OPERATEURS DE CALCUL PARALLELE BASES SUR LES RESEAUX DE PETRI A FLUX DE DONNEES

B. BARBAGELATA - P.ABELLARD

LABORATOIRE D'AUTOMATIQUE ET D'INFORMATIQUE APPLIQUEES
UNIVERSITE DE TOULON,Chateau Saint Michel,83130 LA GARDE

RESUME

Intrinsèquement conçue pour la résolution de problèmes de calcul parallèle, et issue de la décomposition d'algorithmes concurentiels qui modélisent un processus, l'architecture à flux de données se caractérise par une gestion asynchrone d'un ensemble de tâches/opérations simples. Ces opérations réalisées par des processeurs élémentaires sont synchronisées par les flux de données incidents minimisant ainsi les contraintes matérielles inhérentes à la concurrence.

Le but de cet article est d'abord de présenter un modèle de représentation d'une telle architecture : les Réseaux de PETRI à flux de données. Puis une application est proposée pour la résolution des systèmes d'équations linéaires. On montre sur un exemple emprunté à la robotique comment utiliser ce modèle afin de réaliser un coprocesseur de calcul pour l'inversion de la matrice jacobienne d'un robot, en réduisant le temps de calcul par rapport aux méthodes classiques. Enfin l'implémentation des résultats théoriques sur un processeur à flots de données est exposée.

SUMMARY :

Specifically designed for the solving of parallel computing problems and derived from the decomposition of concurrent algorithms modelling a process, the data flow architecture is characterized by an asynchronous management of a set of a simple operations. These operations carried out by elementary processors are synchronized by incidental data flow in order to reduce the material constraints inherent to competition.

The aim of this paper, is first to show a model of representation of such an architecture the data flow Petri nets. Then an application of those nets is proposed for the solving of linear systems of equations. We explain with an example in robotics how to use this model in order to build up a computation coprocessor for the Jacobian matrix inversion of a manipulator arm. This coprocessor allows the competition time to be reduced with respect to traditionnal software methods. At last, the implementation of theoritecal results on a data flow processor is discussed.

1. INTRODUCTION

A l'heure actuelle, des problèmes aussi différents que le filtrage numérique ou la commande dynamique de robots voient leurs applications indusrielles parfois retardées à cause de la grande quantité d'opérations à réaliser sur des systèmes de calculs rapides à faible coût.

La résolution complète de ces problèmes passe par l'obtention d'algorithmes efficaces, de modélisations performantes et la réalisation d'opérateurs de calculs parallèles rapides de type multiprocesseurs.

Actuellement leurs principales limitations sont :
- Une concurrence limitée : l'utilisation de plusieurs processeurs de type Von Neuman, caractérisés par un mode de fonctionnement séquentiel et un partage des zones mémoires, diminue la concurrence entre les différentes opérations du programme et aini le nombre de tâches exécutées simultanément.
- Système de contrôle complexe : pour gérer les communications processeurs-mémoires et processeurs-processeurs ou pour résoudre les conflits d'accès aux ressources, il faut un système de contrôle qui devient rapidement complexe avec l'importance des opérations. Des phénomènes d'étranglement et de saturation de bus compliquent ce contrôle et diminuent son efficacité.
- Difficultés de programmation : dans les machines multiprocesseurs classiques, un problème important est d'éviter les interférences entre les instructions du programme.

Dans une architecture à flux de données, ces trois problèmes n'existent pas. Cette architecture est structurellement différente des machines multiprocesseurs traditionnelles et elle n'implique pas de contraintes matérielles supplémentaires sur l'exécution des programmes parallèles.

Parmi ces modèles de représentation des calculs parallèles, nous avons choisi une extension des réseaux de Petri ordinaires: Les réseaux de Petri à Flux de Données.

2. DESCRIPTION QUALITATIVE DU MODELE :

2.1. Opération :

Une opération est réalisée avec un opérateur et un ensemble de variables que l'on représente par le réseau de Petri de la figure 1.

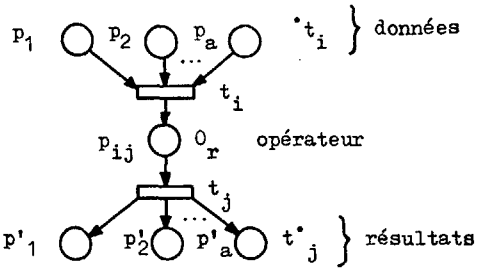


Figure 1 : **Réseau de Petri à flux de données.**

- t_i et t_j sont appelées respectivement transitions d'entrée et de sortie de l'opérateur Or associé à la place p_{ij}.
- Les places

$^\circ t_i=\{p_1,p_2,....,p_a\}$ et $t^\circ_j=\{p'_1,p'_2,....,p'_a\}$ représentent respectivement les données nécessaires à l'opérateur Or et les résultats obtenus.

2.2.Marquage :

Un marque déposée dans une place "variable" signifie que la valeur de la variable est écrite dans la mémoire. Une marque déposée dans une place opérateur signifie que l'opérateur est activé.

On suppose qu'une place peut contenir au plus une marque, donc les réseaux sont saufs.

3. DEFINITIONS MATHEMATIQUES :

3.1. Réseau de Petri à flux de données :

Un réseau de Petri à flux de données RdPFD est un 7 uplet

$$<R,\varphi,\zeta,\psi,\chi,o,c>$$

dans lequel :
- R est un réseau de Petri à places biparties conformes

$<P,T,\alpha,\beta>$ tel que P est un ensemble de places biparties P_v pour les variables et P_o pour les opérateurs, T est l'ensemble des transitions du réseau, et α,β sont les relations d'incidence avant et arrière.

$p_v \cup p_o = p$ et $p_v \cap p_o = \varnothing$

si $^\circ t_i \subset p_o \rightarrow t_i^\circ \subset p_v$ et si $^\circ t_i \subset p_v \rightarrow t_i^\circ \subset p_o$

$\forall_p \in p_v \rightarrow$ CARD $(^\circ p) \leqslant 1$ et CARD $(p^\circ) \leqslant 1$

$\forall_p \in p_o \rightarrow$ CARD $(^\circ p) \geqslant 1$ et CARD $(p^\circ) \geqslant 1$

$\forall_{pi} \in p_o, p_j \in p_o : {}^\circ p_i \cap {}^\circ p_j = \varnothing$ et $p^\circ_i \cap p^\circ_j = \varnothing$

- φ est une application surjective

$\varphi / p_v : p \rightarrow x$, les places p_v prennent leurs valeurs dans l'ensemble X.

$\varphi / p_o : p \rightarrow o$, les places p_o prennent leurs définitions dans l'ensemble O.

telle que $\forall_{pj} \in p_o, p_i \in p_o$ et $\varphi(p_i) = \varphi(p_j)$

pour $i \neq j \Rightarrow \forall_{tl} \in {}^\circ p_i,\ t_k \in {}^\circ p_j$

alors $\{\varphi(^\circ t_l)\} \neq \{\varphi(^\circ t_k)\}$

- ζ est une application injective $\zeta : X \rightarrow M = \{ME_1, \ldots, ME_u\}$

telle que $\forall_p \in p_v$, $ME \in M \Rightarrow ME = \zeta(\varphi.(p))$

M est appelé ensemble de zones mémoires

- ψ est une application surjective $\psi : T \rightarrow C$

- $X = \{x_1, x_2, \ldots, x_u\}$ est un ensemble de variables (réelles, entières, logiques) prenant leurs valeurs respectivement dans les domaines D_1 et $D_2, \ldots, D_u$.

- $O = \{O_1, O_2, \ldots, O_u\}$ est un ensemble fini d'opérateurs définis comme des applications internes de $D_1 * D_2 * \quad D_u$

- $c = \{c_1, c_2, \ldots, c_u\}$ est un ensemble de conditions (prédicats) sur les variables de X.

3.2. Graphe de marquage :

Soit R un Réseau de Petri et M_0 un marquage initial tel que la classe des marquages conséquents soit finie. Le graphe de marquage des marquages conséquents est le graphe orienté {S,l} dans lequel $S = \{\vec{M}_0\}$ est l'ensemble des sommets et L l'ensemble des arcs tels que :

$M_i \in \vec{M}_0$ et $M_j \in \vec{M}_0 \exists\ t_i$ telle que t_i soit une transition de R par laquelle $M_i \xrightarrow{t_i} M_j$. Les arcs du graphe sont étiquetés par les transitions du réseau.
Les figures 2 et 3 montrent un exemple de RdPFD correct dans lequel toutes les opérations sont exécutées et tous les résultats désirés obtenus, associés au calcul d'un terme de produit matriciel

$$c_{11} = a_{11} \times b_{11} + a_{12} \times b_{21}$$

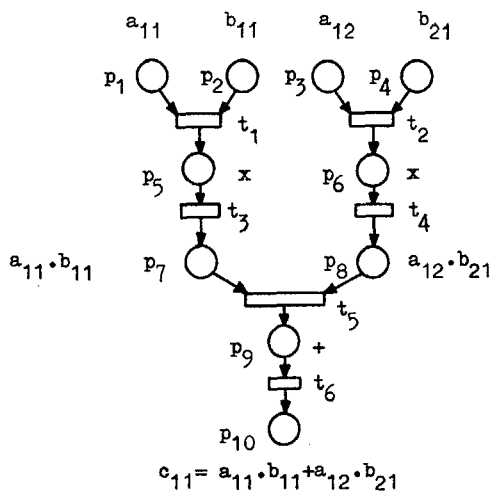


Figure 2: **RdPFD du calcul** $c_{11}=a_{11}.b_{11}+a_{12}.b_{21}$

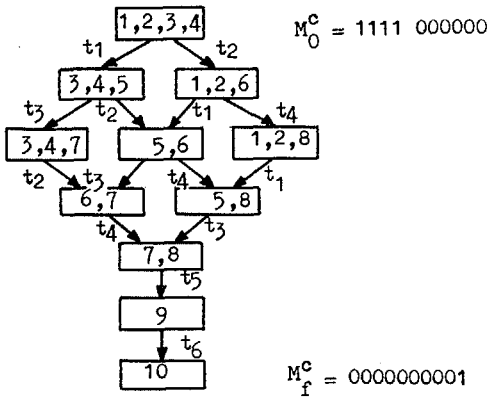


Figure 3: **Graphe des marquages du calcul** $c_{11}=a_{11}.b_{11}+a_{12}.b_{21}$

4. APPLICATION :

4.1 Résolution de [K] = [A] * [X] par décomposition LU

Le problème de la résolution des systèmes d'équations linéaires [K] = [A] * [x] dans lequel [A] et [K] sont des matrices connues par le calcul de $[A]^{-1}$ devient très rapidement contraignant dès que l'ordre de [A] dépasse 3. La décomposition LU peut alors être avantageusement utilisée pour cette résolution. Pour l'obtention de matrices triangulaires inférieures et supérieures L et U uniques la décomposition [A] = [L] x [U] nécessite le calcul de :

pour i = 1,2,.....

$u_{ik} = a_{ih} - \sum_{j=1}^{i-1} l_{ij} \cdot u_{jk}$ pour $i \leqslant k \leqslant n$;

$l_{ki} = (a_{ki} - \sum_{j=1}^{i-1} l_{kj} \cdot u_{ji}) / u_{ii}$ pour $i \leqslant k \leqslant n$;

$l_{ii} = 1$

4.2. Application à la commande cinématique d'un manipulateur articulé

4.2.1. Exemple :

La figure 4 montre un manipulateur articulé à deux rotations et une translation et sa coordination à partir du modèle cinématique. Les variations $\Delta\Theta$ des variables articulaires satisfont la relation

$\Delta\Theta = J^{-1} . \Delta X$ où J est le jacobien

En posant $\cos\Theta_1 = C_1$; $\sin\Theta_1 = S_1$ etc.....
La matrice jacobienne s'écrit :

$$J = \begin{vmatrix} -C_1C_2(L_2+T) & S_1S_2(L_2+T) & S_1C_2 \\ -S_1C_2(L_2+T) & -C_1C_2(L_2+T) & C_1C_2 \\ 0 & C_2(L_2+T) & S_2 \end{vmatrix}$$

4.2.2. Décomposition LU

Les termes à calculer sont alors :

$u_{11} = J_{11}$; $u_{12} = J_{12}$; $u_{13} = J_{13}$

$l_{21} = J_{21}/J_{11}$; $l_{31} = J_{31}/J_{11}$

$u_{22} = J_{22} - l_{21} . J_{12}$

$u_{23} = J_{23} - l_{21} . J_{13}$

$l_{32} = (J_{32} - l_{31} . J_{12}) / u_{22}$

$u_{33} = J_{33} - l_{31} . J_{13} - l_{32} . u_{23}$

Pour cela nous dessinons le RdPFD en associant un opérateur à chaque opération à réaliser, en les dessinant sur des branches parallèles lorsque les calculs peuvent être simultanés, et en propageant les données entre opérateurs suivant les calculs à réaliser.

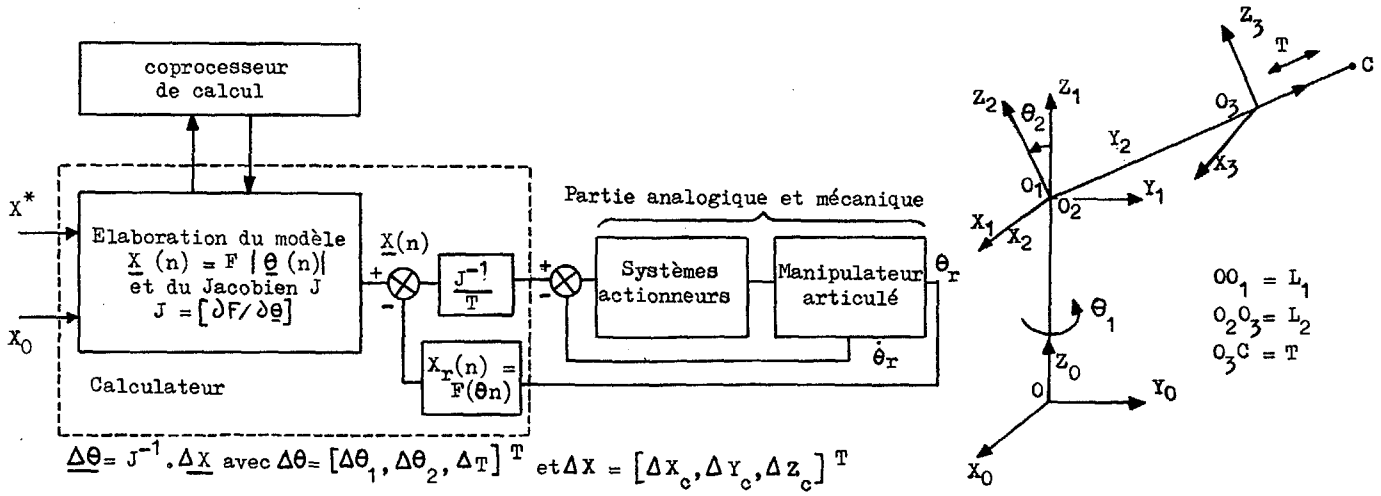


Figure 4: **Exemple d'un manipulateur articulé et de sa coordination à partir dún modèle cinématique.**

La figure 5 montre le RdPFD de la décomposition

[J] = [L] x [U]

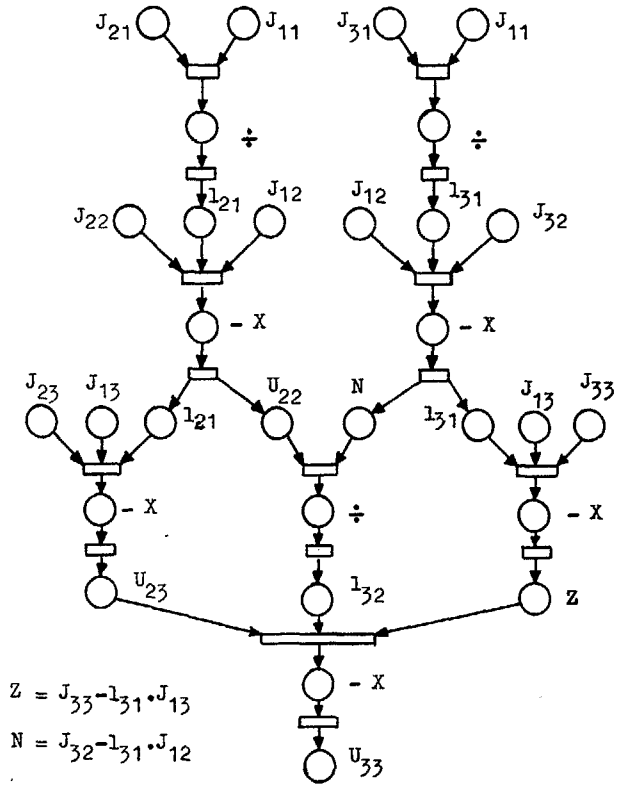


Figure 5 : **RdPFD de la décomposition** [J] = [L] x [U]

Ce réseau met en évidence le parallélisme des calculs et la nature de la tâche réalisée par chaque opérateur. On voit par exemple que pour faire circuler le flot de données de l'entrée du réseau vers la sortie, les opérateurs ont parfois à réaliser seulement un calcul, et parfois en plus à propager des données.

4.2.3 Résolution de $\Delta X = J . \Delta\Theta$

Considérons l'ensemble des variables auxiliaires données par la matrice unicolonne [Y] = [U]. [ΔΘ]

Nous obtenons alors :

[J] . [ΔΘ] = [L].[U].[ΔΘ]. = [L].[Y] = [ΔX]

Cette dernière équation nous donne la série de relations suivantes

$\Delta X_1 = Y_1$

$\Delta X_2 = l_{21} . Y_1 + Y_2$

$\Delta X_3 = l_{31} . Y_1 + l_{32} . Y_2 + Y_3$

$\Delta X_n = l_{n1} . Y_1 + l_{n2} . Y_2 + ... Y_n$

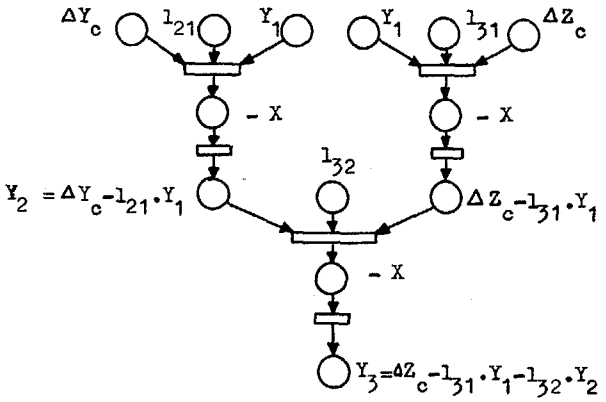


Figure 6 : **RdPFD du calcul** [ΔX]=[L].[Y]

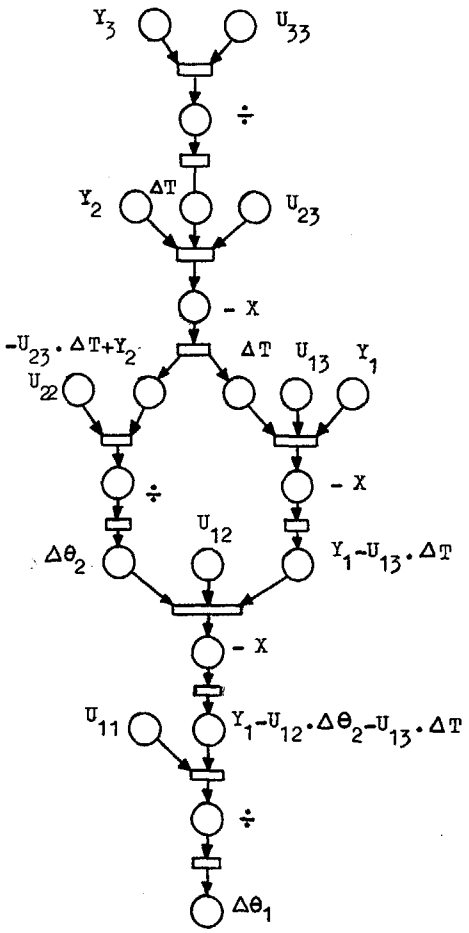


Figure 7 :**RdPFD du calcul** [Y]= [U].[ΔΘ]

Ainsi pour le manipulateur articulé considéré nous obtenons :

$Y_1 = \Delta Xc$

$Y_2 = \Delta Yc - l_{21} Y_1$

$Y_3 = \Delta Zc - l_{31} \cdot Y_1 - l_{32} \cdot Y_2$

En procédant de la même façon que précédemment nous obtenons alors le réseau de la figure 6

Maintenant connaissant [Y], l'équation $[U].[\Delta\Theta] = [Y]$ donne :

$U_{nn} \cdot \Delta\Theta_n = Y_n$

$U_{n-1,n} \cdot \Delta\Theta_n + U_{n-1,n-1} \cdot \Delta\Theta_{n-1} = Y_{n-1}$

$U_{1n} \cdot \Delta\Theta_n + U_{1,n-1} \cdot \Delta\Theta_{n-1} + U_{12} \Delta\Theta_2 + U_{11} \Delta\Theta_1 = Y_1$

La figure 7 montre les calculs à effectuer dans l'exemple considéré :

$\Delta T = Y_3 / U_{33}$

$\Delta\Theta_2 = (Y_2 - U_{23} \cdot \Delta T) / U_{22}$

$\Delta\Theta_1 = (Y_1 - U_{12} \cdot \Delta\Theta_2 - U_{13} \cdot \Delta T) / U_{11}$

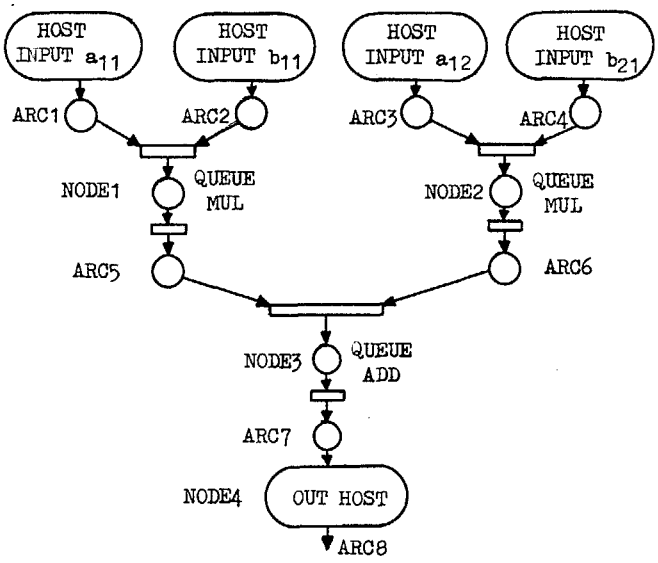
Le réseau complet de la commande cinématique est composé de l'ensemble de ces 3 sous-réseaux.

On voit donc sur cet exemple comment les RdPFD peuvent modéliser des calculs parallèles.

5. IMPLEMENTATION SUR PROCESSEUR A FLOTS DE DONNEES

Ce modèle théorique des réseaux de Petri à flux de données est directement implémentable sur le processeur à flots de données µPD 7281 de NEC.

La figure 8 donne un exemple d'implémentation du calcul $c_{11} = a_{11} \times b_{11} + a_{12} \times b_{21}$ présenté au 3.2.




```
1  .  EQUATE    HOST=0;
2  .  MODULE    EXONE=1;
3  .  INPUT     ARC1,ARC2,ARC3,ARC4;
4  .  OUTPUT    ARC8;
5  .  LINK      ARC5=NODE1(ARC1,ARC2);
6  .  LINK      ARC6=NODE2(ARC3,ARC4);
7  .  LINK      ARC7=NODE3(ARC5,ARC6);
8  .  LINK      ARC8=NODE4(ARC7);
9  .  FUNCTION  NODE1=MUL,QUEUE(QUE1,1);
10 .  FUNCTION  NODE2=MUL,QUEUE(QUE2,1);
11 .  FUNCTION  NODE3=ADD,QUEUE(QUE3,1);
12 .  FUNCTION  NODE4=OUT1(HOST,0);
13 .  MEMORY    QUE1=AREA(1);
14 .  MEMORY    QUE2=AREA(1);
15 .  MEMORY    QUE3=AREA(1);
16 .  START;
17 .  DATA      EXEC(EXONE,ARC1);
18 .  DATA      EXEC(EXONE,ARC2);
19 .  DATA      EXEC(EXONE,ARC3);
20 .  DATA      EXEC(EXONE,ARC4);
21 .  END.
```

Figure 8 : **Implémentation du calcul : $c_{11} = a_{11} \times b_{11} + a_{12} \times b_{21}$ sur processeur à flots de données 7281**

6. COMPARAISONS AVEC LES SOLUTIONS ACTUELLES

On trouve chez tous les constructeurs de robots modernes, une architecture multiprocesseur dans laquelle plusieurs modules dotés d'un microprocesseur ou non sont connectés sur un bus commun et travaillent en parallèle.

Chaque module comporte les ressources qui lui sont nécessaires en RAM, E/S etc... lui seul pouvant y accéder. Les conflits de partage de ressources sont ainsi en partie résolus.

Toutefois, certains modules ont des informations à échanger comme par exemple des résultats de calculs. Cet échange est résolu par la présence d'une zone RAM accessible à tous, les transferts ayant lieu par l'intermédiaire du bus principal.

Des dispositifs sont prévus au niveau logiciel et matériel :
-Tout module qui veut avoir accès au bus principal pour communiquer avec un autre doit avoir la possibilité au niveau logiciel de verrouiller le bus pendant tout le temps qu'il l'occupe.
- l'accès au bus se fait sur la base de niveau de priorité.

Le MULTIBUS est très bien adapté à ce type de structure dont le fonctionnement est asynchrone et qui permet à des unités centrales différentes (8.16 bis) travaillant à des vitesses différentes de collaborer.

L'utilisation d'un ou plusieurs processeurs 7281 comme coprocesseurs de calcul permet d'augmenter sensiblement les vitesses de calcul.

A titre indicatif, le modèle géométrique direct d'un manipulateur articulé à 6 degrés de libertés, est obtenu dans le formalisme de Denavit. Hartenberg en 48 µs avec 1 processeur et 3 µs avec une machine de 14 processeurs, pour des matrices 4 x 4 et le passage d'un répère Rn au suivant Rn + 1.

De plus l'utilisation d'une architecture à flux de données permet de résoudre les problèmes évoqués précédemment.

7. CONCLUSION :

L'architecture à flux de données est parfaitement bien adaptée aux calculs concurrentiels pour les raisons suivantes :
- facilités d'implantation du calcul parallèle ;
- toutes les opérations constituant le programme sont locales et dépendent uniquement des opérandes dont elles ont besoin ;
-il n'y a pas d'interférence entre les différentes tâches exécutées simultanément, car les zones mémoires ne sont jamais partagées.

Elle peut donc être utilisée partout où des calculs rapides et volumineux doivent être réalisés comme en robotique, filtrage numérique...

BIBLIOGRAPHIE :

1- µ pd 7281 user's manual.

2- BARBAGELATA B. et ABELLARD P. : Data flow petri nets for data flow processors. The Petri net newsletter, No 24, pp 18 - 20, August 1986.

3 - ALMANAH J. : Modélisation par réseaux de Petri à flux de données. Application à la synthèse de l'opérateur de Ricatti rapide. Thèse de doctorat d'Etat, Université d'Aix-Marseille 3 , juin 1982.

4-BARBAGELATA B. et ABELLARD P. : Parallel processing modelling with data flow petri nets. First european workshop on parallel processing techniques for simulation, 28 - 29 October 1985 UMIST Manchester.

5- MESHACH W ; Data flow IC makes short work of though processing chores, electronic design No 17 pp 191-206, may 1984

6- VAUTHERIN J. Un modèle algébrique basé sur les réseaux de Petri pour l'étude des systèmes parallèles. Thèse Docteur Ingénieur Université de Paris Sud Juin 1985

7- KRISNA M KHAVI, BILL P BUCKLES, U NARAYAN BHAT A formal definition of data flow graph models IEEE Vol C35 N°1pp 940-948 Nov 1986

8- P. LOPEZ - J.N. FOULC . Commande des systèmes robotiques. Editests.