des valeurs très faibles des paramètres.

Par ailleurs, pour une fréquence d'échantillonnage de 25,6 kHz, pour obtenir une fréquence d'oscillation f telle que

comme réponse impulsionnelle de la cellule, il faut que K satisfasse la relation :

$$6.10^{-6} \leqslant K \leqslant 3,54 \quad .$$

Pour avoir une précision de 1% sur la fréquence du signal (cela correspond à une précision de l'ordre du comma sur la hauteur du son) sur toute la gamme de variation, nous devons représenter K avec 25 bits en format virgule fixe.

Nous n'avons pas fait la même étude pour le paramètre Z, mais les contraintes paraissent moins importantes.

Enfin nous pouvons signaler que les contraintes dynamiques ne sont pas importantes au point d'imposer le choix d'un format flottant.

D'après toutes ces considérations nous avons choisi un format de représentation interne en virgule fixe sur 32 bits. Les coefficients A, B, C du calcul rapide sont représentés comme des nombres semi-fractionnaires. Leur signe est représenté en complément à deux (**Figure 30**).

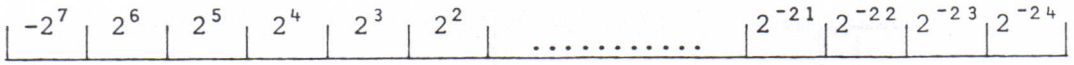


Figure 30

Les variables sont aussi représentés comme des nombres semi-fractionnaires, en arithmétique virgule fixe, complément à deux, mais la position de la virgule binaire est variable et dépend de la simulation. Par convention l'amplitude maximale crête à crête de la vibration acoustique est égale à 1 en format interne. Dans le cas où la virgule binaire est à gauche du bit de poids le plus fort, nous aurons le maximum de précision absolue sur la vibration acoustique, mais pas de possibilité de représenter des distances supérieures à 1 entre deux points de la structure par exemple. Dans les autres cas, quand la virgule se déplace vers la droite, la précision absolue diminue, mais les possibilités de représentation de distances supérieures à 1 augmentent.

Si nous voulons donner une signification physique aux unités internes, nous pouvons considérer que l'amplitude de la vibration acoustique réelle est de l'ordre du millimètre. Nous pouvons alors faire correspondre 1 mm, à l'unité interne 1 pour les déplacements. Si nous considérons maintenant des distances réalistes entre plusieurs structures vibrantes d'une même simulation, ou entre l'excitateur et la structure vibrante, nous pouvons nous limiter à des distances de l'ordre de 10 à 20 cm. Dans le cas d'une telle simulation, les variables de déplacement pourront être représentées par exemple dans le format de la **figure 31.** Ces distances seront alors comprises entre - 12,8 cm et + 12,8 cm.

Figure 31

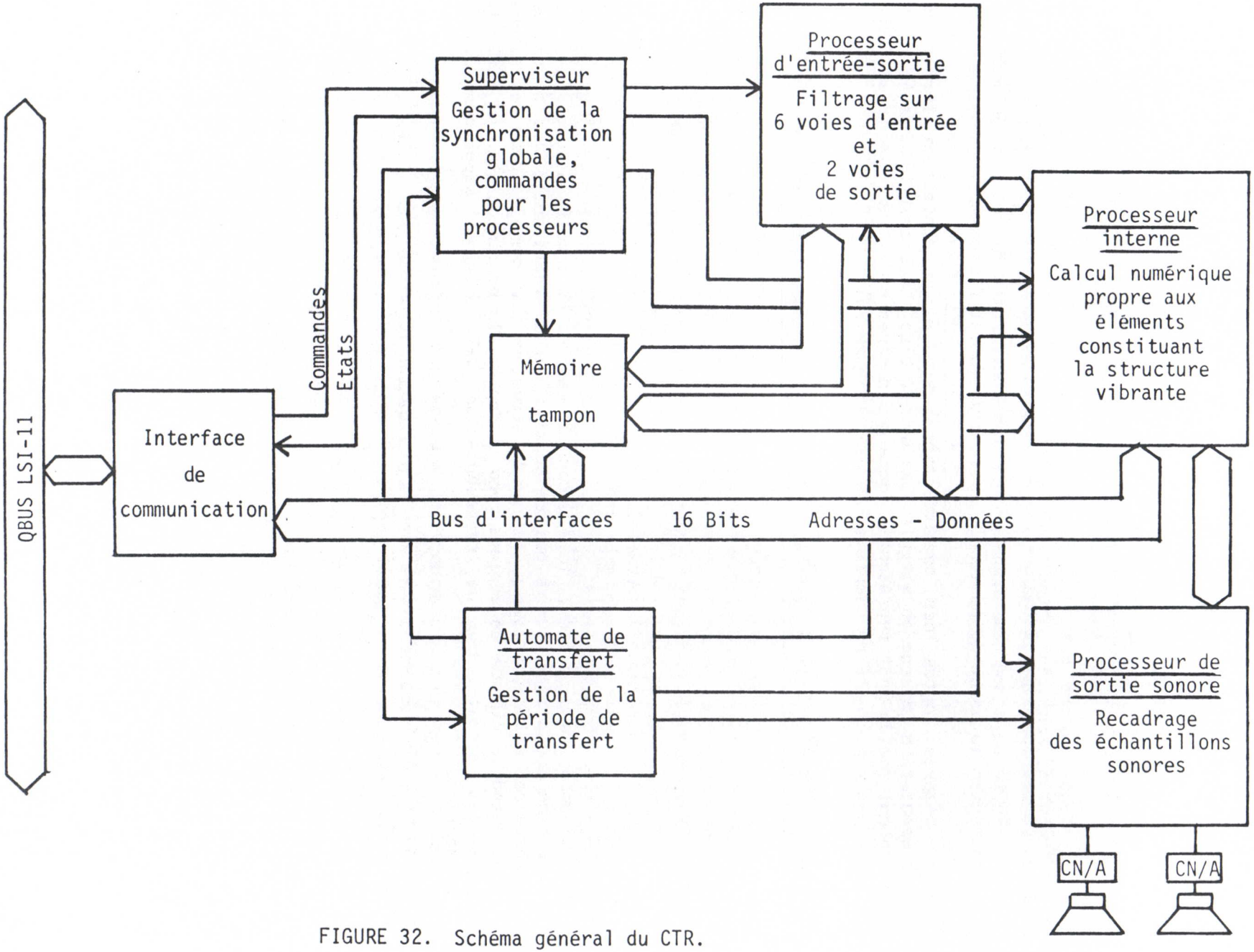


FIGURE 32. Schéma général du CTR.

B/ STRUCTURE ET FONCTIONNEMENT DU CTR

1. Description générale

Le CTR sous tension peut se trouver soit dans un état "ARRET" soit dans un état "CALCUL".

Dans l'état "ARRET", le CTR n'effectue pas de calculs des échantillons sonores. Le mini-ordinateur peut communiquer avec le CTR soit pour transférer des données, soit pour contrôler ou commander l'état du calculateur. Les données peuvent correspondre à des instructions de programme, à des valeurs numériques pour les paramètres mécaniques de la structure ou aux valeurs initiales de certaines variables. Toutes les mémoires du CTR peuvent communiquer avec le LSI 11 dans l'état "ARRET". C'est dans cet état qu'ont lieu les phases de prestructuration qualitative et quantitative.

Dans l'état "CALCUL", les opérateurs et les mémoires sont sollicités pour le calcul interne et inaccessibles pour le mini-ordinateur. Des données en provenance ou à destination des transducteurs gestuels peuvent cependant être transmises en cours de calcul. L'état de "CALCUL" du CTR correspond à la phase de jeu. Dans cet état les échantillons calculés sont envoyés aux convertisseurs numérique/analogique des voies de sortie sonore.

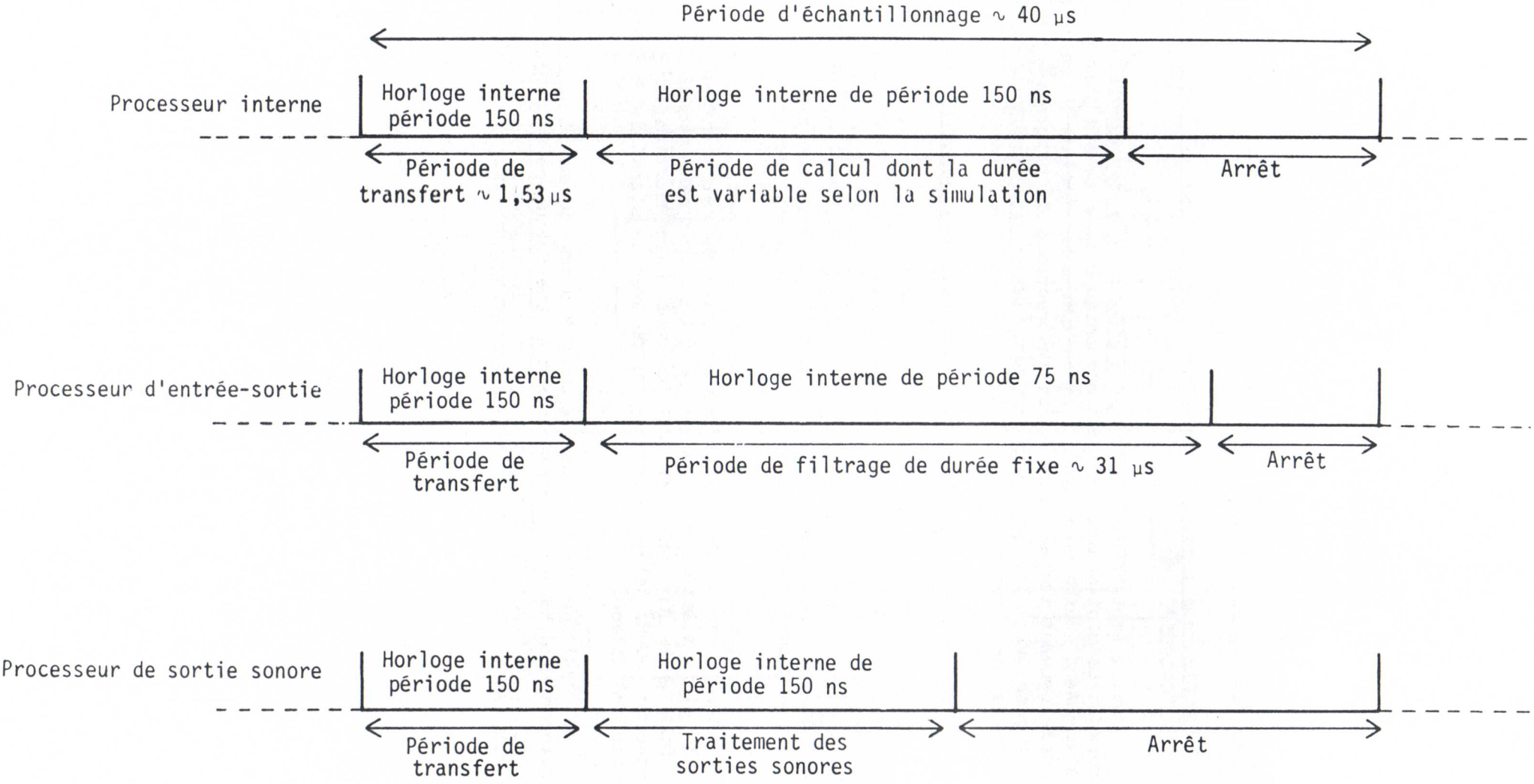
La fréquence de base pour le calcul des échantillons sonores est égale à 25,6 kHz. Cette fréquence indique le rythme du calcul rapide correspondant à la structure vibrante, alors que le calcul lent correspondant à l'excitateur se fait à la fréquence de 100 Hz dans le miniordinateur. Un signal de fréquence 100 Hz issu du CTR, synchrone avec le signal de 25,6 kHz est envoyé au miniordinateur.

Le cycle de calcul interne du CTR a une durée d'environ 150 ns (plus exactement 152,58 ns), correspondant à une fréquence de calcul de 6,536 MHz. Le signal de 6,536 MHz est synchrone avec le signal de 25,6 kHz. 256 cycles de calcul peuvent avoir lieu pour un cycle d'échantillonnage.

Avec une fréquence d'échantillonnage de 25,6 kHz la structure la plus complexe que nous puissions réaliser est une structure formée de 11 cellules et 10 élements de liaison avec une sortie sonore. Pour réaliser des structures plus complexes nous avons la possibilité de diminuer la fréquence d'échantillonnage par paliers jusqu'à 6,4 kHz. Le nombre total de fréquences différentes est égal à sept.

La structure générale du CTR apparaît sur la **figure 32.**

FIGURE 33.


Période d'échantillonnage ∿ 40 µs
Processeur interne
Horloge interne période 150 ns
Horloge interne de période 150 ns
Période de transfert ∿ 1,53 µs
Période de calcul dont la durée est variable selon la simulation
Arrêt
Processeur d'entrée-sortie
Horloge interne période 150 ns
Horloge interne de période 75 ns
Période de transfert
Période de filtrage de durée fixe ∿ 31 µs
Arrêt
Processeur de sortie sonore
Horloge interne période 150 ns
Horloge interne de période 150 ns
Période de transfert
Traitement des sorties sonores
Arrêt

Le CTR possède trois organes principaux qui sont :

- **le processeur interne (PI),**
- **le processeur d'entrée-sortie (PES),**
- **le processeur de sortie sonore (PSS).**

Il possède en outre une **interface de communication** avec le LSI 11, une **mémoire tampon** permettant une communication asynchrone en cours de calcul (en phase de jeu) entre le LSI 11 et le CTR, et un **automate de transfert** permettant la communication entre les différentes parties du CTR.

Les trois processeurs ont des rôles différents et déterminés.

Le processeur d'entrée-sortie effectue un filtrage sur les 6 voies d'entrée lente du CTR.

Le processeur interne effectue le calcul des échantillons selon les algorithmes de simulation.

Le processeur de sortie sonore effectue un cadrage des données destinées aux convertisseurs numérique/analogique pour la sortie sonore.

La **figure 33** montre une période d'échantillonnage du CTR dans l'état "CALCUL".

Quand les processeurs sont inactifs, les différentes commandes internes ont des états de repos. L'automate de transfert peut cependant établir des chemins de communication entre les différentes mémoires et registres internes de chaque processeur. Quand les processeurs sont actifs les voies de communication sont coupées, l'automate de transfert devient inactif et les processeurs travaillent simultanément pour le calcul, le filtrage et le cadrage des données.

Un cycle d'échantillonnage commence toujours par un temps de transfert (de durée égale à 10 cycles de calcul soit 1,53 microsecondes) pendant lequel les données calculées au cycle d'échantillonnage précédent sont transférées

- de la mémoire tampon vers le processeur d'entrée-sortie,
- du processeur d'entrée-sortie vers le processeur interne,
- du processeur interne vers le processeur de sortie sonore,
- du processeur interne vers la mémoire tampon.

Quand le transfert est terminé, l'automate génère un signal qui rend actif les trois processeurs simultanément. Le processeur d'entrée-sortie effectue le filtrage sur les 6 voies d'entrée (de manière multiplexée) en 31 microsecondes environ. Au bout de ce temps, il s'arrête en attendant le début du cycle d'échantillonnage suivant. Le calcul interne est plus ou moins long selon la complexité de la structure simulée. Pour une structure formée d'une cellule, d'une entrée d'excitation en force, d'une sortie sonore, nous avons un temps de calcul de 2,85 microsecondes environ ; alors que pour une structure de 11 cellules et 10 éléments de liaison et une sortie sonore, il est égal à 37 microsecondes. L'instruction FIN qui figure obligatoirement à la fin d'un programme est la dernière instruction decodée à chaque cycle d'échantillonnage. C'est elle qui remet le processeur interne à l'état inactif, en attendant le prochain cycle d'échantillonnage. Le cadrage des échantillons par le processeur de sortie sonore se fait en un temps plus ou moins long selon la position de cadrage desirée, mais ce temps est court par rapport à la période d'échantillonnage. Ce processeur se met à l'état inactif

à la fin du cadrage en attendant le prochain cycle, pendant ce temps les échantillons sonores sont envoyés aux convertisseurs sur les deux voies de sortie.

Nous allons voir maintenant la structure de chacun des organes du CTR de manière plus détaillée.

2. L'interface

L'accès au CTR par le LSI se fait en mode programmé, par une suite d'instructions (pour le LSI) de transfert des informations, de la mémoire centrale de l'ordinateur vers les registres de l'interface. L'adressage des mémoires du CTR est un adressage indirect.

Les informations transmises sont de quatre sortes :

a/ des commandes à destination du CTR,
- pour démarrer le calcul quand il est à l'état d'arrêt, et pour arrêter le calcul,
- pour faire effectuer un cycle unique de calcul (commande très utile pour la mise en route et la maintenance de la machine),
- pour initialiser les registres internes et les bascules en mode programmé,
- pour autoriser les interruptions sur les deux lignes d'interruption,

b/ des adresses pour toutes les mémoires du CTR y compris la mémoire tampon,

c/ des données de préstructuration (instructions, paramètres...),

d/ des données de jeu (variables,paramètres ...).

Le bus de communication entre l'interface et le CTR possède 16 lignes pour les adresses et les données, qui sont alors multiplexées.

Pour chacun de ces quatre types d'information, le LSI s'adresse à un registre différent. Il y a donc quatre registres d'interface.

LE REGISTRE D'ETAT ET DE CONTROLE

C'est un registre de 16 bits dont l'adresse en octal est égale à 170200. La signification des données inscrites dans ce registre respecte la philosophie générale des mots d'état et de contrôle des autres périphériques du LSI. La signification des bits est la suivante :

bit 15 : Demande d'interruption sur la ligne A,
bit 14 : Autorisation d'interruption sur la ligne A,
bit 7 : Demande d'interruption sur la ligne B,
bit 6 : Autorisation d'interruption sur la ligne B,
bit 3 : Commande d'initialisation des registres et bascules du CTR,
bit 2 : Commande pour un cycle de calcul unique,
bit 1 : Commande d'arrêt,
bit 0 : Commande de démarrage.

LE REGISTRE D'ADRESSE

C'est un registre de 16 bits dont l'adresse en octal est égale à 170202. Les 16 lignes d'interface utilisées pour les adresses, permettent d'adresser toutes les mémoires du CTR. Celles-ci se séparent en cinq groupes :

- la mémoire de programme (MP),
- la mémoire de variables (MESI),
- la mémoire de paramètres (MC),
- la mémoire de coefficients pour les filtres (COEF),
- la mémoire tampon (TAMPON).

Les trois premières font partie du processeur interne, la quatrième appartient au processeur d'entrée-sortie, alors que la mémoire tampon sert au stockage provisoire des données transmises entre le LSI et le CTR en cours de jeu.

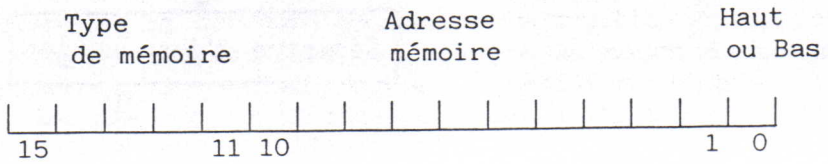


Figure 34

Le format du registre d'adresse apparaît sur la **figure 34.** Les 5 lignes de poids fort servent à choisir une mémoire parmi les cinq, et les 11 lignes de poids faible servent à adresser la mémoire choisie. La représentation des données dans le CTR se fait sur 32 bits. Seul les instructions s'écrivent sur 16 bits. Toutes les mémoires du CTR sauf la mémoire de programme, sont formées de mots de 32 bits. Pour la communication d'une donnée entre le LSI et le CTR, il faut deux transferts successifs. Chaque donnée possède une partie basse et une partie haute. La partie basse est transmise la première, suivie de la partie haute. Le bit de poids le plus faible du registre d'adresse, sert à adresser la partie basse ou la partie haute d'une mémoire. Pour permettre de charger les mémoires du CTR de manière rapide, sans avoir à préciser une adresse pour chaque donnée, l'interface possède un système d'incrémentation automatique. Si une adresse est suivie de plusieurs données, la première donnée est chargée à cette adresse, la deuxième à l'adresse suivante etc..

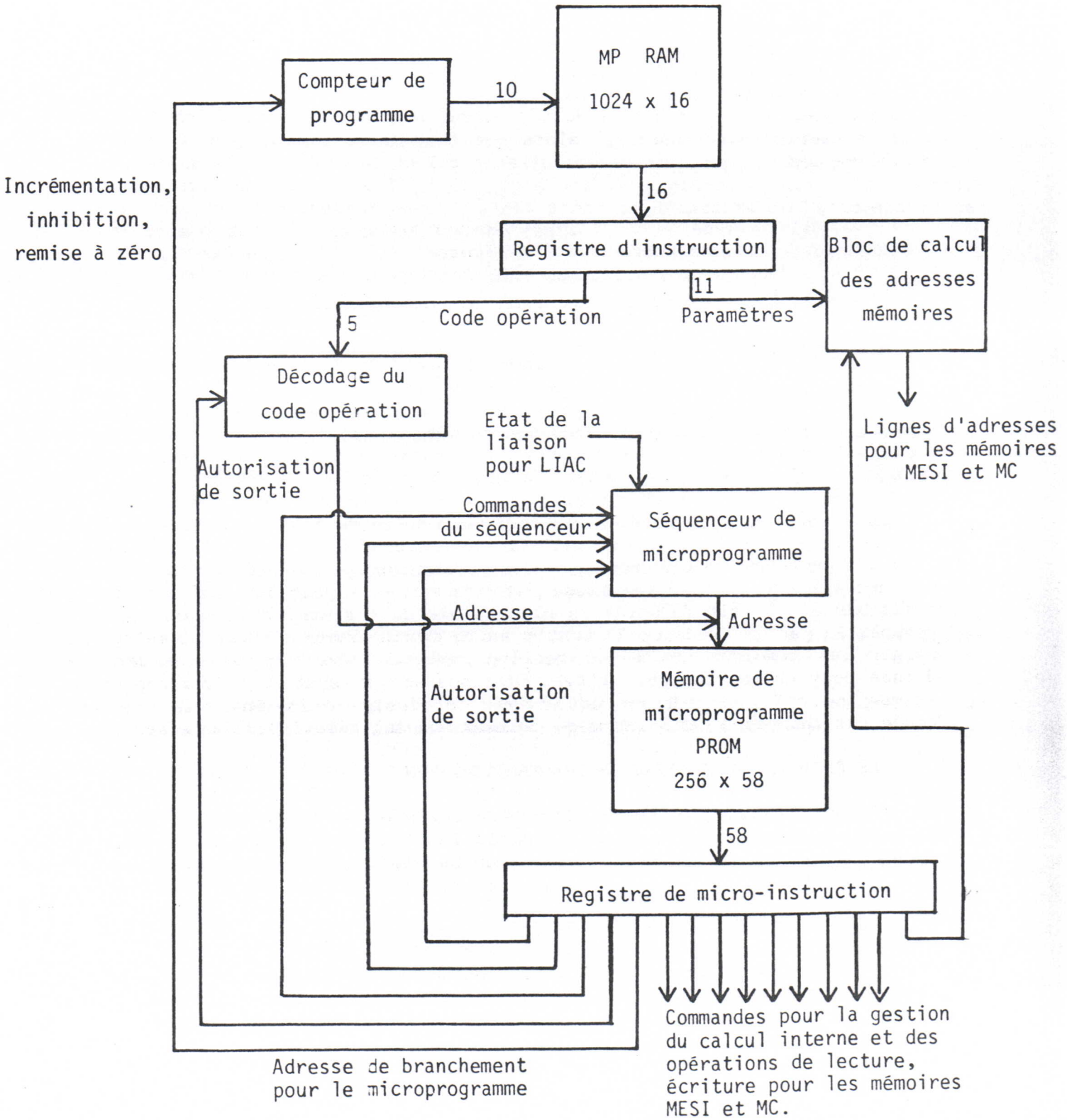
LE REGISTRE DE DONNEES DE PRESTRUCTURATION

C'est un registre de 16 bits dont l'adresse en octal est 170204. Il contient tantôt la partie basse, tantôt la partie haute d'une donnée. Le transfert des données de prestructuration ne peut se faire que quand le CTR est à "l'ARRET".

LE REGISTRE DE DONNEES DE JEU

C'est un registre de 16 bits qui contient la partie basse ou la partie haute d'une donnée. Son adresse en octal est égale à 170206. Le transfert des données peut se faire dans l'état "ARRET" ou dans l'état "CALCUL".

FIGURE 35.


MP RAM
1024 x 16
Compteur de programme
10
Incrémentation, inhibition, remise à zéro
16
Registre d'instruction
11
Bloc de calcul des adresses mémoires
5
Code opération
Paramètres
Décodage du code opération
Etat de la liaison pour LIAC
Lignes d'adresses pour les mémoires MESI et MC
Autorisation de sortie
Commandes du séquenceur
Séquenceur de microprogramme
Adresse
Adresse
Autorisation de sortie
Mémoire de microprogramme PROM
256 x 58
58
Registre de micro-instruction
Commandes pour la gestion du calcul interne et des opérations de lecture, écriture pour les mémoires MESI et MC.
Adresse de branchement pour le microprogramme

3. Le processeur interne

Le processeur interne possède trois fonctions :
- la fonction de mémorisation et déroulement d'un programme,
- la fonction de mémorisation des variables et paramètres,
- la fonction de calcul.

3.1. Fonction de mémorisation et de déroulement d'un programme

La **figure 35** montre les organes qui assurent cette fonction.

Les instructions du CTR sont des macro-instructions dans le langage CORDIS. La lecture et le décodage de chaque instruction et de ses paramètres active un microprogramme spécifique qui élabore au cours du temps toutes les commandes nécessaires au calcul et à l'enchaînement avec la suite du programme. La lecture et le décodage d'une instruction se font pendant l'exécution de l'instruction précédente. Les instructions sont de deux sortes :

a/ Les instructions simples, exemple : CEL i.

Dans ce cas, le seul paramètre à transmettre est le numéro de la cellule. Ces instructions occupent un mot de 16 bits dans la mémoire de programme.

b/ Les instructions doubles, exemple : LIA i,j,k.

Dans ce cas les paramètres à transmettre sont le numéro de la liaison et les numéros des cellules qui sont liées entre elles. Ces instructions occupent deux mots de la mémoire de programme.

3.1.1. Le compteur de programme

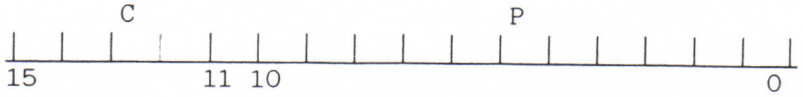
Le compteur de programme est un compteur synchrone qui fournit une adresse à la mémoire de programme. Il est mis à zero par une ligne spéciale, à la mise sous tension du LSI ou à chaque lancement d'un programme exécutable sur le LSI ou encore par programmation d'une initialisation générale pour le CTR. Un programme démarre toujours à l'adresse zéro. Le compteur est incrémenté une fois pour une instruction simple, deux fois pour une instruction double. L'instruction FIN remet le compteur à zéro à chaque cycle d'échantillonnage.

3.1.2. La mémoire de programme

C'est une mémoire RAM statique de taille 1024 x 16. Un seul programme exécutable est chargé à la fois dans MP. Un programme se déroule toujours avec un ordre croissant des adresses, c'est-à-dire qu'il n'y a pas de boucle dans le programme.

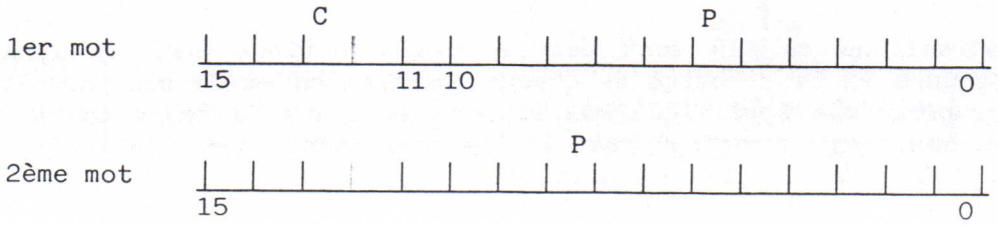
3.1.3. Le format des instructions

Le format d'une instruction simple est le suivant :



C est le code instruction,
P est le champ de paramètres.

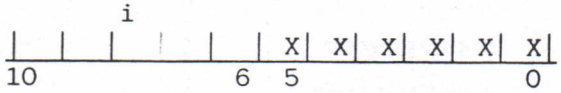
Le format d'une instruction double est le suivant :



C est le code instruction,
P est le champ de paramètres.

Le code instruction sur 5 bits permet de coder 32 instructions différentes. La liste des instructions s'arrête à 22 actuellement, il est possible de l'étendre ultérieurement. Le code instruction constitue une adresse pour une mémoire morte (PROM) de taille 32 x 8 qui délivre une adresse sur 8 bits pour la mémoire de microprogramme. Le champ de paramètres de l'instruction prend une signification différente selon le type d'instruction.

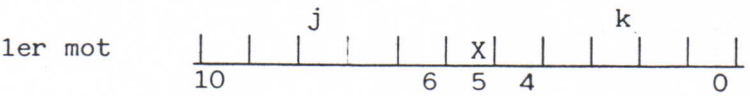
CEL i



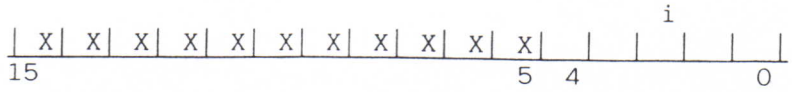
i est le numéro de la cellule, i=0.....31,
X est une donnée indéterminée (les lignes correspondantes peuvent prendre indifféremment l'état 1 ou l'état 0).

LIA i,j,k

LIAC i,j,k

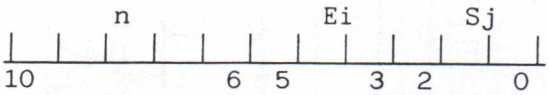


2ème mot



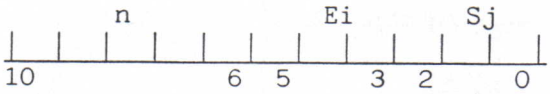
i est le numéro du module de liaison, i=0.....31,
j est le numéro du premier module matériel, j=0.....31,
k est le numéro du deuxièmè module matériel, k=0.....31.

PL, PLF n,Ei,Sj



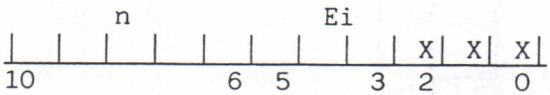
n est le numéro de module matériel attribué à PL,PLF, n=0.....31,
Ei est le numéro de la voie d'entrée, Ei=0....5,
Sj est le numéro de la voie de sortie, Sj=6 ou 7.

FL, FLP n,Ei,Sj



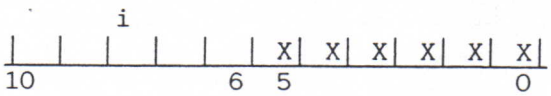
n est le numéro du module matériel lié à FL,FLP, n=0.....31,
Ei est le numéro de la voie d'entrée, Ei=0.....5,
Sj est le numéro de la voie de sortie, Sj=6 ou 7.

CLPX n,Ei



n est le numéro du module matériel ou de liaison dont on contrôle le paramètre, n=0.....31,
Ei est le numéro de la voie d'entrée, Ei=0.....5.

SORO, SOR1 i



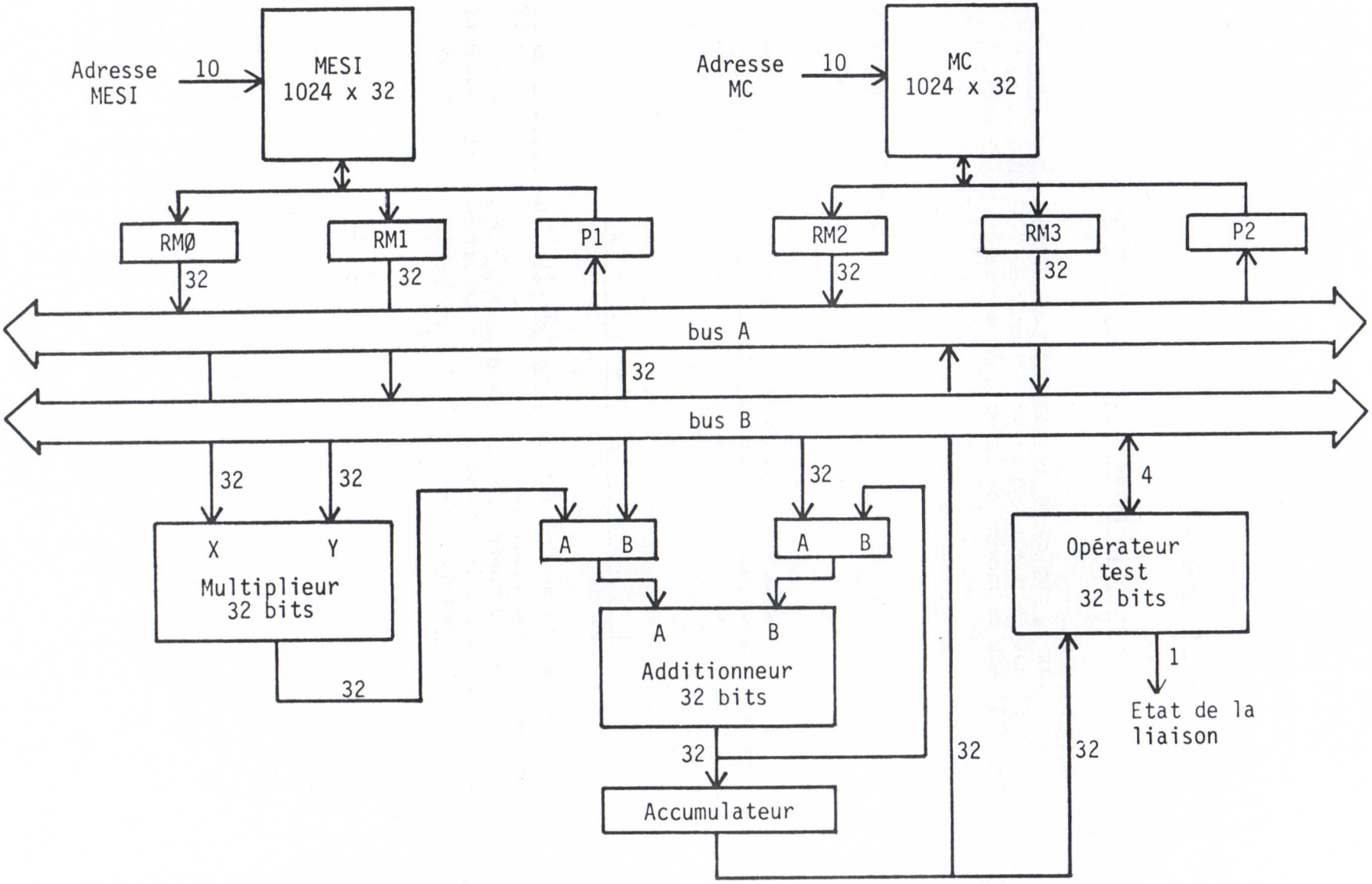
i est le numéro de la cellule dont on utilise la sortie, i=0.....31.

L'instruction FIN ne possède pas de paramètres.

3.1.4. La microprogrammation

La mémoire de microprogramme est une mémoire morte (PROM) de taille 256 x 58. Elle contient une suite de micro-instructions pour chaque instruction. Exemple : 9 micro-instructions pour CEL,
4 micro-instructions pour SORO,
14 micro-instructions pour LIA.

FIGURE 36.


Adresse MESI
10
MESI 1024 x 32
Adresse MC
10
MC 1024 x 32
RMØ
RM1
P1
RM2
RM3
P2
32
bus A
bus B
X
Y
Multiplieur 32 bits
A
B
Additionneur 32 bits
Accumulateur
Opérateur test 32 bits
4
1
Etat de la liaison

A chaque cycle de calcul un nouveau mot de 58 bits est lu dans la mémoire et le mot lu au cycle précédent est exécuté. Nous n'avons pas introduit de codage au niveau de la micro-instruction. Chaque bit correspond directement à une commande. Ainsi la taille de la micro-instruction est importante mais le temps d'exécution est réduit. La signification des microcommandes est la suivante :

8 bits servent à indiquer une adresse de branchement conditionnel dans le séquencement du microprogramme,

7 bits sont des lignes d'adresse partielle pour les mémoires MESI et MC,

14 bits servent à contrôler le calcul des adresses pour les mémoires MESI et MC, et le déroulement du programme et du microprogramme,

7 bits servent à contrôler les registres d'entrée et les registres de sortie des mémoires MESI et MC,

22 bits servent à commander tous les opérateurs.

3.2. Fonction de mémorisation des variables et paramètres

L'ensemble des organes qui assurent cette fonction ainsi que la fonction de calcul sont montrés sur la **figure 36.** Le processeur interne possède deux bus de données de 32 lignes chacun.

Dans le système CORDIS, les variables sont soit des forces, soit des positions, soit des vitesses. Les paramètres **utilisateur** sont des constantes de raideur, constantes de frottement, des longueurs au repos etc.. Pour le CTR les coefficients du calcul sont déduits des paramètres utilisateur par un calcul fait dans le LSI.

Considérons l'algorithme de la cellule :

$$X(n) = Fe(n) + A.X(n-1) + B.X(n-2) + C$$

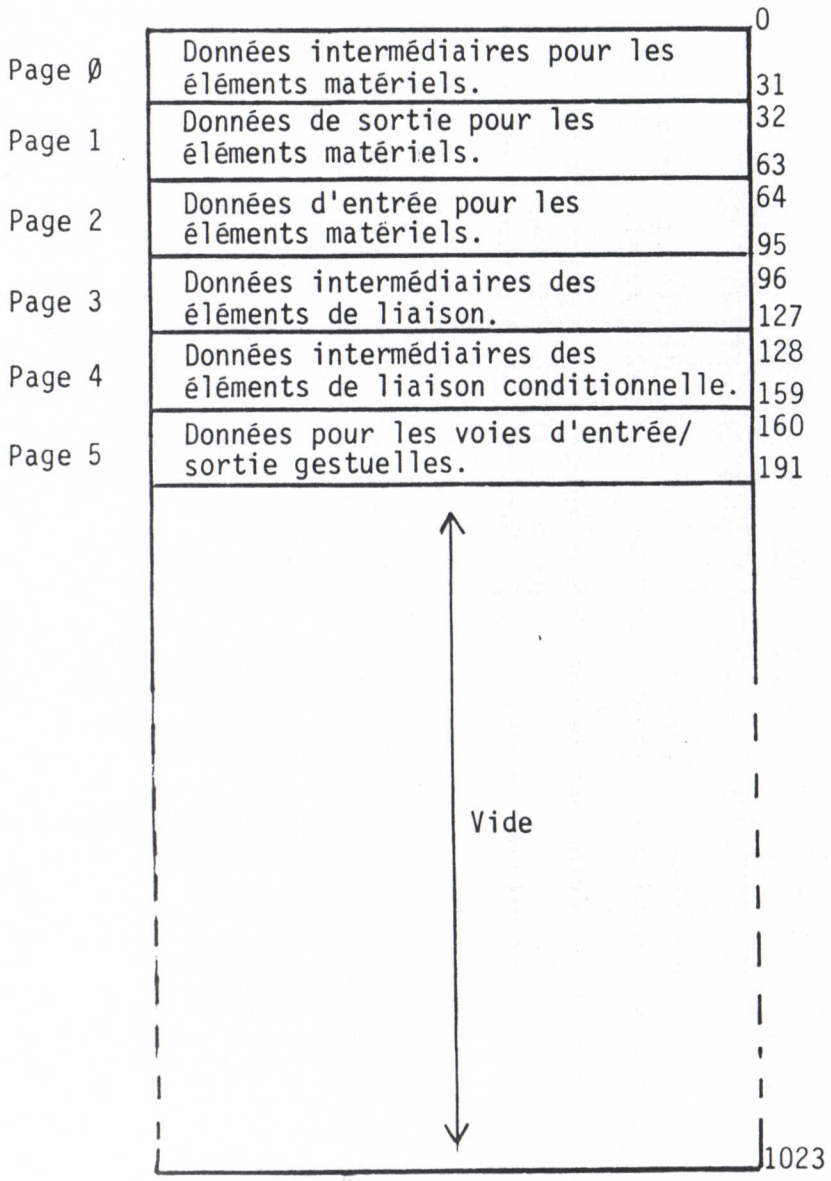
Les paramètres sont A, B, C et les valeurs des variables qui servent pour le calcul à l'instant n sont X(n-1), X(n-2), Fe(n).

Fe(n) est la variable d'entrée,
X(n-1) est la variable de sortie (à l'instant n-1),
X(n-2) est la variable intermédiaire (sortie à l'instant n-2).

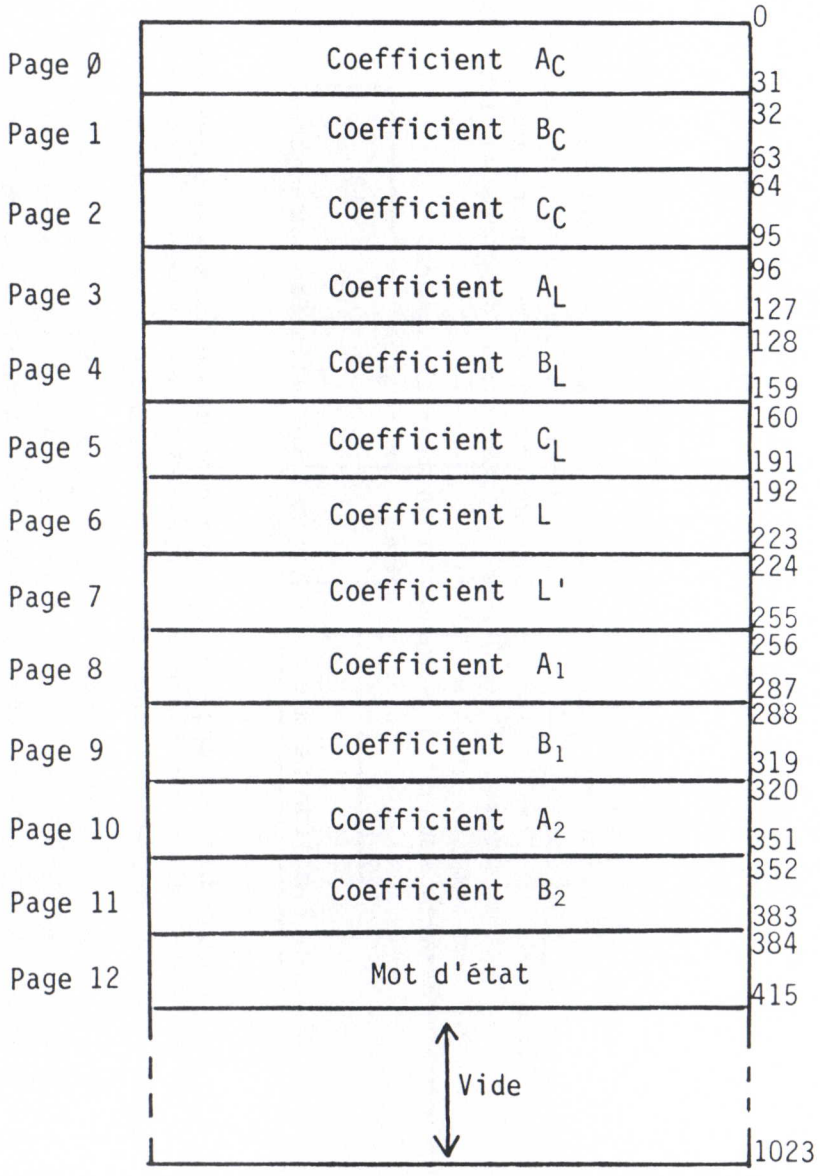
Le calcul à tout instant n a donc besoin de trois valeurs pour les variables et trois pour les paramètres, soit au total 6 valeurs différentes, donc 6 cases mémoires pour les stocker. En étudiant ainsi tous les algorithmes, nous avons déterminé la taille minimale des mémoires que nous utilisons. De plus, l'étude de l'ensemble des algorithmes nous montre, que deux variables ne sont jamais multipliées entre elles, de même que deux paramètres de calcul. La présence de deux mémoires différentes : l'une pour les variables et l'autre pour les paramètres, permet de lire simultanément deux valeurs pendant un cycle de calcul et de commencer la multiplication au cycle suivant. Les deux mémoires de données sont : la mémoire MESI (entrée-sortie-intermédiaire) et la mémoire MC (coefficients). Elles ont toutes les deux une taille de 1024 x 32. Leur structure est montrée sur la **figure 37.**

FIGURE 37.

Structure de la mémoire MESI



Structure de la mémoire MC



Elles sont structurées en zones ou pages ayant chacune 32 mots contigus. Dans MESI une page est destinée à un type de variable déterminé, par exemple "la page de la variable d'entrée pour la cellule". Le numéro i de la cellule (i=0,.....31), permet d'accéder dans cette page à la valeur de la variable d'entrée pour la cellule numéro i. Dans MC une page est destinée à un type de paramètre déterminé, par exemple "la page du paramètre A pour les modules de liaison". Le numéro i de la liaison (i=0,.....31), permet d'accéder dans cette page à la valeur du paramètre A pour le module de liaison numéro i. Les numéros des pages dans lesquelles des accès sont effectués à chaque cycle de calcul, sont indiqués pour chacune des mémoires par la micro-instruction courante.

Les mémoires de données du CTR sont donc très spécialisées. Pendant la phase de chargement, les données quantitatives sont chargées à des adresses prédéterminées. Pendant le calcul, les adresses mémoires sont calculées très facilement par juxtaposition d'un numéro de page, issu du microprogramme et d'un numéro de module, issu du champ de paramètres de l'instruction.

Chacune des mémoires a deux registres de type parallèle en sortie : un sur le bus A, un sur le bus B. Pendant qu'une valeur lue précédemment est envoyée à un opérateur, une autre lecture en mémoire est possible. Pour écrire dans l'une quelconque des mémoires on se sert uniquement du bus A.

3.3. Fonction de calcul

Nous appelons **opérateurs** les organes du processeur interne servant au calcul proprement dit. Nous distinguons trois opérateurs dans le processeur interne.

3.3.1. L'unité arithmétique

Sa structure est montrée sur la **figure 35.**

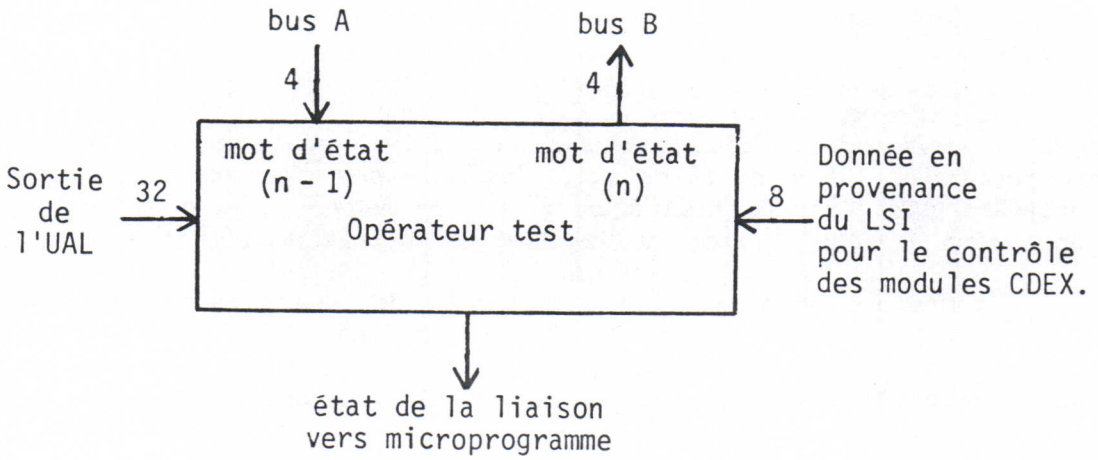
L'unité arithmétique fait des opérations d'addition et de soustraction sur 32 bits. Sa structure est tout à fait classique : mise en parallèle d'unités identiques opérant chacune sur 4 bits et report accéléré de la retenue. Sur chacune des entrées A et B de l'unité arithmétique (voir figure) il y a un registre de type parallèle sur 32 bits possédant à son tour deux entrées. L'entrée A du registre A est reliée à la sortie du multiplieur; l'entrée B est reliée au bus A. L'entrée A du registre B est reliée au bus B, l'entrée B est reliée à la sortie de l'unité arithmétique elle-même. Cette configuration permet de faire des sommes de produits sans avoir de cycle vide entre deux sommes. La sortie de l'unité arithmétique est en outre envoyée vers le bus A pour l'écriture en mémoire et vers l'entrée de l'opérateur test.

3.3.2. L'opérateur test

L'opérateur test est conçu pour déterminer l'état de la liaison à l'instant n, pour une liaison conditionnelle (**Figure 38**). Il est activé chaque fois que l'une des instructions UNL, ECH, CDEX est exécutée. Pour UNL ou ECH, l'opérateur test ne tient pas compte de la donnée sur huit bits provenant du

LSI. Cette donnée ne concerne que les modules CDEX. Les microprogrammes correspondant aux modules de liaison conditionnelle s'effectuent en deux phases.

FIGURE 38.



Première phase : Détermination de l'état de la liaison

La détermination de l'état de la liaison se fait selon l'organigramme de la figure 16. L'opérateur test est capable d'effectuer les tests suivants :

- égale à zéro,
- supérieure ou égale à zéro,
- inférieure ou égale à zéro,
- inférieure à zéro,
- supérieure à zéro,

ainsi que des opérations logiques sur les résultats de plusieurs tests sur des données différentes. Par exemple : le mot 1 est supérieur ou égal à zéro
et
le mot 2 est inférieur ou égal à zéro.
Les données sur lesquelles un test est effectué, proviennent de l'opérateur arithmétique et logique.

Une fois que les conditions de liaison sont testées, en tenant compte des seuils L et L', pour les modules UNL et ECH, l'opérateur test tient compte de l'état précédent de la liaison, et compare le signe de la variable (X1 - X2) au cycle n avec son signe au cycle n-1. L'élaboration de l'état de la liaison dure plusieurs micro-instructions. Pour le module CDEX, les conditions de la liaison ne sont pas élaborées par l'opérateur test. Celui-ci tient compte simplement de la donnée sur huit bits qui lui parvient du LSI. La suite de l'organigramme se déroule de la même manière que pour les deux autres modules. Le mot d'état de 4 bits contient les informations suivantes à chaque cycle :

bit 0 : indique l'état de la liaison,
bit 1 : indique le signe de X1 - X2,
bit 2 : est l'inverse du bit 1,
bit 3 : indique si X1 - X2 est nul.

Deuxième phase : calcul

La liaison se trouve maintenant dans l'état 1 ou dans l'état 2 comme déterminé dans la première phase. Le calcul se fait avec une série de paramètres correspondant à l'état de la liaison, et selon un algorithme identique à celui d'une liaison simple.

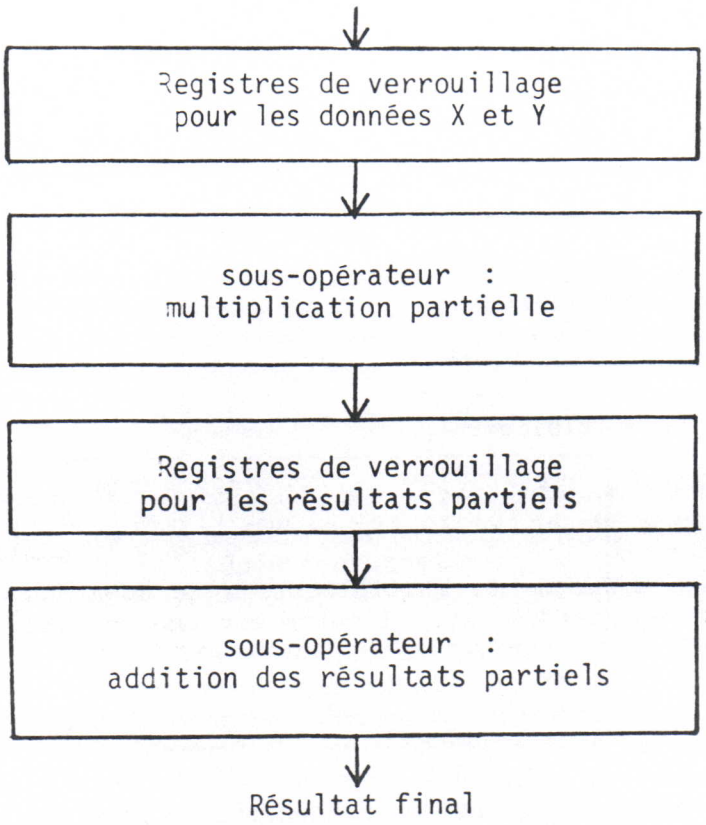
L'état de la liaison constitue une entrée pour le séquenceur de microprogramme, qui peut effectuer un branchement conditionnel. L'adresse de branchement est donnée par la micro-instruction courante.

3.3.3. Le multiplieur

La structure du multiplieur, considéré comme un opérateur pipe-line, apparaît sur la **figure 39a** et sur la **figure 39b.**

Le multiplieur effectue la multiplication de deux données de 32 bits. Une des données est un coefficient, l'autre est une variable. L'entrée X du multiplieur est reliée au bus A, l'entrée Y est reliée au bus B. Il est construit avec 4 boitiers de multiplieurs, de type parallèle 16 bits, qui calculent des résultats partiels en opérant séparément sur la partie basse et sur la partie haute de 2 mots de 32 bits. Le principe de la multiplication apparaît sur la **figure 40.** Les 4 multiplieurs calculent simultanément. Les résultats partiels sont additionnés en respectant le rang de chaque chiffre et avec un report de la retenue pendant l'addition. L'étage de multiplication et l'étage d'addition sont séparés par des registres de type parallèle. Le multiplieur est conçu comme un opérateur pipe-line à deux étages. Alors que la multiplication totale dure 2 cycles de calcul (300 ns), un résultat est fourni à chaque cycle (150 ns).

FIGURE 39.


Registres de verrouillage pour les données X et Y
sous-opérateur : multiplication partielle
Registres de verrouillage pour les résultats partiels
sous-opérateur : addition des résultats partiels
Résultat final

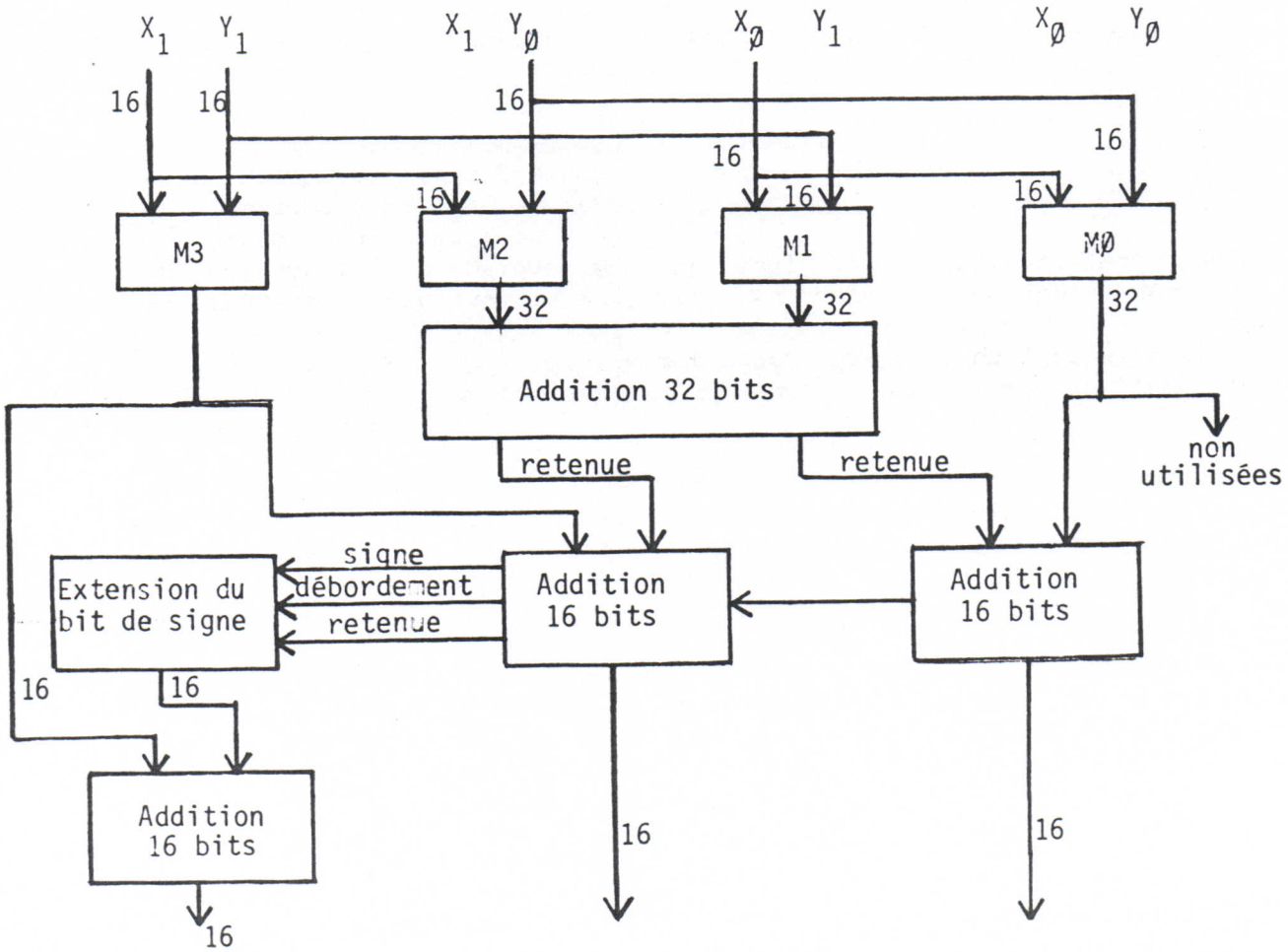

X1 Y1
X1 Yø
Xø Y1
Xø Yø
16
M3
M2
M1
Mø
32
Addition 32 bits
retenue
non utilisées
signe
débordement
retenue
Extension du bit de signe
Addition 16 bits
Addition 16 bits
Addition 16 bits

FIGURE 40.

$X = X_{31} \ldots X_{16}X_{15} \ldots X_0$

$Y = Y_{31} \ldots Y_{16}Y_{15} \ldots Y_0$

Soit le résultat : $P = P_{63} \ldots\ldots P_0$ 64 bits avant troncature

Considérons maintenant les multiplications partielles :

MØ $\quad X_{15} \ldots X_0$

$\times \quad Y_{15} \ldots Y_0$

$SP_Ø = P_{0,31} \ldots P_{0,16}\, P_{0,15} \ldots P_{0,0}$ sous-produit Ø sur 32 bits est un nombre non signé

M1 $\quad X_{15} \ldots X_0$

$\times \quad Y_{31} \ldots Y_{15}$

$SP_1 = P_{1,31} \ldots P_{1,0}$ sous-produit 1 sur 32 bits est un nombre signé

M2 $\quad X_{31} \ldots X_{15}$

$\times \quad Y_{15} \ldots Y_0$

$SP_2 = P_{2,31} \ldots P_{2,0}$ sous-produit 2 sur 32 bits est un nombre signé

M3 $\quad X_{31} \ldots X_{15}$

$\times \quad Y_{31} \ldots Y_{15}$

$SP_3 = P_{3,31} \ldots P_{3,0}$ sous-produit 3 sur 32 bits est un nombre signé

P sera obtenu en additionnant les résultats partiels :

zéros ⟶ $P_{0,31} \ldots P_{0,16}P_{0,15} \ldots P_{0,0}$

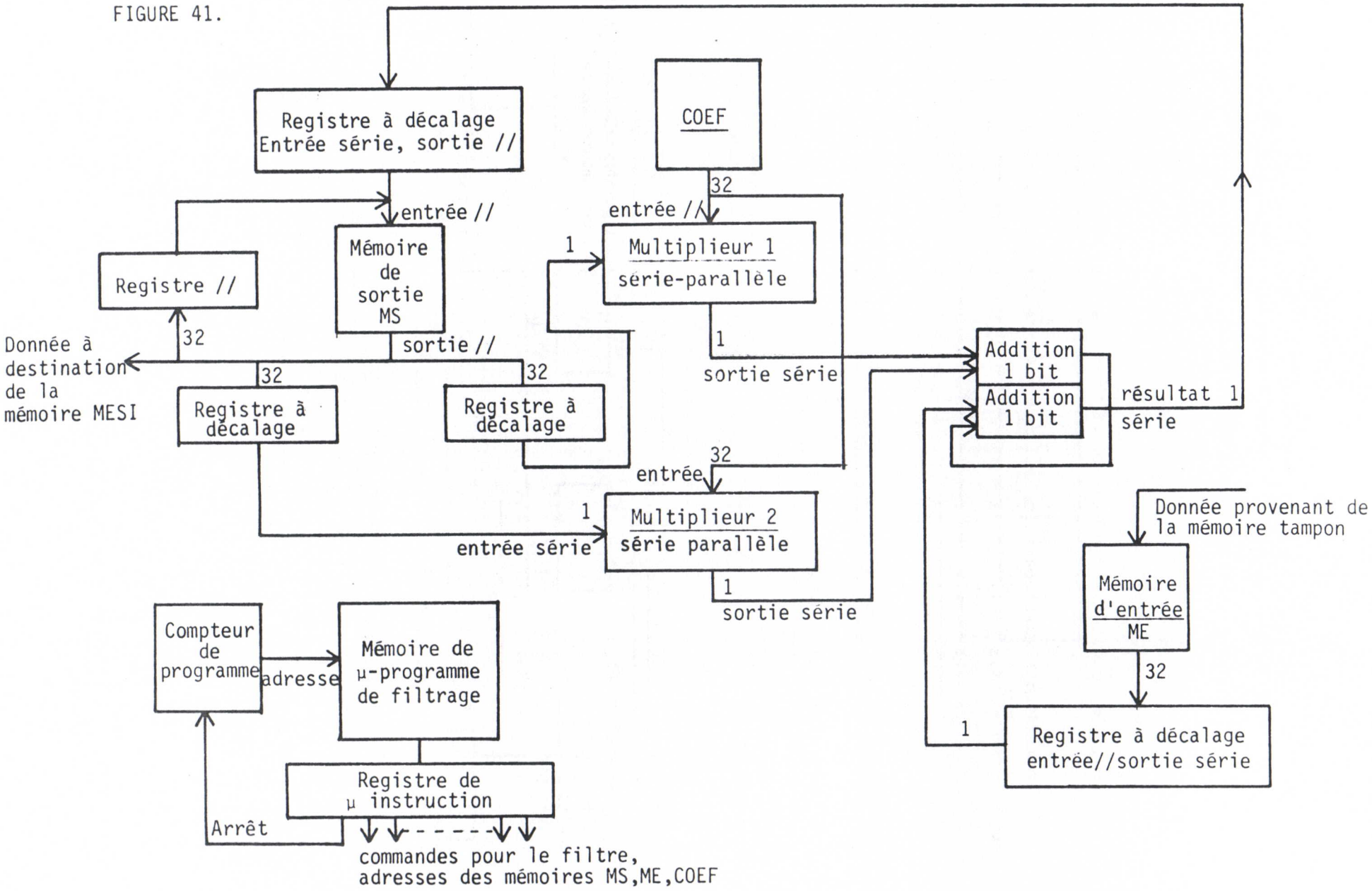
extension de signe ⟶ $P_{1,31} \ldots P_{1,16}P_{1,15} \ldots P_{1,0}$

extension de signe ⟶ $P_{2,31} \ldots P_{2,16}P_{2,15} \ldots P_{2,0}$

$P_{2,31} \ldots P_{3,16}P_{3,15} \ldots P_{3,0}$

résultat sur 64 bits

FIGURE 41.


Registre à décalage
Entrée série, sortie //
COEF
entrée //
entrée //
32
Mémoire
de
sortie
MS
1
Multiplieur 1
série-parallèle
Registre //
Donnée à
destination
de la
mémoire MESI
32
sortie //
32
32
1
sortie série
Addition
1 bit
Addition
1 bit
résultat 1
série
Registre à
décalage
Registre à
décalage
32
entrée
1
Multiplieur 2
série parallèle
entrée série
Donnée provenant de
la mémoire tampon
Mémoire
d'entrée
ME
1
sortie série
Compteur
de
programme
adresse
Mémoire de
µ-programme
de filtrage
32
1
Registre à décalage
entrée//sortie série
Registre de
µ instruction
Arrêt
commandes pour le filtre,
adresses des mémoires MS,ME,COEF

4. Le processeur d'entrée-sortie

Sa structure est montrée sur la **figure 41.**

Le processeur d'entrée-sortie effectue un filtrage numérique passe-bas du second ordre sur les 6 entrées gestuelles. Il y a un seul filtre, les entrées sont multiplexées. La fréquence de coupure du filtre (- 3 DB) est fixée à 50 Hz. Son équation est la suivante :

$$y(n) = x(n) + A.y(n-1) + B.y(n-2) \quad .$$

Nous utilisons le filtre dans le cas de l'amortissement critique. Alors la condition :

$$A^2 = -4 . B$$

doit être réalisée.

Soit $\omega_c = 2 \pi \frac{F_c}{F_e}$ la pulsation de coupure réduite.

Alors :

$$A = 2 \frac{(\sqrt{2} - \cos \omega_c) - ((\cos \omega_c - \sqrt{2})^2 - (\sqrt{2} - 1)^2)^{1/2}}{\sqrt{2} - 1}$$

$$B = - \frac{A^2}{4}$$

$$GT_0 = \frac{4}{(2 - A)^2} \quad , \quad GT_0 \text{ est le gain à l'origine.}$$

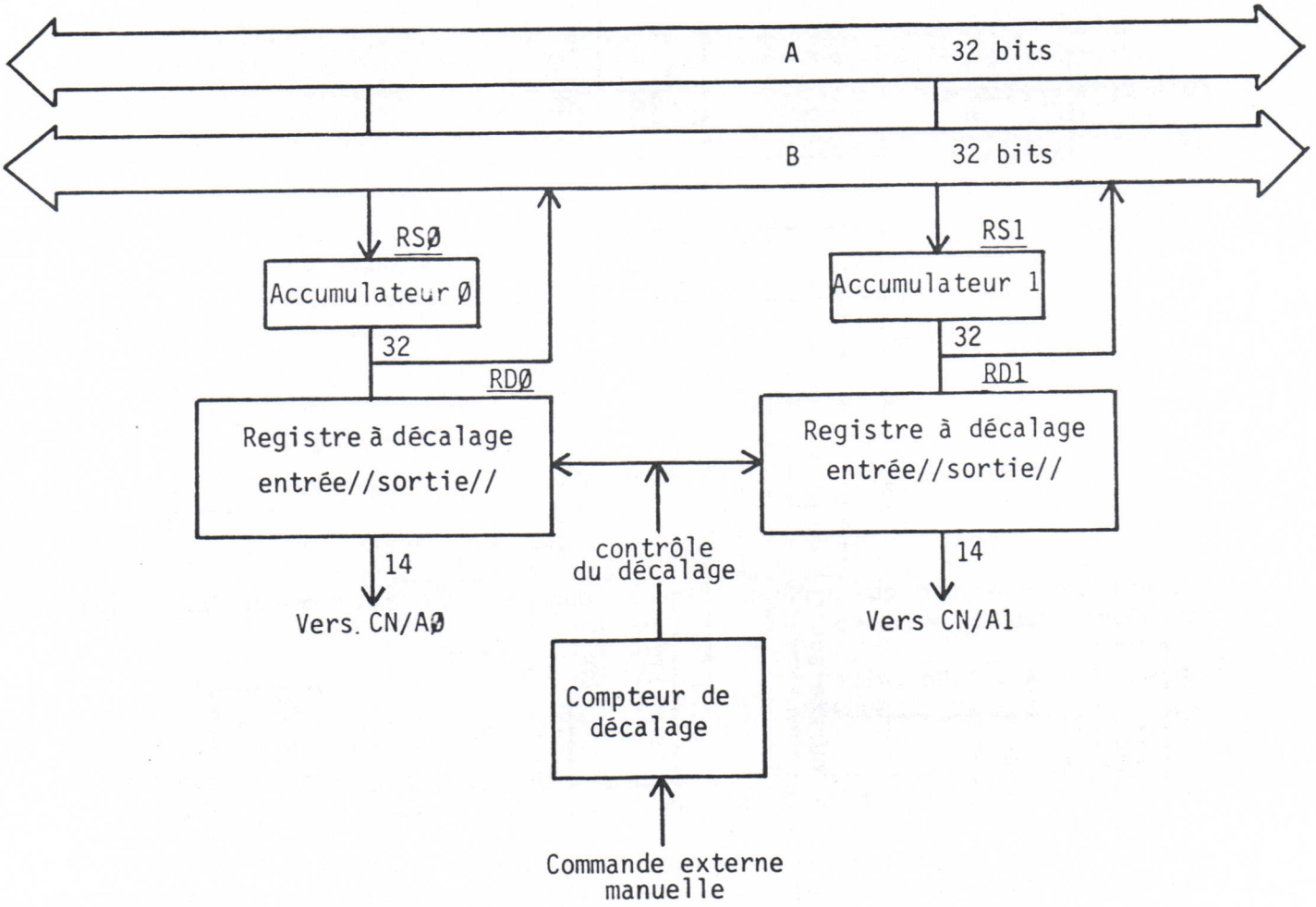
Les valeurs standard que nous utilisons pour les coefficients du filtre sont calculées pour $F_c = 50$ Hz , $F_e = 25{,}6$ kHz .

Alors $A = 1{,}962242603$,
$B = -0{,}96259897$.

Le filtre fonctionne de la manière suivante : dans la mémoire de coefficients, la valeur de B est lue et chargée dans le premier multiplieur. En même temps, la valeur de l'échantillon y(n-2) est lue dans la mémoire de sortie. Cette valeur est chargée dans l'un des deux registres à décalage. Pendant les cycles de calcul suivants, le registre à décalage va charger la valeur de y(n-2), bit par bit, dans le multiplieur 1, en commençant par le bit de poids le plus faible. Au cycle suivant le chargement du coefficient B dans le multiplieur 1, le coefficient A est lu, et chargé dans le multiplieur 2. En même temps la donnée y(n-1) se trouvant dans la mémoire de sortie, est chargée dans l'autre registre à décalage. Après conversion en série, cette donnée est chargée dans le multiplieur 2, bit par bit.

Les multiplieurs délivrent un bit du résultat à chaque cycle de calcul. Les bits de même rang dans chacun des résultats partiels B.y(n-2) et A.y(n-1) sont additionnés au fur et à mesure, avec report de la retenue. La donnée x(n) se trouvant dans la mémoire d'entrée est convertie en série et additionné bit par bit au résultat partiel B.y(n-2) + A.y(n-1). Le résultat final y(n),

FIGURE 42.


A
32 bits
B
32 bits
RSØ
Accumulateur Ø
32
RDØ
RS1
Accumulateur 1
32
RD1
Registre à décalage
entrée//sortie//
Registre à décalage
entrée//sortie//
14
Vers. CN/AØ
contrôle
du décalage
14
Vers CN/A1
Compteur de
décalage
Commande externe
manuelle

obtenu sous forme de donnée série, est convertie en donnée parallèle et chargé dans la mémoire de sortie. Ce résultat reste dans MS pendant deux cycles d'échantillonnage.

Le fonctionnement du filtre est identique pour les six voies. Le séquencement se fait par un microprogramme qui fournit les commandes pour les multiplieurs, les registres, les mémoires. La dernière micro-instruction, pour la sixième voie, arrête le microprogramme et met le compteur de microprogramme à zéro. Après la période de transfert du cycle d'échantillonnage suivant, celui-ci va redémarrer quand il recevra le signal "fin de transfert".

Toutes les données y compris les coefficients du filtre sont représentées sur 32 bits.

5. Le processeur de sortie sonore

Sa structure est montrée sur la **figure 42.**

Le processeur de sortie sonore possède 2 registres, de type parallèle, RSO, RS1 de 32 bits, qui ont leurs entrées sur le bus A et leurs sorties sur le bus B. Ils servent d'accumulateurs pour les voies de sortie. A chaque cycle après la période de transfert, leur contenu devient nul.

Si nous voulons que le signal de sortie soit la somme des positions de plusieurs cellules, sur une des voies, (par exemple les cellules numéros i et j, sur la voie numéro zéro), le programme devra contenir les instructions **SORO i, SORO j.** Le calcul correspondant fera à chaque cycle d'échantillonnage une lecture dans le registre RSO. La valeur contenue initialement dans ce registre est nulle. La valeur de la position de la cellule numéro i, sera ajoutée au contenu du registre. Ensuite, le nouveau contenu du registre RSO sera lu, et la valeur de la position de la cellule numéro j, sera ajoutée à ce contenu. Le résultat restera dans RSO jusqu'à la période de transfert, pendant laquelle elle passera dans le registre à décalage RDO. RSO sera remis à zéro.

Par un bouton de commande extérieur à 16 positions, l'utilisateur peut choisir parmi les 32 bits du mot de sortie, les 14 bits significatifs constituant le signal de sortie. Il dispose alors de 16 positions de cadrage différentes. La donnée correctement cadrée, est envoyée vers les convertisseurs numérique/analogique.

6. L'automate de transfert

C'est l'organe qui gère les transferts entre les trois processeurs. Il est microprogrammé. La période de transfert dure 10 cycles de calcul. A la fin de la période de transfert, le signal "fin de transfert" est envoyé aux trois processeurs.

Transferts entre la mémoire tampon et le processeur d'entrée-sortie

La mémoire tampon est une mémoire RAM, de taille 16 x 32. Les données qui s'y trouvent en stockage provisoire, correspondent aux fonctions gestuelles. Les 6 premières cases de la mémoire sont réservées aux 6 voies d'entrée de jeu. Le LSI peut accéder à la mémoire tampon, quel que soit l'état du CTR, pour y écrire des données. En cours de jeu ces données peuvent varier à la cadence de 100 Hz. A chaque cycle d'échantillonnage, ces données sont transférées dans la mémoire d'entrée du processeur d'entrée-sortie.

Transferts entre le processeur d'entrée-sortie et le processeur interne

A chaque période de transfert, les données constituant la sortie du filtre pour chacune des voies, sont envoyées dans la mémoire MESI. Ceci se fait simultanément avec les transferts précédents.

Transferts entre le processeur interne et la mémoire tampon

Les deux cases suivantes de la mémoire tampon, sont réservées aux données de retour vers le LSI. Celui-ci viendra les lire à la cadence de 100 Hz. Ces données passent à chaque cycle, de la mémoire MESI dans la mémoire tampon.

Transferts entre le processeur interne et le processeur de sortie

Il s'agit du transfert des données de sortie, calculées au cycle précédent. Elles passent dans les registres à décalage RD0 et RD1 du processeur de sortie sonore.

Enfin, la mémoire tampon contient aussi, la donnée de 8 bits, concernant les modules CDEX de liaison conditionnelle. Cette donnée est transférée de la mémoire tampon, vers un registre de 8 bits se trouvant à l'entrée de l'opérateur test.

C/ REALISATION

1. Méthode de réalisation. Câblage automatique

Le CTR est réalisé entièrement par la technique de **wrapping.** Le nombre total de boîtiers de circuits intégrés est égal à 500 environ. Ces circuits sont répartis sur trois grandes cartes de circuit imprimé, de dimension 400 mm x 200 mm, et une petite carte de longueur deux fois plus faible. A cela il faut rajouter, la carte interface qui a un format conforme aux normes du LSI. Sur une grande carte, le nombre de fils de connexion varie entre 1500 et 1800. Vu ce nombre élevé, le procédé de câblage manuel est à la fois un générateur important d'erreurs (qu'on ne peut quasiment pas repérer par la suite) et un procédé fastidieux et très coûteux en temps. Profitant d'une première expérience faite au laboratoire à propos de la reproduction du calculateur **SCH,** nous avons adopté la méthode de câblage automatique pour notre prototype. Les phases successives de cette méthode sont:

- la réalisation d'un ensemble de schémas électriques très détaillés où tout doit figurer et doit être numéroté,
- la description de chaque carte en langage UBU, (UBU est le programme d'aide à la conception de systèmes et de câblage automatique),
- l'utilisation du programme UBU, les corrections nombreuses et les modifications de l'implantation sur la carte,
- l'obtention d'un listing final où se trouvent tous les renseignements concernant la carte qu'on manipule et d'un ruban perforé qui va commander la machine de câblage,
- le câblage effectif. Le pistolet est automatiquement positionné et la longueur de fil à utiliser est indiquée.

Des possibilités d'erreur existent avec cette méthode. Elles sont dues en général à l'opérateur, mais elles sont minimes car des voyants et un signal sonore avertissent l'opérateur de chaque anomalie. L'apprentissage de la méthode est assez rapide. Nous pouvons affirmer que le procédé (y compris la correction des quelques erreurs qui ont eu lieu) est moins coûteux en temps et beaucoup plus intéressant à réaliser qu'un procédé manuel, même pour un prototype. En outre l'existence de nombreux documents archivés permet de reproduire d'autres exemplaires de la machine ou de la modifier aisément.

2. Conditionnement. Alimentation. Circuits imprimés

Le courant maximal consommé par le CTR est égal environ à 25A sous 5V. L'alimentation en courant continu de 5V est fournie par trois alimentations à découpage, d'une capacité de 10A chacune. La boîte du CTR contient deux étages séparés par une paroi. Les alimentations se trouvent à l'étage inférieur, et les cartes à l'étage supérieur. Le boîtier a des dimensions qui lui permettent d'être placé dans l'armoire où se trouvent le LSI et les convertisseurs numérique/analogique. Le refroidissement est assuré par deux ventilateurs placés à l'arrière. Les cartes sont fixées sur quatre cadres pivotants. Nous

avons conçu des circuits imprimés spéciaux que nous avons fait réaliser par un sous-traitant. Leurs dimensions externes sont conformes aux cadres utilisés. Elles comportent des colonnes équidistantes ayant chacune des trous métallisés équidistants. Cette configuration permet d'implanter un grand nombre de circuits, avec des nombres de broches différents. Des points d'alimentation et de masse sont prévus pour chaque circuit dans son environnement immédiat. Chaque colonne reçoit deux condensateurs de découplage, ce qui fait un condensateur en moyenne pour trois boîtiers. La numérotation des colonnes permet de repérer un circuit facilement.

3. Mise en route

La phase de test et de mise en route du CTR a duré 6 mois. Les tests ont été faits d'abord fonction par fonction en commençant par l'interface ; ensuite en couplant deux ou plusieurs fonctions entre elles. Pour chaque fonction testée les signaux externes à la fonction ont été générés soit en maintenant des niveaux constants, soit à l'aide d'appareils électroniques divers (générateurs de signaux carrés, d'impulsions etc.). Après l'interface, nous avons testé toutes les fonctions du processeur interne, réunies sur deux grandes cartes : la carte mémoire et la carte opérateur. Nous avons testé dans l'ordre les fonctions suivantes :

- l'écriture et la lecture dans les mémoires de données et dans la mémoire de programme,
- le séquencement du programme; le bon déroulement des commandes de démarrage, d'arrêt, de cycle unique, d'initialisation,
- le séquencement du microprogramme, pour les instructions sans branchement d'abord, avec branchement ensuite,
- le calcul des adresses pour les mémoires de données.

Nous avons vérifié ensuite le fonctionnement des opérateurs. Pour cela nous avons considéré les diverses opérations élémentaires dans l'ordre hiérarchique suivant :

- transfert des données de mémoire à mémoire,
- addition,
- multiplication,
- tests pour les liaisons conditionnelles.

Pour les transferts de données, nous avons utilisé les instructions PL, PLF, CLPX. En effet ces instructions ne mettent en jeu aucune autre opération. Pour vérifier les additions nous avons utilisé les instructions FL, FLP. L'instruction CEL nous a servi à vérifier et à mettre au point le multiplieur. Enfin les instructions UNL, ECH, CDEX nous ont permis de vérifier le bon fonctionnement de l'opérateur test. Chacune des instructions a été utilisée de manière isolée au moyen de petits programmes écrits en langage machine directement.

Après la mise en route du processeur interne, nous avons procédé à la mise en route des horloges internes et de l'horloge d'échantillonnage. Nous avons vérifié ensuite le fonctionnement de l'automate de transfert.

Après vérification du processeur de sortie sonore, nous avons utilisé l'instruction CEL pour obtenir des sons sur les deux voies de sortie. En dernier lieu nous avons vérifié le fonctionnement des filtres d'entrée.

Enfin le programme de P. LACORNERIE a permis de passer à la phase d'utilisation du CTR et au langage CORDIS en s'affranchissant du langage machine.

CONCLUSION

Comme nous l'avons déjà signalé dans la première partie du document, les travaux sur la définition du cahier des charges du CTR, ont commencé au début de l'année 1980, et les travaux pour la réalisation, un an après. Nous avons accordé beaucoup d'importance à la décision de démarrer la réalisation du CTR, car à partir de ce moment-là, nous avons délibérément écarté, toutes les idées pouvant remettre en cause, de manière fondamentale, le cahier des charges. Dans la phase de réalisation, les contraintes techniques ont joué un rôle quantitatif surtout, comme par exemple la diminution de la complexité de la structure ou la diminution du nombre d'entrées lentes. Nous allons examiner maintenant le produit fini sous deux aspects.

a/ Comme point de départ pour la définition du cahier des charges, nous avons considéré les objectifs fondamentaux, que nous avons exposés en six points dans la première partie de ce document. Les objectifs ainsi formulés, sont eux-mêmes le fruit d'un travail d'analyse des idées initiales et collectives au sujet du CTR. Le produit fini représente une réduction du projet initial, dûe aux contraintes techniques. L'une des questions que nous pouvons nous poser maintenant est : "Quel est l'écart entre le produit que nous avons fabriqué et celui qui répondrait intégralement au cahier des charges initial ?". Nous allons répondre à cette question, en plusieurs points :

- Limitation de la complexité de la structure pouvant être simulée en temps réel. A titre d'indication, nous pouvons dire qu'au départ nous pensions (naïvement) pouvoir simuler 64 cellules ou plus et autant de liaisons en temps réel, à 25,6 kHz. Nous avons essayé, dans ce but, de simplifier les algorithmes, d'introduire des techniques d'accélération des calculs, de supprimer les niveaux de codage intermédiaires. Mais nous avons rencontré des limites, imposées par la technologie et le format de calcul choisis. Nous avons du tenir compte aussi des contraintes d'encombrement, de coût, de consommation, etc.. Le CTR peut simuler actuellement une structure formée de 11 cellules et 10 éléments de liaison, avec une entrée lente et une sortie rapide, à 25,6 kHz. Il peut aussi simuler en temps réel à 6,4 kHz, une structure formée de 32 cellules et autant d'éléments de liaison.

- Suppression des modifications qualitatives de structure en temps réel. Bien que prévue dans la version 2 du logiciel CORDIS, cette fonction n'a pas encore été exploitée. Elle pose des problèmes de continuité de l'état d'une structure, après la modification. Avec le CTR nous avons la possibilité de préparer à l'avance plusieurs structures, que nous pouvons charger rapidement.

Par un geste unique, par exemple une action sur une touche, nous pouvons arrêter le calcul en cours, charger une nouvelle structure et redémarrer le calcul, le tout étant fait rapidement. Par ailleurs nous pouvons ramener très souvent les cas de modification qualitative de structure à des modifications de paramètres.

- Suppression du contrôle rapide de paramètres. De même que précédemment, cette fonction prévue dans la version logicielle, n'a pas encore été exploitée. Elle permet de contrôler un paramètre, par exemple la raideur d'une liaison, par une variable, par exemple la position de l'une des cellules rattachées à l'élément de liaison. Elle permet d'introduire ainsi des non-linéarités. Nous avons supprimé cette fonction dans le CTR, parce qu'elle ne nous a pas paru indispensable dans un premier temps.

- Limitation du nombre de voies d'entrée-sortie lente. La limitation du nombre des entrées provient de la contrainte du filtrage. Avec six voies dont deux rétroactives, si nous utilisons deux entrées d'excitation, il ne reste plus que quatre voies pour le contrôle de paramètres. Or si nous voulons contrôler la raideur d'une liaison par exemple, il nous faut agir sur les paramètres A et C de la liaison, donc utiliser deux voies. Si nous voulons contrôler la constante de frottement, il nous faut agir sur les paramètres A et B, donc utiliser encore deux voies. Nous voyons que nous serons vite limités, en ce qui concerne le contrôle de paramètres. Avec la structure actuelle du processeur d'entrée-sortie et de l'automate de transfert, ces deux organes travaillent toujours pour un même nombre de voies (6 en entrée et 2 en sortie), quel que soit le nombre effectif de voies d'entrée-sortie utilisées. Pour les voies non utilisées, les données traitées n'ont aucune signification, et ne sont pas lues par le processeur interne. Cette structure ne permet pas d'optimiser le temps de calcul effectif pour une simulation, en fonction du nombre de voies d'entrée-sortie utilisées. Une meilleure structure, qui fonctionnerait selon le nombre de voies effectif, permettrait par exemple de diminuer le nombre des accès pour des structures complexes, ou d'augmenter ce nombre pour des structures plus simples. Comme exemple nous pouvons considérer les cas d'une corde formée de nombreuses cellules et éléments de liaison, que nous pourrions pincer en un seul endroit (variable), et dont nous pourrions contrôler un seul paramètre à la fois, et de plusieurs cellules indépendantes, excitées séparément, dont nous pourrions contrôler séparement la raideur. Dans le premier cas les accès sont en nombre limité, dans le second ils sont nombreux.

b/ Considérant l'évolution de notre pensée (collective), parallèlement à l'élaboration du CTR, nous pouvons nous poser la question suivante : "Si nous voulions refaire maintenant une machine temps réel, quels seraient nos objectifs ?".

Le CTR actuel a été conçu comme une machine uniquement "temps réel", fonctionnant comme esclave du LSI, dans le cadre de l'exploitation du système CORDIS tel qu'il était défini à l'époque. Il ne possède aucun moyen de stockage des échantillons sonores. L'opérateur test a été défini à partir des liaisons conditionnelles particulières. Il pourrait permettre l'implantation d'autres liaisons conditionnelles, à condition que celles-ci suivent le même organigramme, la différence se situant au niveau des conditions de liaison. Par ailleurs la nature de l'évolution de ces liaisons montre qu'il est pratiquement impossible d'intégrer cette évolution dans le CTR.

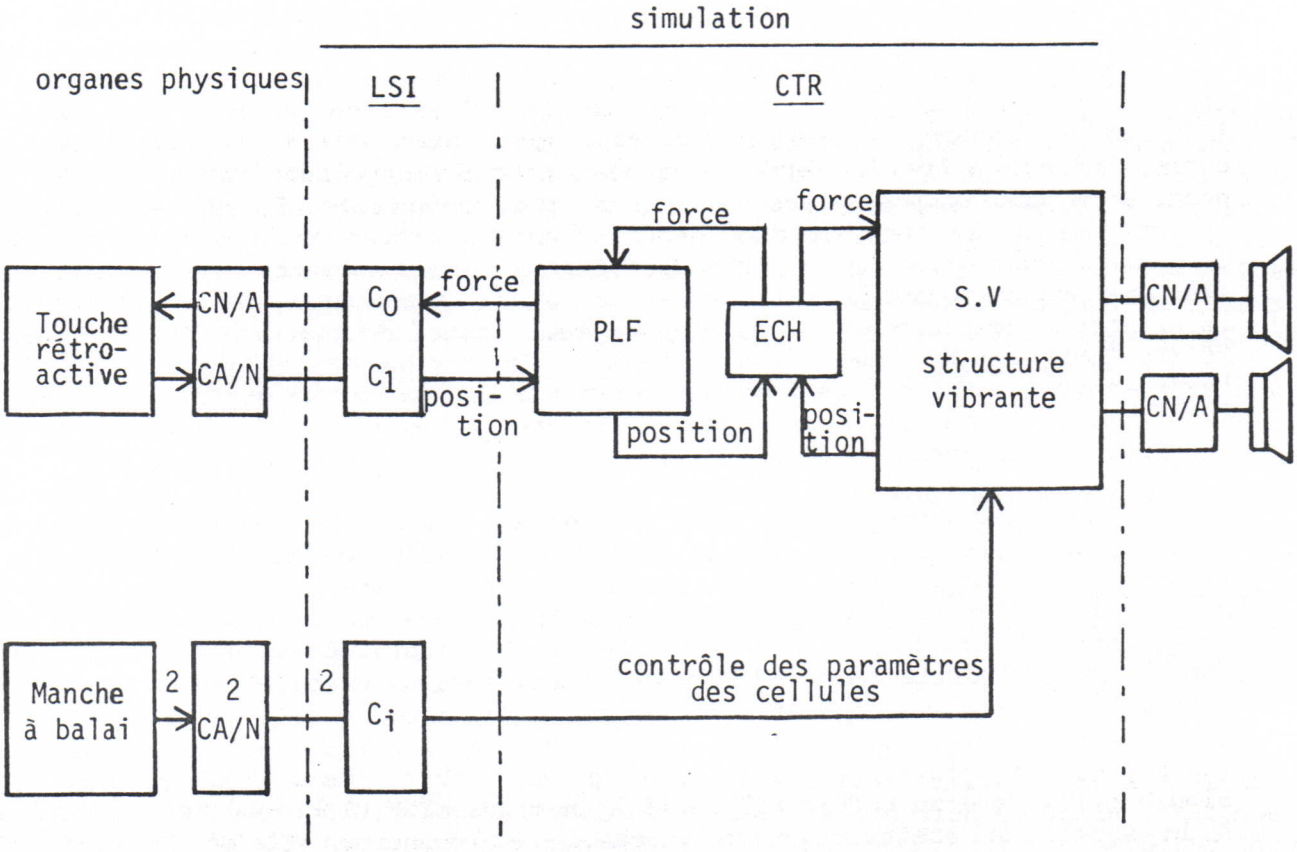
Si nous avions à redéfinir un projet maintenant, nous définirions avant tout un opérateur test plus général et non strictement câblé. Nous définirions pour la machine des cas de fonctionnement en dehors des limites prévues au départ, c'est à dire une possibilité de fonctionnement en temps différé avec des moyens de stockage pour les échantillons sonores, et une possibilité d'évolution pour l'évaluation du sytème CORDIS lui-même. Mais nous attirons l'attention sur le fait que nous ne pouvons pas demander à une même petite machine d'être à la fois une machine temps réel performante (rapide) et une machine pouvant traiter des gros volumes de calcul en temps différé. Il faut réaliser un compromis. Quand le volume d'informations à traiter augmente, il faut augmenter la taille des mémoires locales. Alors les temps d'accès aux mémoires augmentent aussi, les cycles de calcul deviennent plus longs et la rapidité diminue.

Tel qu'il est actuellement, nous nous proposons d'utiliser le CTR pour un certain nombre d'essais. Si l'idée de ces essais existe depuis longtemps, nous n'avons pas pu faire d'exploration systématique jusqu'à maintenant, car la contrainte du calcul en temps différé devient à la longue décourageante. Les premiers essais que nous nous proposons de faire avec le CTR concernent l'étude systématique de la perception sonore reliée à la causalité de l'événement sonore. Nous envisageons par exemple le problème de la correspondance entre la forme du réseau pour la structure vibrante, et la pertinence sonore, les autres paramètres étant invariants. De même nous posons le problème de la correspondance entre l'excitateur et la perception sonore, le geste d'excitation ou le geste de modification et la perception sonore. Un deuxième aspect constitue plus une situation de véritable "jeu" en temps réel. Une "structure-instrument" étant donnée, nous essaierons de la faire "s'exprimer" par le jeu. Il s'agit aussi de trouver les structures les plus "expressives". Enfin nous pouvons imaginer tous les cas de simulation, compte tenu des contraintes du CTR (limitation en complexité, liaisons conditionnelles particulières etc.).

Pour finir nous allons présenter un cas concret de simulation réalisée avec le CTR. Le programme a été fait par P. LACORNERIE. Il s'agit d'une **structure vibrante** dont nous pouvons choisir entièrement les aspects qualitatifs et quantitatifs. La structure vibrante est **excitée** en un de ses points par une excitation de type **pincement.** Le point d'excitation peut être choisi par l'utilisateur. Le geste d'excitation s'applique à la **touche rétroactive.** Un seul contrôle de paramètres physiques est possible pour la structure vibrante. Il s'agit des paramètres K **ou** Z des cellules. La **figure 43** montre cette simulation à l'aide de diagrammes fonctionnels. Le pincement est simulé à l'aide d'un module PLF relié à un module ECH d'un coté, et de l'autre à la sortie de position de la touche. Les grandeurs C1 et C2 sont des constantes de calibrage.

Bien qu'ayant posé la question : "Si nous faisions un CTR maintenant, comment serait-il défini ?", nous devons signaler que nous n'envisageons pas la construction d'un deuxième prototype dans l'immédiat. L'utilisation du CTR actuel de la manière que nous venons de définir, conjointement à un travail algorithmique à l'aide d'un processeur vectoriel, doit nous permettre d'avancer dans notre travail de recherche. Nous pourrons néanmoins commencer à envisager dès maintenant la définition d'un poste de travail CORDIS TEMPS REEL dans une perspective d'application à la création musicale.

FIGURE 43.



★ ★
★

BIBLIOGRAPHIE

LIVRES

P. SCHAEFFER (1966)
Le traité des objets musicaux
Seuil, Paris

P. SCHAEFFER (1973)
La musique concrète
Que sais-je, P.U.F.

M. MATHEWS, J.E. MILLER, F.R. MOORE, J.R. PIERCE, J.C. RISSET (1969)
The technology of computer music
The M.I.T. Press, USA

B. GOLD, L. RABINER (1975)
Theory and application of digital signal processing
Prentice Hall, New-York

B. GOLD, M. RADER (1969)
Digital processing of signals
Mac Graw Hill

A.V. OPPENHEIM, R. SCHAFER (1975)
Digital signal processing
Prentice Hall, New-York

J.P. MEINADIER (1971)
Structure et fonctionnement des ordinateurs
Larousse, Paris

J.M. BERNARD, J. HUGON (1979)
De la logique câblée aux microprocesseurs
Eyrolles, Paris

THESES

P. ABAUZIT (1980)
Etude et réalisation d'un calculateur rapide
Application au traitement, en temps réel, du signal de parole
Thèse de Docteur-Ingénieur, Grenoble

C. CADOZ (1979)
Synthèse sonore par simulation de mécanismes vibratoires
Application aux sons musicaux
Thèse de Docteur troisième cycle, Grenoble

D. DEGRYSE (1976)
Etude et réalisation d'un calculateur spécialisé
pour le traitement du signal de parole
Thèse de Docteur-Ingénieur, Grenoble

J.L. FLORENS (1978)
Coupleur gestuel interactif pour la commande et le contrôle
de sons synthétisés en temps réel
Thèse de Docteur troisième cycle, Grenoble

S. SANKAR (1974)
Conception de filtres numériques et analyse des erreurs
résultant de leur réalisation en arithmétique fixe
Thèse de Docteur-Ingénieur, Grenoble

RAPPORTS DE D.E.A

T. BERBERYAN (1979)
Application des processus récursifs à la synthèse sonore
D.E.A. d'électronique, I.N.P.G.

P. LACORNERIE (1982)
Dialogue homme-machine pour la synthèse sonore
par simulation de mécanismes vibratoires en temps réel
D.E.A. d'électronique, I.N.P.G.

ARTICLES ET RAPPORTS DE L'ACROE

C. CADOZ (1982)
Description de CORDIS-LOGICIEL version 2
Rapport interne A.C.R.O.E.

C. CADOZ (1981)
Le rapport instrumental dans l'ordinateur
appliqué à la création musicale
Rapport interne A.C.R.O.E.

C. CADOZ, J.L. FLORENS (1978)
Fondement d'une démarche de recherche informatique/musique
Revue d'acoustique N° 45

C. CADOZ, A. LUCIANI, J.L. FLORENS (1981)
Synthèse musicale par simulation des mécanismes instrumentaux
Transducteurs gestuels rétroactifs pour l'étude du jeu instrumental
Revue d'acoustique N° 59

C. CADOZ, A. LUCIANI, J.L. FLORENS, T. BERBERYAN (1982)
The control channels of instrumental playing in computer music
Real time in computer music Incidence on the choice of the basic models
International Computer Music Conference, Venise

ARTICLES

J. CHOWNING (1973)
The synthesis of complex audio spectra by means of frequency modulation
J.A.E.S., Vol. 21, 526-534

G.L. KRATZ, W.W. SPROUL, E.T. WALENDZIEWICZ (1974)
A microprogrammed approach to signal processing
I.E.E.E. Trans., C-23, 808-816

S.R. REDFIELD (1971)
A study in microprogrammed processors : A medium sized microprogrammed processor
I.E.E.E. Trans., C-20, 743-750

J.C. MAJITHIA (1976)
Some comments concerning design of pipeline arithmetic arrays
I.E.E.E. Trans., C-25, 1132-1139

F.G. HALLIN, M.J. FLYNN (1972)
Pipelining of arithmetic functions
I.E.E.E. Trans., C-21, 880-?

J.L. BAER (1976)
Multiprocessing systems
I.E.E.E. Trans., C-25, 1271-1277

F.F. KUO, J.F. KAISER (1966)
System analysis by digital computer
John Wiley, New-York

B. GOLD, C.M. RADER (1966)
Effects of quantization noise in digital filters
Proceedings Spring Joint Computer Conference

B. LIU, T. KANEKO (1969)
Error analysis of digital filters with floating point arithmetic
Proceedings I.E.E.E., vol. 57, N° 10

I.N. SANDBERG (1967)
Floating point roundoff accumulation in digital filter realization
Bell System Technical Journal 46

A.V. OPPENHEIM, C. WEINSTEIN (1969)
A comparison of roundoff noise in floating point and fixed point digital filter realizations
Proceedings I.E.E.E., vol. 57, N° 6

A.V. OPPENHEIM (1970)
Realization of digital filters using block-floating-point arithmetic
I.E.E.E. Trans., AU-18, 130-136

AUTORISATION DE SOUTENANCE

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

VU les rapports de présentation de Messieurs

. R. CARRE, Directeur de recherche et C. CADOZ

. H. COUTURIER, Ingénieur

Madame BERBERYAN Talin épouse DARS

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

Fait à Grenoble, le 19 novembre 1982

Le Président de l'I.N.P.-G.

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

P.O. le Vice-Président,

MADAME TALIN DARS - BERBERYAN

DOCTEUR - INGENIEUR

6 décembre 1982

I.N.P.G.-E.N.S. ELECTRONIQUE ET RADIOÉLECTRICITÉ DE GRENOBLE

RESUME

Il s'agit de la conception et de la réalisation d'un calculateur spécialisé pour la synthèse sonore en temps réel. Ce travail s'inscrit dans le cadre plus général des travaux de l'A.C.R.O.E. La méthode de synthèse utilisée implique une analyse mécanique préalable de mécanismes instrumentaux et la mise en oeuvre de processus de calcul récursifs pour leur simulation. Le calculateur que nous avons conçu et réalisé fonctionne en temps réel à 25,6 kHz, avec un format interne de 32 bits. Il possède une architecture de type multiprocesseur. Les trois processeurs travaillent de manière simultanée et sont microprogrammés. Le calculateur est connecté à un ordinateur hôte.

MOTS - CLES

SYNTHESE SONORE PAR ORDINATEUR - SIMULATION MECANIQUE - PROCESSEUR SPECIALISE - CALCULATEUR SPECIALISE - CALCUL EN TEMPS REEL - PROCESSUS RECURSIFS