HUITIEME COLLOQUE SUR LE TRAITEMENT DU SIGNAL ET SES APPLICATIONS

NICE du 1er au 5 JUIN 1981

MODULARISATION ET SYNCHRONISATION DES TRAITEMENTS DANS UNE MACHINE MULTIPROCESSEUR
MODULARIZATION AND SYNCHRONISATION OF DATA PROCESSING IN A MULTIPROCESSOR MACHINE

A. DEMEURE ET S. BRAUDEL

THOMSON-C.S.F.-DASM CHEMIN DES TRAVAILS - 06802 CAGNES SUR MER

RESUME

Cet exposé traite de l'organisation des traitements dans une structure multiprocesseur orientée traitement de signal.

Les caractéristiques principales de la machine (déjà exposées au GRETSI 1979) sont résumées en introduction :

- ressources "calcul" (opérateurs, mémoires) partageables par tous les processeurs ,
- reliant ces ressources, bus unique piloté par un "Ordonnanceur"regroupant les unités de contrôle des processeurs,
- configurant la machine, gestionnaire définissant à chaque processeur son rôle dans une chaîne de traitement.

La programmation de cette structure a dû résoudre deux problèmes :

1) Modularisation des traitements : choix du quantum de traitement (processus) exécutable au niveau processeur, sans référence à l'activité des autres processeurs. Ce choix devait :

- exploiter le parallélisme de la machine,
- permettre, à partir d'un répertoire minimum de processus, de constituer un maximum de traitements globaux.

2) Synchronisation : choix d'un mécanisme d'enchaînement des processus assurant la simultanéité optimum des exécutions.

Les solutions exposées permettent de considérer chaque processeur comme spécialisé dans l'exécution d'un certain nombre de macro-instructions opérant sur des tableaux de données.

Le mécanisme d'enchaînement s'apparente beaucoup au principe de "séquencement par les données", l'exécution d'un processus étant subordonnée à la disponibilité de tableau(x) de résultats.

SUMMARY

This paper deals with the organization of data processing in a multiprocessor architecture applied to signal processing.

The most important characteristics of this machine (already presented at GRETSI 1979) are summed up in the first part :

- processing resources (processing units, memory blocks) shared by every control unit (processor)
- linkingthese resources, a mono-bus driven by a scheduler which is connecting all the control units,
- management unit configuring the machine by assigning to every control unit its role in a chain of data processing.

Program analysis of this structure of machine has brought out two problems :

1) Modularization : the choice of the quantum of program (processus) which can be executed by a processor, without considering the activity of others processors. This choice implied the following optimisations :

- taking advantage of the parallelism of the machine,
- allowing the constitution of a maximum number of global data processings with the smallest number of processus.

2) Synchronization : the choice of a chaining procedure which authorizes the best simultaneity in processus executions.

With the strategy adopted here, each processor is specialized in executing a few macroinstructions, processing data arrays.

The chaining mechanism is very similar to the basic principle of data driven computing. The execution of a processus is indeed dependent on the availability of the preceding results.

MODULARISATION ET SYNCHRONISATION DES TRAITEMENTS DANS UNE MACHINE MULTIPROCESSEUR
MODULARIZATION AND SYNCHRONIZATION OF DATA PROCESSING IN A MULTIPROCESSOR MACHINE

INTRODUCTION

Les principes d'organisation matérielle de la machine OPOSSUM ont été présentés au GRETSI 1979. A cette date, l'étude était trop récente pour que les détails fins d'architecture des traitements soient clairement perçus. Les deux années écoulées ont permis de décanter les problèmes liés à cette structure multi-processeur. Actuellement, des programmes opérationnels "tournent" depuis plusieurs mois sur ce type de machine.

L'essentiel de la machine est rappelé ci-dessous mais dans l'optique Utilisateur (au sens informatique). La structure permet de faire collaborer jusqu'à 16 processeurs. Il s'agit de processeurs différenciés. Chacun :

- par le contenu de sa mémoire microprogramme,
- par ses Opérateurs de calcul sur les Données,

se spécialisera dans un (ou plusieurs) type de transformation, par exemple :

- processeur FFT,
- processeur Normalisation,
- processeur Filtrage, etc...

Les processeurs travaillent en simultanéité. Ils se transmettent de proche en proche les Données transformées en partageant des zones communes dans les blocs de Mémoire Données.

La même structure linéaire se retrouve dans chaque processeur :

- Unité de Commande (UC) qui, par lecture des micro-instructions successives de son programme, définit la séquence d'actions régissant une transformation,
- Unité de Calcul d'Adresse (UCA) qui décode les micro-instructions et les traduit en terme d'ordres pour
- les opérateurs et les mémoires qui travaillent au niveau des Données.

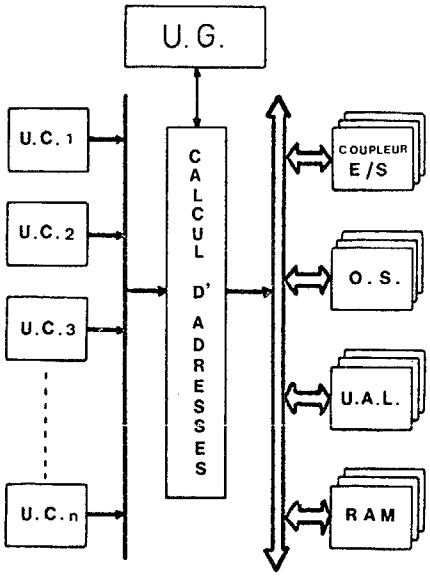

FIGURE 1

Dans cette structure, on trouve des éléments communs à tous les processeurs : l'Unité de Calcul d'Adresse se multiplexe entre les différentes Unités de Commandes ; le bus de communication entre Opérateurs et Mémoires relie en fait tous les moyens mis en oeuvre à ce niveau "Données" sans préjuger de l'affectation d'un moyen à un processeur.

A ce stade de la description, on peut se contenter de détailler les points particuliers de l'organisation qui ont permis :

- a priori, la cohabitation matérielle des processeurs,
- a posteriori, la coopération des processeurs dans la constitution des chaînes de traitement.

Partage des ressources :

Les processeurs sont amenés à partager des moyens au niveau "Données" : zone d'échange dans les mémoires, éventuellement Opérateurs. Cette mise en commun de ressources nécessite un mécanisme de partage, une ressource ne pouvant servir simultanément plusieurs processeurs L'allocation dans le temps de ressources partagées nécessite des prises de décision rapides (les demandeurs attendent tant que la décision n'est pas prise). L'approche purement programmée, par sémaphore, étant trop pénalisante, la solution adoptée (figure 2) est hybride :

- demande, occupation, libération des ressources explicitées dans les programmes des Unités de Commande,
- comparaison des demandes aux disponibilités

effectives, allocation au demandeur le plus prioritaire, par dispositif câblé.

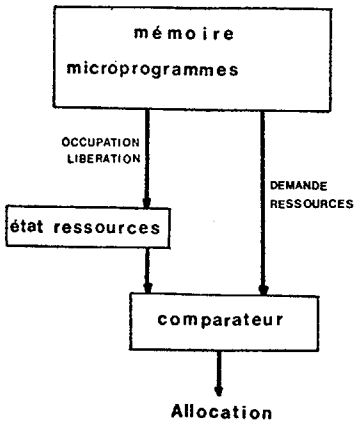

FIGURE 2

Dans le langage de microprogrammation adoptée, la gestion d'une ressource s'exprime par les phrases :

- WITH R(X) ON % Attendre que la ressource R(X) soit libre
- SET R(X) OFF % occuper la ressource R(X)
- SET R(X) ON % libérer la ressource R(X)

Temporisation :

Dans le déroulement de son microprogramme, une unité de commande doit respecter des intervalles de temps correspondant aux temps de réponse des Opérateurs. Plutôt que de recourir à des comptes-rendus câblés : opérateurs vers UC, qui préjugeraient de l'affectation des moyens de calcul aux processeurs, il est préférable de donner à chaque UC la notion de temps écoulé.

Chaque UC dispose d'un temporisateur qui recopie un champ particulier de la micro-instruction. Cette "mise en sommeil" programmable s'écrit dans le microprogramme par :

- WAIT (n) % l'UC attend pendant n cycles machine (1 cycle : 0,1 µs).

Cette "mise en sommeil" d'une UC est importante pour la cohabitation des processeurs. L'activité d'un processeur dont l'UC est en sommeil continue aux niveaux de ses opérateurs ; par contre, des UC moins prioritaires peuvent accéder aux organes multiplexés : UCA, bus Données.

A ce stade de la description, l'utilisateur dispose de processeurs ayant la capacité :

- d'effectuer en fonction des microprogrammes et des opérateurs implantés, un certain jeu de transformations (processus),
- de partager des ressources,
- de mettre leur UC temporairement en sommeil.

Il reste à introduire :

1) Le paramétrage des processus, les microprogrammes étant écrits sous la forme la plus générale. Exemple : le processus FFT requiert pour arguments : le nombre de points, l'adresse du tableau Opérande, etc... L'activité calcul découlant de l'exécution d'un processus dûment argumenté constituera la Tâche.

2) Les conditions d'exécution liées à des événements externes (acquisition temps réels) ou internes (achèvements d'autres tâches).

L'ensemble des informations 1) et 2) constitue le niveau Programme. A la différence d'un processeur traditionnel, le travail y est quantifié non en instructions mais en tâches ; par ailleurs, leur ordre de rangement ne préjuge absolument pas de l'ordre d'exécution dans le temps.

Le support du niveau Programme est pris sur une zone réservée des mémoires vives de l'UCA : mémoire X. Le chargement du Programme (figure 3) s'effectue en dynamique par un microprocesseur 6800 qui constitue l'Unité de Gestion (UG). Celle-ci a la capacité de générer du Programme à partir de commandes macroscopiques : changement de mode, reconfiguration, etc...

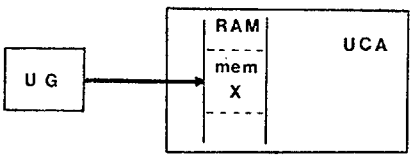

FIGURE 3

REPRESENTATION GRAPHIQUE DE L'ENCHAINEMENT DES TACHES.

Pour exprimer les relations qui conditionnent l'exécutabilité d'une tâche à l'exécution d'autres tâches, on a développé un formalisme qui rappelle les réseaux de Pétri.

Ce formalisme est décrit ci-dessous avec son vocabulaire propre. Les entités évoquant une analogie avec les réseaux de Pétri seront mentionnées par RdP = ...

L'enchaînement des tâches est représenté par un graphe orienté. Les sommets du graphe traduisent :

- soit des tâches symbolisées par un rectangle (RdP = Transitions),
- soit des variables de déclenchement, V.D., symbolisées par un rond (RdP = Place).

Les arcs ne relient que des sommets de "races" différentes.

Le graphe est renseigné (exemple figure 4) en portant :

- dans les rectangles, le sigle de la tâche, sa durée, le processeur exécutant, éventuellement les ressources Calcul utilisées,
- à côté des ronds, le sigle de la variable de déclenchement.

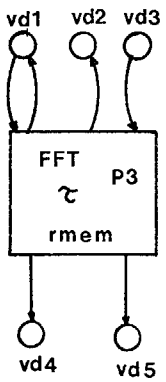

FIGURE 4

Les V.D. sont du type binaire. Elles prennent la valeur 1 ou 0. L'état global de ces variables (RdP = Marquage) évolue suivant des règles spécifiées par le graphe.

Les arcs issus "par le haut" des rectangles mettent les V.D. à 0. Ceux issus "par le bas" mettent les V.D. à 1.

Les arcs issus des sommets "ronds" sont sans influence directe sur l'état des V.D.. Ils définissent les conditions d'exécutabilité des tâches : une tâche est exécutable quand toutes les V.D. émettant un arc vers cette tâche sont à 1.

Ce graphe a, en fait, sa traduction directe en schéma logique à base de bascules, portes ET, monostables. L'état des bascules coïncide avec celui des V.D. ; reliées aux entrées de ET, elles conditionnent sur la transition 0 → 1 de la sortie des ET, le déclenchement des monostables réglés sur la durée des tâches, la RAZ de certaines bascules. La transition différée en sortie du monostable met à 1 d'autres bascules.

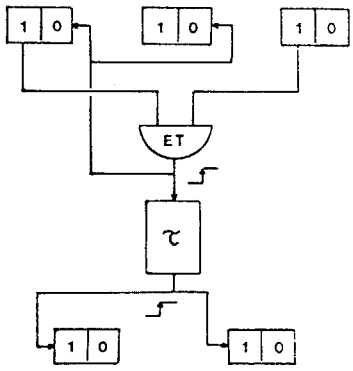

FIGURE 5

Historiquement, le graphe découle des interconnexions de cet automate logique et non pas d'une interprétation des réseaux de Pétri.

L'évolution temporelle du marquage est partiellement explicitée par le graphe. Localement au niveau Tâche, on voit que si à l'instant T_o, la tâche démarre, au même instant, les V.D. atteintes par des arcs issus par le haut du rectangle correspondant, sont mises à 0 ; à T_o + durée Tâche, les V.D. atteintes par le bas du rectangle seront mises à 1.

Le graphe comporte toutes les informations pour :

- faire vivre sur simulateur l'automate correspondant,

- alimenter un compilateur qui sort en final les programmes implantés dans la machine.

MECANISME D'ENCHAINEMENT DES TACHES AU NIVEAU PROCESSEUR

Le choix du mécanisme d'enchaînement devait satisfaire deux buts :

1) Ne pas mobiliser un processeur pour cette seule fonction Moniteur. La notion de processeur Maître et processeurs Esclaves implique une dissymétrie dans la structure matérielle qui n'existe pas dans la machine : la fonction Moniteur doit être répartie. Au niveau microprogramme, les processeurs ont strictement les mêmes procédures.

2) Ne pas s'éloigner de la vocation Traitement du Signal. La charge Moniteur doit rester négligeable devant la charge Calcul. En particulier, un processeur oisif ne doit pas dériver une part de la puissance de l'U.C.A. dans sa quête d'une activité.

Etats d'un processeur :

On a considéré qu'un processeur avait trois états possibles :

- ACTIF : le processeur accomplit une tâche.

- RECEPTIF : le processeur est oisif. Il est à l'écoute d'un événement Fin de Tâche.

- CREATIF : le processeur recherche parmi les tâches qui lui incombent si l'une est exécutable.

L'évolution entre ces trois états est donnée figure 6.

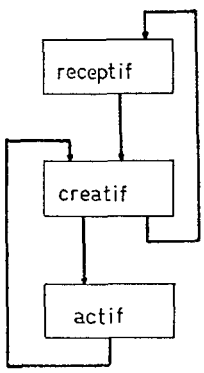

FIGURE 6

1) Passage en Actif : le processeur était précédemment en Créatif, où il a trouvé une tâche exécutable.

2) Passage en Réceptif : comme ci-dessus mais recherche négative.

3) Passage en Créatif, deux cas :

- précédemment en Actif, le processeur va rechercher une tâche rendue exécutable soit par la tâche qu'il vient d'achever, soit par des tâches terminées par d'autres processeurs pendant qu'il était occupé en Actif.

- précédemment en Réceptif, le processeur a été averti qu'une tâche sensibilisant l'exécution d'une de ses propres tâches s'est terminée.

Dans l'état Réceptif ou Créatif, le processeur n'exécute exclusivement que des procédures Moniteur. Dans l'état Actif, les procédures Moniteur correspondent aux transitoires Entrée en Actif, Sortie d'Actif qui encadrent l'exécution proprement dite de la tâche.

Avant d'examiner ces procédures, on décrit la traduction du graphe d'enchaînement en "instructions" machine.

Tables Moniteur :

Le graphe est générateur de deux familles de tables ; de chaque tâche résultent :

- une table en entrée : liste des variables de déclenchement conditionnant la tâche puis nom de la tâche.

- une table en sortie : liste des variables de déclenchement mises à zéro par la tâche, puis liste des V.D. mises à un (en pratique, ces deux listes sont séparées par les arguments calcul de la tâche) ; et enfin, liste des processeurs sensibles à l'événement fin de cette tâche.

En final, on regroupe pour chaque processeur, les tables en entrée des tâches qui lui incombent. Ce regroupement constitue la table de création du processeur.

L'ensemble ci-après (figure 7) montre un graphe d'enchaînement des tâches et ses tables associées.

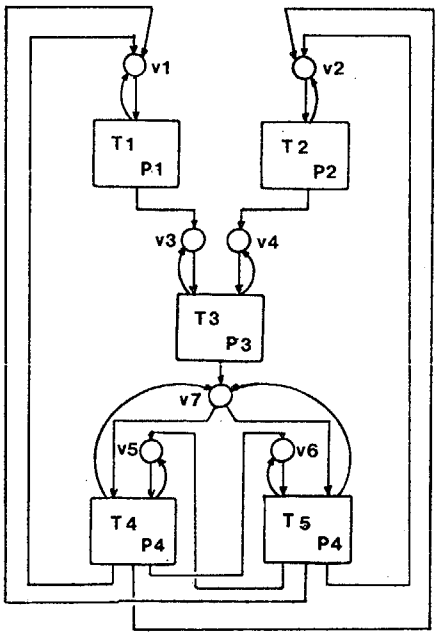

FIGURE 7

Tables en entrée (création) :

P1 : V1
T1
P2 : V2
T2
P3 : V3
V4
T3
P4 : V5
V7
T4
V6
V7
T5

Tables en sorties :

T1 : v1
arguments
V3
P3
T2 : v2
arguments
V4
P3
T3 : V3
V4
arguments
V7
T4 : V5
V7
arguments
V1
V2
V6
T5 : V6
V7
arguments
V1
V2
V5

Le déroulement temporel de l'enchaînemnt des tâches est récurrent avec alternance des tâches T4 et T5 une récurrence sur 2, ce qui donne le diagramme des temps suivant :

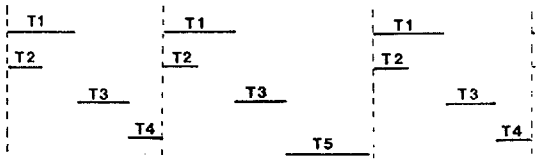

Figure 8

Procédures moniteur.

Pour l'état Créatif et l'état Actif, les procédures Moniteur reviennent pratiquement à exécuter en logique microprogrammée, les mécanismes logiques qui font évoluer les entités du graphe d'enchaînement :

- dans l'état Créatif, recherche d'une fonction ET dont les entrées (variables de déclenchement) sont toutes à 1.

- en Entrée ou Sortie Actif, modification de variables de déclenchement (RAZ en Entrée, RAU en Sortie).

L'état Réceptif n'est, lui, lié qu'à l'aspect multiprocesseur. Ces procédures mettent en oeuvre les dispositifs de Partage des Ressources, et de Temporisation décrits dans l'introduction.

Ressource V.D. : elle désigne globalement les variables de déclenchement. Son rôle est de n'autoriser à un instant donné, qu'un seul processeur à consulter (en Créatif) ou à modifier (en Actif) l'état de l'automate équivalent au graphe. Cet accès non partageable équivaut à rendre instantanées consultations ou modifications. On évite ainsi les aléas propres aux logiques asynchrones.

Ressource Réveil : elle permet à un processeur sortant de l'état Actif de signaler aux processeurs Réceptifs l'événement Fin de Tâche. L'utilisation du mécanisme "Partage de Ressource" garantit ici que les processeurs Réceptifs, bloqués dans l'attente du Réveil, laissent toute la disponibilité de l'U.C.A. aux processeurs Actifs.

Un processeur sortant de l'état Actif active la ressource puis arme une temporisation permettant aux seuls processeurs se trouvant dans l'état réceptif d'avoir accès à l'U.C.A. et de passer à l'état créatif s'ils sont sollicités ou bien de rester dans l'état réceptif. A la fin de la temporisation, la ressource retombe et les processeurs en état actif ont de nouveau accès à l'U.C.A.

Les organigrammes correspondant aux trois états d'un processeur sont donnés en figures 9, 10 et 11.

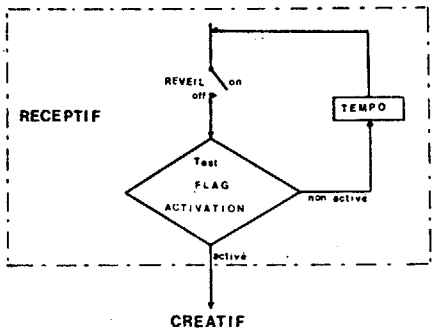

FIGURE 9

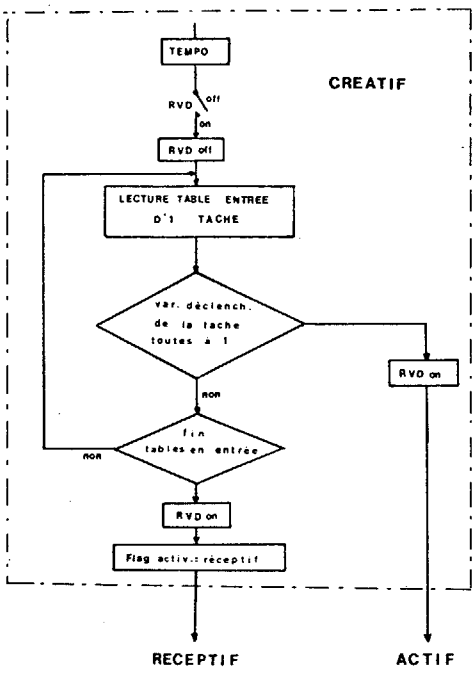

FIGURE 10

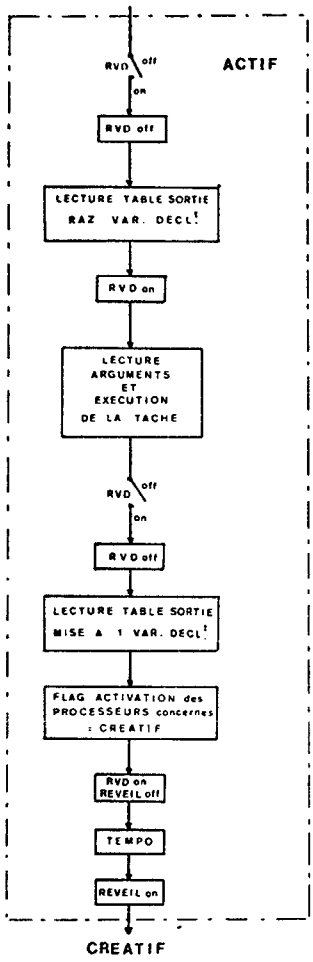

FIGURE 11

Simulateur d'enchaînement des tâches.

Pour vérifier que la description obtenue par le graphe est bien conforme aux spécifications du traitement représenté, on utilise un simulateur qui joue le rôle du moniteur et permet de générer le déroulement temporel de l'enchaînement des tâches. Cet outil permet de vérifier en particulier que les contraintes temps réel sont tenues dans la machine. Il permet de faire ressortir la simultanéité entre les tâches et éventuellement les conflits pour l'occupation d'un opérateur.

En entrée du simulateur, le graphe est décrit à l'aide d'un langage spécifique qui permet de préciser pour chaque tâche :

- le processeur concerné,
- la durée,
- les variables qui permettent de la déclencher,
- les variables à mettre à 0 quand elle démarre,
- les variables à mettre à 1 quand elle se termine ainsi que les processeurs à activer,
- les opérateurs qu'elle utilise.

Cette description permet par ailleurs, la génération automatique des tables Moniteur (tables en entrée tables en sortie).

On peut aussi cadencer les traitements avec une ou plusieurs horloges programmables qui simulent les synchronisations externes de la machine.

Le simulateur est lancé pour une durée déterminée et permet l'observation au cours du temps de plusieurs types d'informations :

- Pour chaque processeur, quelle est la tâche qu'il exécute à chaque instant (voir figure 12).
- Pour chaque opérateur implanté sur le Bus, quelle est la tâche qui l'utilise à chaque instant.
- Le taux d'occupation global de chaque opérateur pendant la durée de la simulation (voir figure 13).

Conclusion.

En ce qui concerne la description d'une chaîne de traitement, la représentation par graphe s'avère tout à fait adaptée aux problèmes posés par une machine multiprocesseur en temps réel : c'est une méthode simple d'analyse et de validation de traitements parallèles à l'aide du simulateur.

De plus, le moniteur d'enchaînement des tâches décrit ici, s'intègre très bien dans la machine et convient parfaitement au type de calculs qu'on rencontre en traitement du signal :

- en effet, les procédures moniteur nécessitent un nombre d'instructions relativement faible (moins de 40),
- au point de vue charge temporelle, le moniteur occupe l'unité de calcul d'adresse de la machine entre 1 et 5 millièmes du temps.

Cependant, la méthodologie présentée dans cet article ne doit pas être considérée comme ayant atteint un état de développement définitif.

Il serait souhaitable dans un premier temps, de développer un outil complémentaire pour éviter l'étape de traduction du graphe dans un langage adapté au simulateur. On pourrait, de plus, envisager un outil de CAO pour aider à la génération du graphe, qui permette d'optimiser le nombre et l'utilisation des opérateurs et des mémoires.

Bibliographie :

- Systèmes de commande en temps réel :

Robert VALETTE - Marc COURVOISIER (SCM)

- Structure programmable pour l'exécution simultanée des processus rencontrés en traitement du signal : A. DEMEURE (G.R.E. T.S.I. 1979).

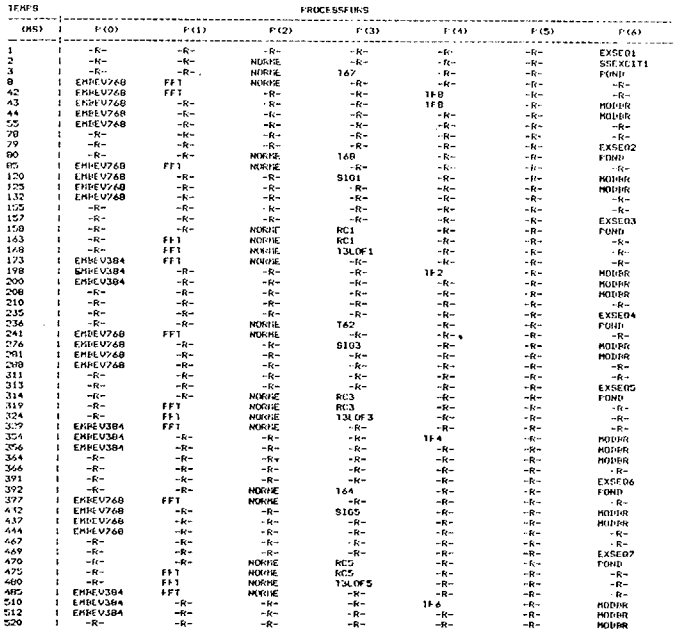

FIGURE 12

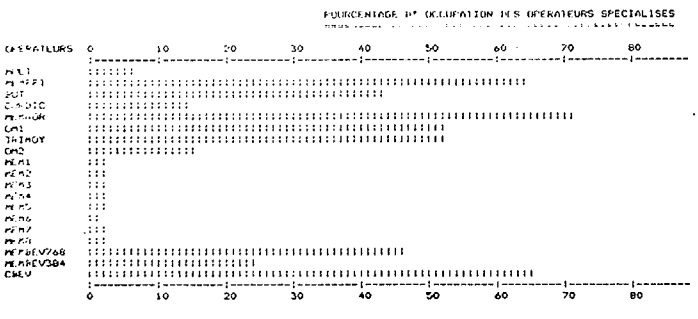

FIGURE 13