

THESE

présentée

M0022/87A

A L'UNIVERSITE DES SCIENCES ET DE LA
TECHNOLOGIE HOUARI BOUMEDIENE

pour l'obtention du grade de

MAGISTER

EN ELECTRONIQUE DES SYSTEMES

par

المدرسة الوطنية المتعددة التقنيات
BIBLIOTHEQUE — المكتبة
Ecole Nationale Polytechnique

Fatima OULEBSIR Epouse BOUMGHAR

Conception et réalisation d'un système
multiprocesseur pour la simulation d'un processus
physique continu : Algorithmes numériques

Soutenue le 03 Juillet devant la commission d'examen :

MM. A. DAHEL

Président

A. ADANE

Examineurs

H. TEDJINI

R. TOUMI

R. OUIGUINI

B. SANSAL

Rapporteur

11002287

THESE

présentée

المدرسة الوطنية المتعددة التقنيات
BIBLIOTHEQUE — المكتبة
Ecole Nationale Polytechnique

**A L'UNIVERSITE DES SCIENCES ET DE LA
TECHNOLOGIE HOUARI BOUMEDIENE**

pour l'obtention du grade de

MAGISTER

EN ELECTRONIQUE DES SYSTEMES

par

Fatima OULEBSIR Epouse BOUMGHAR

**Conception et réalisation d'un système
multiprocesseur pour la simulation d'un processus
physique continu : Algorithmes numériques**

Soutenue le 03 Juillet devant la commission d'examen :

MM. A. DAHEL

Président

A. ADANE

Examineurs

H. TEDJINI

R. TOUMI

R. OUIGUINI

B. SANSAL

Rapporteur

A mon père

A ma mère

A mes frères et soeurs

المدرسة الوطنية المتعددة التقنيات
BIBLIOTHEQUE — المكتبة
Ecole Nationale Polytechnique

A Mohammed

A Yasmine

J'exprime tous mes remerciements à Monsieur le Professeur DAHEL pour m'avoir fait l'honneur , d'assumer la présidence de ce jury .

Que Monsieur le Professeur B.SANSAL soit assuré de toute ma gratitude et ma reconnaissance, pour m'avoir accueillie dans la post-graduation Electronique des Systèmes , pour m'avoir dirigée et , encouragée tout au long de ce travail .

J'adresse ma sincère reconnaissance à Monsieur BESSALAH Directeur du CDTA au CEN , pour avoir mis à ma disposition les moyens du Laboratoire Architecture des Systèmes , ainsi que pour ses précieux conseils .

J'exprime tous mes remerciements à Monsieur H.TEDJINI , pour avoir suscité en moi l'intérêt pour le multiprocessing et , pour m'avoir guidée dans la phase initiale du projet .

J'adresse tout particulièrement à Monsieur le Professeur R.TOUMI mes sincères remerciements pour ses conseils , et , ses critiques déterminantes , pour mener à terme ce projet .

Je ne manquerais pas d'exprimer mes vifs remerciements ainsi que ma reconnaissance à Monsieur R.OUIGUINI , pour l'immense aide qu'il a prodigué au travail mené .

J'exprime tous mes remerciements à Monsieur le Professeur ADANE , Directeur de l'Institut d'Electronique , pour avoir accepté de participer à ce jury .

Je remercie très chaleureusement Mademoiselle A.HELIFA pour sa participation sérieuse au projet et , à qui j'exprime ma profonde gratitude .

Je ne saurais pas oublier le précieux concours de Mademoiselle N.TAFAT , au cours de la réalisation du système ; qu'elle trouve ici ,le témoignage de ma sincère reconnaissance.

Enfin , je remercie toutes les personnes , en particulier le personnel du laboratoire Architecture des systèmes , qui ont de près ou de loin , aider à la mise en forme du projet .

- SOMMAIRE -

INTRODUCTION

CHAPITRE I

LES ARCHITECTURES PARALLELES

I. Classification des calculateurs à hautes performances

I.1. Structure de contrôle séquentiel

I.2. Structure de contrôle parallèle

II. Les réseaux d'interconnexion

II.1. Réseaux d'interconnexion statiques

II.2. Réseaux d'interconnexion reconfigurables

III. Quelques exemples de calculateurs spécialisés

CHAPITRE II

STRUCTURES MATERIELLES DES SYSTEMES MULTIMICROPROCESSEURS

I. Architecture des multimicroprocesseurs

I.1. Structure logique

I.2. Structure physique

I.3. Mode d'interaction

I.4. Mode de traitement

II. Evaluation des performances

II.1 Capacité de traitement

II.2 Partage des données communes

II.3. Délai de transmission des messages

III. Amélioration des performances

- CHAPITRE III -

DESCRIPTION DE LA MACHINE MULTIMICROPROCESSEUR

I. Définitions des entités intervenants dans un exécutif

II. Architecture du système multiprocesseur

II.1. Choix de la structure multiprocesseur

II.2. Présentation du système Multimu

1. Le processeur principal

2. Le processeur secondaire

3. Le bus interprocesseurs

4. Le logiciel de gestion

III. Description de la machine - Unités de traitement

III.1. Description matérielle

III.2. Structure logicielle

- CHAPITRE IV -

LES UNITES DE TRAITEMENT

I. Méthodes numériques

II. Traitement numérique

II.1 Position du problème

II.2 Choix de la méthode

II.3 Algorithmes numériques et organigrammes

II.4 Graphe de dépendance

- CHAPITRE V -

RESULTATS ET PERFORMANCES

CONCLUSION

ANNEXES

BIBLIOGRAPHIE

- I N T R O D U C T I O N -

La simulation de phénomènes physiques peut s'effectuer, à l'aide des équations du processus en utilisant des outils tels :

- le calculateur analogique .
- le calculateur hybride .
- le calculateur numérique classique .
- le calculateur spécialisé .

Les lois de base d'un processus physique continu , peuvent s'exprimer par un système d'équations différentielles représentant le modèle mathématique du processus à simuler . Il existe d'autres modes de représentation (telles les équations aux dérivées partielles , les fonctions de transfert , les équations d'état , etc ...) .

Le mathématicien peut , avec des conditions initiales préalablement posées , prouver qu'il existe une solution unique au système d'équations régissant le phénomène et décrivant le processus de façon complète . Mais il est souvent impossible , de trouver une expression analytique de la solution exacte , mis à part quelques cas simples . Les méthodes numériques peuvent approcher la solution de façon aussi précise qu'on le désire par augmentation du volume de calcul arithmétique .

Le système d'équations différentielles représentant le processus physique peut se ramener à la forme :

$$\left\{ \begin{array}{l} d(X_i(t))/dt = f_i(t, X_1(t), X_2(t), \dots, X_n(t), r(t)) \\ i=1, \dots, n \end{array} \right.$$

où $X_i(t)$ sont les variables d'état du système ,
et $r(t)$ les entrées ou commandes .

Nous élaborons tout d'abord , une méthode numérique approchée et , pour cela , nous supposons que les entrées peuvent être échantillonnées avant chaque évaluation des variables d'état [15] .

Le pas d'intégration numérique , h , doit être suffisamment petit pour approcher au mieux la solution . D'autre part , il ne doit pas être trop faible , pour pouvoir assurer la prérogative temps réel .

Pour chaque point du réseau déduit de la discrétisation du champ physique , plusieurs données numériques doivent être emmagasinées et constamment remises à jour dans la mémoire du calculateur .

De plus , la contrainte temps réel qui apparaît dans certains domaines d'application , a contribué à stimuler l'accroissement des performances des calculateurs .

La puissance des ordinateurs conventionnels n'a cessé de croître grâce , en partie , à l'utilisation de technologies très rapides qui ont permis de diminuer les temps de cycle (unité centrale et mémoire) . Néanmoins , les besoins en traitement de certaines classes de problèmes sont estimés à une opération flottante , sur 64 bits , par nanoseconde soit mille millions d'opérations flottantes par seconde (1000 MFlops) voire 10 000 MFlops !

La technologie ne peut à elle seule , répondre au problème posé . Des améliorations architecturales ont été alors apportées et ont permis d'atteindre un bon débit de calcul .

Il existe , à l'heure actuelle , des supercalculateurs fournissant de bonnes performances . Les plus puissants : CRAY XMP , VP 200 (FUJITSU) , S 810/20 (HITACHI) , SX-1 (NEC) ont des vitesses théoriques qui varient entre 400 et 800 MFlops. Or , ces machines ont été essentiellement conçues pour traiter des problèmes dont les algorithmes de résolution sont parallélisables . L'exécution d'algorithmes récursifs ou séquentiels constitue un facteur limitatif des performances globales de ces calculateurs .

Les problèmes à traiter sont souvent difficiles à vectoriser ou ne sont que partiellement vectorisables ; le rendement chute alors jusqu'à environ 10% (voire moins) .

Ainsi , il ne sert à rien de vouloir augmenter indéfiniment les performances vectorielles si l'on ne peut paralléliser le traitement scalaire .

L'une des voies possibles , pour faire du traitement scalaire en parallèle est le MIMD [2,4,6,10,27] , où un flot d'instructions traite un flot de données .

Dans la présente étude , nous nous sommes intéressés à la simulation de processus physiques continus dont le modèle mathématique nous a conduit à choisir un calculateur spécialisé de type parallèle . Un système multimicroprocesseur , le Multimu , a été conçu et réalisé pour lever les contraintes de

temps et de précision .

La machine Multimu est d'architecture parallèle de type MIMD . Le contrôle et l'exécution sont parallèles .

Ce système est constitué d'un processeur principal , ou maître , de quatre processeurs secondaires ou esclaves et d'une mémoire commune . Les échanges interprocesseurs s'effectuent à travers un bus commun [25,26] .

Les processeurs secondaires sont chargés d'intégrer, numériquement , un système d'équations différentielles .

Notre contribution a porté , essentiellement sur la conception et la réalisation des unités de traitement numérique de la machine Multimu .

Ce mémoire s'articule autour de cinq chapitres :

- le CHAPITRE I introduit les architectures parallèles .
- au CHAPITRE II , les structures matérielles des machines multimicroprocesseurs seront décrites .
- le CHAPITRE III expose le système Multimu , son architecture et son système d'exploitation .
- au CHAPITRE IV , les unités de traitement numérique seront détaillées .
- le CHAPITRE V fournira les résultats obtenus à l'aide de la machine Multimu . La comparaison avec la solution approchée théorique sera faite .

Quelques perspectives seront proposées en conclusion .

- CHAPITRE I -

LES ARCHITECTURES PARALLELES

Le développement récent des calculateurs à hautes performances répond à des besoins considérables en puissance de calcul de certaines classes d'applications .

Cette puissance concerne la précision des données , exprimées en virgule flottante sur 64 bits , ainsi que la contrainte de temps pour laquelle un nombre d'opérations arithmétiques effectuées à la seconde doit être imposé .

Les problèmes qui se posent dans les domaines de l'aérodynamique , la simulation nucléaire , la météorologie , la sismologie , le traitement d'images , la physique des plasmas ... , sont caractérisés par leur complexité numérique . Les temps d'exécution sont alors prohibitifs sur les machines conventionnelles (séquentielles) , de l'ordre de plusieurs jours , ce qui est souvent incompatible avec le problème lui-même (exemple du traitement d'images de satellite en continu) .

Les chercheurs ont pensé alors à de nouvelles architectures ; un certain degré de parallélisme a été ainsi introduit .

On compte dans le monde plus de 150 projets de machines à structure parallèle . Certains , ont dépassé le stade de projet en passant à la phase de la construction puis , dans quelques cas à l'étape de la commercialisation .

Les limites du traitement parallèle ne sont pas toutes connues et , des questions restent posées :

- quelle est l'architecture la plus adaptée pour certaines classes de problèmes et , quel est le coût et l'avantage d'une structure spécifique par rapport à celle d'un calculateur universel .

- quel est le réseau d'interconnexion adéquat entre les processeurs .

La classification de Flynn [20] concernant la nature de l'exécution d'un programme se divise en quatre catégories :

- * la machine SISD exécute une seule instruction sur une seule donnée . C'est la machine de Von Neumann .
- * la machine MISD , ou plusieurs instructions traitent une seule donnée ; l'utilité est perçue par les processeurs pipelines qui segmentent les calculs en stations consécutives .
- * la machine SIMD , où une seule instruction traite plusieurs données (donnée vectorielle) . Ce type de machine a une unité de contrôle pour les adresses et les processeurs . Les processeurs vectoriels , matriciels et réseaux en sont un exemple .
- * la machine MIMD où plusieurs données sont traitées simultanément par plusieurs instructions . Un ensemble de processeurs exécute chacun sa propre instruction ou programme .

Dans le présent chapitre , la classification des calculateurs parallèles à hautes performances est fournie avec

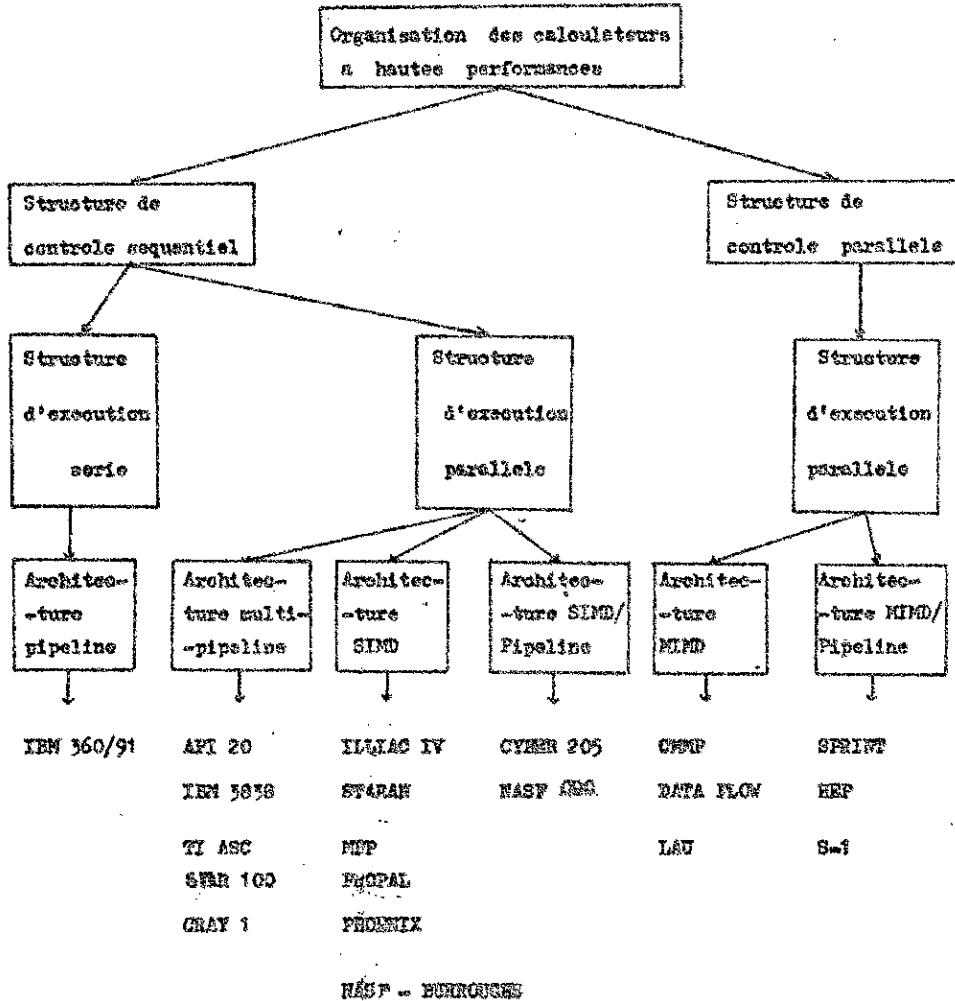
quelques réseaux d'interconnexions possibles et enfin, des exemples de calculateurs spécifiques seront cités.

I. CLASSIFICATION DES CALCULATEURS A HAUTE PERFORMANCE.

Le séquençage des instructions se fait dans un calculateur, grâce à sa partie contrôle; l'exécution de ces instructions se fait par la partie opérative. Ces deux parties sont appelées respectivement structures de contrôle et d'exécution.

La figure I.1 donne une classification arborescente [6] à trois niveaux, des calculateurs à hautes performances :

FIGURE I.1. CLASSIFICATION DES CALCULATEURS A HAUTES PERFORMANCES.



1.1. Structure de contrôle séquentiel .

L'interprétation des instructions se fait séquentiellement . Ce type de contrôle peut agir sur deux structures d'exécution : série et parallèle , lesquelles peuvent fonctionner soit en mode synchrone , soit en mode asynchrone .

1.1.1. Structure d'exécution série ou pipeline [6,7,27]:

Elle repose sur le découpage d'un travail en tâches élémentaires , exécutée chacune par un opérateur spécialisé , de la même façon qu'un produit manufacturé est traité en passant par plusieurs postes de travail dans une chaîne industrielle .

L'opérateur spécialisé est une unité d'exécution . Ainsi , l'exécution d'une tâche sur l'unité d'exécution correspondante , pourra être représentée par un délai Δt , généralement différent pour chaque unité . Un buffer placé à l'entrée de chaque unité reçoit les résultats produits par l'unité précédente (Fig.1.2) .

Le délai est le même pour toutes les unités lorsque le système pipeline est synchrone . Lorsque les unités sont complexes et les délais différents , un fonctionnement asynchrone est le plus adapté.

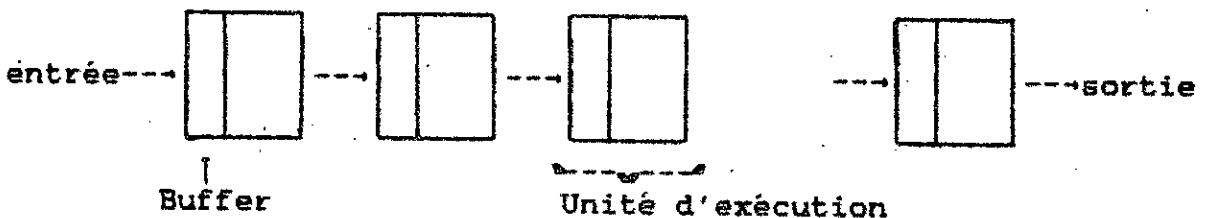


FIGURE 1.2. ARCHITECTURE PIPELINE .

Une unite d'exécution peut aussi bien être un circuit combinatoire qu'un véritable processeur spécialisé . Cela est précisé par l'application .

L'avantage du traitement pipeline est l'élimination du temps d'accès mémoire , lié au fait que la recherche d'opérandes s'effectue en même temps que l'exécution de l'instruction précédente .

Théoriquement , la puissance de calcul est multipliée par le nombre d'unités en service .

En pratique , le logiciel conditionne l'obtention de bonnes performances . Deux contraintes logicielles apparaissent :

* les instructions de branchement dégradent les performances d'un pipeline sur les instructions .

* les vecteurs courts ne permettent pas un fonctionnement optimal d'un pipeline sur les données régulières qui sont des vecteurs ou tableaux pleins .

* les données scalaires constituent un facteur limitatif des performances globales de la machine .

Pour atteindre un débit de 100 Mflops , la solution monopipeline nécessiterait des temps de cycle du pipeline inférieurs à 10 ns , ce qui est difficile à atteindre avec les technologies actuelles . (CRAY1 avec une structure plus performante et une technologie ECL obtient un cycle de base égal à 12,5 ns) .

La solution multipipeline est alors nécessaire .

I.1.2. Structure d'exécution parallèle .

Un travail est réparti sur un ensemble d'unités d'exécution parallèle indépendantes les unes des autres .

Trois architectures appartiennent à la structure parallèle et sont :

- l'architecture multipipeline .
- l'architecture SIMD .
- l'architecture hybride SIMD/multipipeline .

1.1.2.a. L'architecture multipipeline .

Les unités sont en général spécialisées dans des fonctions telles la multiplication , la division , les opérations flottantes etc ... Les mêmes principes et contraintes que l'architecture pipeline régissent l'architecture multi-pipeline .

L'interconnexion , la synchronisation et la mise en oeuvre de plusieurs systèmes pipelines spécialisés sont difficiles sur le plan matériel . D'autre part , il n'est pas aisé de réaliser un compilateur capable de faire un découpage fonctionnel optimal .

Les performances de l'architecture multipipeline , tout en étant supérieures à celles d'un monopipeline , restent limitées . Son rapport coût-performance rend cette architecture intéressante pour des applications moyennes . Lorsque les pipelines sont sophistiqués , nous obtenons des machines complexes et chères qui permettent de traiter des problèmes de grande dimension (CYBER 205 , CRAY1 , TI ASC) .

Nous citons quelques exemples de machines :

* le TI ASC [6] est constitué de quatre pipelines identiques spécialisés dans le traitement vectoriel . L'efficacité des

pipelines dépend du degré de vectorisation du programme , car les données scalaires dégradent les performances de ces unités . La détection d'opérations vectorisables est fonction à la fois de la nature du problème et du nombre d'unités pipelines . Pour améliorer les performances du système , des processeurs périphériques exécutent le système d'exploitation . Les programmes FORTRAN traversent une phase d'optimisation de code (vectoriseur) avant d'être compilés . Le système débite 100 Mflops max à pleine charge .

* le STAR 100 [6] est constitué de deux processeurs pipelines différents . L'un comporte un additionneur pipeline , une unité fonctionnelle et un diviseur . L'autre est constitué d'un additionneur pipeline et d'un multiplieur .

Un développement d'outils de détection et de transformation de boucles en instructions vectorielles STAR permet d'utiliser au maximum l'efficacité du matériel disponible . Un langage FORTRAN étendu permet la programmation directe d'opérations vectorielles ; en outre le programmeur a la possibilité de coder son application en langage symbolique STAR de façon à améliorer les performances . En conditions idéales , le système débite 100 Mflops .

Pour améliorer les performances réelles des systèmes multipipeline , un processeur scalaire y est parfois ajouté . C'est le cas des machines CRAY 1 et CDC CYBER 205 .

* le CYBER 205 est constitué d'un processeur scalaire composé de 5 unités fonctionnelles pipeline et d'un processeur vectoriel de quatre unités arithmétiques pipeline . Sa puissance théorique est de 400 Mflops sur 64 bits ou 800 Mflops sur 32 bits .

Ce calculateur est autonome , mais il peut être relié à un calculateur hôte (CYBER 170) pour optimiser son rendement .

La programmation directe d'opérations vectorielles se fait à l'aide d'un FORTRAN étendu . Ce compilateur détecte également les boucles et les transforme en instructions vectorielles .

* Le CRAV 1 est inspiré de l'architecture du CDC 7600 . Il est constitué de douze unités fonctionnelles pipeline , alimentées à partir de 16 registres tampon , 8 pour les scalaires et 8 pour les vecteurs . Il possède un compilateur CFT .

Son mode de fonctionnement est autonome mais il peut être relié à un calculateur hôte . Ses performances théoriques sont supérieures à 200 Mflops .

I.1.2.b. L'architecture SIMD .

L'unité de contrôle , pilote les unités d'exécution parallèles que sont les processeurs élémentaires , ou PE . Tous les processeurs effectuent la même instruction vectorielle en synchronisme .

L'arrangement des PE est , en général , matriciel et chaque processeur a sa propre mémoire de travail . Cette architecture présente cependant , des inconvénients :

- la taille de la matrice des PE est en général inférieure

à celles des matrices traitées , ce qui entraîne un découpage et rend ainsi l'écriture du compilateur plus complexe .

- le parallélisme est effectif au niveau des données du programme . Même lorsque des instructions parallèles existent dans le programme , elles sont traitées séquentiellement .
- les intercommunications dégradent les performances du système (conflits d'accès) ; les communications se font suivant les axes NORD-SUD EST-OUEST . Celles-ci limitent l'efficacité du traitement .

L'une des premières machines SIMD à vocation scientifique est l'ILLIAC IV [6].

* L'ILLIAC IV comporte 64 processeurs élémentaires structurés en matrice (8x8) . Un PE est constitué d'une unité arithmétique et logique , d'une mémoire de données et d'une logique de communication avec sa mémoire , l'unité de contrôle et ses PE voisins . La technologie MSI la plus complexe a été utilisée ; ce qui est pénalisant .

Un logiciel IVTRAN défini comme un FORTRAN étendu , est utilisé .

* Le BSP [6] : 16 PE sont spécialisés dans le traitement vectoriel . Ils sont synchrones et sont reliés à 17 bancs mémoire par l'intermédiaire d'un Crossbar . Un processeur scalaire se charge du traitement scalaire et de l'interprétation du programme . Le B7700 est prévu comme hôte .

I.1.2.c. L'architecture SIMD / pipeline

Elle regroupe une exécution globale SIMD et une implémentation pipeline des unités d'exécution ; c'est une structure qui profite du parallélisme vectoriel .

* NASF CDC [6] est constitué de quatre processeurs pipeline qui, dans le traitement vectoriel , fonctionnent en SIMD , d'un processeur scalaire et d'un processeur assurant la fiabilité . Un système mémoire à trois niveaux assure les transferts importants de données : une mémoire centrale de 8 Mmots en technologie ECL (très rapide , consommation excessive) avec un temps d'accès à 50 ns , une mémoire intermédiaire de 32 Mmots en technologie MOS et une mémoire de masse de 128 Mmots en technologie CCD (remplace les mémoires magnétiques) . Ce projet est destiné à des applications de l'aérodynamique qui nécessitent des performances supérieures à 1000 MFlops effectifs.

* Les machines systoliques [5,19] constituent une classe particulière de machines pipelines . Un système systolique est un réseau de processeurs au travers duquel circulent des données . Celles-ci sont injectées à l'aide d'un ordinateur "hôte" . Les processeurs fonctionnent de façon synchrone par "battement" par analogie avec la systole qui , en cardiologie représente une contraction du coeur .

Au cours d'un "battement", chaque processeur reçoit des données qu'il traite et transmet à ses plus proches voisins .

Cette machine a la régularité d'une organisation synchrone SIMD. Son principal intérêt est de permettre de traiter des données au "vol" , ce que ne fait pas naturellement une architecture SIMD .Ceci explique les recherches très actives dans ce domaine.

I.2. Structure de contrôle parallèle .

La disponibilité d'exécution des instructions guide l'interprétation du programme qui n'est plus séquentiel . La détection des instructions ou tâches (ensemble logique d'instructions) prêtes à être exécutées , est prise en charge par l'unité de contrôle . Les contraintes de synchronisme ou de ressources et les relations de dépendance entre les instructions (ou tâches) déterminent la disponibilité d'exécution de ces instructions (ou tâches) .

Le contrôle de type MIMD exprime tous les parallélismes contenus dans un programme ; nous citerons le parallélisme entre opérations arithmétiques et logiques , et celui des instructions de contrôle .

Le supercalculateur de type MIMD est un vrai multiprocesseur dans le sens où plusieurs PE exécutent en parallèle des instructions différentes sur des données différentes .

Dans la conception d'une machine MIMD , il existe deux façons de définir le mécanisme de détection du parallélisme :

- * la détection explicite des instructions (ou tâches) exécutables , s'effectue par interprétation d'une séquence de synchronisation générée soit , par le traducteur du langage de programmation soit , par l'utilisateur .

* la détection implicite par Data flow se fait à partir des données prêtes : une instruction (ou tâche) s'exécute dès que ses données d'entrée sont calculées .

I.2.1. Architecture MIMD - Détection explicite .

Citons quelques exemples de telles architectures :

* le CMMP [6] est un multiprocesseur pour applications non scientifiques . Il est malgré tout un bon exemple de ce type d'architecture .

* NASF Burroughs [6] n'est pas destiné à fonctionner en mode MIMD ; cependant , il dispose d'un réseau de communication entre PE et mémoire permettant un asynchronisme total entre PE . Le mode de fonctionnement global est de type SIMD , où tous les PE sont lancés au même instant sur un ensemble d'instructions (un corps de boucles DO par exemple) , et l'exécution se déroule alors de façon autonome sur chaque PE jusqu'à la fin de la séquence . Il atteint des performances de 1 Gflops max à pleine charge .

I.2.2. Architecture MIMD / multipipeline .

La mise en oeuvre de PE pipelines permet de profiter au maximum du parallélisme vectoriel . Citons quelques machines de ce type :

* S-1 [6] est constitué de 16 PE reliés à 16 bancs mémoire par l'intermédiaire d'un réseau Crossbar . Un PE regroupe deux pipelines spécialisés : l'un sur les instructions , le second sur les données ; chacun est constitué d'unités fonctionnelles pipelinées .

Un précompilateur , orienté vers des langages de haut niveau , prend en compte le jeu d'instructions des PE et, facilite l'exploitation du parallélisme .

Le S-1 est autonome , multiprogrammé et peut débiter jusqu'à 400 Mflops dans des conditions idéales .

* HEP Denelcor [20] , est un système multiprocesseur modulaire allant jusqu'à 16 PE . Sur chacun d'entre eux coexistent un certain nombre de séquences de code indépendantes ou coopérantes appelés processus . L'interprétation de ces processus est effectuée de façon pipeline . Un Fortran étendu prend en compte les principes d'exécution , liés à la notion de processus .

Ce système est autonome , multiprogrammé , dont les performances avoisinent les 160 Mflops .

* SPRINT -System Control [6] , est un réseau d'array processeurs API 20 , extensible jusqu'à 8 .

Le langage Fortran est muni de six commandes qui servent , à transférer des données et à implémenter diverses opérations arithmétiques sur des matrices , vecteurs et sous-ensembles de matrices . Le système est prévu pour un fonctionnement multi (ou mono) utilisateur .

I.2.3. Architecture MIMD , approche Data -Flow .

Le principe du Data Flow est basé sur la disponibilité des données ; une instruction est déclarée exécutable dès que ses données d'entrée sont calculées . La notion de compteur

ordinal perd son sens . Des mécanismes d'interprétation totalement nouveaux font alors apparaître des architectures multiprocesseurs originales .

I.2.3.a. Le Data Flow pur

Un graphe data flow permet de représenter un programme . Deux types de noeuds apparaissent : les chaînages et les acteurs . Des "pions" , véhiculés par les arcs du graphe , transportent les valeurs d'un acteur à un autre . Selon cette structure , l'exécution d'un programme est décrite par une séquence de flashes . Chaque flash se traduit par la mise en opérations de noeuds activés par l'affectation de valeur à leurs pions d'entrée . Ce type de machine est étudié au MIT , à l'Université de Californie et à Irvine , entre autres .

I.2.3.b. Le Data Flow à assignation unique .

Une contrainte logicielle est ajoutée au cadencement par les données . En effet , toute variable ne peut recevoir qu'une valeur au plus durant l'exécution d'un programme . Nous citons pour exemple la machine LAU [21,35] CERT-DERI , qui a permis de démontrer la faisabilité d'une machine data flow , programmée dans un langage évolué , exprimant de façon naturelle le parallélisme d'exécution MIMD .

II. LES RESEAUX D'INTERCONNEXION .

Le choix d'une structure d'interconnexion adéquate est délicat , d'autant que le nombre de processeurs est élevé .

Si le système contient N processeurs et , que nous choisissons un réseau où chaque processeur est relié aux autres alors , le nombre de connexions est égal à $[N * (N-1)]$, égal environ à $(N * N)$; le réseau est irréalisable pour une valeur élevée de N .

Il existe des schémas d'interconnexion plus économiques (Fig.I.3) .

II.1. Réseaux d'interconnexion statiques .

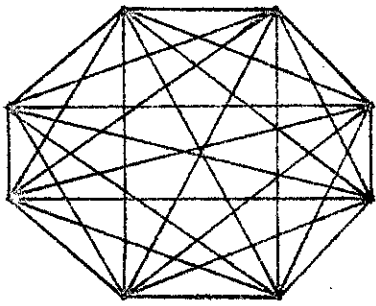
Un schéma d'interconnexion sous forme d'anneau a l'avantage de ne nécessiter que deux connexions par processeur quelquescit le nombre N de processeurs . Mais un message devra traverser en moyenne , $N/2$ processeurs avant d'atteindre son destinataire ,ce qui nécessite trop de temps , pour N grand .

La structure en arbre est utilisée dans beaucoup d'architectures prototypes : chaque processeur peut communiquer avec trois autres ; les processeurs placés dans le bas de l'arbre ("feuilles") ne communiquent pas facilement entre eux .

Les processeurs peuvent être connectés en parallèle (à travers des portes d'accès) , sur un groupe de lignes omnibus appelé Bus , à travers lequel ils communiquent (Fig.I.4.a). On suppose qu'il existe un mécanisme de contrôle des accès au bus , ou chemin partagé .

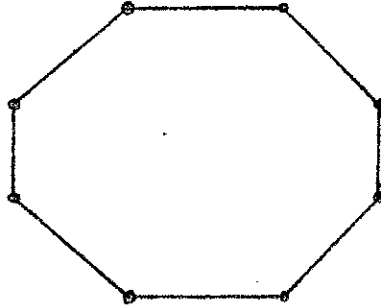
Les processeurs peuvent être aussi , connectés en matrice . Le nombre de chemins est réduit , mais le nombre de connexions croît avec $N*N$.

FIGURE I.3. RESEAUX D'INTERCONNEXION .

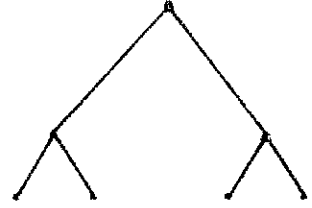


MAILLAGE

COMPLET (impraticable pour n grand)

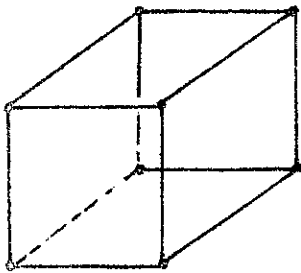


ANNEAU (lent)

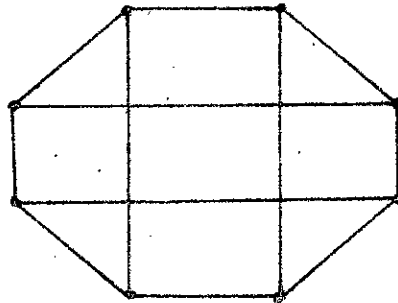


ARBRE

(intelligence artificielle)



3 - CUBE

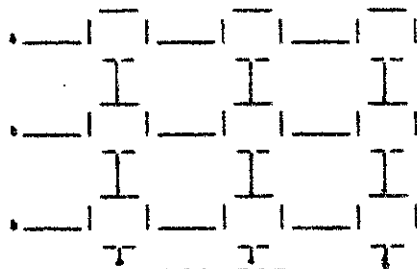


3 - CUBE EN ANNEAU



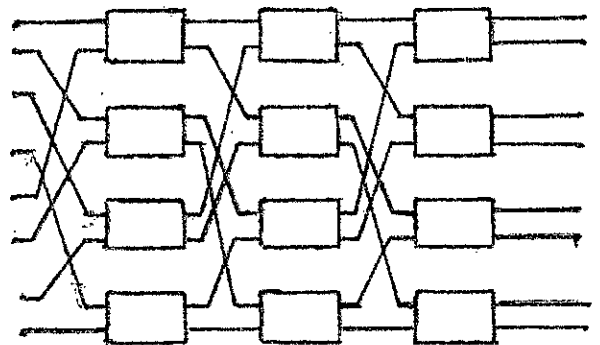
COMMUTATEUR

PROGRAMMABLE



RESEAU CROSSBAR

(permet d'interconnecter deux)
(processeurs quelconques , mais)
(coûteux en commutateurs : n^2)

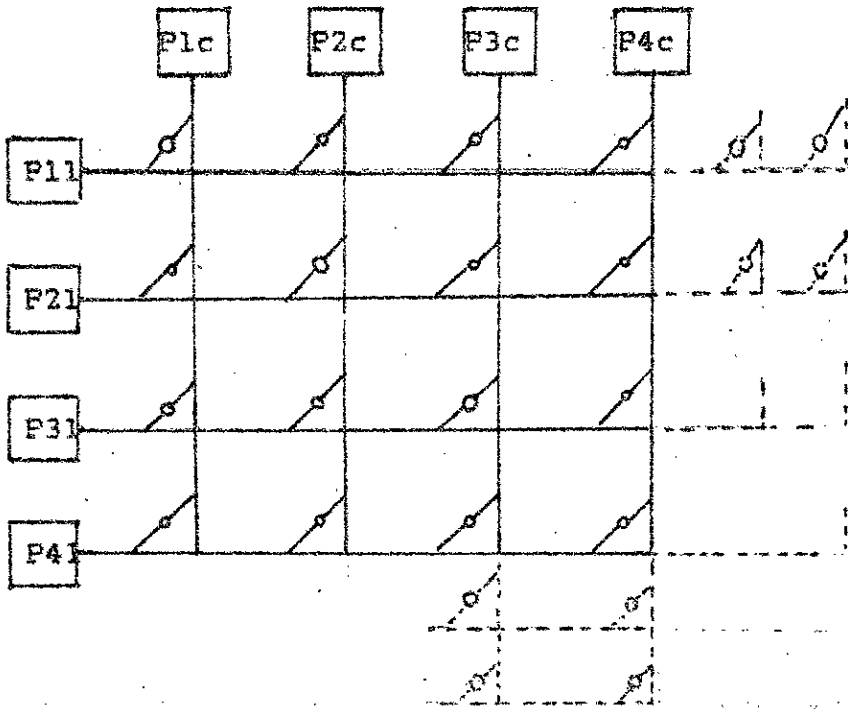


RESEAU OMEGA

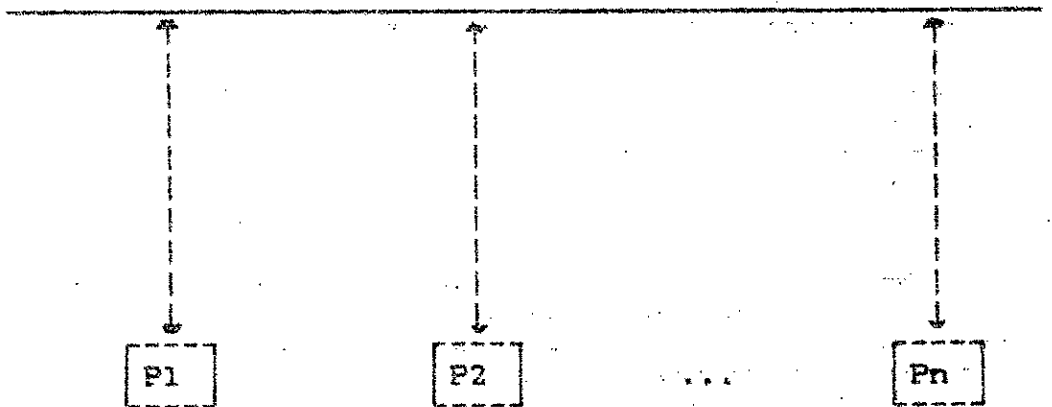
($n \log n$ commutateurs sont)
(nécessaires pour n)
(processeurs)

FIGURE I.4. STRUCTURES A COMMUNICATION DIRECTE

RESSOURCE PARTAGEE , AUTOCONTROLE .



b. Matrice .



a. Bus à accès parallèle .

Une structure très prometteuse est l'hypercube . Elle est constituée d'un cube dans un espace de dimension quelconque. En dimension trois , l'hypercube est le cube de la géométrie classique . Chaque processeur est alors relié à trois autres processeurs et , se trouve à une distance maximale de trois connexions de tout autre processeur . Cette structure peut être généralisée dans un espace de dimension k et portera le nom de k -cube binaire :

le cube contient $(2)^k$ sommets et , chacun (ou processeur) est relié à ses k voisins immédiats soit, à un voisin par dimension du réseau . Cette structure a été retenue par les chercheurs du CALTECH (California Institute of Technologie) . Elle a été conçue et réalisée en 1983 sous le nom d'ISPC/d6 et commercialisée en Février 1985 par INTEL . C'est une machine du type MIMD composée de 32 , 64 ou 128 microprocesseurs (80286 d'INTEL) connectés comme un 6-cube binaire . Son rapport performance / coût est très intéressant . Le taux réel d'utilisation des processeurs (ou rendement) est souvent supérieur à 0,8 , ce qui est bien accepté .

II.2. Réseaux d'interconnexion reconfigurables .

Les processeurs sont reliés par l'intermédiaire de commutateurs programmés . Dans les architectures comportant des milliers de processeurs , le réseau reconfigurable (dynamique) ne sera pas utilisé -car coûteux - malgré sa grande souplesse .

Par contre , dans les architectures SIMD , des commutateurs seront utilisés entre processeurs et entre processeurs et mémoire . Le réseau est alors reconfiguré par programme entre chaque instruction et, les communications sont synchrones.

Le BSF (Burroughs Scientific Processor) devait posséder 512 processeurs connectés dynamiquement à 521 modules de mémoire (cette machine n'a pas été commercialisée pour des raisons financières) [5] .

CEDAR est une machine de l'Université de l'Illinois [5] (projet) qui possède deux niveaux d'interconnexion . Les processeurs sont organisés en groupes , eux-mêmes , interconnectés par un réseau reconfigurable .

III. QUELQUES EXEMPLES DE CALCULATEURS SPECIALISES .

.....

Le traitement numérique d'images est une technique dont l'importance ne cesse de croître . De nombreux secteurs de pointe y font appel : en médecine (scanner) , robotique , photos transmises par satellite pour la météorologie , la recherche minière , l'archéologie ou l'agriculture . Ces images sont loin de pouvoir être exploitées en totalité .

Ces applications nécessitent une masse considérable de calculs , souvent répétitifs , sur des images numérisées . La qualité d'une image est obtenue à l'aide de plusieurs millions de points ; la contrainte temps réel associée à la qualité d'une image , conduit à des architectures de type parallèle.

Pour effectuer des calculs sur une image , celle-ci est codée en un tableau de nombres qui représentent l'intensité

lumineuse en chaque point de l'image , appelé pixel . En associant à chaque pixel , un processeur relié à ses voisins immédiats , une architecture parallèle très régulière adaptée aux calculs locaux entre un point et ses voisins , est obtenue.

Nous citons quelques exemples de calculateurs spécialisés .

Dans le traitement d'images :

* STARAN [18,38] est un calculateur à mémoire associative . Il est constitué d'un certain nombre de mémoires (au plus 32) . Un PE est prévu pour chaque mot . Les opérations d'entrée-sortie sont prises en charge par l'hôte . STARAN n'est programmable qu'en langage assembleur , appelé APPLE . Il est capable de débiter à 200 Mflops .

* MPP [18] a été conçu pour la NASA . Il est constitué de 16384 processeurs connectés en matrice 128 x 128 . Ils ont une architecture SIMD . Cette machine utilise le parallélisme image . Une mémoire peut échanger jusqu'à 320 M octets / s avec la matrice de PE , soit l'équivalent de 320 images de 1024 sur 1024 points . Un processeur de contrôle commande les PE et exécute les instructions scalaires . Les PE exécutent les instructions vectorielles (les processeurs sont regroupés par huit sur un même circuit intégré) . Chaque processeur a une mémoire de 1Kbits contenant les données image et , relié à ses plus proches voisins E , O , N , S sous le contrôle du programme . Les colonnes de gauche et de droite peuvent être connectés en cylindre , ainsi que la première et la dernière ligne . Les performances sont de 6553 M d'additions / s sur des entiers et 216 M d'opérations / s pour une multiplication flottante sur 32 bits . Le cycle de base est de 100 ns permettant les

entrées-sorties à cette vitesse .

* CYTOCOMPUTER [18,38] est constitué de deux séries parallèles de processeurs connectés en pipeline . Les processeurs sont synchrones et suivent une architecture MIMD . La première série est constituée de 88 processeurs connectés en pipeline pour traiter les images binaires , alors que la deuxième série contient 25 processeurs connectés en pipeline travaillant en parallèle par rapport à la première série pour les images multiniveaux de gris . La vitesse de calcul est de 1,6 M octets / s .

* CLIP IV [18] est une machine bâtie autour d'une matrice 96 x 96 processeurs booléens . Une fenêtre correspondant à une trame de 96 x 96 x 6 bits peut être mémorisée dans six registres à décalage (1 par plan bit) . Le plan bit est converti en une matrice binaire 96 x 96 pouvant être transférée dans la mémoire correspondante au tableau matriciel . Une instruction de contrôle commune , est envoyée à tous les processeurs qui sont parfaitement synchronisés . CLIP est performant pour les opérations logiques portant sur une fenêtre 3 x 3 ; l'opération dilatation / rétraction est réalisée en un cycle (10 microsecondes) . Une convolution 3 x 3 exige par contre , 2000 cycles , soit 20 ms .

* PASM [18,38] est une machine parallèle qui peut être structurée en une ou plusieurs machines SIMD et/ou MIMD indépendantes . C'est un système reconfigurable , pour exploiter au mieux les parallélismes inhérents au traitement d'images et à la reconnaissance des formes .

En traitement du signal (télécommunications ,...):
un algorithme de traitement du signal consiste à répéter un même calcul , souvent complexe , sur une suite de données . Cela est le cas , pour une représentation codée de la parole , d'images à transmettre ou encore d'une série de mesures effectuées sur un système physique ou biologique que l'on cherche à observer . Dans ce cas , il ne s'agit plus d'associer un processeur par point de mesure puisqu'il y en a souvent une infinité . Par contre , les données doivent circuler au rythme où elles arrivent . C'est ainsi qu'est né le concept d'architecture systolique .

Les algorithmes de la reconnaissance de la parole sont très coûteux en temps de calcul , mais leur régularité permet une mise en oeuvre systolique . Plusieurs étapes caractérisent la reconnaissance de la parole : reconnaissance de phonèmes puis de mots et enfin de la phrase :

* API 89 [5] de l'IRISA de Rennes , est constitué de 89 PE . Un processeur est associé à chaque point d'une grille dont les colonnes correspondent aux phonèmes du mot recherché . Cette machine permet de reconnaître 2000 mots en temps réel (1000 fois plus rapide que sur un miniordinateur du commerce) .

* WARP [5] , en construction à l'université Carnegie Mellon , a une architecture systolique linéaire programmable . Elle permet le calcul des principaux algorithmes de traitement du signal (transformée de Fourier rapide , convolution , filtrage ...) . Elle peut atteindre une puissance de 10 M instructions/s avec 10 processeurs seulement .

Les recherches liées aux processus parallèles concerneront bientôt tout le champ de recherches en informatique .

Des architectures multiprocesseurs , massivement parallèles , permettent d'obtenir , grâce à l'avènement de microprocesseurs très performants , un rapport performance - coût très intéressant . Ce ratio ira en croissant avec la technologie WSI (Wafer Scale Integration) , qui intègre plusieurs processeurs sur une puce .

- CHAPITRE II -

STRUCTURE MATERIELLE DES SYSTEMES MULTIMICROPROCESSEURS

La demande sans cesse croissante de performances de calcul a abouti , à la conception de calculateurs à architecture multiprocesseur .

L'avènement des microprocesseurs bon marché et performants (16 bits voire 32 bits) a encouragé cet engouement et, permet de penser en termes d'architectures massivement parallèles comportant un nombre très important de processeurs d'une puissance égale à celle d'un petit microordinateur . Ce qui importe , à présent , est l'organisation et la façon dont les processeurs échangent l'information .

En effet , pour que deux processeurs puissent communiquer , une synchronisation s'impose . Et , ceci nécessite du temps .

D'autre part , la communication s'effectue à travers un support de communication physique qui pourrait être plus coûteux que les processeurs eux-mêmes si l'on n'y prend pas garde .

I. STRUCTURES MATERIELLES DES MULTIMICROPROCESSEURS .

I.1. Structure logique .

Il existe deux relations logiques :

I.1.1. L'organisation verticale

Dans sa forme simple , une organisation verticale a un seul maître et plusieurs esclaves . Elle présente les caractéristiques suivantes :

- les éléments ne sont pas tous semblables .
- à un instant donné , seul un élément joue le rôle du maître ; cependant , plusieurs éléments peuvent avoir la potentialité de le devenir .
- toutes les communications doivent passer à travers le maître ou être initialisées par le maître .
- le hardware des esclaves peut être identique et , chacun peut être spécialisé dans une tâche par le software . On peut rencontrer une configuration maître-esclave pyramidale .

I.1.2. L'organisation horizontale requiert plus de coordination . Elle a les caractéristiques suivantes :

- tous les éléments sont logiquement équivalents .
- chaque élément est susceptible d'être maître .
- tous les éléments peuvent communiquer entre eux .

En général , l'organisation horizontale est plus flexible que la première . Cependant , elle n'est pas plus efficace pour des applications ayant des tâches très différentes .

I.2. Structure physique [1] .

Cela concerne la méthode d'échange d'information . Elle est fonction de l'organisation de la communication inter-processus et de la topologie d'interconnexions .

I.2.1. Organisation des communications .

I.2.1.1. Mode de communication .

Les communications entre les processeurs peuvent être directes (Fig.II.1) ou indirectes .

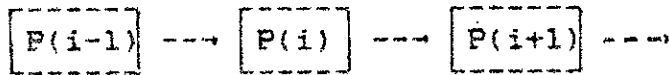


FIGURE II.1. Communication directe.

La communication de $P(i-1)$ à $P(i+1)$ est directe si $P(i)$ n'est pas un organe de contrôle .
Sinon , nous avons un processeur $P(i, j)$ qui doit décider du routage des messages ; il joue le rôle d'organe de contrôle et nous avons le schéma de la figure II.2 .

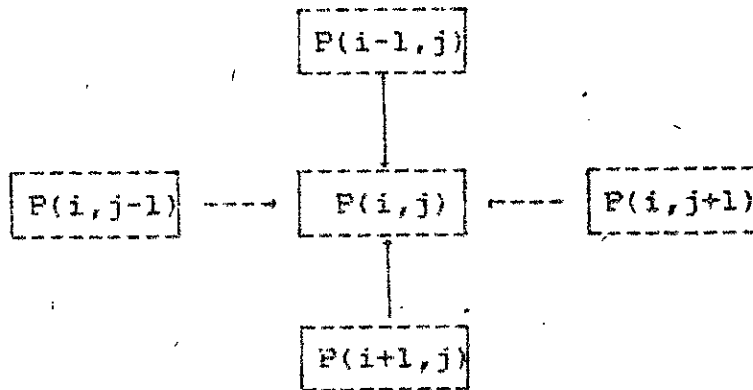


FIGURE II.2. Communication indirecte .

I.2.1.2. Contrôle de la communication (indirecte):

Si le routage des messages est assuré par un seul organe de décision , le contrôle est dit centralisé . Dans le cas contraire , il est décentralisé ou alors nous avons un mécanisme d'autocontrôle .

I.2.1.3. Chemins de communication [1] :

Un chemin de communication est de type partagé s'il comporte plus de deux points d'accès . Sinon , il est réservé ou privé .

I.2.1.4. Ressources partagées :

Les transferts de données entre processeurs peuvent être échangés :

- soit à travers une structure mémoire commune dite centralisée sans accès direct en mémoire commune .
- soit à travers une structure de bus dite distribuée où un lien logique est établi pour créer un chemin de communication entre les éléments .

Dans les systèmes où les transferts sont fréquents et importants , les organisations précédentes ne sont pas efficaces car les conflits d'accès augmentent pour la ressource partagée . Ce problème est plutôt aggravé dans les systèmes à microprocesseurs par le goulot d'étranglement mémoire - processeur , et par la limitation des capacités d'entrée - sortie .

I.2.2. Topologie d'interconnexion .

Les quatre schémas de base sont :

- Bus Commun .
- Etoile .
- Anneau .
- Pleine connexion .

D'autres topologies sont des combinaisons de base ou des variations des quatre schémas précédents (voir Chap.I).

I.3. Mode d'interaction

Malgré les innombrables schémas d'interconnexion existants , les systèmes peuvent être classés selon le degré de couplage et la nature de l'intercommunication entre processeurs .

Le couplage se réfère à la capacité de partager des ressources communes , avec deux possibilités extrêmes : les systèmes couplés faiblement et , les systèmes fortement couplés (Fig II.3') .

I.3.1. Les systèmes faiblement couplés ou Réseaux [3]

Ils sont caractérisés par deux calculateurs ou plus , dispersés géographiquement . Les différents calculateurs du système sont interconnectés à travers une interface de communication . La communication intercalculateurs suit un protocole rigide .

Chaque ordinateur fait son traitement indépendamment des autres . Il a son propre programme et ses données stockées dans sa mémoire , mais il est relié aux autres par les entrées-sorties de données , le contrôle , la communication et / ou l'utilisation commune des périphériques .

Les structures de réseau les plus communément utilisées sont en :

- étoile .
- anneau .
- bus partagé .

La figure II.4 nous les schématise .

FIGURE II.3. SYSTEMES MULTIPROCESSEURS

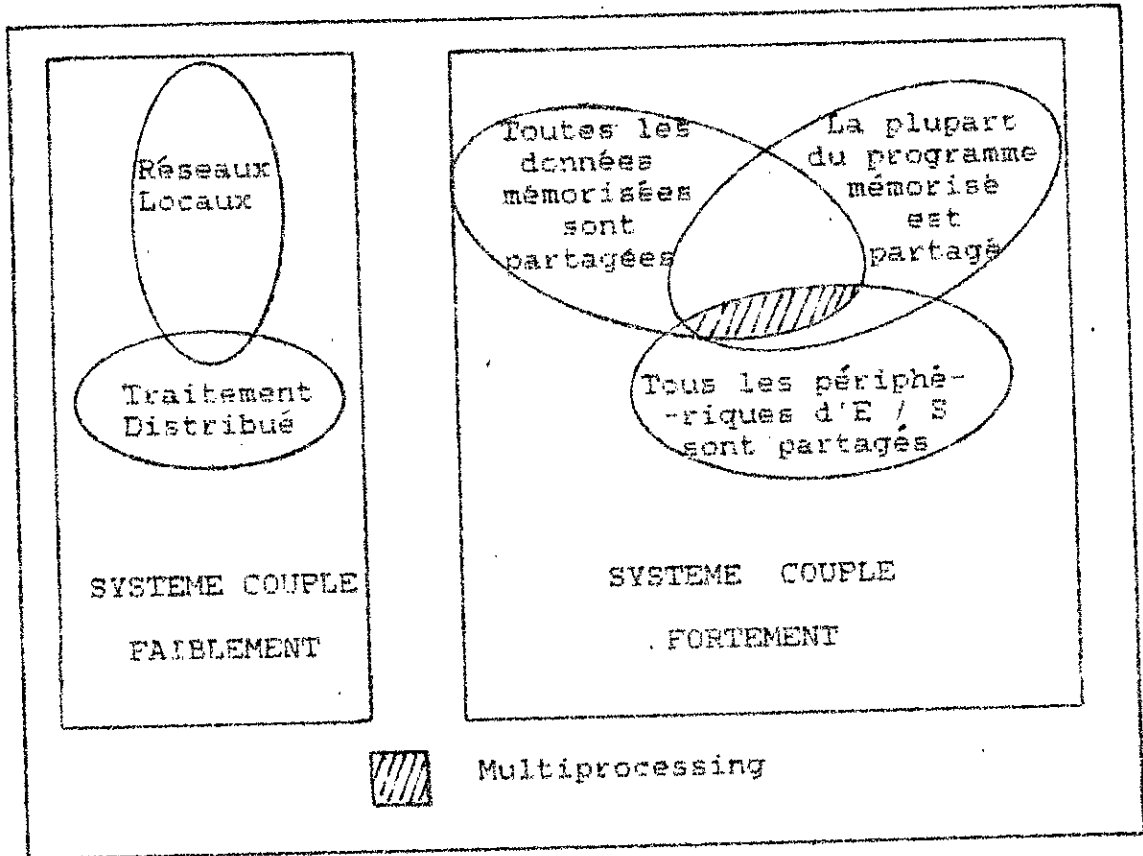
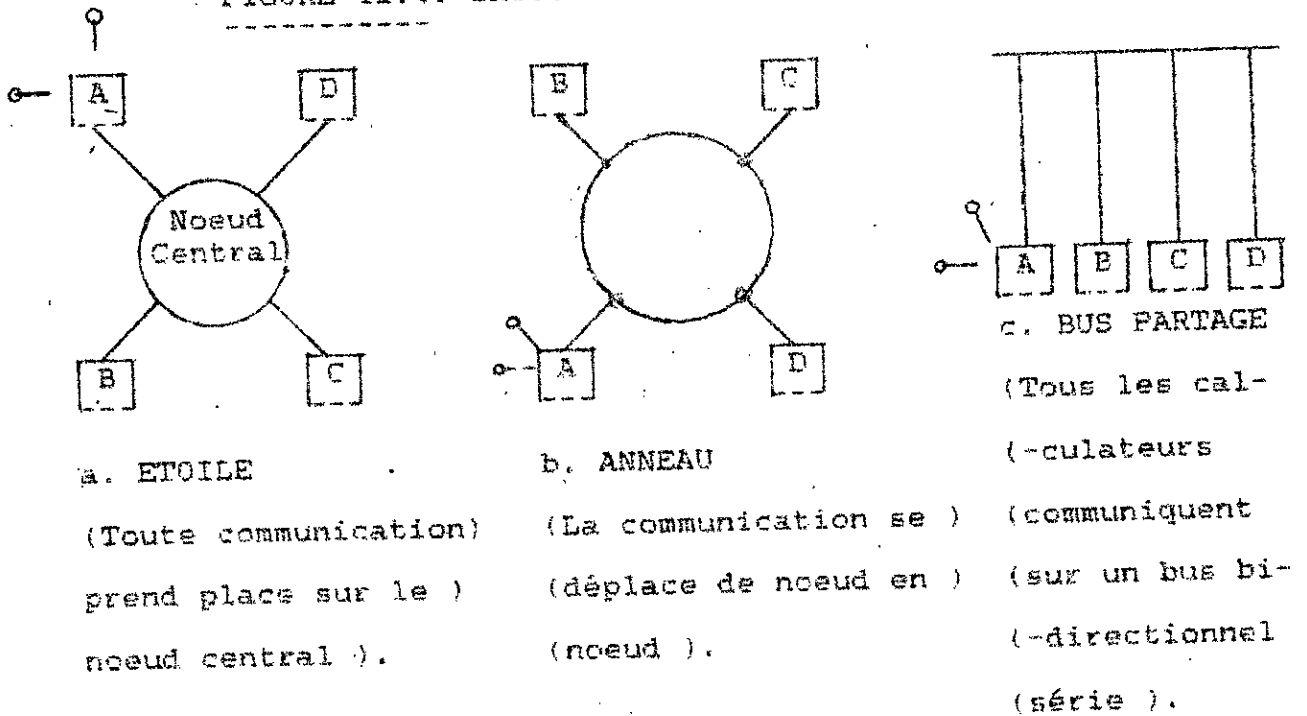


FIGURE II.4. TROIS DES RESEAUX LES PLUS UTILISES .



Un réseau faiblement couplé utilise , de façon typique , des chaînages de données série . Cela peut être aussi simple que la Figure II.5.a ou très complexe , comme dans les environnements de contrôle industriel . Ceux-ci sont constitués de plusieurs niveaux avec plusieurs types de réseaux (Fig.II.5.b) .

L'information circule entre les processeurs sous la forme de paquets de données qui incluent de façon typique l'information concernant la source et la destination de la donnée .

Les systèmes de faible couplage proposent des processeurs dispersés ou distribués , par lesquels le traitement est réalisé et ceci de façon concurrente et asynchrone , tout en ayant des processeurs coopératifs .

Chaque site de calcul a ses propres processeurs , mémoire , ressources locales , système d'exploitation et chemins de communication . Cette structure peut résoudre certains problèmes ; si un noeud tombe en panne , les autres processeurs continuent à travailler et peuvent même , diagnostiquer le problème au noeud défaillant .

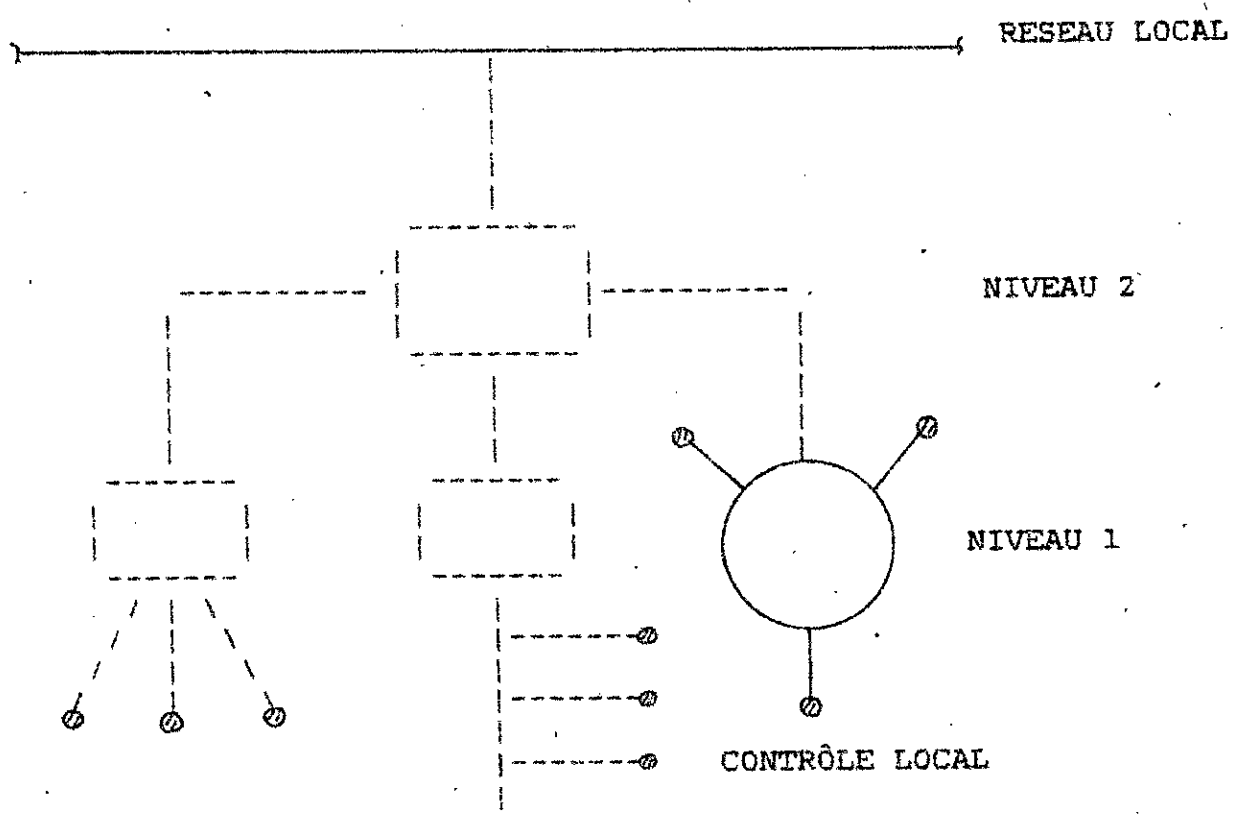
Le réseau le plus important , existe aux USA avec "The Advanced Research Projects Agency Network " qui connecte environ 50 noeuds à travers les USA .

Les organisations des réseaux de calculateurs ont une communication interprocesseurs rigide et des noeuds de traitement extensible , ce qui n'est directement pas applicable aux systèmes à microprocesseurs . Cependant avec l'évolution technologique des microprocesseurs , des versions modifiées des réseaux de calculateurs seront bientôt utilisées .

FIGURE II.5. RESEAU FAIBLEMENT COUPLE



a. Interfaçage entre un terminal intelligent et un ordinateur hôte .



b. Systèmes de contrôle industriel à noeuds multiples .

La circulation série des données , dans les systèmes faiblement couplés , pose des problèmes . En effet , si la chaîne série de données est encombrée , le temps de réponse et l'utilisation de la base de données pourraient baisser considérablement . Ce qui peut être un handicap sérieux dans les applications temps réel . Un système fortement couplé est alors plus approprié .

1.3.2. Les systèmes fortement couplés .

Ils sont connus également comme systèmes multi-microprocesseurs fortement couplés [3] . Ils contiennent deux ou plusieurs processeurs avec :

- une mémoire commune (principale) qui est partagée entre tous les processeurs du système ainsi que la possibilité de posséder une mémoire locale .
- un système d'exploitation commun qui contrôle et coordonne toute interaction entre processeurs et processus .
- des ressources partagées : des facilités d'entrées-sorties et d'autres ressources du système sont généralement partagées entre les processeurs . Cependant , quelques ressources peuvent être attribuées à des processeurs spécifiques .
- la même puissance de calcul : les processeurs sont configurés de façon symétrique et exhibent des capacités similaires .
- le partage de la charge (load sharing) est dynamique [17,33].

La distribution dynamique de la charge d'un processeur saturé est uniforme à travers tous les processeurs :

* autonomie de processeurs : chacun des processeurs (coopératifs entre eux) peut exécuter des traitements significatifs individuellement .

* synchronisation : elle est nécessaire pour des processeurs qui coopèrent .

La plus grande limitation d'une organisation multi-processeur fortement couplée est l'existence des conflits d'accès en mémoire commune . Cette restriction tend à limiter le nombre de processeurs qui peuvent effectivement être supportés par le système d'exploitation .

La plupart des réseaux de connexion processeur - mémoire tendent à réduire ces conflits d'accès en mémoire commune .

Les trois types d'organisation processeur - mémoire les plus fondamentales sont :

- le Bus Commun , où tous les éléments sont connectés à un seul bus .
- le Crossbar Switch , où les éléments sont connectés à un module distinct appelé le " Crossbar Switch " qui peut fournir plusieurs connexions simultanées entre éléments.
- la mémoire multiport , où chaque élément de mémoire a plus d'un port d'accès , est connectée aux autres éléments à travers un système multibus .

Grâce à l'organisation multimicroprocesseur , plusieurs processeurs lents font le travail d'un processeur très rapide . Cependant , le goulot d'étranglement processeur-mémoire , caractéristique des microprocesseurs , limite son application

aux organisations multiprocesseurs .

Des systèmes multiprocesseurs utilisant des micro-processeurs peuvent être implémentés dans des cas spéciaux ou un grand nombre de processus similaires , relativement indépendants , échangent des faibles quantités de données .

I.3.3. Les systèmes distribués à microprocesseurs [2] .

Des structures qui combinent au mieux chacune des deux représentations précédentes (couplages faible et fort) sont plus appropriées aux systèmes à microprocesseurs . Ces structures multimicroprocesseurs modérément couplés s'appellent les DIMSs (Distributed Intelligence Microcomputer Systems) . Dans un système distribué à microprocesseurs , la charge de travail est partitionnée en tâches relativement indépendantes pour être assignées aux divers éléments du système . Un tel système a les principales caractéristiques suivantes :

- les éléments sont autonomes : chaque élément est constitué en général d'un CPU , d'un programme propre , d'une mémoire de données et peut utiliser ou contrôler des périphériques additionnels .
- les processeurs sont spécifiques à une tâche . Théoriquement, chaque élément est dédié à une tâche spécifique qui détermine sa relative complexité .
- les processeurs sont de complexité différente . La configuration du système n'est pas nécessairement symétrique puisque les éléments ne sont pas de même complexité .

- il y a une optimisation hardware et software à effectuer .

En effet , chaque élément a un hardware et un software étudiés en fonction de la tâche spécifique qu'il doit réaliser.

- il y a communication de données : la communication interprocesseurs se fait souvent au niveau des données .

Cependant , dans certaines situations , ces dernières peuvent contenir des commandes .

- processeurs de contrôle et de communication : en général, chaque élément peut gérer le contrôle des entrées-sorties et la communication du système . Dans le cas où la communication est beaucoup plus importante , une des fonctions précédentes peut être attribuée à un autre processeur .

- le partage de la charge (load sharing) est statique .

Comme les processeurs sont spécifiques (à une tâche) , un système minimal ne peut supporter un partitionnement dynamique; ainsi la répartition des tâches (load balancing) doit être faite durant la phase de conception . Cependant , des unités additionnelles peuvent être introduites pour faire une fraction du partage du chargement [2,17] .

I.4. Mode de traitement .

La classification de Flynn considère l'exécution au niveau seulement de l'instruction . Ceci est trop restrictif

Dans sa forme la plus générale , un système multi-processeurs capable d'exécuter de façon concurrente un nombre donné de tâches , chacune utilisant des groupes différents de données , peut être appelé MIMD (Multiple Task , Multiple Data).

Nous pouvons établir une classification arborescente pour les multiprocesseurs (Fig II.6).

II. EVALUATION DES PERFORMANCES .

II.1. Capacité de traitement du système .

II.1.1. Coût - Performance .

Compte tenu de l'évolution technologique des microprocesseurs , il est intéressant sur le plan économique , d'en utiliser plusieurs pour augmenter les performances du système .

Néanmoins , et sans considérer les coûts du logiciel de haut niveau , les coûts des différents éléments tels les périphériques et les connecteurs ne décroissent pas aussi rapidement que ceux des microprocesseurs et des mémoires . Ainsi , la réduction du rapport coût - performance risque d'être annulée par les autres coûts du système .

II.1.2. Throughput [2,17,27] .

Le "throughput" d'un système , défini comme étant l'inverse du temps nécessaire pour exécuter un groupe donné d'algorithmes , est un indicateur approprié de la performance du système . Il est mesuré en nombre d'opérations par unité de temps .

Théoriquement , le "throughput" augmente avec le nombre de processeurs supplémentaires au système multiprocesseur .

FIGURE II.6. CLASSIFICATION ARBORESCENTE DES MULTIPROCESSEURS .

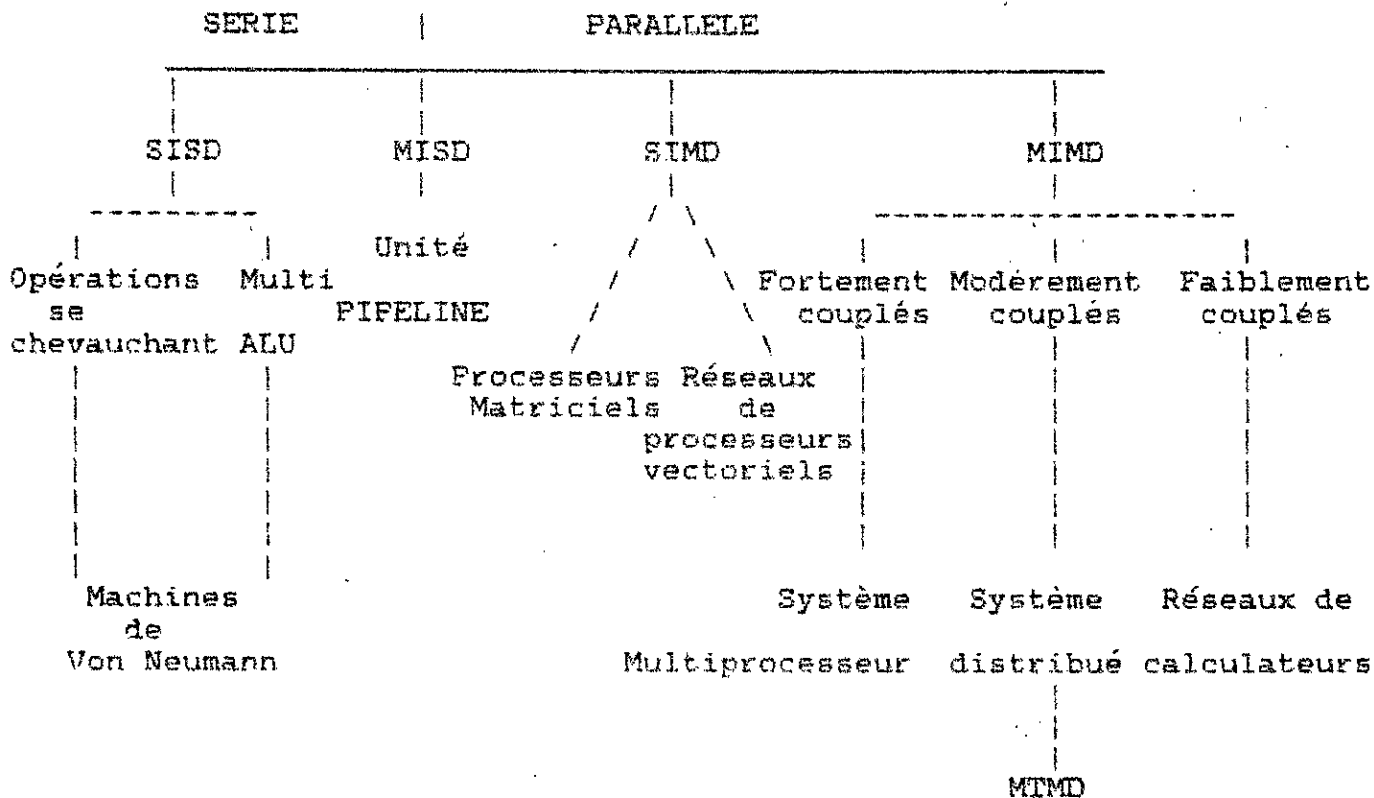
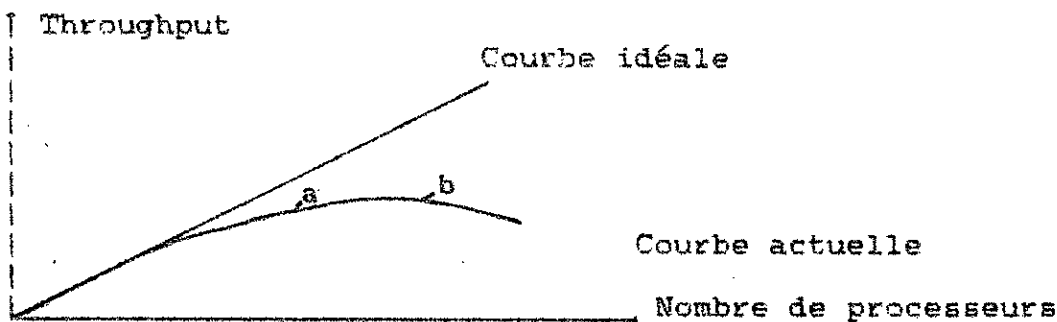


FIGURE II.7. EFFET DE SATURATION .



- a) Dénivellement
- b) Déclin

En pratique , nous avons la courbe actuelle de la Figure II.7 , à cause de la saturation .

Le "throughput" sature à partir d'un nombre incrémental de processeurs ajoutés . Cet effet peut être attribué aux conflits d'accès des ressources partagées , qui augmentent avec le nombre de processeurs .

Les points de dénivellement (levelling off) et de déclin (decay) , peuvent être différents pour des architectures multiprocesseurs différentes .

Théoriquement , le souhait est de travailler sur la partie linéaire de la courbe ; celle-ci peut être étendue en réduisant les conflits d'accès et les communications inter-processeurs . Ce résultat peut être obtenu , en partitionnant le travail principal en petites tâches indépendantes avec une intercommunication minimale .

II.1.3. Ressource partagée .

Cela peut être aussi bien une imprimante , une carte mémoire ou alors un processeur arithmétique à haute vitesse .

Le temps partagé de la ressource s'effectue de façon à éviter la surcharge de la ressource et les conflits d'accès . Des sémaphores peuvent être utilisés (à l'aide de système d'exploitation) comme mécanisme de blocage pour protéger la donnée dans la mémoire partagée .

II.2. Partage des données communes .

Pour éviter la transmission de données volumineuses

d'un processeur à un autre (ou plusieurs autres) , on stocke ces données dans une mémoire commune . Ces conflits d'accès à cette mémoire ralentissent le système .

On peut regrouper les circuits d'accès aux mémoires communes en classes suivantes :

II.2.1. Accès par ligne omnibus

Les processeurs accèdent à la mémoire commune à travers une ligne omnibus sur laquelle ils sont connectés en parallèle . Il peut exister un arbitre pour régler les conflits d'accès . On peut avoir , également , un système d'autoarbitrage . La bande passante du bus limite la fréquence des accès à la mémoire d'un processeur . Cette bande passante doit croître avec le nombre de processeurs pour leur permettre un rythme de travail constant .
L'avantage de cette connexion est sa simplicité matérielle et logicielle .

II.2.2. Accès par réseau de sélection matricielle [1]

La mémoire est fractionnée en bancs M_1, M_2, \dots, M_p bien que fonctionnellement elle puisse être toujours considérée comme un espace continûment adressable . Les processeurs P_1, P_2, \dots, P_n et les bancs de mémoire sont interconnectés, à travers une matrice de connexion ; p parmi les n processeurs peuvent accéder simultanément aux p bancs mémoire . Par contre, la logique d'aiguillage et d'arbitrage devient très compliquée dès que p et n deviennent grands .

II.2.3. Multibus .

C'est une solution intermédiaire par rapport aux deux précédentes . Elle fait le compromis coût - performance ; q parmi les n processeurs peuvent accéder simultanément à q parmi p bancs de mémoire à travers les q bus (Fig.II.8) .

II.2.4. Mémoire multiaccès .

Chacun des bancs mémoire comporte n circuits d'accès auxquels sont reliés les n processeurs . Une logique de priorité câblée permet de hiérarchiser les demandes d'accès simultanées .

II.3. Délai de transmission des messages.

Est fonction de :

* La distance interprocesseur 'd' : elle est mesurée , par exemple , en nombre de noeuds de commutation que doit traverser un message du processeur source au processeur destinataire .

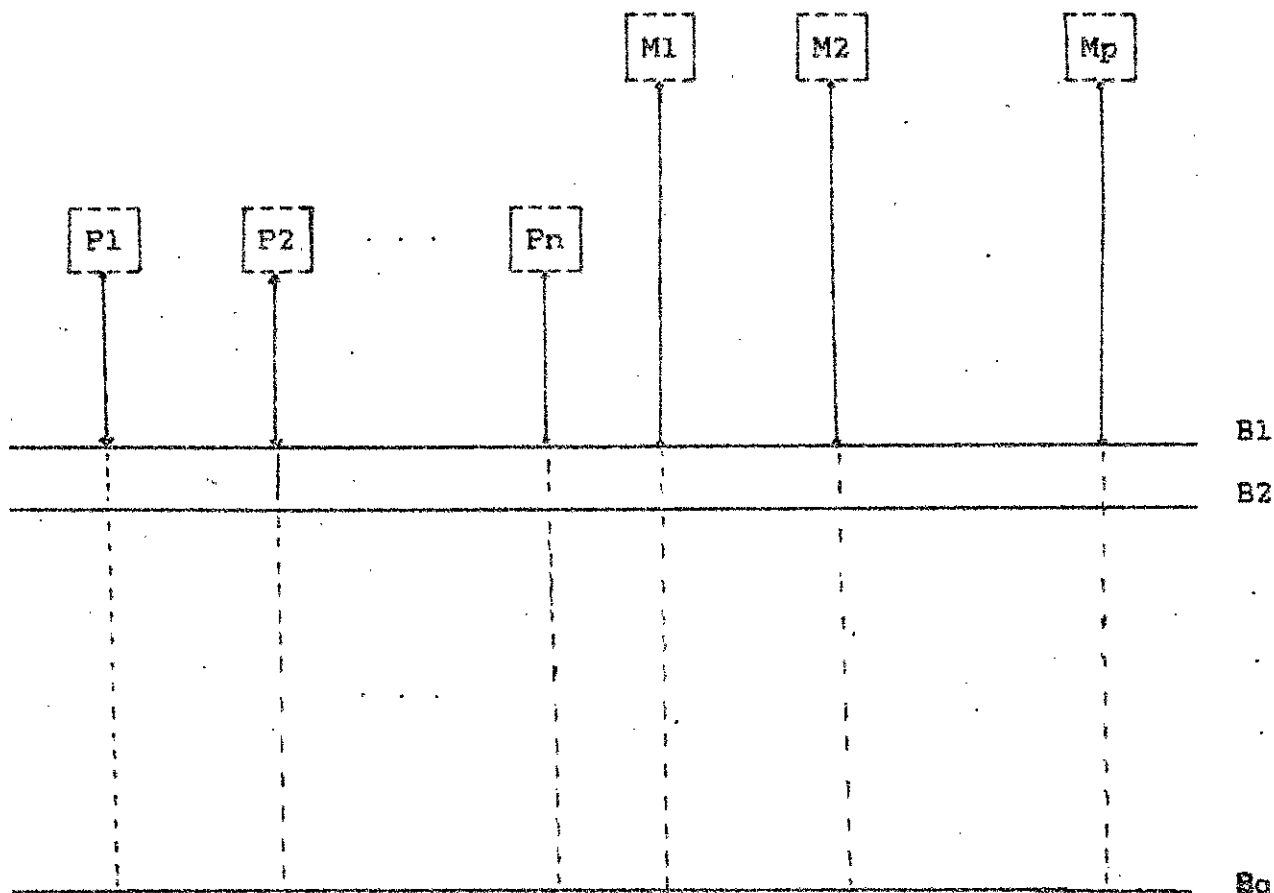
Pour un réseau à interconnexion 2 à 2 , $d=0$.

Pour un réseau à interconnexion matricielle , $d=1$.

Pour un n-cube , 'd' est au maximum égale à n .

Pour un anneau , tout message doit faire le tour de l'anneau pour des besoins de synchronisation du système . Pour un réseau à accès total , comportant N processeurs , Benes a montré qu'il fallait au moins ($N * \log N$) points de commutation , 'd' valant environ $\log N$.

FIGURE II.8. LE MULTIBUS



* temps de séjour dans les files d'attente .

En cas de demandes simultanées d'accès à un même chemin , il se forme une file d'attente .

Le temps d'attente moyen [1] est donné par :

$$E(A) = 1 / (I(B) - \bar{N} * I_m)$$

I_m : nombre moyen d'informations qu'un processeur demande à transmettre , quand il se présente au bus .

\bar{N} : nombre moyen de processeurs utilisateurs de bus à un moment donné .

$I(B)$: nombre d'informations que le bus peut acheminer par unité de temps .

Si le nombre de processeurs augmente , pour un temps d'attente constant , $I(B)$ devra augmenter proportionnellement à \bar{N} . La structure à bus unique est vite limitée par la bande passante de celui-ci .

III. AMELIORATION DES PERFORMANCES .

Le problème de concurrence et de conflit peut être traité indépendamment de la structure physique , en utilisant les concepts de tâche , d'évènement et de communication comme équivalents logiques de structures physiques .

Pour obtenir une opération harmonieuse , le système doit être capable de suivre les aspects de contrôle : arbitrage , allocation et coordination .

III.1. Arbitrage .

Les conflits d'accès au bus des processeurs peuvent être résolus de diverses façons ; nous ne citerons que les dispositifs simples d'allocation de bus (ou arbitres) .

Un arbitre de bus permet une transparence au niveau logiciel ; chaque processeur ignore les autres processeurs ; il affiche des adresses et si le bus n'est pas libre , le déroulement de l'instruction en cours est bloqué jusqu'à ce que le bus lui soit alloué .

Le débit de communication varie avec le nombre de processeurs utilisés , de leur débit de requêtes et du débit des mémoires ou interfaces au bus . Le débit de communication peut être bon si les éléments précédents sont correctement équilibrés .

Deux types d'arbitre peuvent se concevoir :

- arbitres synchrones par multiplexage temporel [9] : le temps est divisé en tranches dont la durée est égale au temps d'accès élémentaire au bus . Chaque tranche est allouée à un même processeur avec une périodicité égale au nombre de processeurs . Cette méthode est simple et peu coûteuse . De plus , dans le cas où les processeurs ont des cycles de lecture / écriture périodiques et où l'on peut faire coïncider les deux périodes , des performances très élevées seront obtenues .

Lorsque le nombre de processeurs croît , les performances chutent .

- arbitres asynchrones [9] : l'arbitre reçoit les requêtes des processeurs et alloue le bus à un processeur en fonction de celles-ci .

Plusieurs types d'arbitre :

* priorité tournante : efficace et peu coûteuse .

* priorité fixe : se justifie lorsque les temps d'accès des processeurs sont très différenciés .

* file d'attente : est optimale car minimise l'espérance du temps de chaque processeur .

Les arbitres asynchrones sont plus performants et plus coûteux que le multiplexage temporel .

(Dans le système Multimu , la gestion des priorités s'effectue par software . Nous n'utiliserons pas d'arbitre de bus proprement dit) .

III.2. Allocation de tâches .

Nous avons vu dans le paragraphe II.1.2 que , le "throughput" décroît à partir d'un certain nombre incrémental de processeurs ajoutés . Ceci peut se produire à partir de 3 ou 4 processeurs si le système est improprement désigné . Cette décroissance est due à la communication interprocesseurs (IPC) excessive .

L'allocation de tâches est le procédé d'attribution de modules aux processeurs avec un IPC minimal [2,17] .

Le partitionnement des tâches se réfère au découpage d'une tâche en plusieurs modules individuels avec une communication intermodule (IMC) minimale .

Le partitionnement des tâches et l'allocation des tâches sont deux étapes nécessaires pour minimiser l'IPC .

Il est commode de considérer un module comme une entité indivisible , la plus petite unité d'opération viable . Une tâche est une simple entité de traitement .

Le procédé du partitionnement des tâches sera réalisé par la partie software et chaque tâche arrivera dans le système

de traitement distribué , déjà partitionnée en groupes de modules . Le partitionnement des tâches est un problème important dans les systèmes de traitement distribué et , il est loin d'être résolu .

Des méthodes sont développées pour accomplir l'allocation de tâches et minimiser le coût total du traitement . Ces méthodes peuvent être statiques ou dynamiques .

III.3. Coordination entre les tâches .

Dans un système multiprocesseur , une coordination entre plusieurs tâches concurrentes est nécessaire . Les problèmes de coordination associés aux processeurs concurrents peuvent être établis en termes de synchronisation , temps mort et exclusion mutuelle .

III.4. Système d'exploitation (ou executive) .

Tous les aspects de contrôle cités ci-dessus sont du ressort d'un système d'exploitation . L'implémentation d'un multimicrocalculateur effectif est très influencée par la conception propre de son "executive" . Celui-ci utilisera quelques concepts déjà utilisés dans les systèmes d'exploitation temps réel pour des systèmes monoprocesseurs .

Il est donc intéressant d'implémenter des systèmes multimicroprocesseurs lorsqu'il s'agira par exemple :

- d'élaborer des applications de contrôle de processus ayant des traitements diversifiés et des contraintes temps réel . Pour

la plupart de ces applications , les traitements volumineux excèdent de loin les capacités d'un système monoprocesseur .

- des applications qui nécessitent une bonne fiabilité mais, à cause des contraintes économique et / ou d'espace , ne peuvent supporter une redondance massive .

- des applications avec un traitement d'E/S extensif .

La nécessité d'interfacer avec des processus d'E/S variés impose généralement un contrôle inacceptable sur un système monoprocesseur et cause une dégradation sévère de la souplesse du système .

Par contre , il sera déconseillé d'utiliser un système multimicroprocesseur dans l'un des deux cas suivants :

* l'application à traiter nécessite seulement une puissance de calcul plus élevée . Il sera alors plus avantageux d'utiliser un CPU plus puissant ou alors ajouter un ou plusieurs processeurs spécialisés pour certaines tâches [2] .

* la tâche globale à exécuter ne peut être partitionnée en tâches relativement indépendantes avec des besoins de communication intertâches les plus faibles .

- CHAPITRE III -

DESCRIPTION DE LA MACHINE MULTIPROCESSEUR

La possibilité de décomposer le modèle d'un processus physique continu nous a conduit à des sous modèles . Ceux-ci sont tels que leur interaction est minimale .

L'implémentation des algorithmes correspondants aux sous-modèles s'effectue au niveau des unités de traitement ou processeurs secondaires , du système multiprocesseur conçu . Un processeur principal - PP ou maître - et quatre processeurs secondaires - PS ou esclaves - constituent ce système .

La fonction contrôle de la machine est dévolue au PP . Le pas de calcul est à cet effet , émis par l'intermédiaire d'une ligne omnibus FACAL à tous les esclaves et , ceci de façon simultanée . Ainsi , le contrôle est parallèle .

Lorsque les transferts de coefficients et paramètres nécessaires au traitement sont réalisés sur la mémoire locale de chaque PS , l'exécution des tâches s'effectue simultanément sur tous les processeurs secondaires . L'exécution est dite parallèle .

Ainsi le système multiprocesseur conçu est de type MIMD (multiple instruction ,multiple data) .

Le maître supervise tout échange interprocesseurs au niveau de la mémoire commune via un bus partagé. La logique

de la structure est verticale car , constituée d'un seul maître et de plusieurs esclaves .

Il est nécessaire avant tout , de présenter quelques notions élémentaires utiles à la compréhension de ce chapitre .

I. DEFINITIONS DES ENTITES INTERVENANTS DANS UN EXECUTIF .

I.1. La tâche .

C'est un agent actif responsable de l'exécution, par une machine, d'un programme . Elle possède un nom et des attributs ; est caractérisée par :

- un contexte physique comprenant l'ensemble des informations nécessaires à son exécution par un processeur .
- un descripteur logique qui rassemble les données nécessaires à sa gestion - nom , priorité , adresse du S.P , etc...

Dans notre cas , la tâche a deux descripteurs :

- * un descripteur statique défini dans la configuration du système et contient les informations statiques de la tâche .
- * un descripteur dynamique appelé PCB ou process control bloc , généré par le système d'exploitation à l'initialisation du système à partir des informations données par le descripteur statique .

I.2. Les ordonnanceurs .

On appelle ordonnanceur , un module chargé de gérer un ensemble de processeurs pour le compte d'une famille de tâches.

Cette gestion consiste à attribuer les processeurs , libres ou libérés , aux tâches prêtes à commencer ou à continuer leur exécution et , à reprendre les processeurs affectés à

certaines tâches . Il existe deux types d'ordonnanceur :

- l'ordonnanceur câblé dont l'algorithme de choix est souvent basé sur les priorités associées aux interruptions :
 - . il examine séquentiellement les priorités à partir de la plus élevée .
 - . sauvegarde le contexte de la tâche immédiate de priorité courante lorsque l'interruption associée à la tâche de priorité p est arrivée et qu'elle n'est pas masquée ; la tâche de priorité p est alors démarrée .

Cet algorithme peut être exécuté par le processeur en tout point d'interruptibilité de la tâche en cours . La fin de l'exécution d'une tâche immédiate , relance la tâche de plus grande priorité prête à être exécutée .

- l'ordonnanceur des tâches différées - ou ordonnanceur programmé - gère les tâches en suivant un algorithme imposé par le type de l'exécutif à réaliser .

1.3. La gestion des tâches .

Selon l'état où se trouve une tâche , à un instant donné , une des opérations ci-dessous est appelée lorsque le processeur doit décider du sort d'une tâche . Il s'agit de :

- . DEMARRER (une tâche) : la tâche passe de l'état " hors service " à l'état " en service " .
- . ARRETER (une tâche) : la tâche est mise dans l'état " hors service " ; pour toute réactivation ultérieure , on utilise l'opération CONTINUER .
- . CONTINUER (une tâche) : la tâche est mise dans l'état

"en service" après avoir été arrêtée .

.SE TERMINER : la tâche courante est retirée à son processeur ; elle est mise à l'état " hors service " . Toute réactivation ultérieure nécessite l'opération DEMARRER .

Le module ordonnanceur se charge de faire appel à l'une de ces opérations . La fig.III.1 présente l'évolution d'une tâche .

I.4. L'évènement .

C'est l'outil de signalisation entre les tâches .

On peut soit signaler un évènement , soit attendre un évènement .

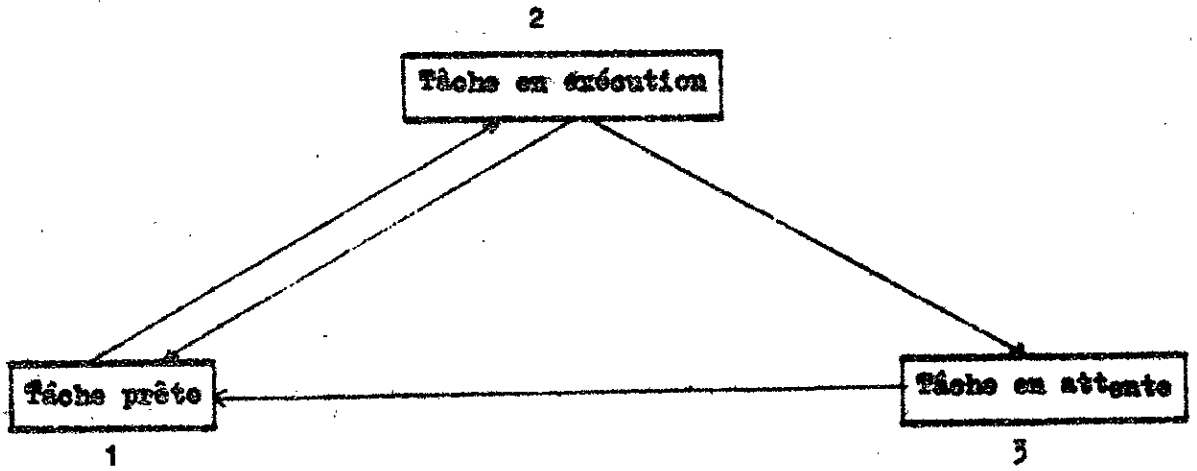
Une tâche émettrice peut ,après avoir effectué un traitement t1 , signaler un évènement à une tâche réceptrice en attente de cet évènement qui lui permettra de réaliser le traitement t2 . Sur le plan hardware , l'évènement est une interruption câblée .

I.5. Gestion des régions .

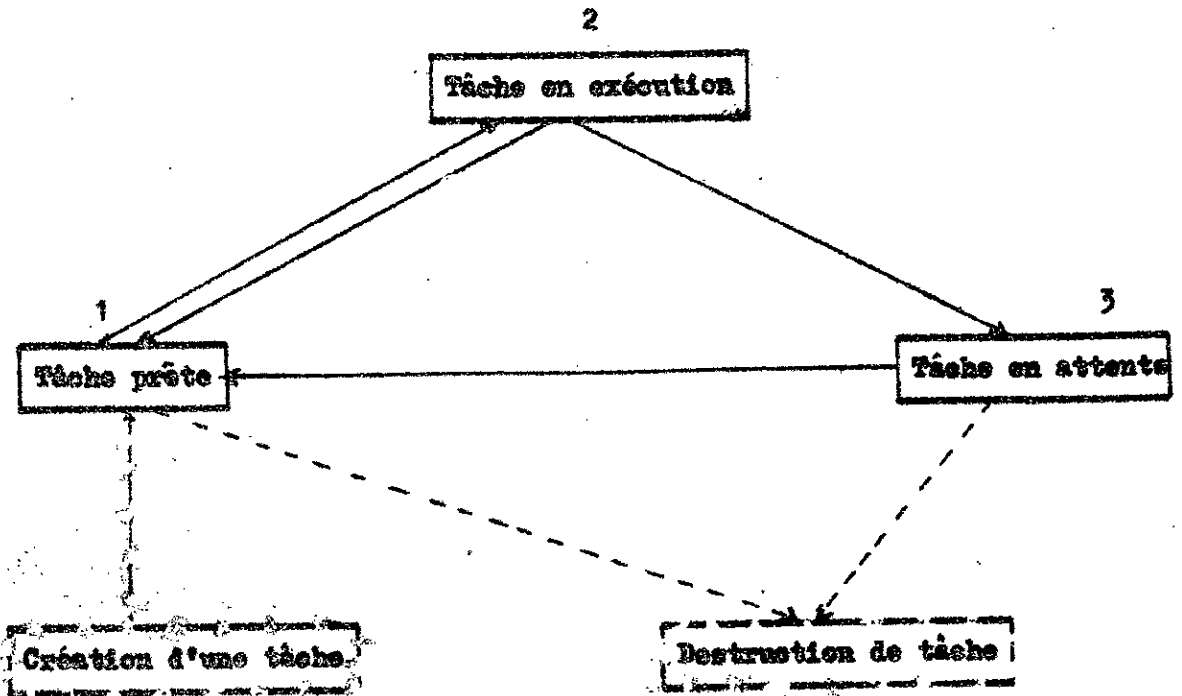
Le parallélisme des tâches induit un phénomène de compétition pour la manipulation des ressources partagées . Un mécanisme d'exclusion mutuelle est nécessaire pour gérer les conflits possibles . Ce mécanisme assure que les exécutions de séquences d'instructions manipulant un même objet partagé soient disjointes dans le temps , indépendamment de l'ordre dans lequel ces séquences sont invoquées . Ces sections sont dites " critiques " .

La région est marquée occupée pendant tout le temps où une tâche est dans une section critique . Ceci est matérialisé par un masquage des interruptions (avec l'analogie évènement -

FIGURE III.1. ETATS D'UNE TACHE .



1.a. tâche permanente



1.b. tâche temporaire

interruption) . Lorsqu'un évènement est arrivé , le traitement d'interruption qui en découle nécessite l'exécution d'une "section critique" qui implique l'entrée dans une région (masquage des interruptions) . A la sortie de cette région , on effectuera un démasquage des interruptions (I.T) .

La manipulation des régions se fait par les deux opérations suivantes : ENTRER (dans) et SORTIR (de) la région .

I.6. File .

C'est une stratégie de gestion de communication entre tâches . Nous avons la file des tâches en attente , la file des messages etc ...

Le noyau gère les files à stratégie FIFO . Les opérations de manipulation des files sont :

- ENVOYER et/ou RETIRER (élément , file) .

Le paramètre élément peut aussi bien être une tâche (c'est alors son PCB qui est manipulé) qu'un message .

II . ARCHITECTURE DU SYSTEME MULTIPROCESSEUR .

II.1. Choix de la structure multiprocesseur .

La simulation de processus physiques continus peut se ramener à la simulation de sous-systèmes représentés par des équations différentielles . Ces sous-systèmes seront attribués chacun à un processeur , pour le traitement numérique .

Ces sous-modèles étant en général interdépendants , un couplage interprocesseurs assez important s'impose . Nous obtenons alors une structure multiprocesseur à fort couplage .

Chaque processeur sera chargé d'intégrer une équation

différentielle . Cette opération sera subdivisée en tâches . L'enchaînement des tâches doit être ordonné . Ceci nous conduit à un contrôle centralisé ou à une structure maître - esclave .

Pour accélérer les temps de réponse dans le traitement des informations , nous avons préféré attribuer à chaque processeur son propre ordonnanceur et les objets du noyau au niveau de sa mémoire privée ou locale .

L'évolution des tâches implémentées sur les différentes unités de traitement appelées processeurs secondaires (PS) se fait de manière indépendante avec transfert d'informations entre les PS , sous le contrôle d'un processeur principal (PP) . Celui-ci impose un pas de calcul commun de façon à obtenir un démarrage synchrone . Durant ce pas , l'exécution est asynchrone .

Les transferts d'information entre PS se font dans la mémoire commune (MC) à travers un bus partagé . Une demande d'accès à la MC est formulée par le PS en envoyant une interruption au PP qui supervise l'accès en MC .

Le bus partagé est l'un des schémas d'interconnexion multiprocesseur les plus classiques ; il reste intéressant pour la connexion d'un petit nombre de processeurs , vue sa bande passante limitée .

II.2. Présentation du système Multimu .

Le simulateur est conçu autour d'un PP qui contrôle quatre PS et une MC à travers un bus partagé . Une extension à huit processeurs (voire seize) est possible .

Tout dialogue PS - PS passe par la MC sous le contrôle du PP .

II.2.1. Le processeur principal .

Il a pour fonction :

- d'assurer l'entrée des paramètres du modèle du processus à simuler .
- de prendre en compte les commandes destinées à définir les conditions de la simulation .
- de distribuer les tâches aux PS .
- de lancer la simulation .
- de contrôler l'accès à la MC et au bus commun .
- de synchroniser l'ensemble à travers le pas de la simulation .

Il se compose :

- d'un microprocesseur MC6800 .
- d'une mémoire privée .
- d'une interface série ACIA pour la liaison avec une console de service .
- d'une interface série pour l'imprimante .
- d'un timer programmable pour synchroniser l'évolution des tâches au rythme d'un pas de calcul .
- d'un dispositif d'entrées-sorties parallèles destiné à la génération du bus interprocesseurs .
- de circuits de gestion d'interruptions telles que les demandes d'accès à la MC ,etc ...

II.2.2. Le processeur secondaire .

Il assure :

- la recherche en MC des paramètres de la simulation (PARO).
- le calcul numérique correspondant à la tâche qui lui incombe (TACAL) .

- La recherche en MC des résultats intermédiaires calculés par les autres PS (FAR1) .
- Le transfert des résultats obtenus par ce PS vers la MC pour leur utilisation par les autres PS (TARES) .

Les sous-tâches PAR0 et FAR1 appartiennent à la tâche TAFAR que nous définissons ultérieurement .

Le PS est constitué :

- d'un microprocesseur MC6800 .
- d'une mémoire locale de programmation et de données (8Kbytes) .
- d'une logique de dialogue interprocesseurs .
- d'un coprocesseur arithmétique Am9512 et sa logique d'interfaçage au microprocesseur .

II.2.3 . Le bus interprocesseurs .

Il se compose de deux parties :

- * le bus mémoire partagée , contient les signaux habituels d'adresses , de données , et de contrôle .
- * le bus d'interconnexion , contient l'ensemble des signaux de synchronisation .

Lorsque l'un des processeurs utilise le bus , les autres doivent être en haute impédance par rapport au bus ; ils ne peuvent alors travailler qu'en local . Le PP synchronise l'évolution parallèle des tâches assignées aux PS . De plus , il élimine les conflits d'accès entre les PS .

Pour ce faire , le partage des ressources communes est réalisé au moyen de lignes d'interruptions à priorité entre le PP et le PS .

Les signaux d'interruption assurant l'interconnexion sont :

- le signal DACC : demande d'accès à la MC de la part d'un PS à la fin de son transfert d'information .
- le signal PACAL : donne le pas de calcul aux PS .Il est généré par le PP .
- le signal ACQ : acquittement envoyé par le PP au PS qui a formulé une demande d'accès . Le PP vérifie préalablement si l'accès en MC est libre . Ce signal consiste à autoriser le PS à accéder au bus commun pour effectuer un transfert de données en MC .
- le signal APP : est un appel envoyé par le PP au PS pour signaler que des informations sont prêtes en MC . Une sélection d'adresse doit être faite par le PP de façon à ce que ce signal ne soit reconnu que par le PS sollicité .

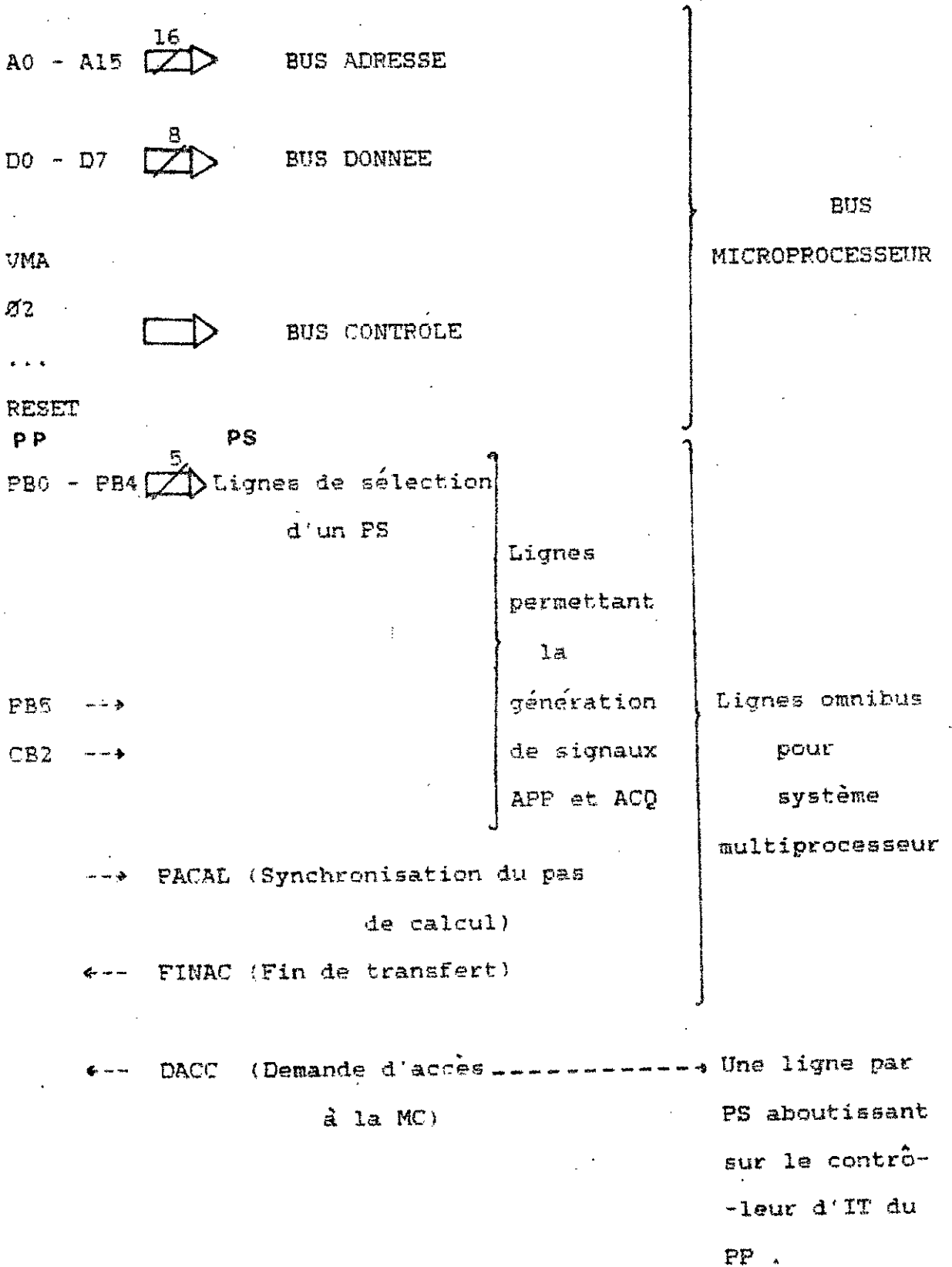
Cette logique permet d'adresser chacun des PS (de l'adresse 0 à 7) .

Le bus interprocesseurs a pris forme à partir du bus Exerciser de Motorola ;celui-ci est un bus monoprocesseur . Cependant , les lignes non utilisées de ce bus ont été répertoriées et ensuite affectées aux signaux précédents de façon à utiliser ce bus en multiprocessing (Fig.III.2) .

II.2.4. Le logiciel de gestion .

L'évolution des tâches avec la contrainte du pas de calcul et les contraintes aléatoires dues aux interruptions entre PP et PS , nous a orientés vers un système d'exploitation multi-tâches temps réel distribué .

FIGURE III.2. SIGNAUX DU BUS INTERPROCESSEURS



Ceci nous permettra entre autres de gérer les interruptions et la synchronisation de l'accès aux ressources partagées en temps réel . Cet exécutif se compose essentiellement :

- d'un noyau conçu pour le microprocesseur MC6800 , implémenté dans toutes les unités (PP et PS) .
- de tâches propres à chaque processeur .
- d'un protocole de dialogue PP/PS .
- de la gestion des périphériques au niveau du PP .

II.2.4.a. Le noyau .

Il a pour rôle principal d'assurer la synchronisation des accès aux ressources partagées , de résoudre les problèmes de concurrence entre les tâches grâce au principe de priorité et gestion des interruptions . Le noyau manipule les files suivantes :

- file des tâches prêtes (Fig.III.3) : onze niveaux constituent cette file (standard 8 bits) .
- file des tâches en attente d'une même ressource (Fig.III.4)

Les opérations relatives à la synchronisation sont appelées primitives , pour leur rôle prépondérant ; ce sont les opérateurs P et V correspondants à la sollicitation et à la libération d'une ressource définis en 1968 par Dijkstra .

Ces primitives agissent sur un mécanisme logiciel appelé sémaphore ; il s'agit d'une structure de données de deux octets qui est définie dans la configuration du système .

FIG.III.3 FILE DES TACHES PRETES MANIPULEE PAR L'ORDONNANCEUR

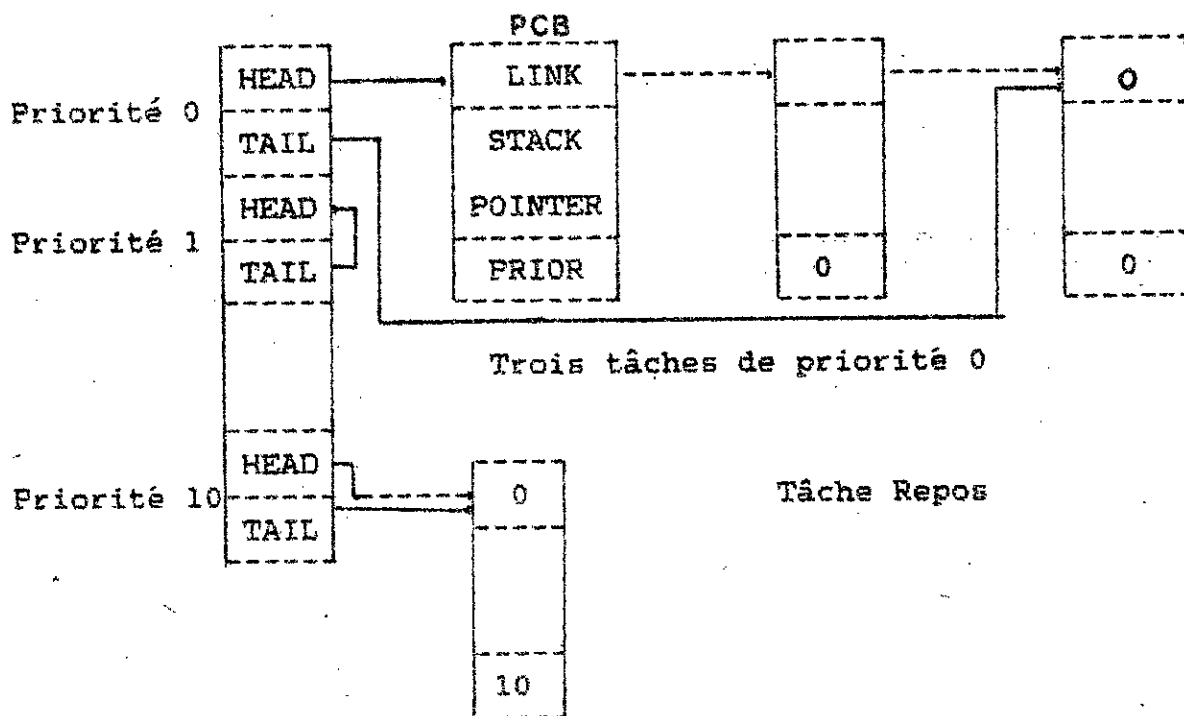
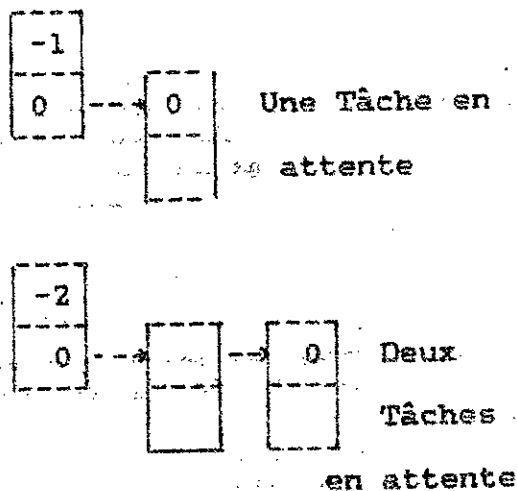


FIG.III.4. FILE DES TACHES EN ATTENTE D'UNE MEME RESSOURCE MANIPULEE PAR LES PRIMITIVES P ET V



a. Ressource libre

Un sémaphore est défini lorsqu'une ressource est partagée entre 2 ou plusieurs tâches. A chaque ressource partagée correspond un sémaphore.



b. Ressource occupée

III.2.4.b. Les tâches du système .

Chaque PS est chargé de résoudre de manière itérative une équation différentielle relative au sous-système. Après un pas de calcul , chaque PS transfère en MC la valeur de la variable obtenue à la fin de ce pas . D'autre part , selon la méthode de résolution numérique choisie , d'autres transferts seront à faire .

Le PP distribue alors ces valeurs aux PS adéquats .
Le travail confié au système est découpé en tâches .

Au niveau du PP :

- tâche de lancement de calcul : TAPAS
- tâche d'acquiescement : TACACQ
- tâche d'appel : TACAPP
- tâche de conversion : CONVER
- tâche du pas à pas
- etc ...

Au niveau du PS :

- tâche de transfert des coefficients et paramètres : TAPAR
- tâche de calcul : TACAL
- tâche de transfert des résultats : TARES

III.2.4.c. Protocole de dialogue PP/PS

Il est essentiellement basé sur la gestion des interruptions (signaux DACC , FINAC , APP , ACQ , PACAL) .
Chaque fois qu'une tâche du PS a besoin d'une ressource , elle fait appel à la primitive P et reste en attente jusqu'à ce que du côté PP , un signal soit émis pour réveiller cette tâche .

III. DESCRIPTON DE LA MACHINE ET DES UNITES DE TRAITEMENT

III.1 Description matérielle de la machine .

Comme cela a été dit précédemment , la structure bus partagé a été adoptée avec un contrôle centralisé de type maître - esclave . Le PP attribue la MC au PS qui la sollicite, une fois que cette ressource est libérée . Nous pouvons dire que la structure a un accès en MC , supervisé par le PP .

Le PP assure également la gestion de plusieurs unités périphériques -console de visualisation , imprimante , traceur de courbes - dans le cadre d'un dialogue interface - user .

Les PS ont tous la même structure matérielle . Cet aspect de symétrie dans la conception du Multimu permet l'interchangeabilité des unités et la facilité de leur maintenance .

La figure III.5 donne un synoptique de la machine . Les PSi sont les unités de traitement de Multimu .

III.1.1. Synoptique du PP .

Il est donné en figure III.6

III.1.2. Description du PS .

Chaque processeur secondaire occupe une carte sur laquelle un bus privé permet la circulation des informations entre les processeurs et les circuits . Un décodeur permet d'adresser la mémoire locale constituée de 8 Koctets (RAM et ROM) . Il adresse également un processeur arithmétique qui effectue des opérations en virgule flottante sur 32 bits . Un schéma bloc est donné en figure III.7 .

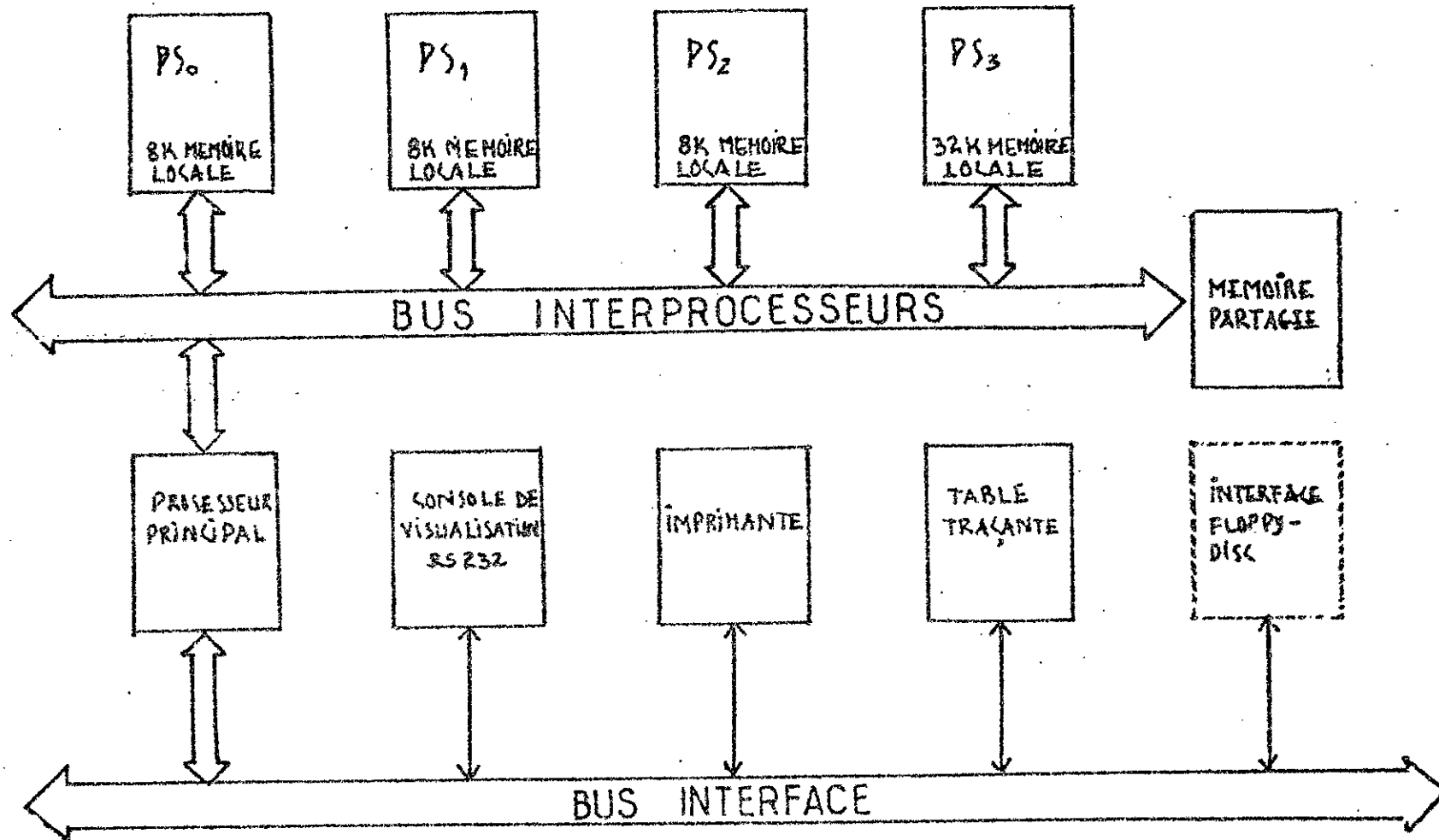


FIGURE III.5. SYNOPTIQUE DU SYSTEME MULTIPROCESSEUR

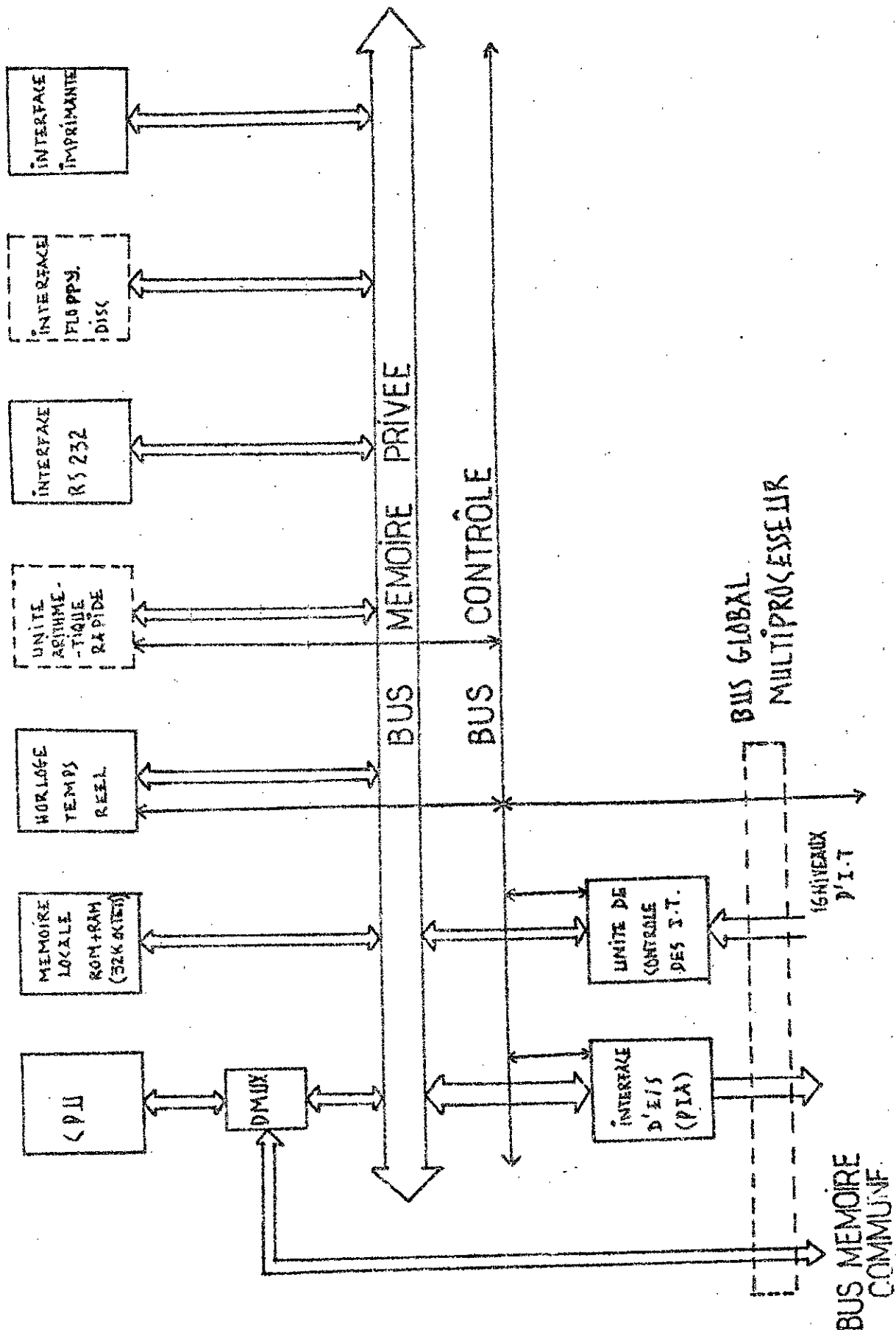


FIGURE III.6. SYNOPTIQUE DU PROCESSEUR PRINCIPAL

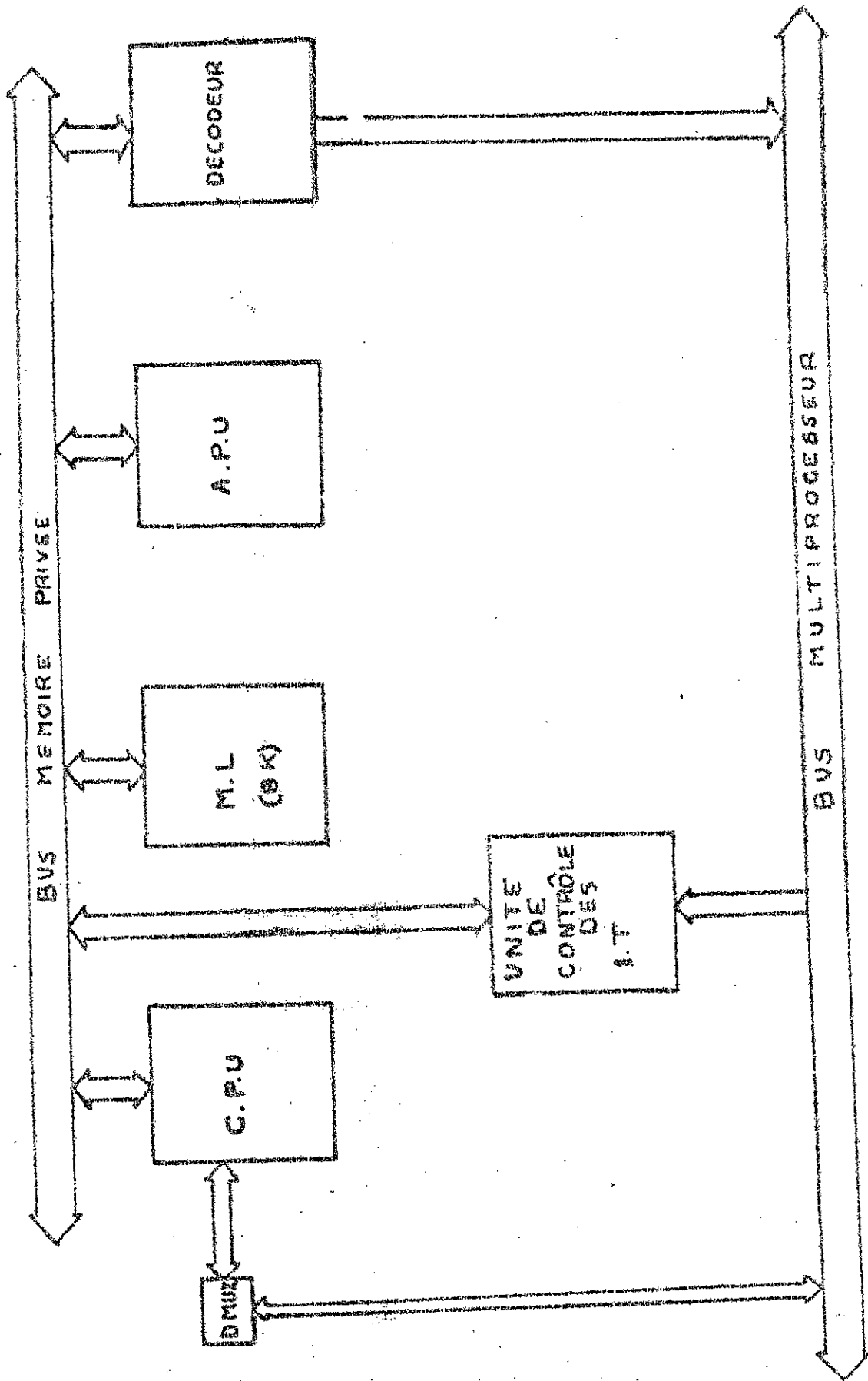


FIGURE III.7. SCHEMA BLOC DU PROCESEUR SECONDAIRE

Le processeur arithmétique utilisé est l'Am 9512 d'Advanced Micro Devices . Ce coprocesseur a été interfacé avec le microprocesseur MC 6800 .

III.1.2.1. Caractéristiques du coprocesseur .

L'Am 9512 ou APU permet de réaliser les quatre opérations élémentaires (+ , - , * , /) en virgule flottante (32 bits ou 64 bits) . Le format des opérandes est compatible IEEE . Sa technologie est en MOS canal N . Ses entrées/sorties sont compatibles TTL . Il se présente sur un support 24 pins ; il est présenté en figure III.5 (Annexe D)

III.1.2.2. Description fonctionnelle .

Le diagramme bloc de la fig. III.9 fait apparaître les unités fonctionnelles de L'APU . Les opérations s'effectuent sur l'unité arithmétique . Celle-ci reçoit un de ses opérandes de la pile , qui contient jusqu'à huit mots de 17 bits avec 2 ports d'entrée et une priorité LIFO (dernier entré , premier sorti) . Le deuxième opérande de l'unité arithmétique est fourni par un bus interne de 17 bits . Ce bus peut transférer le résultat à partir de la sortie de l'unité arithmétique .

Une ROM contient les constantes qui permettent de réaliser les opérations mathématiques , alors que les registres de calcul sont utilisés comme zone mémoire pour les résultats intermédiaires . Un bus bidirectionnel de 8 bits permet les échanges avec l'extérieur .

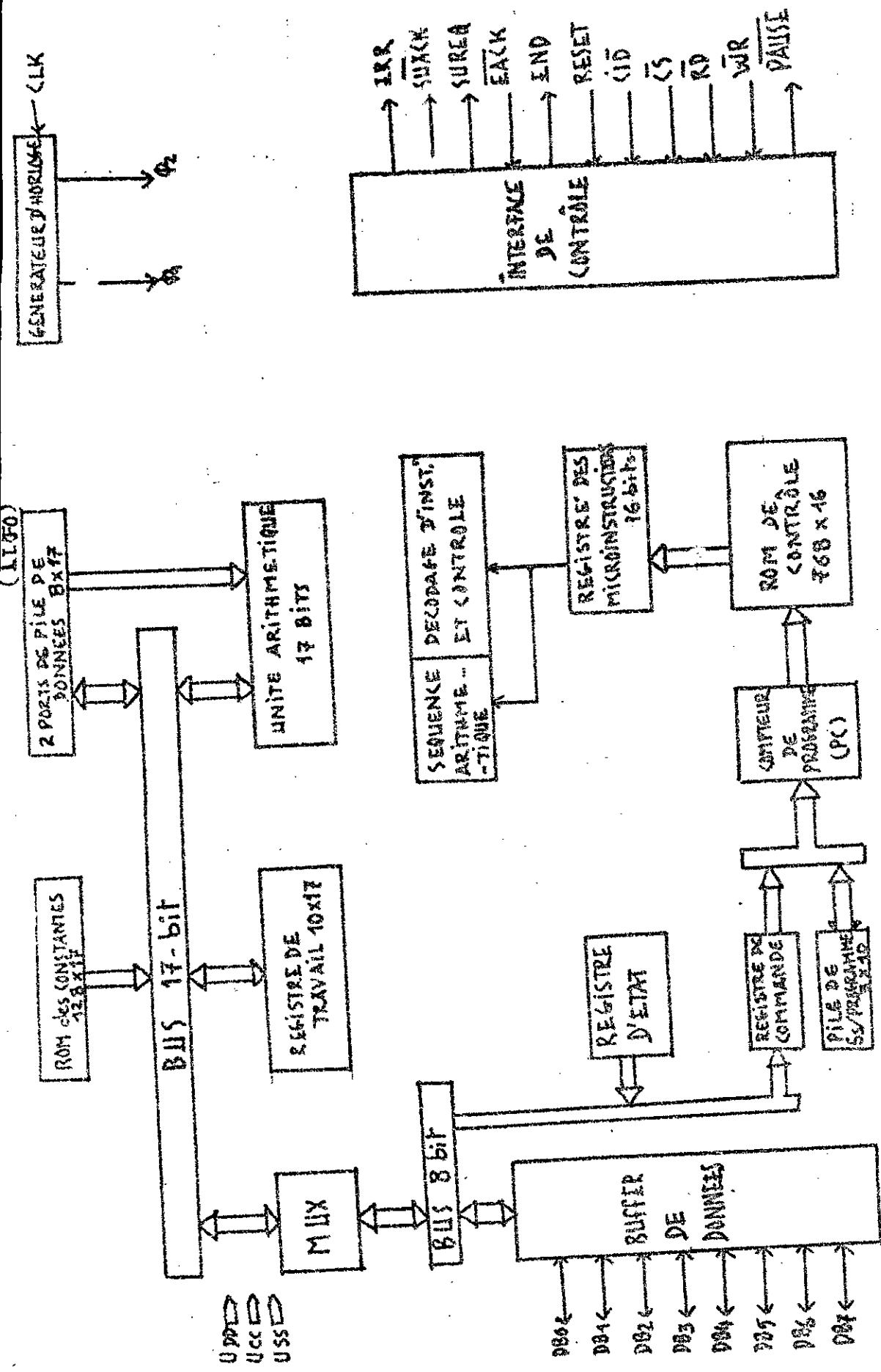


FIGURE III.9. SCHEMA BLOC DE L'AFU

III.1.2.3. Signaux de l'APU .

DB0-DB7 : ce sont les 8 bits du bus de donnée . Ce bus permet l'envoi d'un mot de commande et des opérandes sur l'Am 9512 ; il permet également la lecture du mot d'état et du résultat d'une opération . Notons que les transferts de données avec l'APU peuvent se faire soit en mode programmé soit par accès direct mémoire (DMA) .

CLK : est une entrée horloge reliée à $2 \times f_0$ de l'horloge du MPU . La fréquence de l'APU est alors de 2 MHz .

\overline{CS} : est une entrée qui permet d'activer l'APU , l'autorisant ainsi à communiquer avec le bus de données . Cette entrée conditionne les commandes \overline{RD} et \overline{WR} .

La sortie $\overline{CS5}$ du décodeur adresse l'APU par l'adresse A000 .

\overline{RD} : est la commande de lecture active par son niveau bas lorsque le "chip select" (CS) est activé .

\overline{WR} : est la commande d'écriture active par son niveau bas lorsque $\overline{CS} = 0$.

C/D (command/data) : cette entrée est un bit de sélection de registre . Elle est reliée au bit A0 du microprocesseur . En conjonction avec les entrées \overline{RD} et \overline{WR} , nous obtenons les opérations suivantes :

C/D	\overline{RD}	\overline{WR}	Opération réalisée
0	1	0	Ecriture d'un octet de donnée dans la pile
0	0	1	Lecture d'un octet de donnée dans la pile
1	1	0	Ecriture d'un mot de commande
1	0	1	Lecture d'un mot d'état

PAUSE : cette sortie est un bit d'état autorisant l'échange de données entre le processeur hôte (le MC 6800) et l'APU . Cette sortie est à zéro tant que le transfert n'est pas terminé .Ce signal est envoyé sur l'entrée CLEAR d'une bascule D . Celle-ci , par sa sortie Q fournit le signal à envoyer sur l'entrée "Memory Ready" (MR) de l'horloge MC 6871 du MPU . Ainsi l'état "1" de $\phi 2$ est prolongé .

III.1.2.4. Format du mot de commande .

L'opération effectuée par l'Am 9512 est contrôlée par le processeur hôte qui envoie un mot de commande au coprocesseur . Nous donnons ci-dessous les mots de commande correspondants aux opérations utilisées dans le projet :

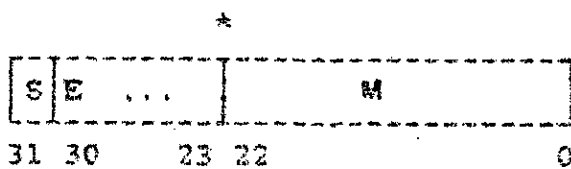
OPERATION	MOT DE COMMANDE	DESCRIPTION
SADD	01	Addition de TOS et NOS sur 32 Bits. Le résultat est sur TOS .
SSUB	02	Soustraire TOS de NOS sur 32 Bits. Le résultat est sur TOS .
SMUL	03	Multiplier NOS par TOS sur 32 Bits. Le résultat est sur TOS .
SDIV	04	Diviser NOS par TOS sur 32 Bits. Le résultat est sur TOS .
FTOS	06	Pousser l'opérande de 32 Bits de TOS vers NOS .

TOS (Top Of Stack) : sommet de la pile de l'APU .

NOS (Next On Stack) : adjacent au sommet de la pile.

III.1.2.5. Format des données .

Les données occupent 32 Bits et sont structurées selon le format suivant :



* Bit implicite qui vaut 1 .

S : Bit de signe de la mantisse .

E : 8 Bits d'exposant .

M : 23 Bits pour la mantisse .

$$N = (-1)^S \cdot 2^{E - (2^7 - 1)} \cdot (1.M)$$

L'exposant est biaisé .

Pour toute opération , le processeur arithmétique restaure le bit implicite . Une normalisation est à effectuer à la fin de chaque opération . Le bit implicite est ensuite retiré avant de restituer le résultat . .

III.1.2.6. Interfacage de l'APU au microprocesseur

La figure III.10 schématise cet interfacage .

III.1.2.7. Programmation de l'APU .

Lorsque l'APU est sollicité pour effectuer une opération deux opérandes sur 32 bits lui sont transmis ainsi que le code opératoire sur 8 bits (un byte) .

Les opérandes sont stockés dans une zone mémoire référencée par le registre d'index ; le transfert de tout opérande vers la pile de l'APU requiert le sous-programme LOGAPU . Le LSB de l'opérande est inséré en premier .

La restitution du résultat nécessite quand à elle , le sous-programme RETAPU . Le MSB du résultat est restitué en premier . Ces deux sous-programmes ont été mémorisés sur les EPROM des PS ; le langage assembleur a été utilisé .

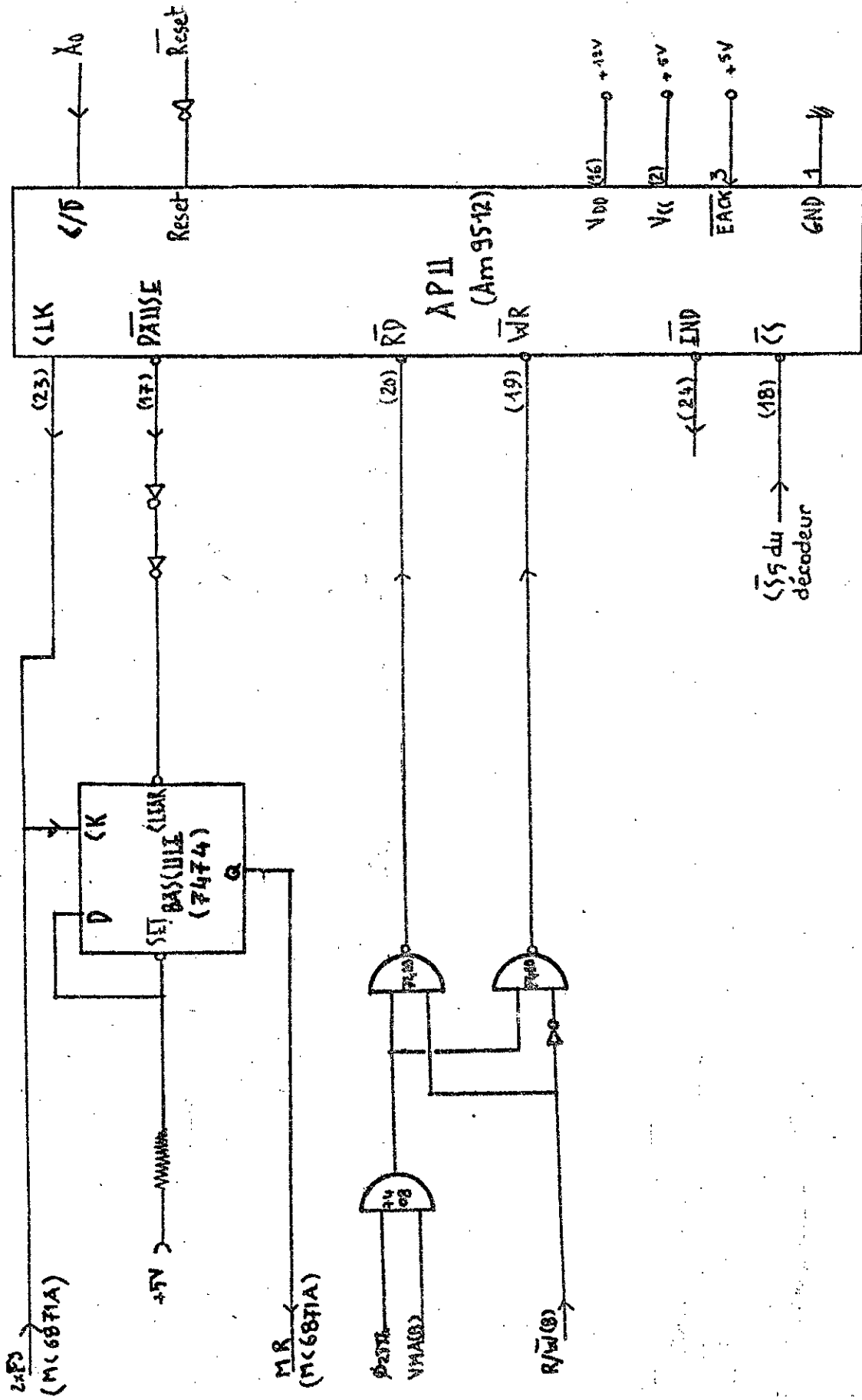


FIGURE III.10. INTERFACE DE L'AM 9512 AVEC LE MICROPROCESSEUR MC 6800 SUR LE PS

Le mot de commande est envoyé de la façon suivante :

LDA A # "CD"

STA A APUCOM

ou APUCOM EQU 0A001H correspondant à l'adresse du registre de commande de l'APU (A0 est mis à 1 , d'où $C/\bar{D} = 1$) .

III.1.3. Interconnexion PS-PP .

Le dialogue entre le processeur principal et les processeurs secondaires est réalisé à l'aide de signaux générés par une logique d'interconnexion sur toutes les unités .

Grâce au port B de son PIA , le PP adresse jusqu'à huit PS d'adresse 0000 à 0111 (sorties PB3 , PB2 , PB1 , PB0) . Le CB2 et le PB5 du même FIA permettent en se combinant avec le signal de sortie du comparateur (OA=OB) du PS, de générer les signaux APPEL (\overline{APP}) et ACQUITTEMENT (\overline{ACQ}) (FIG.III.11) .

Ces signaux d'interruption sont mémorisés à l'aide de bascules D avant d'être injectés sur un contrôleur d'interruption (I.T) du PS .

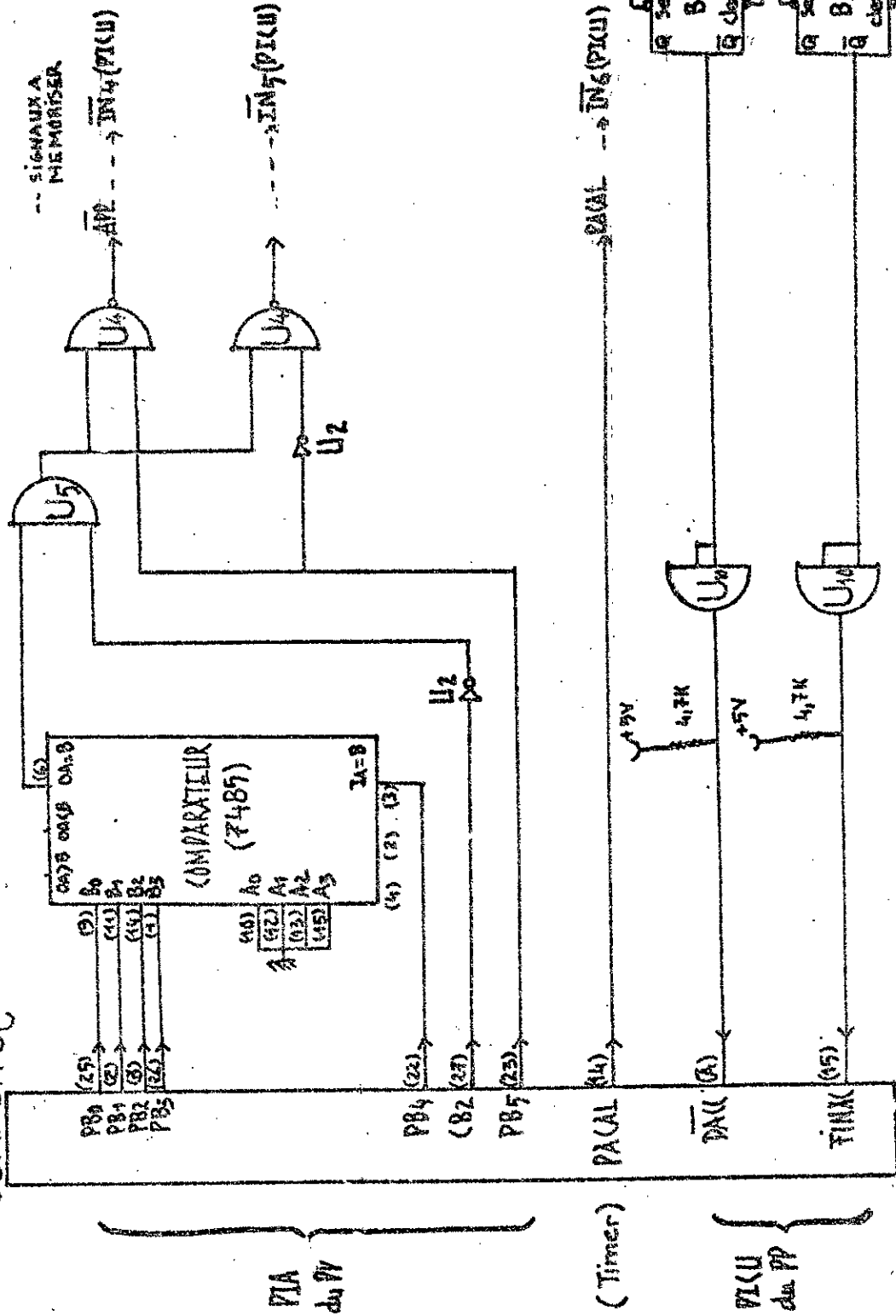
D'autre part , le PS émet des interruptions au PP lorsqu'il désire accéder (ou sortir) à (de) la mémoire commune . Il s'agit des signaux de demande d'accès (\overline{DACC}) ou de fin d'accès (FINAC) (FIG.III.11) .

II.2. Structure logicielle .

Les EPROM de chaque processeur ont été programmées de façon à contenir la configuration du système , le noyau et les tâches attribuées à chaque processeur .

Le noyau initialise le hardware de la machine Multimu ,

CONNECTEUR
COMMUN JC



LEGENDE :

- U10 : 7409
- U2 : 7404 (INVERSEUR)
- U4 : 7409
- U5 : 7408 (AND)
- PICU : 6828
- PAK : P50 → A
- P51 → Z
- P52 → Y
- P53 → X
- adresse A3 A2 A1 A0
- P50 → 0000
- P51 → 0001
- P52 → 0010
- P53 → 0011
- B1 B2 : Bascules D 7474

FIGURE III.11. SCHEMA D'INTERCONNEXION PS / PP

puis, le software . Pour ce dernier , il initialise d'abord la Ready List - ou file des tâches prêtes manipulée par le scheduler . Le noyau se réfère alors au module configuration pour dénombrer les tâches et affecter celles-ci dans la file des tâches prêtes avec le PCB de chacune d'entre elles . La onzième priorité correspond à la tâche REPOS - ou IDLE TASK; elle est exécutée par le scheduler lorsqu'aucune tâche n'est prête en Ready List .

Le noyau crée et initialise les sémaphores . Le compteur et le lien de chaque sémaphore sont donc mis à zéro ce qui signifie qu'à l'initialisation , la ressource est occupée et aucune tâche n'y est en attente . Cette étape est nécessaire pour une bonne synchronisation . Le scheduler scrute la Ready List pour détecter les tâches à exécuter .

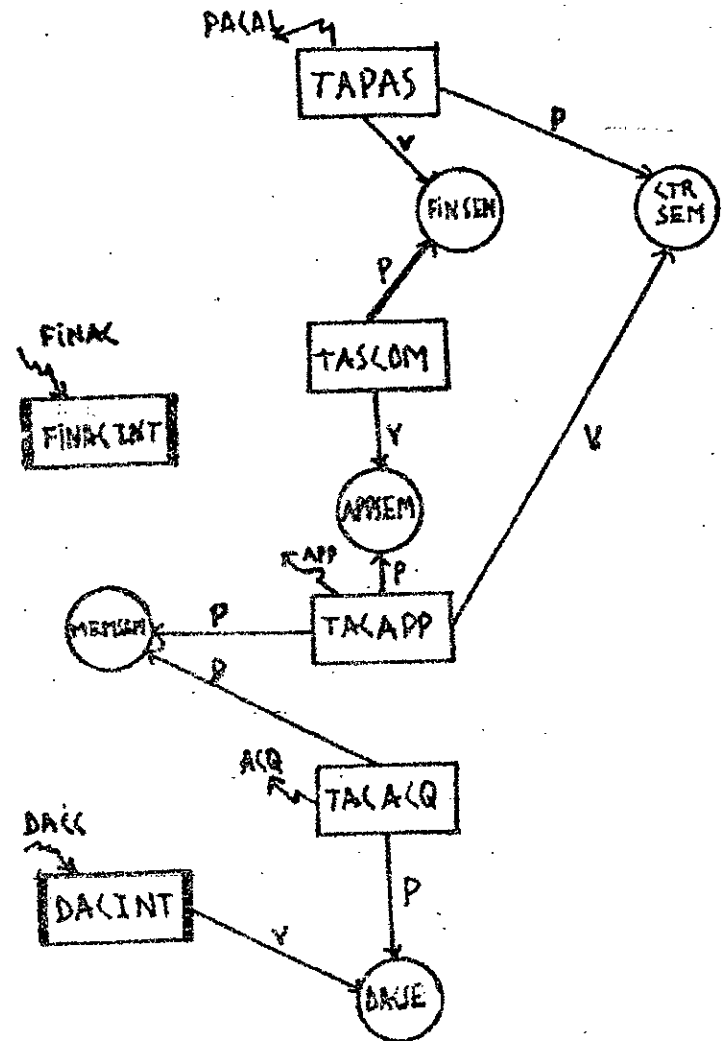
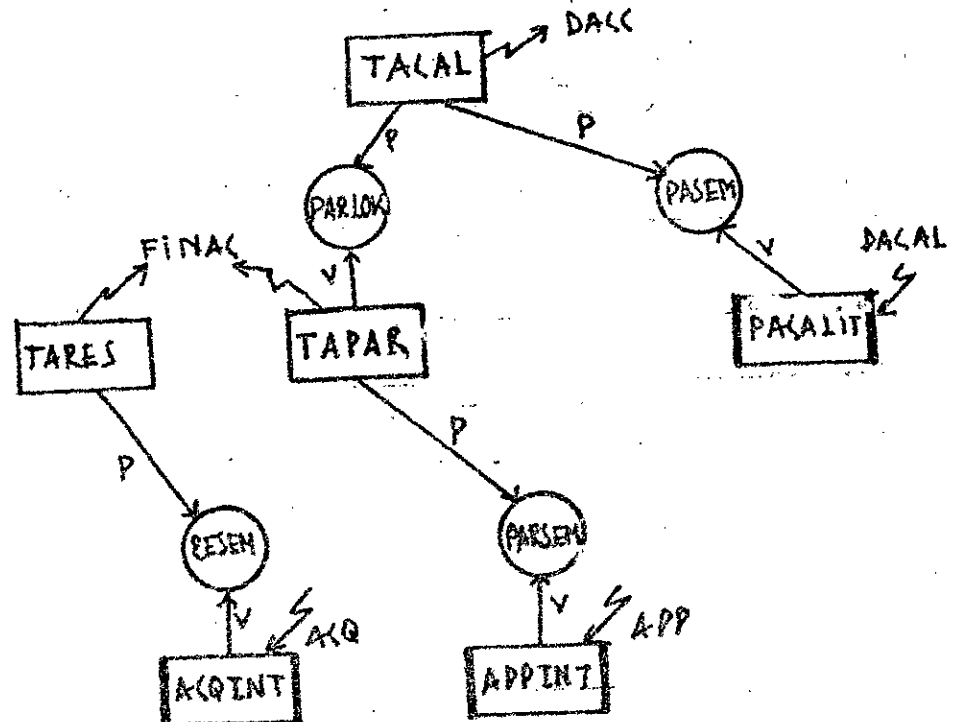
Lorsqu'une tâche sollicite une ressource , elle utilise la primitive P qui décrémente le compteur du sémaphore correspondant à la ressource sollicitée et se met dans la file des tâches en attente du sémaphore . Et ceci , jusqu'à ce que la tâche occupant la ressource la libère par l'intermédiaire de la primitive V . Cette dernière a pour rôle d'incrémenter le compteur du sémaphore et permet ainsi la libération de la ressource .

Sur le système multiprocesseur , le noyau est identique sur tous les processeurs . Le module configuration est le même sur tous les PS et différent sur le PP .

Le module configuration fait apparaître les tâches et

PS

PP



LEGENDE:






	SEMAPHORE		PRIMITIVE
	TÂCHE		INTERRUPTION
	TÂCHE D'INTERRUPTION		

FIGURE III.12. FONCTIONNEMENT DYNAMIQUE DU SYSTEME MULTIMEDI

sémaphores que nous indiquons ci-après .

Sur le PF :

* les tâches sont entre autres : INIT , TAPAS , TACAQ0 , TACAQ1 , TACAQ2 , TACAQ3 , TAPF0 , TAPP1 , TAPP2 , TAPP3 , CONVER .

* les sémaphores sont : MEMSEM , DACSE0 , DACSE1 , DACSE2 , DACSE3 , LOCK , LOCK10 , LOCK12 , LOCK2 , LOCK30 , LOCK32 , CTRSEM , DEBSEM , FINSEM et FAPSEM .

Sur le PS :

* les tâches sont :

- TACAL0 , TAPAR0 , TARESO sur le PS0

- TACAL1 , TAPAR1 , TARES1 sur le PS1

- TACAL2 , TAPAR2 , TARES2 sur le PS2

- TACAL3 , TAPAR3 , TARESE sur le PS3

* les sémaphores sont : PASEM , PARLOK , PARSEM et RESEM .

La fig.III.12 donne le fonctionnement dynamique du multimu .

Le système multiprocesseur Multimu est de structure relativement simple . Cependant , l'accent a été mis sur l'aspect logiciel de cette machine .

La programmation du Multimu a été effectuée en langage assembleur .

Un système d'exploitation a permis de résoudre le problème des sections critiques . Pour les différents processeurs secondaires , les sections critiques sont :

- la table de mémorisation des différents Xi , HEAD1 .

- la table qui mémorise les fi,m (i=0,1,2,3 ; m=0,1,2) , FTAB;

relative à la méthode d'intégration numérique que l'on détaillera dans le chapitre suivant .

Ces deux tables se situent dans la mémoire commune . Celle-ci est alors considérée comme section critique du système multiprocesseur Multimu .

Les sémaphores gérés par le système d'exploitation ont permis de résoudre ce problème de section critique .

Quand au problème de synchronisation entre les tâches , les primitives P et V ont permis sa résolution .

CHAPITRE IV

LES UNITES DE TRAITEMENT NUMERIQUE

L'intégration des équations différentielles ordinaires du modèle de simulation représentant un processus physique continu , a été réalisée à l'aide de la méthode numérique de Runge et Kutta d'ordre 4 .

Cet outil de résolution , permet d'obtenir une intégration assez précise . Son implantation sur le multiprocesseur Multimu est relativement aisée .

Nous présentons dans un premier paragraphe , les méthodes numériques adaptées à la résolution d'équations différentielles représentant un problème non raide .

Le traitement numérique effectué par les processeurs secondaires est alors décrit , en faisant apparaître les algorithmes et organigrammes associés . Un graphe de dépendance des tâches , permettra ensuite , de décrire l'exécution globale correspondant à l'intégration du système d'équations différentielles .

I . METHODES NUMERIQUES .

I.1. Représentation classique des systèmes d'équations .

Beaucoup de problèmes - ou modèles - de physique , chimie , biologie conduisent à la résolution de systèmes d'équations différentielles de la forme :

(1) { dXi/dt = fi(t, X1, X2, ... , Xn) avec i=1,2,...,n
Xi(0) : conditions initiales

où t est compris dans l'intervalle de temps [t0 , t0+T] .

Le système d'équations différentielles (1) est appelé problème de Cauchy .

L'étude des propriétés mathématiques du problème de Cauchy conditionne le choix des méthodes numériques à utiliser .

Le théorème de Cauchy-Lipschitz donne les conditions d'existence et d'unicité de la solution X :

- * si f(. , .) est continue sur [t0 , t0+T] x R^n
* si ||f(t , X) - f(t , X*)|| < L || X - X* || sur [t0 , t0+T] (2)
quelquesoient X et X* appartenant a R^n , L > 0

(conditions de Lipschitz)

alors le problème de Cauchy admet une solution unique .

Pour qu'un problème soit bien posé numériquement , il a été montré que max [exp(L(t))] ne doit pas être trop grand ; t appartenant à l'intervalle [t0 , t0+T] .

I.2. Les méthodes à un pas .

I.2.1. La méthode d'Euler explicite .

C'est la plus simple des méthodes . Sa formulation a été

proposée en 1840 , et reste cependant à la base de nouvelles méthodes de résolution .

On subdivise l'intervalle de résolution $[t_0, t_0+T]$ en N intervalles $[t(k), t(k+1)]$ $k = 0, 1, \dots, N-1$ et l'on pose $h(k) = t(k+1) - t(k)$ pour le pas k ; la solution est de la forme récurrente :

$$X_i(t(k+1)) = X_i(t(k)) + P_k ,$$

avec $P_k = \int_{t(k)}^{t(k+1)} f_i(t, X_1, X_2, \dots, X_n) dt$

L'intégrale sera approchée par $h_k * f(t(k), X(t(k))) + \epsilon_k$

En négligeant l'erreur de consistance ϵ_k , nous avons :

$$\begin{cases} X(k+1) = X(k) + h(k) * f(t(k), X(k)) , k = 0, 1, \dots, N-1 \\ X_0 : \text{conditions initiales données} \\ \epsilon_k = o(h(k)) \end{cases}$$

L'erreur de méthode $e_k = X(t(k)) - X(k)$

La méthode d'Euler est convergente si

$\max |e_k|$ tend vers zéro , $k = 0, 1, \dots, N-1$, lorsque $h = \max h_k$ tend vers zéro .

Deux stratégies ont été mises en place pour le choix de h_k :

* l'extrapolation à la limite de Richardson [12] permet d'améliorer la précision de la méthode .

* le contrôle local du pas [11, 12] permet de minimiser le coût de résolution du problème différentiel et de suivre la régularité de la solution $X(t)$ en adaptant le pas à la régularité .

I.2.2. Problèmes bien conditionnés .

Lorsque la constante de Lipschitz est très grande , le problème est raide . La méthode d'Euler explicite n'est pas adaptée à ce type de problème .

I.2.3. Principe des méthodes à un pas .

Les méthodes à un pas calculent une valeur approchée de $X(t(k+1))$ - ie $X(k+1)$ - à partir de t_k , de $h(k+1)$ et de la valeur approchée X_k de $X(t_k)$ obtenue au pas précédent . Ces méthodes sont définies par des équations récurrentes du type :

$$(3) \begin{cases} X(k+1) = X_k + h_k * \phi (t_k , X_k , h_k) , & k=0,1,\dots,N-1 \\ X_0 \text{ valeurs initiales données dans } R \end{cases}$$

et , nous supposons que ϕ est une fonction à trois variables, continue de $[t_0 , t_0+T] \times R \times [0 , h]$, $h > 0$, à valeurs réelles , qui ne dépend que de la fonction f . Le système (3) est une approximation de la solution du système (1) .

La méthode d'Euler est la méthode à un pas qui correspond à $\phi (t , X ; h) = f (t , X)$; ϕ est indépendant de h .

Dans la méthode d'Euler , l'erreur de discrétisation , $e_k = X(t_k) - X_k$ est proportionnelle à h ($e_k = o(h)$) si f est régulière .

En général , dans les méthodes à un pas , e_k est proportionnelle à h^p avec $p > 1$. La méthode est dite alors d'ordre p .

Le coût d'un pas de calcul , d'une méthode d'ordre p augmente avec p mais ceci est largement compensé par le fait

que pour une même précision , le nombre de pas demandés diminue avec p . Nous obtenons des méthodes qui nécessitent moins de calculs et qui accumulent alors , moins d'erreurs d'arrondis .

Il existe deux types de méthodes à un pas :

- les méthodes explicites où $X(k+1)$ s'obtient directement à partir de (X_k, t_k) ;
- les méthodes implicites comme la méthode d'Euler implicite (rétrograde) où $X(k+1) = X(k) + h_k * f[t(k+1), X(k+1)]$ qui , nécessite la résolution d'une équation algébrique implicite à itérations .

Les méthodes les plus connues sont celles basées sur le développement de Taylor de $X(t_k+h_k)$ au voisinage de t_k et , les méthodes de Runge et Kutta (RK) .

Parmi les méthodes à un pas adaptées aux problèmes réguliers , citons :

- la méthode de Runge et Kutta d'ordre 2 (RK2) donne les approximations suivantes :

$$\begin{cases} X(k+1) = X_k + (h_k/2) * [f(t_k, X_k) + 2 * f(t_k + h_k/2, X_k + (h_k/2) * f(t_k, X_k))] \\ X_0 : \text{conditions initiales données .} \end{cases}$$

Deux évaluations de f sont nécessaires à chaque pas . L'erreur $(X(t_k) - X_k)$ est en $o(h_k^2)$

- la méthode de Runge et Kutta d'ordre 4 (RK4) donne :

$$X(k+1) = X_k + (h_k/6) * [f(t_k; X(k,1)) + 2 * f(t_k + h_k/2; X(k,2)) + 2 * f(t_k + h_k/2; X(k,3)) + f(t_k + h_k; X(k,4))]$$

avec $X(k,1) = X_k$

$$X(k,2) = X_k + (h_k/2) * f(t_k; X(k,1))$$

$$X(k,3) = X_k + (h_k/2) * f(t_k + h_k/2; X(k,2))$$

$$X(k,4) = X_k + h_k * f(t_k + h_k/2; X(k,3))$$

Remarque

Pour la programmation des méthodes de Runge et Kutta ,
il est plus agréable de les écrire sous la forme équivalente
suivante :

$$f_0 = f(t_k, X_k)$$

$$f_1 = f(t_k + h/2, X_k + (h/2) * f_0)$$

$$f_2 = f(t_k + h/2, X_k + (h/2) * f_1)$$

$$f_3 = f(t_k + h, X_k + h * f_2)$$

$$\text{alors } X(k+1) = X_k + (h/6) * [f_0 + 2 * f_1 + 2 * f_2 + f_3]$$

La méthode de Runge et Kutta d'ordre 4 est
adaptée aux problèmes bien conditionnés et réguliers . De plus
elle est stable .

Pour des problèmes mal conditionnés , les méthodes
implicites sont plus adéquates .

Citons la méthode d'Euler implicite où ,

$$X(k+1) = X_k + h * [\theta * f(t_k, X_k) + (1-\theta) * f(t(k+1), X(k+1))]$$

avec $\theta \neq 1/2$, la méthode est d'ordre 1 .

et $\theta = 1/2$, la méthode est d'ordre 2 .

Nous obtenons le θ -schema ; la méthode est stable ,
notamment lorsque $(h * L) < 1$.

Pour une précision fixée , la mise en oeuvre des
méthodes de R.K pour la résolution de problèmes réguliers ,
dépend du contrôle du pas et du choix de l'ordre de la
méthode . La méthode est alors adaptée à la régularité de la
solution et minimise son coût calcul .

Des méthodes de Runge et Kutta emboîtées R.K pp' ($p' > p$)
permettent de diminuer le coût de n (valeur approchée de n) .

I.3. Les méthodes d'Adams .

Contrairement aux méthodes à un pas , les méthodes multiples , calculent une valeur approchée de $X(k+1)$ à l'instant $t(k+1)$ en utilisant plusieurs valeurs approchées X_k , $X(k-1)$,, obtenues aux pas précédents . Parmi les méthodes multiples , celles d'Adams sont à l'heure actuelle , les plus efficaces lorsque le problème est bien conditionné .

Les méthodes d'Adams permettent d'obtenir , par rapport aux méthodes à un pas , pour une précision donnée , des coûts moins élevés .

Cependant , la mise en oeuvre des méthodes d'Adams est très fastidieuse , d'autant plus que le traitement numérique est réalisé à l'aide de microprocesseurs . Nous ne porterons donc pas , d'intérêt à ces méthodes .

I.4. Méthode de Crank Nicholson .

Cette méthode intègre l'équation différentielle donnée sous la forme $\dot{X}(t) = A * X(t) + B * U(t)$.

La méthode de Crank Nicholson est particulièrement intéressante pour une large distribution de valeurs propres de la matrice A , ce qui est souvent le cas dans les problèmes instables .

II. TRAITEMENT NUMERIQUE .

II.1. Position du problème .

Le processus physique choisi pour tester la structure de Multimu , est celui de deux pendules identiques couplés par un ressort .

Ces deux pendules oscillent dans un même plan vertical. La longueur des pendules est l . Le point de fixation du ressort est à une distance a de l'axe de rotation. Les pendules sont de même masse M .

Pour des petits mouvements autour de la position d'équilibre, le processus est décrit par deux équations différentielles du second ordre, données ci-après :

$$(I) \begin{cases} J \ddot{\theta}_1(t) + (k a^2 + b) \dot{\theta}_1(t) = k a^2 \theta_2(t) \\ J \ddot{\theta}_2(t) + (k a^2 + b) \dot{\theta}_2(t) = k a^2 \theta_1(t) \\ \text{conditions initiales données.} \end{cases}$$

θ_1 : position angulaire du pendule 1 ;

θ_2 : position angulaire du pendule 2 ;

$J = M l^2$, moment d'inertie d'un pendule par rapport à l'axe de rotation ;

$b = M g l$;

k : constante de raideur du ressort .

A l'aide de la transformation canonique :

$$\begin{cases} X_0 = \theta_1 \\ X_1 = \dot{\theta}_1 \\ X_2 = \theta_2 \\ X_3 = \dot{\theta}_2 \end{cases}$$

nous passons à la représentation d'état . Le système (I) devient :

$$(II) \begin{cases} \dot{X}_0 = X_1 \\ \dot{X}_1 = -C_2 X_0 + C_1 X_3 \\ \dot{X}_2 = X_3 \\ \dot{X}_3 = -C_2 X_2 + C_1 X_0 \\ \text{conditions initiales données,} \end{cases}$$

$C_1 = (k a^2)/J$ et $C_2 = (k a^2 + b)/J$

Il est nécessaire de choisir une méthode d'intégration numérique pour la résolution du système d'équations (II).

II.2. Choix de la méthode numérique

Le processus des deux pendules couplées est un système stable. De plus, en fixant les conditions initiales relatives au processus physique, nous obtenons un problème bien conditionné. Les méthodes explicites d'Euler, de Runge et Kutta et d'Adams sont alors adaptées à la résolution du système d'équations différentielles représentant le processus.

Les méthodes d'Adams sont pour une précision donnée, de coût moins élevé que les méthodes à un pas (Euler, R.K, ...). Cependant leur mise en œuvre sur multiprocesseur est laborieuse. Pour cette raison, nous écartons les méthodes multipas.

La méthode de Runge et Kutta d'ordre 4 (RK4) a été choisie pour sa bonne précision et son implémentation possible sur le Multimu.

Pour réduire le temps de calcul, nous fixons le pas d'intégration h , au cours de la résolution d'une équation, comme une constante égale, environ, à $T/20$.

II.3. Algorithmes numériques et organigrammes

Le système (II) peut être schématisé par la représentation classique (1) donnée au paragraphe I.1 de ce chapitre.

La méthode de Runge et Kutta d'ordre 4 permet de donner à chaque itération, $X_i(k+1)$ en fonction de $X_i(k)$ du pas précédent et en fonction de $f_{i,l}$ ($l=0,1,2,3$) ; l est relatif à l'ordre de la méthode ; $k=0, \dots, N-1$; N : nombre de pas ; i : numéro de l'équation et par conséquent, du processeur secondaire chargé d'intégrer cette équation ; $i=0, \dots, 3$.

Le système (II) est constitué de quatre équations différentielles du 1er ordre à intégrer.

Les quatre processeurs secondaires du Multimu sont chargés de résoudre ce système par la méthode de RK4. Les résultats partiels ($f(i,m)$; $m=0,1,2$) sont stockés en mémoire commune (MC), et chaque processeur vient y chercher la donnée adéquate à ses calculs. A la fin de chaque pas, le résultat est déposé en MC.

A chaque itération, pour la structure parallèle du Multimu, la méthode de RK4 donne :

$$X_i(k+1) = X_i(k) + h/5 * [f(i,0) + 2*f(i,1) + 2*f(i,2) + f(i,3)]$$

avec $f(i,0) = f_i(t_k, X_0(t_k), \dots, X_3(t_k))$

$$f(i,1) = f_i(t_k + h/2, X_0(t_k) + (h/2)*f(0,0), \dots, X_3(t_k) + (h/2)*f(3,0))$$

$$f(i,2) = f_i(t_k + h/2, X_0(t_k) + (h/2)*f(0,1), \dots, X_3(t_k) + (h/2)*f(3,1))$$

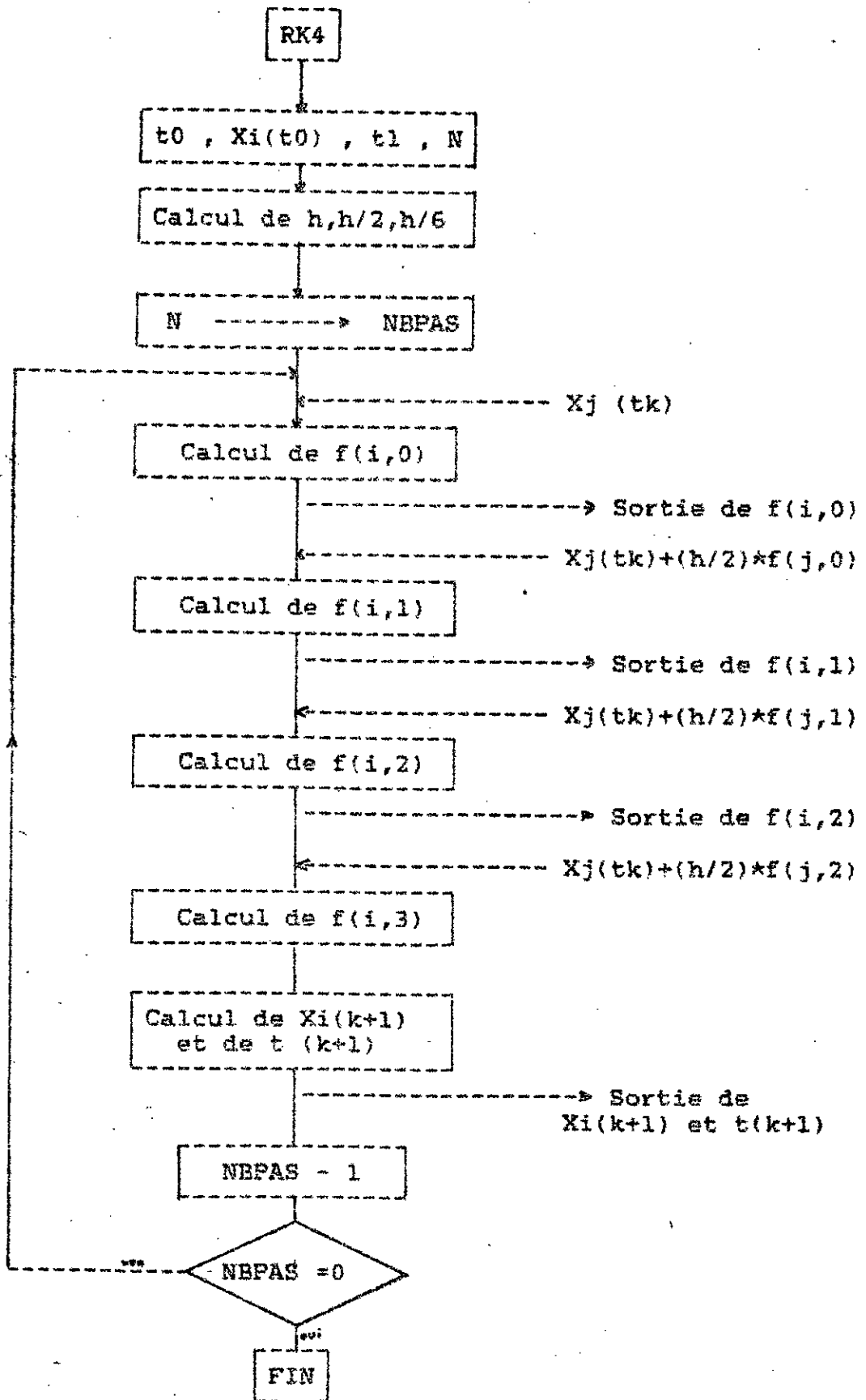
$$f(i,3) = f_i(t_k + h, X_0(t_k) + h*f(0,2), \dots, X_3(t_k) + h*f(3,2))$$

t appartenant à $[t_0, t_0 + T]$, intervalle de temps.

$h = (t_1 - t_0)/N$ est le pas d'intégration numérique.

Le calcul d'un $X_i(t)$ dans la structure parallèle du Multimu est donné dans l'organigramme 1. Le PSI intègre l'équation i suivant cet organigramme ; le PSj lui transmet durant un pas de calcul, les valeurs $f(j,m)$ via la mémoire commune.

ORGANIGRAMME 1



Le système Multimu réalisé , intègre le système d'équations différentielles (II) . Les algorithmes d'intégration des équations $\dot{X}_i = f_i(t, X_0, \dots, X_3)$; $i=0, \dots, 3$ sont implémentés au niveau de chaque P*S*_i .

La méthode de RK4 permet d'obtenir à chaque itération, la valeur de $X_i(k+1)$. Sur chaque P*S* , les algorithmes suivants correspondent aux programmes implantés .

L'intégration de $\dot{X}_0 = X_1$ est attribuée au P*S*₀ . Son intégration à l'aide de la méthode de RK4 donne à chaque itération :

$$X_0(k+1) = X_0(k) + (h/6) * (f(0,0) + 2*f(0,1) + 2*f(0,2) + f(0,3))$$

avec

$$\begin{cases} f(0,0) = X_1(k) \\ f(0,1) = X_1(k) + (h/2) * f(1,0) \\ f(0,2) = X_1(k) + (h/2) * f(1,1) \\ f(0,3) = X_1(k) + h * f(1,2) \end{cases}$$

Les $f(1,m)$ ($m=0,1,2$) sont déposées en MC par le P*S*₁.

Le P*S*₀ vient les récupérer sous le contrôle du P*P* .

L'intégration de $\dot{X}_1 = -C_2 * X_0 + C_1 * X_2$ est attribuée au P*S*₁.

Sa résolution à chaque pas d'intégration donne :

$$X_1(k+1) = X_1(k) + (h/6) * (f(1,0) + 2*f(1,1) + 2*f(1,2) + f(1,3))$$

avec

$$\begin{cases} f(1,0) = -C_2 * X_0(k) + C_1 * X_2(k) \\ f(1,1) = -C_2 * [X_0(k) + (h/2) * f(0,0)] + C_1 * [X_2(k) + (h/2) * f(2,0)] \\ f(1,2) = -C_2 * [X_0(k) + (h/2) * f(0,1)] + C_1 * [X_2(k) + (h/2) * f(2,1)] \\ f(1,3) = -C_2 * [X_0(k) + h * f(0,2)] + C_1 * [X_2(k) + h * f(2,2)] \end{cases}$$

Les $f(0,m)$ et $f(2,m)$ sont respectivement déposées en MC par les P*S*₀ et P*S*₂ . Le P*S*₁ vient chercher ces valeurs sous le contrôle du P*P* .

L'intégration de $X_2 = X_3$ est attribuée au PS2 . A chaque itération , nous avons :

$$X_2(k+1) = X_2(k) + (h/6) * [f(2,0) + 2*f(2,1) + 2*f(2,2) + f(2,3)]$$

avec

$$\begin{cases} f(2,0) = X_3(k) \\ f(2,1) = X_3(k) + (h/2) * f(3,0) \\ f(2,2) = X_3(k) + (h/2) * f(3,1) \\ f(2,3) = X_3(k) + h * f(3,2) \end{cases}$$

Les $f(3,m)$ ($m=0,1,2$) sont déposées en MC par le PS3.

L'intégration de $X_3 = -C_2 * X_2 + C_1 * X_0$ est attribuée au PS3. Le calcul au cours d'un pas d'intégration est donné , par la relation itérative suivante :

$$X_3(k+1) = X_3(k) + (h/6) * [f(3,0) + 2*f(3,1) + 2*f(3,2) + f(3,3)]$$

avec

$$\begin{cases} f(3,0) = -C_2 * X_2(k) + C_1 * X_0(k) \\ f(3,1) = -C_2 * [X_2(k) + (h/2) * f(2,0)] + C_1 * [X_0(k) + (h/2) * f(0,0)] \\ f(3,2) = -C_2 * [X_2(k) + (h/2) * f(2,1)] + C_1 * [X_0(k) + (h/2) * f(0,1)] \\ f(3,3) = -C_2 * [X_2(k) + h * f(2,2)] + C_1 * [X_0(k) + h * f(0,2)] \end{cases}$$

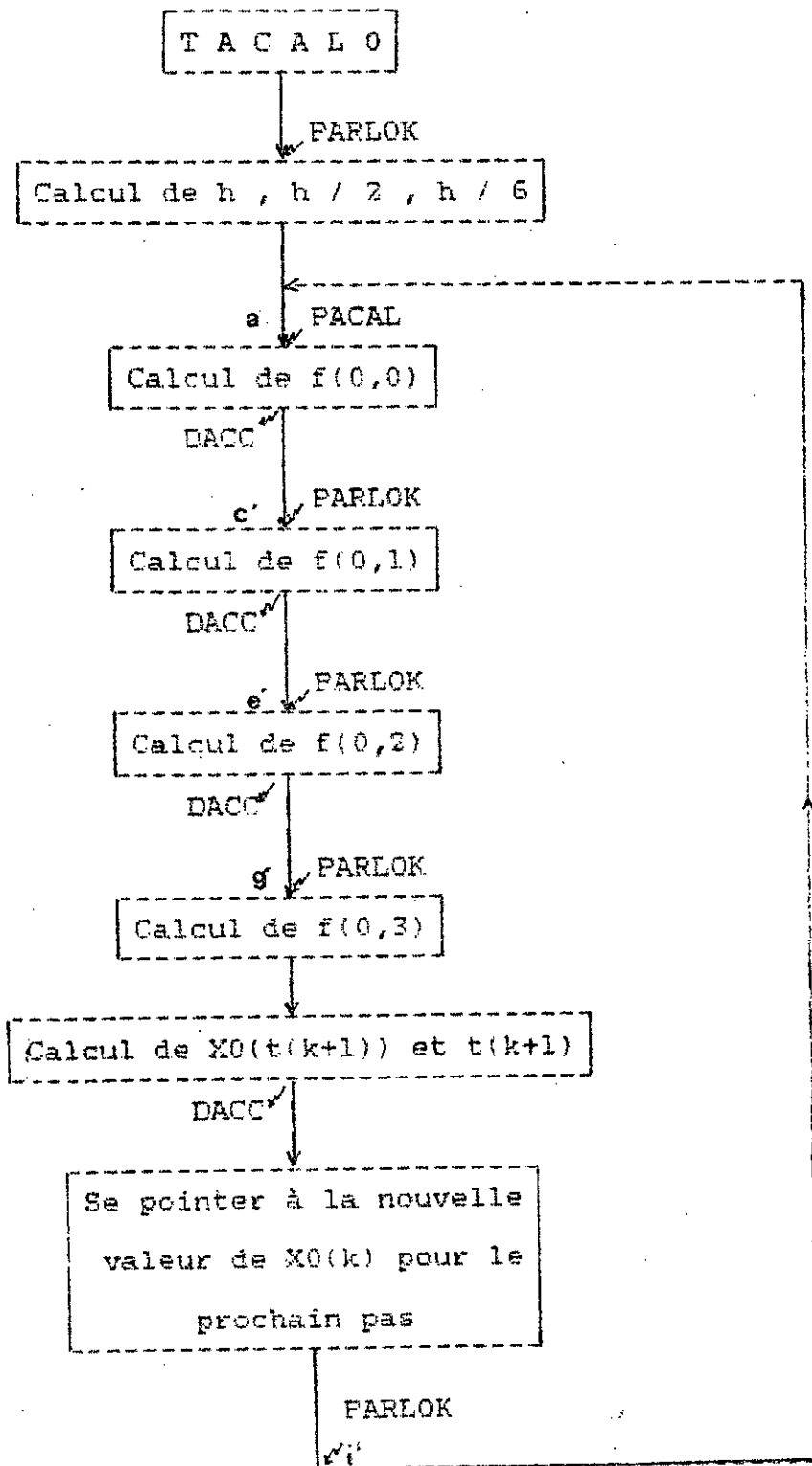
Les $f(0,m)$ et $f(2,m)$ sont respectivement déposées en MC par les PS0 et PS2

L'intégration de chaque équation s'effectue en trois étapes :

- le transfert des conditions initiales , paramètres et résultats intermédiaires ($f(j,m)$) vers le PS i , par la tâche TAFAR i implémentée sur le PS i .
- le calcul au niveau du PS i par la tâche TACAL i
- le transfert vers la MC , des résultats obtenus par le PS i par la tâche TARESi ; $i = j$.

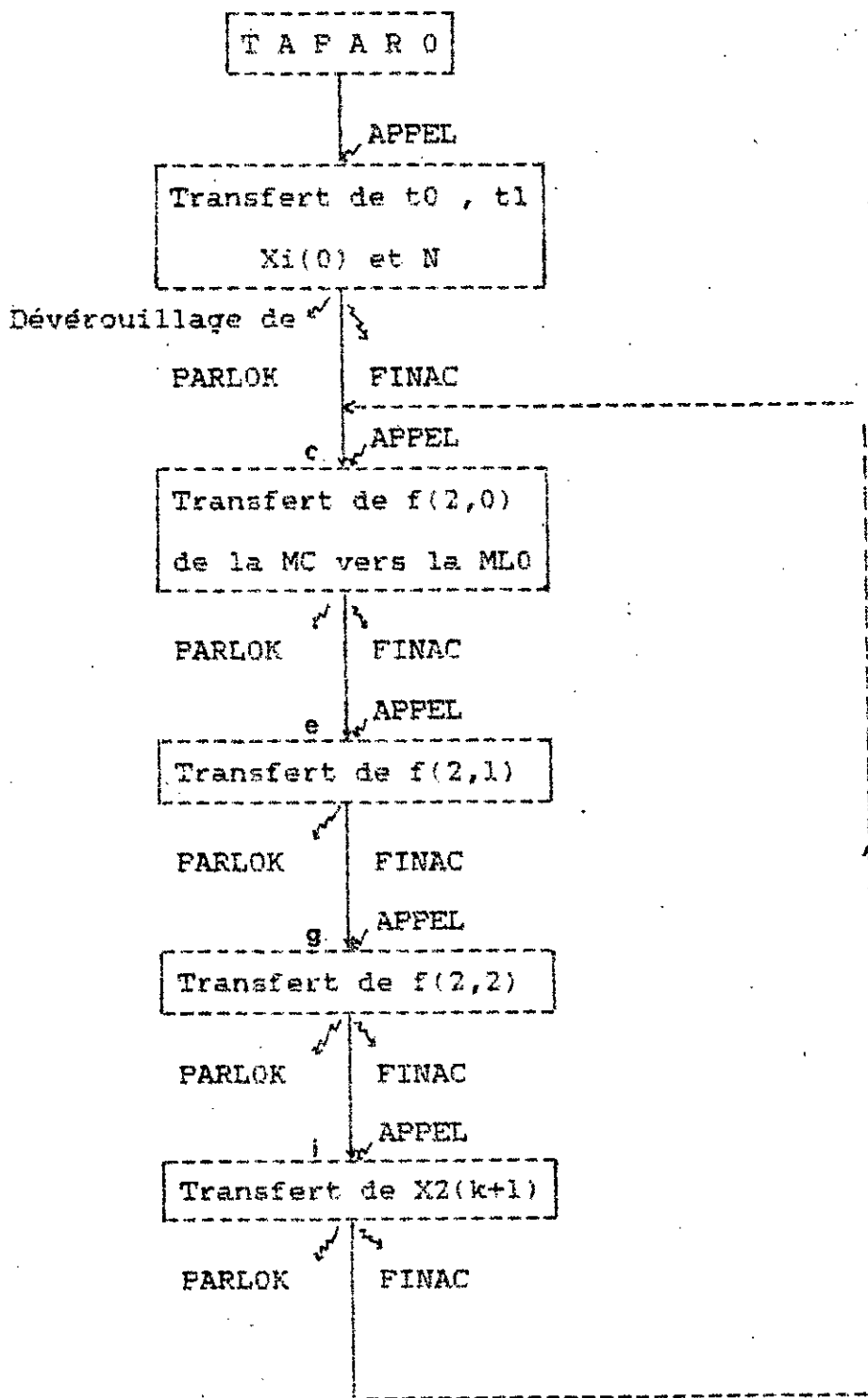
Les organigrammes 2 , 3 et 4 représentent les tâches du PS0 .

ORGANIGRAMME 2

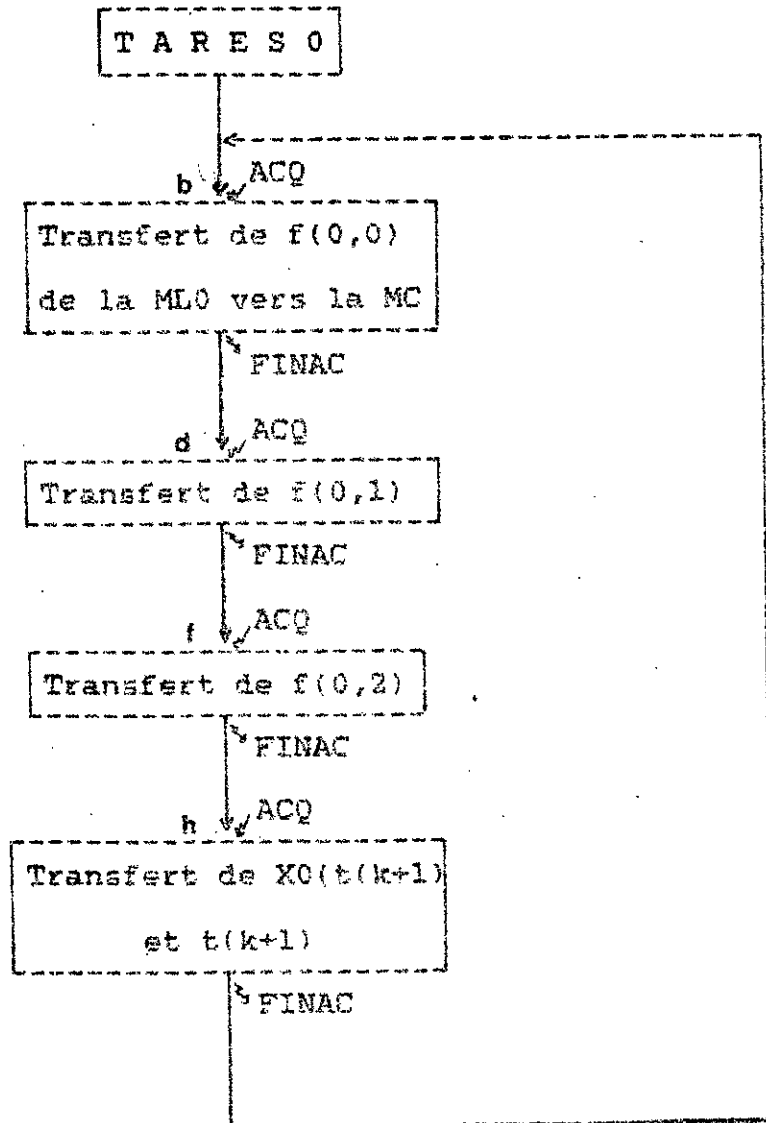


✓ Interruption (Hardware ou Software)

ORGANIGRAMME 3



ORGANIGRAMME 4



Tous les transferts effectués par une tâche TARES_i se déroulent de la MLI vers la MC ; i étant le numéro de l'équation à intégrer (ou du PS) .

La tâche TACAL , de priorité 0 , démarre et se bloque aussitôt sur le sémaphore PARLOK . Ce sémaphore est verrouillé, car les paramètres utiles au calcul n'ont pas encore été transférés par la tâche TAPAR .

La tâche TAPAR , de priorité 1 , démarre à son tour et, se bloque sur le sémaphore PARSEM (celui-ci est déverrouillé lorsqu'un appel est émis par le PP) .

La tâche TARES , de priorité 2 , démarre et se bloque sur le sémaphore RESEM (celui-ci est déverrouillé lorsqu'un acquittement est envoyé par le PP) .

Lorsque le PP émet un appel , la tâche TAPAR s'exécute et transfère les paramètres et conditions initiales utiles au premier pas de calcul ; TAPAR déverrouille alors PARLOK puis émet un signal FINAC (fin d'accès en MC) ; attente du prochain appel sur le sémaphore PARSEM .

La tâche TACAL peut alors exécuter la phase initiale ; et se met en attente du pas de calcul (PACAL) sur le sémaphore PASEM .

a. Lorsque le PP émet un pas de calcul (PACAL) , la tâche TACAL se continue . A la fin du calcul de $f(i,0)$, une demande d'accès est émise par le PS pour le transfert du résultat en MC . Puis la tâche TACAL se bloque sur PARLOK .

b. Lorsque le PP envoie un acquittement (ACQ) , la tâche TARES transfère $f(i,0)$ en MC .

c. Le prochain appel permet de relancer la tâche TAPAR qui recueille les $f(j,0)$ en MC . TACAL calcule alors $f(i,1)$.

d. Un acquittement (ACQ) relance TARES pour le transfert du résultat $f(i,1)$ vers la MC .

e. Un appel permet à TAPAR de recueillir les $f(j,1)$; TACAL calcule alors $f(i,2)$.

f. Un ACQ est émis par le PP pour le transfert de $f(i,2)$ par TARES .

g. APPEL pour recueillir $f(j,2)$. Calcul de $f(i,3)$, $X_i(t(k+1))$ et de $t(k+1)$ par la tâche TACAL .

h. ACQ pour le transfert de $X_i(t(k+1))$ et de $t(k+1)$ de la mémoire locale vers la MC par TARES .

i. Un appel est émis par le PP pour signaler que les résultats $X_j(k+1)$ sont prêts en MC . Au pas suivant l'exécution reprend à partir de a .

Une fin d'accès est émise par le PS à chaque fois qu'un transfert en (de la) mémoire commune est achevé .

La tâche TAPAR déverrouille le sémaphore PARLOK à la fin d'un transfert pour indiquer à la tâche TACAL que les paramètres nécessaires au calcul sont disponibles en mémoire locale du processeur secondaire .

Une demande d'accès en mémoire commune (DACC) est émise par le processeur secondaire pour indiquer qu'un résultat est prêt .

Les $f(j,m)$ sont mémorisées de façon temporaire sur la table FTAB de la MC . Celle-ci contient les $f(j,m)$ d'un pas de calcul . A chaque pas , les $f(j,m)$ du pas précédent sont effacés par simple écriture des nouveaux $f(j,m)$ obtenus au pas courant .

Les $X_i(k+1)$ et $t(k+1)$ sont mémorisées de façon permanente sur la table HEAD1 de la MC .

Les deux tables FTAB et HEAD1 sont initialisées par le PF . Le processeur principal recueille les résultats en virgule flottante sur 32 bits - ou 4 Bytes - et les convertit en décimal avant de les présenter à l'utilisateur sous forme soit , de résultats numériques soit , de courbes .

II.4. Graphe de dépendance .

Les tâches TAPARI , TACALi et TARESi sont exécutées au niveau d'un PSI de façon dépendante . Par contre les tâches TACALi s'exécutent de façon concurrente . Le graphe de dépendance est donné en figure IV .

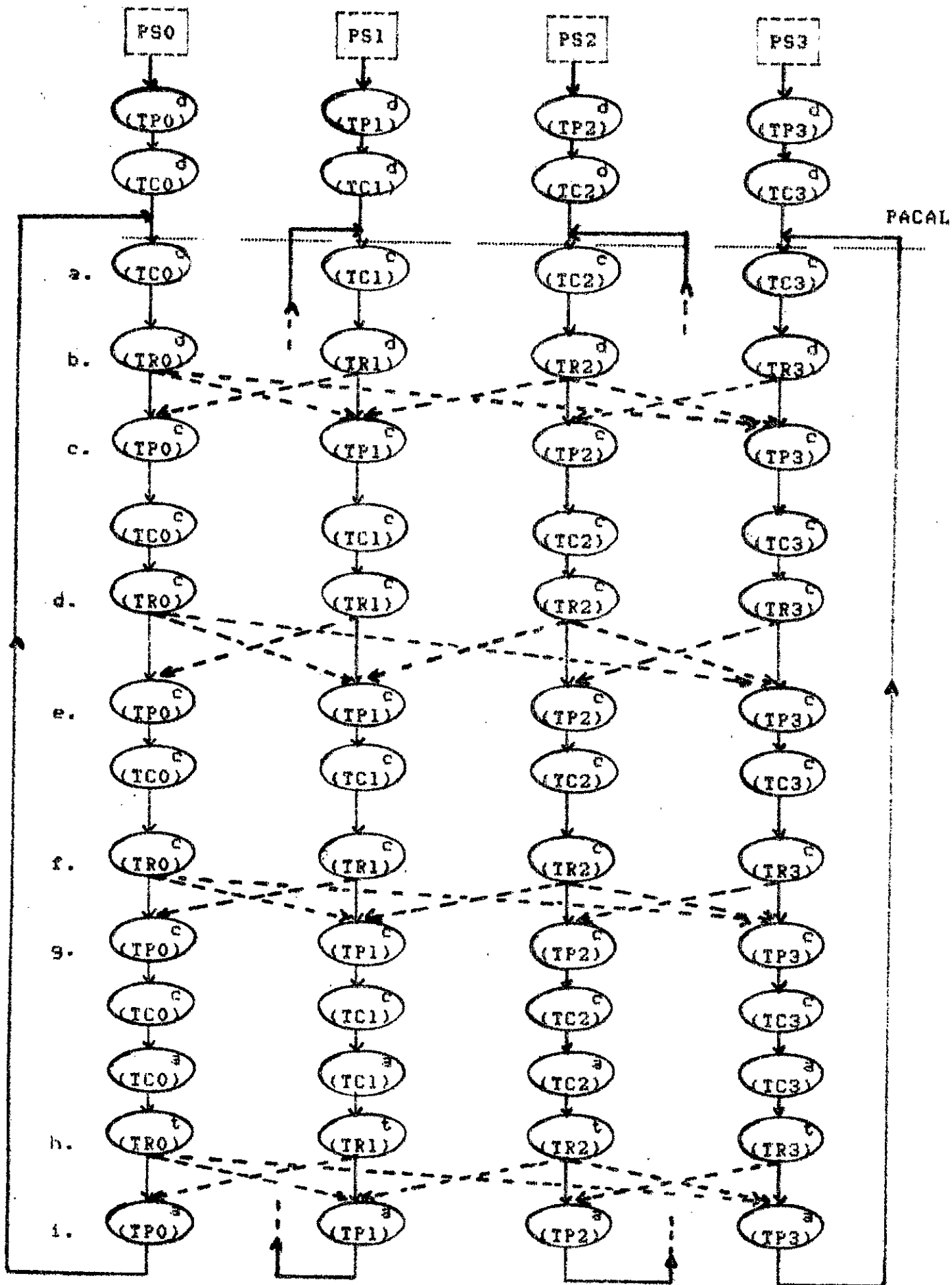
Les relations de dépendances entre une tâche d'un processeur i et une tâche d'un processeur j ($i \neq j$) apparaissent en pointillés .

Remarques

Sur la figure IV , des abréviations ont été utilisées . (TFi) correspond à TAPARI , (TCi) à TACALi et , enfin (TRi) à TARESi . Lorsqu'une tâche démarre , nous porterons l'indice " d " ; lorsqu'elle se continue " c " ; lorsqu'elle se termine l'indice " t " et lorsqu'elle s'arrête l'indice " a " .

Une numérotation alphabétique est portée sur la figure IV , identique à celle choisie au paragraphe II.3 lors de l'interprétation des organigrammes .

FIGURE IV . GRAPHE DE DEPENDANCE .



RESULTATS ET PERFORMANCES

Le système d'équations différentielles (I) , cité au chapitre IV , correspond au modèle mathématique du processus physique de deux pendules couplés P1 et P2 . Ce modèle a été établi pour de petits mouvements autour de la position d'équilibre des deux pendules couplés .

I. RESULTATS

La solution la plus générale du système d'équations représentant le mouvement de P1 et P2 est de la forme :

$$(1) \begin{cases} \theta_1(t) = A_1 \cos(\omega_1 t + \Psi) + A_2 \cos(\omega_2 t + \Psi) \\ \theta_2(t) = -A_1 \cos(\omega_1 t + \Psi) + A_2 \cos(\omega_2 t + \Psi) \end{cases}$$

où A_1 , A_2 , Ψ et Ψ dépendent directement des conditions initiales du mouvement . Selon ces dernières , nous obtenons des solutions particulières . Etudions quelques configurations :

1er cas

A l'instant initial , les deux pendules ont la même élongation initiale et sont abandonnés sans vitesse initiale :

$$\theta_1(0) = \theta_2(0) = \theta_0$$

$$\dot{\theta}_1(0) = \dot{\theta}_2(0) = 0$$

Le calcul effectué à partir de (1) et des conditions initiales permet d'écrire :

$$\Psi = \Psi = 0 ; A_1 = 0 \text{ et } A_2 = \theta_0 \quad \text{d'où}$$

$$(2) \begin{cases} \theta_1(t) = \theta_0 \cos(\omega_2 t) \\ \theta_2(t) = \theta_0 \cos(\omega_2 t) \end{cases}$$

$$\text{avec } \omega_2^2 = \omega_0^2 + (1-Q) \quad ; \quad Q = (k^2 a^4) / (k^2 a^2 + b^2) \quad \text{et}$$

$$\omega_0 = (k a + b) / J$$

les coefficients k , a , b et J auront été définis au paragraphe II.1 du précédent chapitre .

Les deux pendules ont des mouvements semblables , d'ailleurs identiques au mouvement de l'un des pendules battant seul sans ressort de couplage .

La figure V.1.b représente les équations (2) lorsque les paramètres et constantes prennent les valeurs : $k = 4$ USI ; $a = 0.2$ USI ; $m = 0.47$ USI ; $g = 9.8$ USI et l'élongation initiale $\theta_0 = \pi/10$; il s'agit alors de deux sinusoides en phase dont la pulsation est telle que : $\omega_2 = 3.88$ USI .

$$\begin{aligned} \text{La transformation canonique} \quad X_0 &= \theta_1 \\ X_1 &= \dot{\theta}_1 \\ X_2 &= \theta_2 \\ X_3 &= \dot{\theta}_2 \end{aligned}$$

étant choisie (chapitre IV) , le simulateur Multimu obtient , pour les mêmes conditions de fonctionnement , les résultats et courbes de la figure V.1.a ($Y_1=X_0$ et $Y_3=X_2$). Ce sont deux sinusoides en phase, en concordance avec celles de la fig. V.1.b.

2eme cas

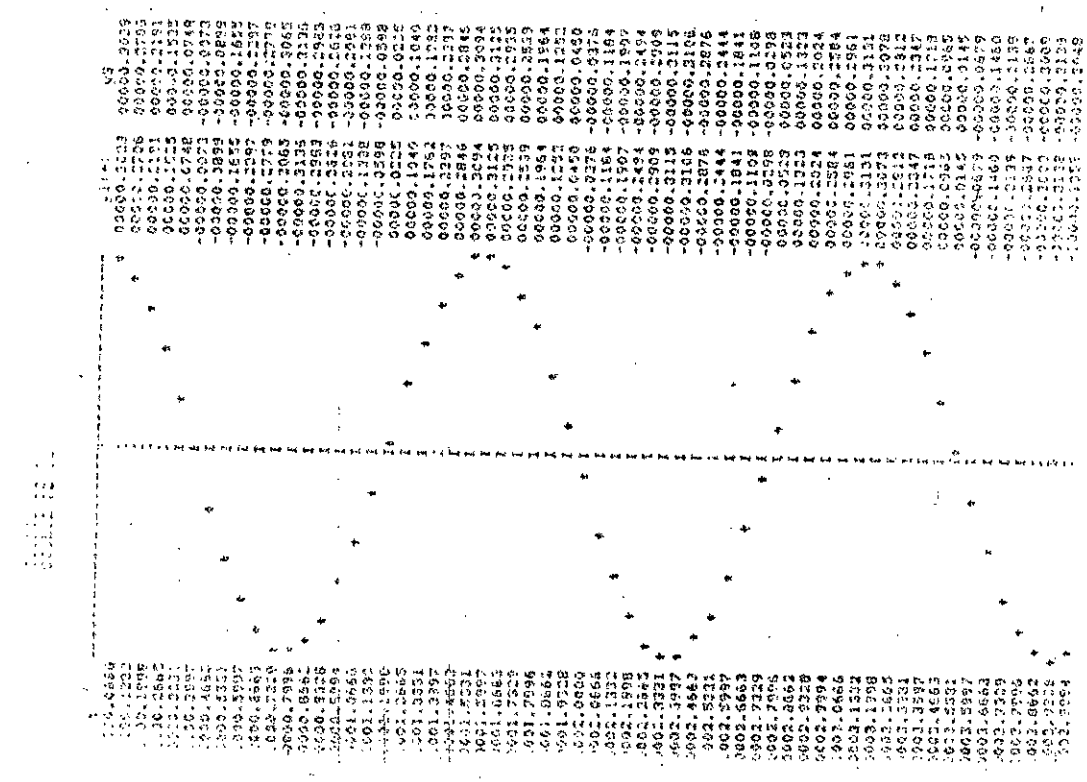
A l'instant initial , les deux pendules ont des amplitudes initiales égales en valeur absolue , mais opposées ; ils sont abandonnés sans vitesse initiale :

$$\text{à } t=0 , \theta_1(0) = -\theta_2(0) \quad \text{et} \quad \dot{\theta}_1(0) = \dot{\theta}_2(0) = 0$$

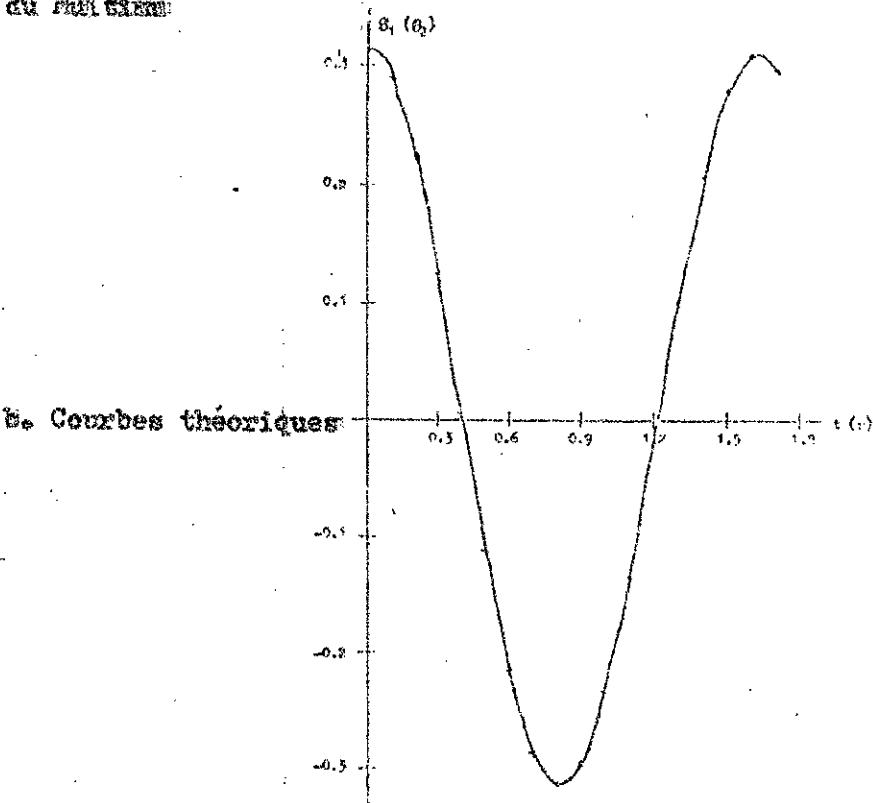
Les équations (1) associées aux conditions initiales permettent de déterminer les constantes A_1 , A_2 , φ et ψ et ainsi trouver :

$$\varphi = \psi = 0 \quad A_1 = \theta_0 , A_2 = 0 . \text{ La solution donnée par les}$$

$$\text{équations (1) devient : } \begin{cases} \theta_1(t) = \theta_0 * \text{COS}(\omega_1 * t) \\ \theta_2(t) = -\theta_0 * \text{COS}(\omega_1 * t) \end{cases} \quad (3)$$



a. Courbes obtenues à l'aide du Multisim



b. Courbes théoriques

FIGURE V.1

Les deux pendules ont des mouvements identiques en opposition de phase . La pulsation de leurs oscillations est telle que :

$$\omega_1 = \omega_0 \sqrt{1+Q}$$

La figure V.2.b représente deux sinusoides en opposition de phase dont les équations sont données par les relations (3) avec $\theta_0 = \pi/10$ et $\omega_1 = 4.085$.

Les conditions initiales insérées dans le Multimu par l'utilisateur sont $X_0(0) = -X_2(0) = \pi/10$ ou $Y_1(0) = -Y_3(0) = \pi/10$

$$\text{et } X_1(0) = X_3(0) = 0 \quad \text{ou } Y_2(0) = Y_4(0) = 0$$

Le système Multimu fournit par l'intermédiaire de la fig. V.2.a deux sinusoides en opposition de phase en concordance avec celles de la figure V.2.b pour , les mêmes conditions de fonctionnement .

Some cas

A l'instant initial $\theta_1(0) = 0$

$$\theta_2(0) = \theta_0 = \pi/10$$

$$\dot{\theta}_1(0) = \dot{\theta}_2(0) = 0$$

La solution analytique est représentée par les équations :

$$(4) \begin{cases} \theta_1(t) = \theta_0 * \text{SIN}(\omega_1 - \omega_2) * t/2 * \text{SIN}(\omega_1 + \omega_2) * t/2 \\ \theta_2(t) = \theta_0 * \text{SIN}(\omega_1 - \omega_2) * t/2 * \text{COS}(\omega_1 + \omega_2) * t/2 \end{cases}$$

$$(5) \begin{cases} \theta_1(t) = \theta_0 * \text{SIN}(\omega_0/2) * (\sqrt{1+Q} - \sqrt{1-Q}) * t * \text{SIN}(\omega_0/2) * (\sqrt{1+Q} + \sqrt{1-Q}) * t \\ \theta_2(t) = \theta_0 * \text{COS}(\omega_0/2) * (\sqrt{1+Q} - \sqrt{1-Q}) * t * \text{COS}(\omega_0/2) * (\sqrt{1+Q} + \sqrt{1-Q}) * t \end{cases}$$

Les deux pendules oscillent en quadrature de phase à la pulsation ω_3 . Leurs oscillations sont modulées . La période des battements des pendules est $T = 4 * \pi / (\omega_0 * (\sqrt{1+Q} - \sqrt{1-Q})) = 4 * \pi / \omega_3$
 $1/T = 1/T_1 - 1/T_2$; $T_1 = 2 * \pi / \omega_1$; $T_2 = 2 * \pi / \omega_2$.

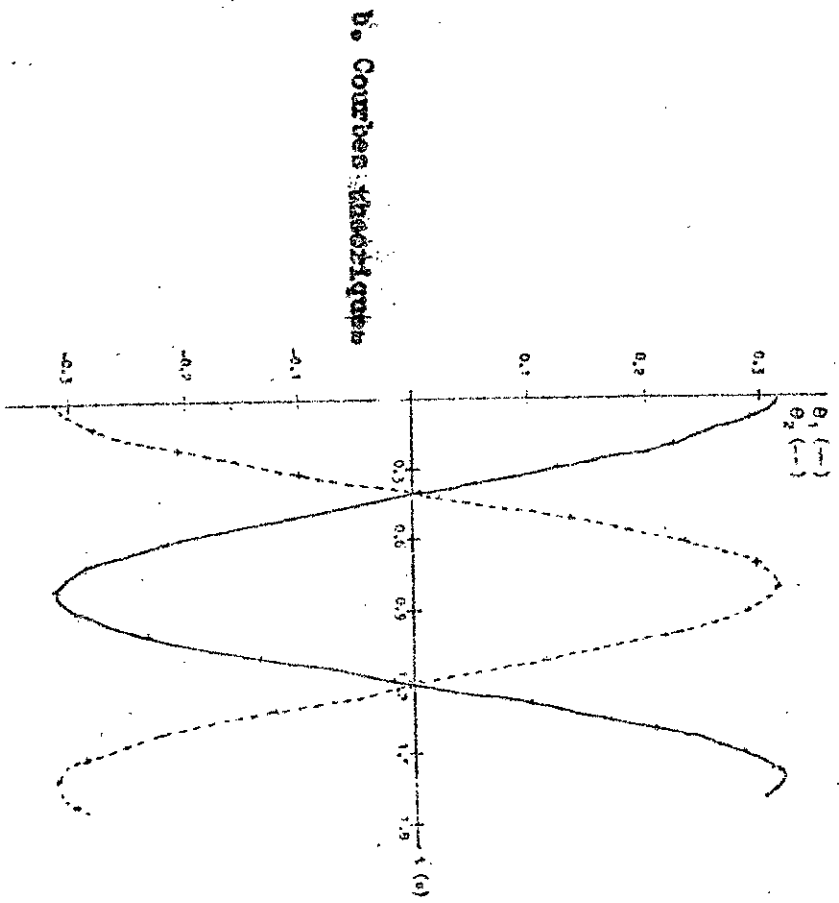
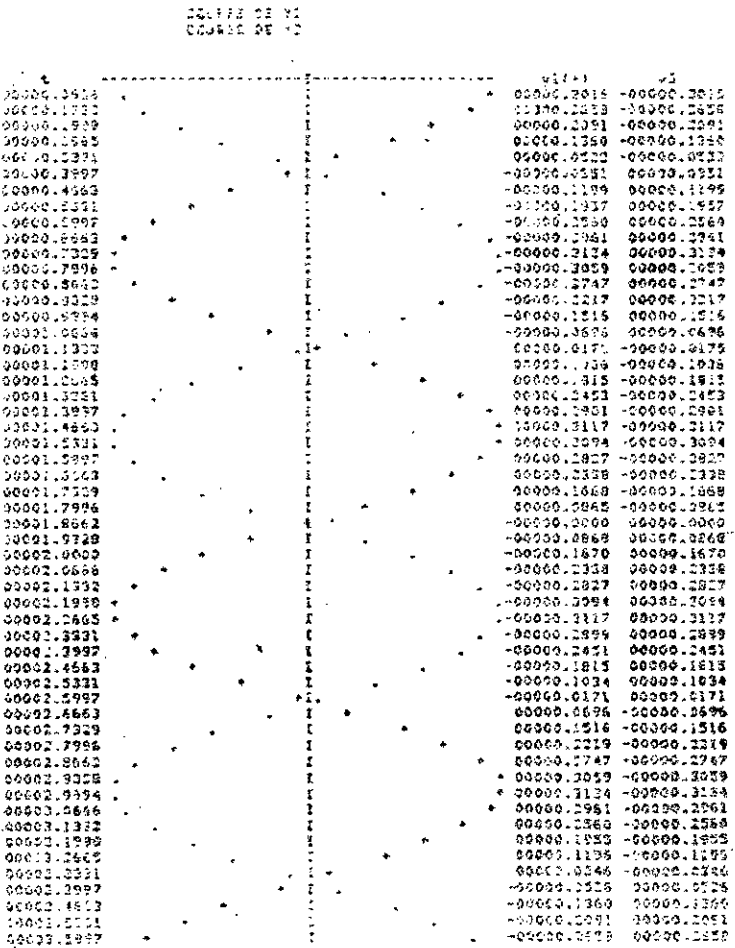


FIGURE V.2

La figure V.3.b représente les élongations θ_1 et θ_2 , respectivement des pendules P1 et P2, où les oscillations sont en quadrature de phase. Une perte d'énergie du pendule P1 est récupérée par le pendule P2.

Pour les mêmes conditions de fonctionnement, le système multiprocesseur réalisé fournit les courbes et résultats de la figure V.3.a. Le phénomène de battement est observable ainsi que la transmission d'énergie du pendule P1 vers le pendule P2.

D'autres configurations ont été simulées par le multiprocesseur Multimu. Des courbes et résultats ont été alors obtenues. Le phénomène de battement a été observé.

La figure V.4 représente le mouvement des deux pendules lorsque: $k = 10$ USI, $\theta_1(0) = \theta_2(0) = \pi/10$ soit $X_0(0) = X_2(0) = \pi/10$ ($Y_1(0) = Y_3(0)$)
 $\dot{\theta}_1(0) = 0$, $\dot{\theta}_2(0) = \pi/10$ soient $X_1(0) = 0$ et $X_3(0) = \pi/10$

La figure V.5 a été obtenue lorsque : $k=10$ USI

$$X_0(0) = Y_1(0) = \theta_1(0) = \pi/10$$

$$X_1(0) = Y_2(0) = \dot{\theta}_1(0) = 0$$

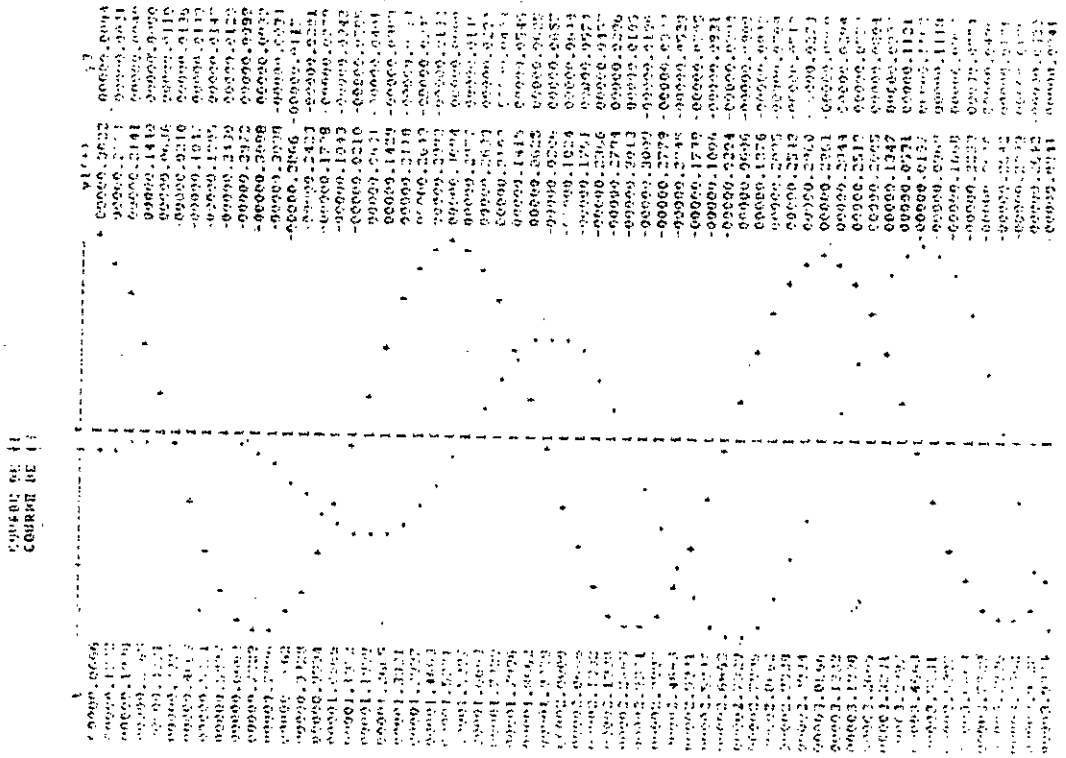
$$X_2(0) = Y_3(0) = \theta_2(0) = 0$$

$$X_3(0) = Y_4(0) = \dot{\theta}_2(0) = -\pi/10$$

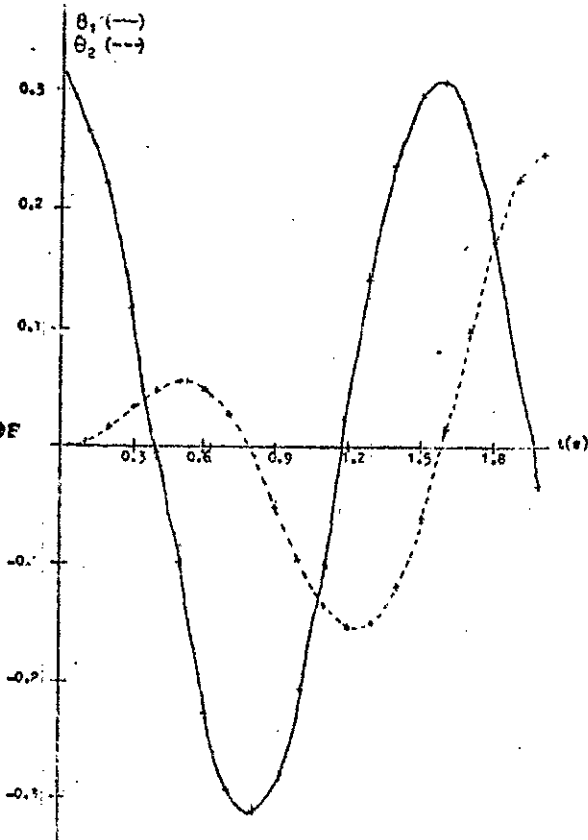
II. INTERPRETATION DES RESULTATS ET PERFORMANCES

Le système Multimu, a fourni des résultats et courbes en harmonie avec ceux obtenus théoriquement.

Ceci prouve que les algorithmes numériques de résolution du système d'équations différentielles à l'aide de la méthode de Runge et Kutta d'ordre 4, ont été implémentées correctement sur la structure parallèle du Multimu.



a. Courbes pratiques



b. Courbes theoriques

FIGURE V.3

COURSÉ DE Y1
COURSÉ DE Y3

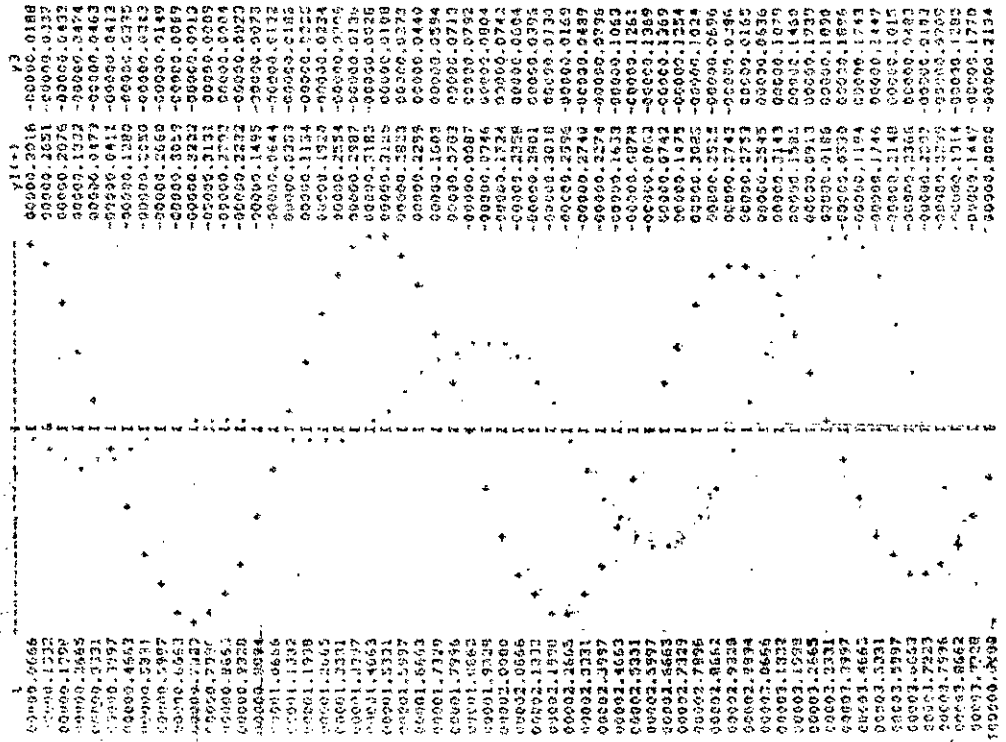


FIGURE 7.5

COURSÉ DE Y1
COURSÉ DE Y3

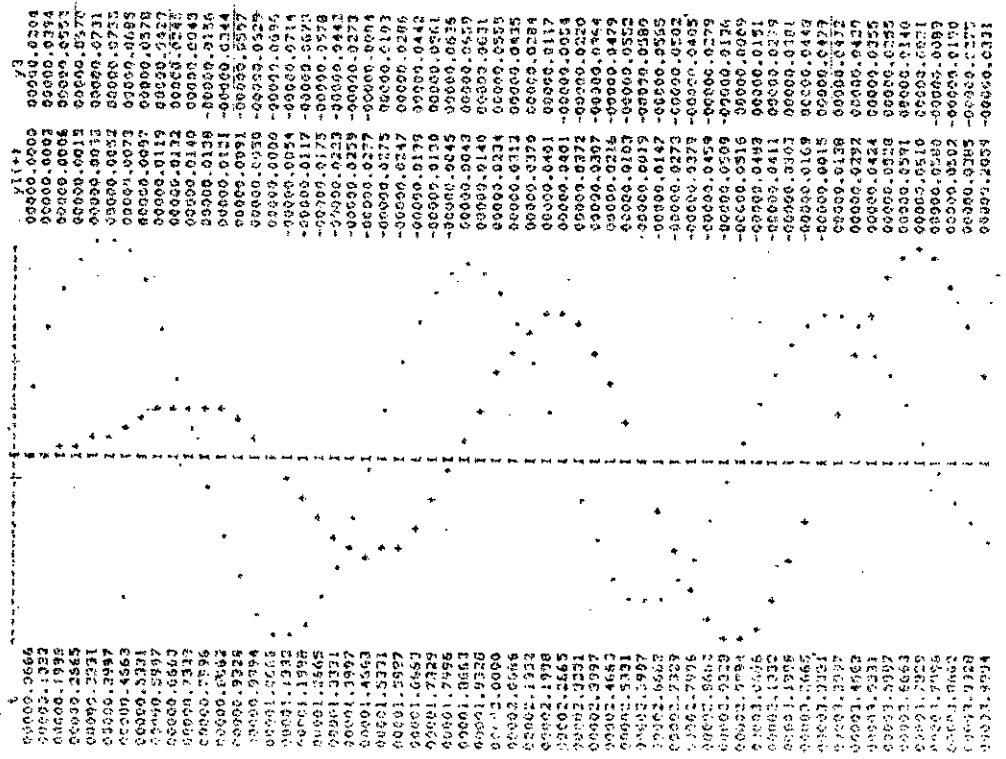


FIGURE 1.4

La synchronisation entre tâches a été supervisée efficacement par le processeur principal ce qui a permis , d'assurer la cohérence dans la dépendance des données .

Les conflits d'accès en mémoire commune ont été également résorbés , grâce à l'utilisation de primitives P et V .

D'autre part , nous nous sommes intéressés à la comparaison des temps de calcul de l'algorithme d'intégration numérique , établis par la machine Multimu et un système monoprocesseur de technologie équivalente .

A cet effet , un timer (MC6840) a été programmé en mode : mesure d'intervalle de temps . Une impulsion est générée par une bascule D qui est mise à zéro au début de l'intégration et remise à 1 à la fin de l'intégration . Cette impulsion commande l'entrée G du timer et permet de déclencher un compteur à la transition négative , puis le bloque à la transition positive . Le nombre de cycles machine nécessaires à un pas de calcul est indiqué par le contenu du compteur .

La même méthode de mesure a été utilisée sur la machine monoprocesseur , constituée d'un processeur secondaire dont l'algorithme d'intégration numérique concerne les quatre équations-différentielles .

Le throughput de la machine Multimu a été alors mesuré et comparé à celui du système monoprocesseur . Un gain de 2.5 a été établi au profit de la machine multiprocesseur .

III. PERSPECTIVES AVEC LES MICROPROCESSEURS 16 BITS

Le facteur du throughput pourrait être nettement amélioré si la technologie du microprocesseur était modifiée .

En effet les microprocesseurs 8 bits sont , de par leur format , adaptés au traitement de caractères codés sur un octet. Mais ils ne sont pas particulièrement adaptés au calcul . Aussi une tendance pour l'utilisation de mots plus longs s'est dégagée depuis quelques temps .

La technologie H-MOS (MOS à canal N) permet une grande intégration et des vitesses de propagation , à travers une porte élémentaire , très élevées . Des microprocesseurs 16 bits tels le 8086 d'INTEL et le 68000 de MOTOROLA utilisent cette technologie .

a. Le 8086 d'INTEL

Ce microprocesseur a été doté de signaux de contrôle spécifiques pour pouvoir être utilisé dans une architecture multiprocesseur . Nous citons quelques uns de ces signaux :

- un signal LOCK (broche de sortie) assure la fonction de verrouillage lorsque l'exclusivité du bus est voulue pendant la durée d'une instruction .
- un signal REQUEST-GRANT donne la possibilité de désélectionner un processeur avec priorité absolue .
- deux sorties additionnelles provenant du 8086 , définissent l'activité de sa file d'attente interne (Q50 et Q51) . Ces lignes sont destinées aux coprocesseurs et permettent de suivre la gestion interne de la file d'attente .

b. Le 68000 de MOTOROLA

Le fonctionnement asynchrone du 68000 permet de travailler aisément avec des systèmes opérant à des vitesses différentes . Il est doté de signaux spécifiques pour évoluer dans une architecture multiprocesseur (BR , PG , DTACK , EGACK).

CONCLUSION

La simulation d'un processus physique continu, dont le modèle mathématique est représenté par un système d'équations différentielles, nous a permis d'étudier et de réaliser une architecture parallèle.

En effet, ce modèle est décrit par des équations différentielles interdépendantes dont l'interaction est, relativement minimale. Ceci, nous a conduit à la parallélisation d'algorithmes d'intégration numérique du système différentiel pour une architecture parallèle de type MIMD.

Au cours de cette étude, nous avons été confrontés à de nombreux problèmes, entre autres, la parallélisation d'algorithmes numériques et, la synchronisation entre tâches partageant une ressource commune.

Ces problèmes ont pu être résolus et nous avons ainsi, intégré un système de quatre équations différentielles représentant un phénomène physique, à l'aide de la machine parallèle Multibus.

Cette étude fait apparaître deux résultats essentiels :

- la parallélisation d'algorithmes d'intégration numérique à partir de la méthode de Runge et Kutta d'ordre 4 a été bien réalisée.
- le throughput du système de traitement multiprocesseur a été nettement amélioré par rapport à celui d'une structure monoprocesseur de technologie équivalente, malgré les temps d'intercommunication supplémentaires.

De plus l'architecture retenue dans la réalisation de la machine Multimu , est très flexible . Elle permet d'apporter des extensions et modifications futures . Entre autres , augmenter le nombre de processeurs élémentaires en veillant à ne pas dégrader le throughput du système . Ainsi , une application décrite par un plus grand système d'équations différentielles , dont le partitionnement de tâches permet une interaction minimale pourrait y être simulée .

D'autre part , des perspectives s'offrent à ce système multiprocesseur :

- diminuer les temps d'accès à la mémoire commune , en la découpant en bancs mémoire . Ce type de mémoire devra alors être géré par des multiplexeurs d'entrée et de sortie .
- on peut également anticiper la lecture des prochaines instructions probables avec leurs opérandes , dans une mémoire très rapide et de faible taille (antémémoire) . Ceci améliore le temps d'accès en mémoire commune .
- alléger le logiciel en sophistiquant la partie hardware et , adjoindre un arbitre de bus pour régler les conflits d'accès en mémoire commune ; les pertes de temps dues à l'exécution d'un système d'exploitation complexe serait alors diminuées .

A l'heure actuelle des possibilités énormes sont offertes pour les architectures multiprocesseurs .

La technologie met à la disposition de concepteurs de systèmes distribués à microprocesseurs , des puces qui simplifient l'interaction et la coordination à l'aide de signaux de contrôle spécifiques .

Les microprocesseurs 16 bits sont plus appropriés au multiprocessing et au traitement numérique .

Des modifications substantielles au niveau technologique peuvent être amenés sur le système Multimu :

* remplacer le microprocesseur 8 bits utilisé , par un microprocesseur 16 bits (voire 32 bits) plus adapté au multiprocessing .

* utiliser un bus multiprocesseur .

* éventuellement , passer de la structure bus partagé à une autre structure matérielle , lorsqu'on dépasse un certain nombre de processeurs .

Il existe également des circuits à forte intégration , contenant plusieurs processeurs sur une seule puce , en technologie WSI (wafer scale integration) et , qui permettraient de probantes améliorations dans les systèmes multiprocesseurs .

- BIBLIOGRAPHIE -

- [1]. TRAN VAN KHAI . " Architectures Multiprocesseurs "
Revue technique Thomson CSE vol.11 no 2 Juin 79
- [2]. Eli I.FATHI et Moshe KRIEGER . " Multiple
Microprocessor Systems : What , Why ,and When "
IEEE March 83 , p.219
- [3]. Tom BALPH and John BLÁCH (MOTOROLA) . " Multiprocessing:
could bring out a system's best "
Electronic Design March 18 , 1982
- [4]. D.COMTE et J.C. SYRE . " Les Supercalculateurs "
La Recherche no 147 Sept.1983
- [5]. Patrice QUINTON . " Les Hyperordinateurs "
La Recherche no 167 vol.16 p.740 Juin 1985
- [6]. D.COMTE et J.C.SYRE . " Structure des calculateurs
avancés " Note de cours 82 . Toulouse .
- [7]. J.P.MEINADIER . " Structure et fonctionnement des
ordinateurs "
Edition Larousse . 1971 . Série Informatique .
- [8]. Ronald LEVINE . " Les Superordinateurs "
Pour la Science . Mars 1982 .
- [9]. Habannakeh Midani HUSSEN . " Conception et réalisation
d'une architecture multiprocesseur flexible -
application au contrôle de processus industriel "
Thèse en Génie Informatique INPG 79 .
- [10]. Jocelyne ERHEL . " Parallélisation d'algorithmes "
Thèse 3eme cycle - PARIS 1982 .
- [11]. M.CROUZEIX - Alain L.MIGNOT . " Analyse numérique des
équations différentielles " EDITION MASSON

- [12]. E. BOUMGHAR . Rapport de notes concernant le séminaire
" Résolution des équations différentielles "
présenté par L. MIGNOT le 03-07-84 au CEN (A.S)
- [13]. B. COMBE . " Analyse Numérique "
Polycopies de cours à SUPELEC PARIS
- [14]. Dennis S. BERNSTEIN . " The treatment of inputs in real
time digital simulation "
Digital Simulation Techniques revue SIMULATION aug.79
- [15]. BROEWAYS . " SPECTRE : proposition de noyau normalisé pour
les exécutifs temps réel "
TSI Janvier 1984
- [16]. G . de GRANDI , TANG YUN LI . " DELASH - Distributed
Operating System For Microprocessors "
Software Practice and Experience
EURATOM ISPRA ITALIE
- [17]. W.W. CHU L.J. HOLLOWAY M.T. LAN K. EEE . " Task allocation
in data distributed processing "
IEEE COMPUTER Nov.1980 p.57
- [18]. Serge CASTAN . " Architectures adaptées au traitement
d'images "
TSI vol.4 no 5 p.431 1985
- [19]. B.J. SMITH . " A pipelined share resource MIMD computer "
International conference on parallel processing Aug.,78
- [20]. M.J. FLYNN . " Some computer organisations and their
effectiveness "
IEEE transactions on computers , Sept.72
- [21]. D. COMTE , N. HIEDI , J.C. SYRE . " Etude et réalisation
d'un système multiprocesseur à assignation unique
Rapport final CERT-DERI, Janv 80 "
- [22]. BATCHER . " Architecture of a massively parallel processor "
The 7th annual Symp. on Computer Architecture ,
La Baule 1980 .
- [23]. A. HELIFA . " Système multimicroprocesseur pour la simula-
-tion de processus physiques continus "
SIM 001-83-AS-1983 CDTA CEN ALGER

- [24]. A. HELIFA . * Système d'exploitation multitâches temps réel pour un système multimicroprocesseur *
SIM 002-84-AS- 1984 CDIA CEN ALGER .
- [25]. H. TEDJINI . * Rapport de présentation du projet concernant un simulateur de processus physiques continus *
CEN - 2- 11- 1982 ALGER .
- [26]. H. TEDJINI . * Multiprocesseur et simulation *
Exposé effectué au sein du laboratoire AS .
ALGER 82
- [27]. S.P. KARTASHEV S. KARTASHEV . Designing and programming modern Computers and Systems . Vol.1
LSI MODULAR COMPUTER SYSTEM
- [28]. P. VERJUS . EDITORIAL : Les recherches sur le parallélisme en France .
TSI Vol.4 no 5 Juin 1985
- [29]. R.W. HOCKNEY C.R. JESSHOPE
PARALLEL COMPUTERS
Editeur ADAM HILGER Ltd BRISTOL 1981
- [30]. C. WEITZMAN . * Distributed micro/minicomputer systems .
Structure , implementation and application *
Prentice Hall Inc., Englewood cliffs ,
New Jersey 1980 .
- [31]. J.C. COMFORT . * The simulation of a master-slave event set processor *
SIMULATION March,84 p.117
- [32]. Ronald M. HIDINGER . * Digital computer Benchmarks of a continuous system simulation *
SIMULATION July,82
- [33]. G.A. KORN J.V. WAIT .
DIGITAL CONTINUOUS SYSTEM SIMULATION
Editeur Mc GRAW-HILL . USA .64
- [34]. Kemal EFE . * Heuristic models of tasks assignment scheduling in distributed systems *
COMPUTER 1982
- [35]. Ph. BERGER , D.COMTE , N.HIEDI , B.PELOIS et J.C.SYRE
* Le système LAU : un multiprocesseur à assignation unique * TSI

- [36]. P.J. DENNING , T.D.DENNIS , J.A.BRUMFIELD
* Low contention semaphores and ready lists *
ACM Oct. 1981, Vol.24 no 10
- [37]. M.J.TRASK , R.O.PETTUS * A microprocessor real-time
executive for fonctionnally-partitioned computer systems *
IEEE 1981 p.444 - CH 1650-1/81 .
- [38]. S.CASTAN . * A multi level architecture for image
processing . Algorithmes and performances expected *
Nato Asi on Computer Italy Juin 83
- [39]. G.de GRANDI , R.OUIGUINI * Simulation d'un système de
mesure nucléaire autour d'un moniteur temps
réel multitâches *
Revue Sciences et Techniques Nucléaires
EUR 8598 FR 1983
- [40]. L.E.SHAMPINE , H.A.WATTS and S.M.DAVENPORT
* Solving nonstiff ordinary differential
equations . The state of the art *
SIAM Revue Vol.18 no 3 July 76
- [41]. A.TODA , M.IMAI , H.INAMORI , K.HIYAMA et M.HATADA
* A parallel processing simulator for a network
system using multimicroprocessors *
Microprocessors and Microsystems Vol.6 no 2 82
- [42]. A.B.KOVALESKI , B.Sc * High speed bus arbiter for
multiprocessors *
IEEE PROC .,Vol 130 , Pt.E no 2 March 83
- [43]. Jean Emmanuel HANNE . * Exécutif temps réel pour les nou-
velles gammes de microprocesseurs *
Minis et Micros no 203 p.37
- [44]. Stan BAKER . * De meilleurs résultats avec la combinaison
16081 / 68000 ?
Minis et Micros no 199 p.36
- [45]. Paul W.BAKER . * The solution of differential equations on
short-word-length computing devices *
IEEE Transactions on Computers , Vol.C-28 ,
no.3 , March 79
- [46]. Farid GUETTACHE . * Définition et réalisation d'un systè-
me multiprocesseur pour la simulation d'un
processus physique continu . Ordonnancement des
tâches . * Thèse de Magister USTHB ALGER Juil.86

A N N E X E A : la machine

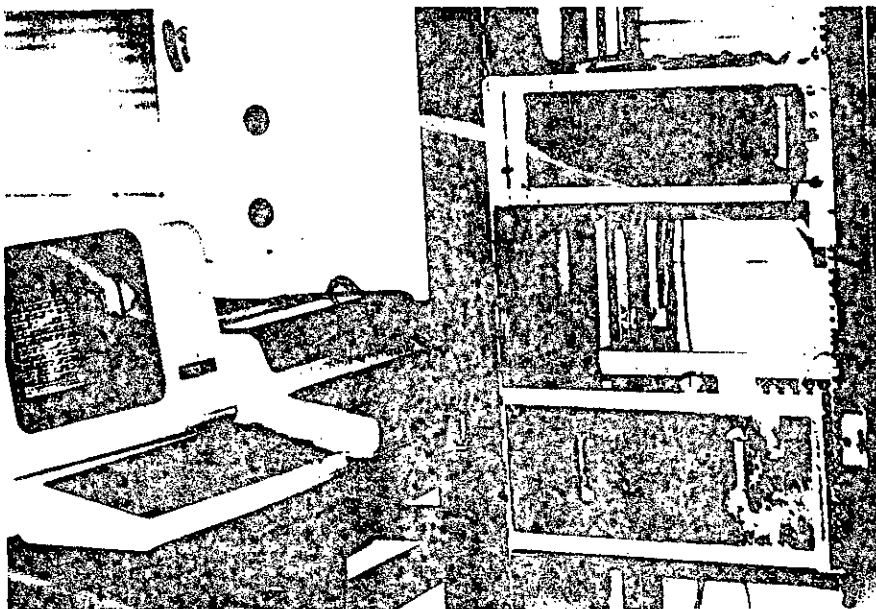


figure A-1 vue générale de la machine.

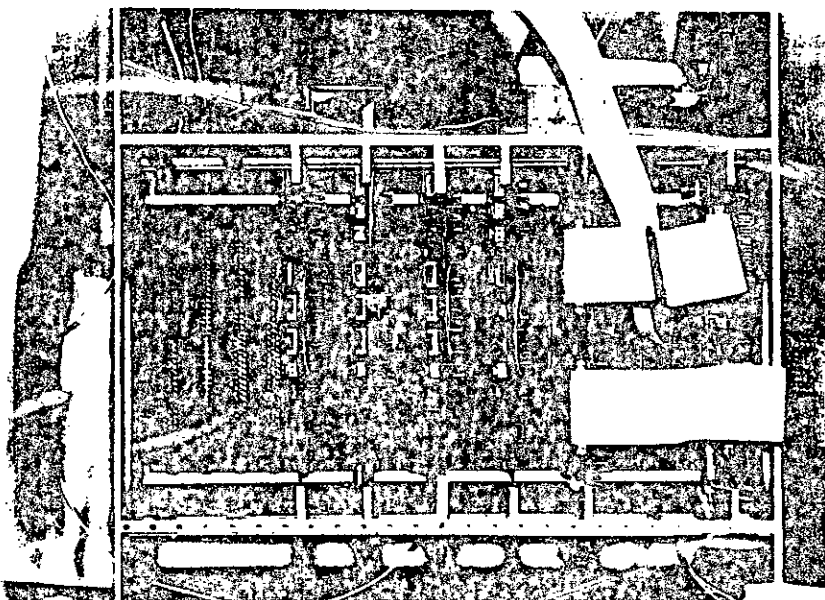


figure A-2 le rack multiprocesseur

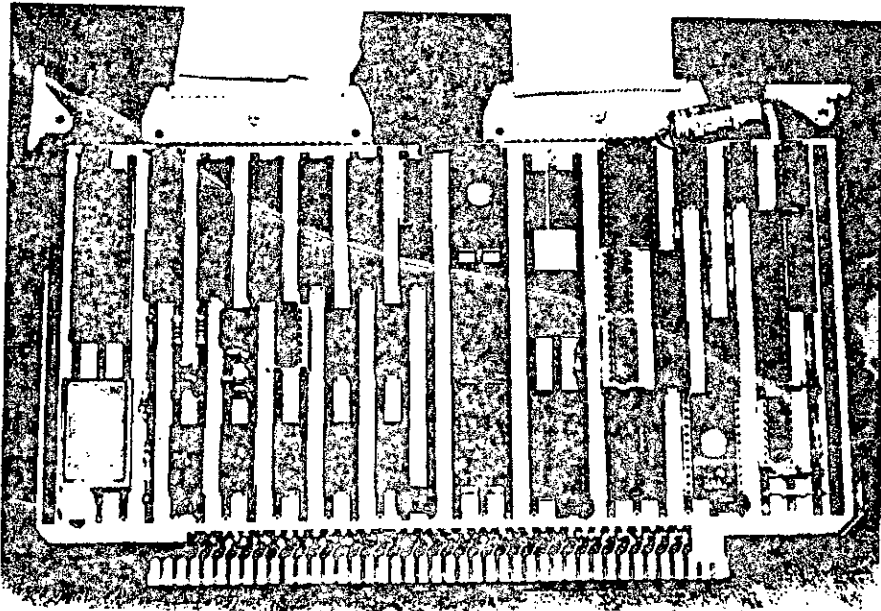


figure A-3 la carte processeur principal

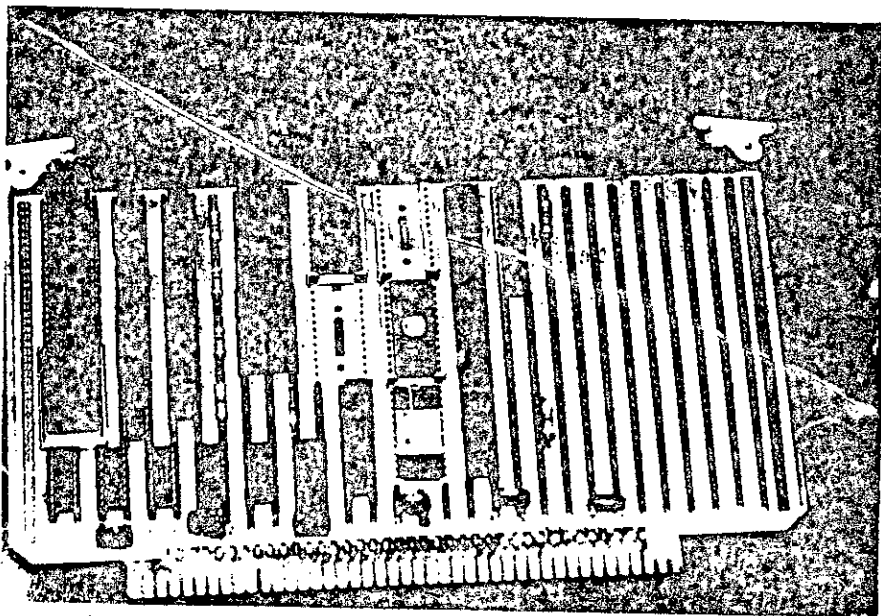


figure A-4 la carte processeur secondaire

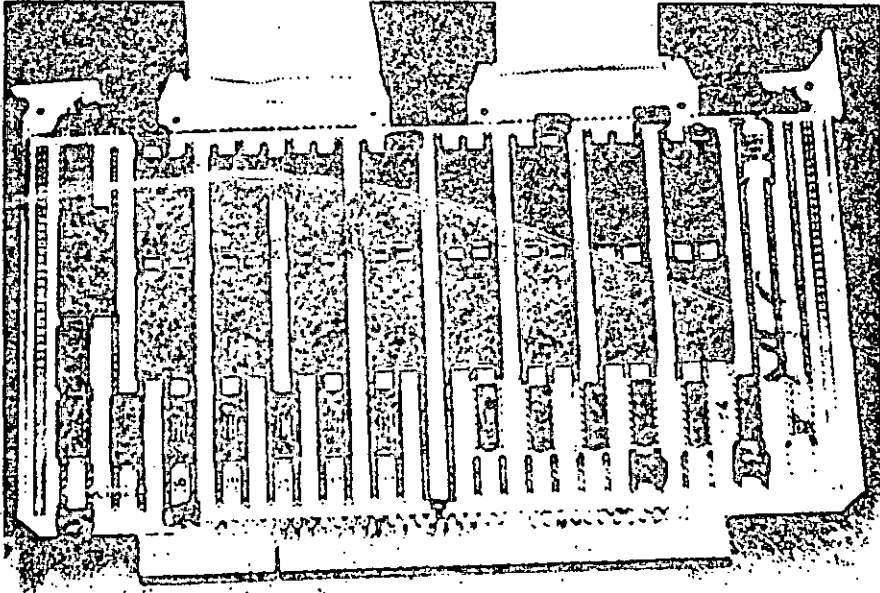


figure A-5 la carte memoire partagee

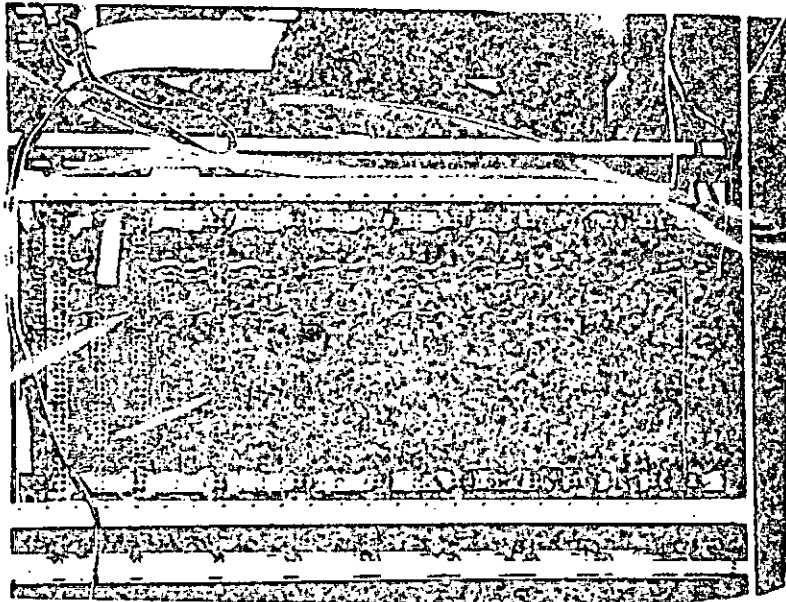


figure A-6 le bus multiprocesseur

A N N E X E B : le jeu de commandes de la
machine

```
..dsIMULATION NUMERIQUE PAR MULTIPROCESSEUR  
c.e.n/c.d.t.a/ 1986  
AS.83.001
```

```
SYSTEM READY
```

```
>equa
```

```
vequ
```

```
.modele du processus a simuler
```

```
dX0/dt = X1
```

```
dX1/dt = -C2*X0 + C1*X2
```

```
dX2/dt = X3
```

```
dX3/dt = -C2*X2 + C1*X0
```

```
  C1 = ka**2/J        C2 = ( ka**2 + mgl )/J
```

```
>equa
```

```
.a : distance du point de fixation du ressort / axe de rotation
```

```
.k : constante de raideur du ressort
```

```
.l : longueur des pendules
```

```
.m : masse des pendules
```

```
.J : moment d'inertie des pendules
```

```
.X0 : elongation du pendule 1
```

```
.X1 : vitesse du pendule 1
```

```
.X2 : elongation du pendule 2
```

```
.X3 : vitesse du pendule 2
```

```
.end
```

```
>simu
```

```
.n=10
```

```
.t0=0
```

```
.t1=1
```

```
.end
```

```
>equa
```

```
.n : nombre de pas
```

```
.t0 : instant initial
```

```
.t1 : instant final
```

```
.end
```

```
>init
```

```
.a=0.2
```

```
.k=4
```

```
.l=0.65
```

```
.m=0.47
```

```
.g=9.8
```

```
ERROR
```

```
.g=9.8
```

```
ERROR
```

```
.y01=0.314
```

```
.y02=0
```

```
.y03=-0.314
```

```
.y04=0
```

```
.end
```

```
>
```


SIN INTEGRATION

```

p3p0p2q0q1q3q2p1p3p0q1p2q3q0q2p1p3p0q1p2q3q0q2p1p3p0q1p2q3q0q2p1p3p0p2
00000.0997 y0= 00000.2896 y1=-00000.4786 y2= 00000.2896 y3=-00000.4786
q1q3q2p1p3p0q1cp2q3q0q2p1p3p0q1p2q3q0q2p1p3p0q1p2q3q0q2p1p3p0p2
00000.1998 y0= 00000.2206 y1=-00000.8834 y2= 00000.2206 y3=-00000.8834
q1q3q2p1p3p0q1cp2q3q0q2p1p3p0q1p2q3q0q2p1p3p0q1p2q3q0q2p1p3p0p2
00000.2996 y0= 00000.1175 y1=-00001.1516 y2= 00000.1175 y3=-00001.1516
q1q3q2p1p3p0q1cp2q3q0q2p1p3p0q1p2q3q0q2p1p3p0q1p2q3q0q2p1p3p0p2
00000.3997 y0=-00000.0032 y1=-00001.2423 y2=-00000.0032 y3=-00001.2423
q1q3q2p1p3p0q1cp2q3q0q2p1p3p0q1p2q3q0q2p1p3p0q1p2q3q0q2p1p3p0p2
00000.5000 y0=-00000.1238 y1=-00001.1412 y2=-00000.1238 y3=-00001.1412
q1q3q2p1p3p0q1cp2q3q0q2p1p3p0q1p2q3q0q2p1p3p0q1p2q3q0q2p1p3p0p2
00000.5997 y0=-00000.2254 y1=-00000.8636 y2=-00000.2254 y3=-00000.8636
q1q3q2p1p3p0q1cp2q3q0q2p1p3p0q1p2q3q0q2p1p3p0q1p2q3q0q2p1p3p0p2
00000.6998 y0=-00000.2922 y1=-00000.4531 y2=-00000.2922 y3=-00000.4531
q1q3q2p1p3p0q1cp2q3q0q2p1p3p0q1p2q3q0q2p1p3p0q1p2q3q0q2p1p3p0p2
00000.7996 y0=-00000.3138 y1= 00000.0273 y2=-00000.3138 y3= 00000.0273
q1q3q2p1p3p0q1cp2q3q0q2p1p3p0q1p2q3q0q2p1p3p0q1p2q3q0q2p1p3p0p2
00000.8997 y0=-00000.2870 y1= 00000.5039 y2=-00000.2870 y3= 00000.5039
q1q3q2p1p3p0q1cp2q3q0q2p0p1p3q0p2q1q3q2p0p1p3p2q0q1q3q2p0p1p3p2
00001.0000 y0=-00000.2157 y1= 00000.9023 y2=-00000.2157 y3= 00000.9023

```

INTEGRATION
RTED;

TEM READY

sn
10

```

00000.0997 y0= 00000.2896 y1=-00000.4786 y2= 00000.2896 y3=-00000.4786
00000.1998 y0= 00000.2206 y1=-00000.8834 y2= 00000.2206 y3=-00000.8834
00000.2996 y0= 00000.1175 y1=-00001.1516 y2= 00000.1175 y3=-00001.1516
00000.3997 y0=-00000.0032 y1=-00001.2423 y2=-00000.0032 y3=-00001.2423
00000.5000 y0=-00000.1238 y1=-00001.1412 y2=-00000.1238 y3=-00001.1412
00000.5997 y0=-00000.2254 y1=-00000.8636 y2=-00000.2254 y3=-00000.8636
00000.6998 y0=-00000.2922 y1=-00000.4531 y2=-00000.2922 y3=-00000.4531
00000.7996 y0=-00000.3138 y1= 00000.0273 y2=-00000.3138 y3= 00000.0273
00000.8997 y0=-00000.2870 y1= 00000.5039 y2=-00000.2870 y3= 00000.5039
00001.0000 y0=-00000.2157 y1= 00000.9023 y2=-00000.2157 y3= 00000.9023

```

d
sg

00000.0997

```

0000 0997
0000 1998
0000 2996
0000 3997
0000 5000
0000 5997
0000 6998
0000 7996
0000 8997
001 0000

```

end

```

>
>
>
>d 0000 00
0001 00
0002 FF
0003 FF
0004 FF
0005 FF
0006 FF
0007 FF
0008 04 66
0008 66
0007 FF
0006 FF
0005 FF
0004 FF
0003 FF
0002 FF
0001 00
0000 00
FFFF E6
FFFE F1
FFFD 32
FFFC E4
FFFB 68
FFFC E4
FFFD 32
FFFE F1 f

```

```

>d 9000q
P=9001 A=b7 B=09 X=9000 C=DE S=A442
>

```

ABORTED;

SYSTEM READY

```

>t
ERROR;n=0?

```

```

>g
ERROR;n=0?

```

```

>hhh
ERROR

```

```

>copn
>copy

```

```

>g
BEGIN INTEGRATIONp1p3p0p2q0q1q3q2p1p3p0q1p2q3q0q2p1p3p0q1p2q3q0q2p1p3p0q
p1p3p2q0q1q3q2p0p1p3p2Fq1q3q0q2p1p3p0p2q1q3q0q2p1p3p0q1p2q3q0q2p0p1p3p2q
p0q1p2q3q0q2p1p3p0p2Pq1q3q0q2p1p3p0q1p2q3q0q2p1p3p0q1p2q3q0q2p1p3p0q1p2q

```

END INTEGRATION

SYSTEM READY

>

ANNEXE C :

TEMPS DE CALCUL DES OPERATIONS EN VIRGULE FLOTTANTE

Nous donnons les temps de calcul des opérations élémentaires en virgule flottante obtenus soit , à l'aide de bibliothèques mathématiques soit , à l'aide du processeur arithmétique AM 9512 d'Advanced Micro Devices .

1. Bibliothèque mathématique de Motorola :

Elle effectue des opérations en virgule flottante sur des opérands de 24 bits .

Addition en virgule flottante :	vitesse typique
32766 + 1	1.20 ms
0.2 + 0.3	0.81 ms
 Soustraction	
32767 - 1	1.32 ms
0.2 - 0.3	0.96 ms
 Multiplication	
32767 x 32767	2.46 ms
1 x 1	1.80 ms
- 32767 x (- 32767)	2.64 ms
 Division	
32767 / 1	2.28 ms
32767 / 32767	2.10 ms

2. Processeur arithmétique AM 9512 (2 Mhz)

	vitesse typique
Addition flottante sur 32 bits (SADD)	220 cycles ou 110 μ s
Soustraction flottante (SSUB)	220 cycles ou 110 μ s
Multiplication flottante (SMUL)	220 cycles ou 110 μ s
Division flottante (SDIV)	240 cycles ou 120 μ s

Une comparaison des temps de calcul entre le processeur AM 9512 et la bibliothèque mathématique d'Intel (Intel FPAL. LIB.) est à établir . En effet cette dernière manipule des données en virgule flottante de même format que celles manipulées par l'AM 9512 .

3. Bibliothèque mathématique d'Intel (2 Mhz)

Les opérandes ont une longueur de 32 bits .

	vitesse typique
Addition (FADD)	2069.6 cycles = 1.03 ms
Soustraction (FSUB)	2259.4 cycles = 1.13 ms
Multiplication (FMUL)	3102 cycles = 1.55 ms
Division (FDIV)	7825.9 cycles = 3.91 ms

4. Exemples traités par 2 et par 3

Des temps de calcul sont donnés pour quelques exemples numériques dans le tableau suivant , en fonction des temps de cycle .

OPERANDE 1		Opérande 2		AM 9512		FPAL.LIB		RATIO
Dec	Hexa	Dec	Hexa	SADD	SSUB	FADD	FSUB	
5	40A00000	6	40A00000	58	89	1430	1734	25 19
5	40A00000	.06	3D75C28F	171	178	2506	2724	15 15
.000012	3749539B	340000	48A60400	475	477	1953	2231	4 5
-1.234	BF9DF3B6	12345	4640E400	284	297	2564	2284	9 8
				SMUL	SDIV	FMUL	FDIV	
5	40A00000	.0006	3A1D4952	234	250	3206	7757	14 31
5	40A00000	.006	3EC49BA6	256	235	3252	7905	13 34
5	40A00000	.06	3D75C28F	198	247	3088	7975	15 32
5	40A00000	60	42700000	200	246	2897	7999	15 32
12345	4640E400	67890	47849900	242	249	3124	7585	13 30
-1.234	BF9DF3B6	12345	4640E400	223	227	3314	7852	15 34

ANNEXE D : Le coprocesseur arithmétique Am 9512 ~

L'Am 9512 permet de fournir au microprocesseur 8 bits MC 6800 de Motorola , un gain de vitesse au minimum égal à 10, dans toutes les opérations de calcul , par rapport à des solutions logicielles .

Le brochage du circuit Am 9512 se présente comme suit :

VSS	1	24	END
VCC	2	23	CLK
$\overline{\text{EACK}}$	3	22	RESET
$\overline{\text{SVACK}}$	4	21	C/D
SVRED	5	20	$\overline{\text{RD}}$
ERR	6	19	$\overline{\text{WR}}$
DO NOT USE	7	18	$\overline{\text{CS}}$
DB0	8	17	PAUSE
DB1	9	16	VDD
DB2	10	15	DB7
DB3	11	14	DB6
DB4	12	13	DB5

D'autres processeurs arithmétiques compatibles 8 bits existent , comme l'AM 9511 qui peut fournir un gain minimum de l'ordre de 20 par rapport aux solutions logicielles . Un problème ayant surgi en dernière minute nous a empêché son utilisation .

D'autres types de processeurs arithmétiques peuvent être utilisés à partir de microprocesseurs en tranches . Ceux-ci sont de technologie bipolaire (rapide) . Les différents éléments d'une unité centrale sont éclatés sur plusieurs circuits et , l'unité arithmétique et logique est découpée en " tranches " qui traite 2 ou 4 bits en parallèle que l'on assemble pour traiter des mots de 4 , 8 , 16 ou 32 bits ou plus . Les circuits

bipolaires dont l'intégration est faible , impose le recours à un tel découpage . Le tableau suivant résume les caractéristiques des principales familles de ce type :

Famille	Intel 3000	AMD 2900	MMI 6700	I.I SBP 400	Motorola 10800
UAL	3002	2901	6700	SBP 400	10800
NB DE BITS	2	4	4	4	4
TECHNOLOGIE	TTL-S	TTL-LS	TTL-LS	I2L	ECL
CYCLE DE BASE	72 ns	85 ns	85 ns	500 ns	40 ns

La famille AMD est celle qui est la plus utilisée . La programmation de tels circuits est assez complexe .