PREMIER COLLOQUE IMAGE
Traitement, Synthèse, Technologie et Applications
BIARRITZ — Mai 1984 —

RÉALISATION D'UN SYSTÈME PARALLÈLE POUR LE GRAPHIQUE
DESIGN OF A PARALLEL SYSTEM FOR COMPUTER GRAPHICS

Mohammed REDJIMI

LA 369 CNRS - Université de Lille I - 59655 VILLENEUVE D'ASCQ - CEDEX

RESUME

Les besoins de performances en temps d'exécution et capacité de traitement et de stockage pour des applications graphiques font évoluer les systèmes informatiques concernés vers des architectures spécialisées.

Un modèle développé au laboratoire d'informatique de LILLE 1 est un multiprocesseur composé de 16 processeurs microprogrammables de traitement, chacun pouvant fonctionner de manière autonome avec sa propre mémoire de commande et accédant à la mémoire de donnée par une fenêtre de 8 bits. Pour les applications parallèles, un processeur maître pilote le système et assure les entrées-sorties.

Des moyens câblés de communication, de routage et de synchronisation ainsi qu'une horloge commune assurent le traitement de tâches parallèles et concurrentes. L'ordre de grandeur des performances du système est de 64 MIPS.

Un processeur hôte basé autour d'un 8085 assure un certain nombre de fonctions logicielles telles que compilation, gestion de fichier etc...

Le noyau de synthèse d'images est hiérarchisé à 3 niveaux :

- *le niveau langage de commande composé d'un interpréteur produisant des macro-commandes,*
- *le niveau macro-commandes qui sont des séquences spécifiques de micro-instructions réalisant des fonctions graphiques,*
- *le niveau micro-instructions qui sont les codes exécutables par les processeurs élémentaires.*

SUMMARY

The need of high performances for computer graphics involves the computer systems to specialized configuration.

A model developped in "Laboratoire d'Informatique de LILLE 1" is a multiprocessor system which is composed of sixteen microprogrammable processing elements. Each of them can perform tasks independantly, it have a proper control memory and can access to a data memory by means of an 8 bit window. For parallel applications the system is driven by a master processor that performs also I/o functions. The performance order of the system is 64 MIPS.

Hardware communications and synchronisations based upon a common clock permit cooperation and concurrency between processors.

For graphical applications the processors operate in a SIMD mode, each of them accessing to the pixel in its window. The master processor issues a data memory address.

A host 8085 based computer performs software functions such as compilations, file management, user and I/o interfacing, microprogramms loading...

Graphical kernel for image synthesis allows a set of macro-command and is distributed at three levels :

- *high level langage (H.L.L), i.e an interpreter for macro-command generation,*
- *macro-command level, i.e a graphical specific set of micro-instructions,*
- *micro-instruction level, i.e code for the processing elements.*

Réalisation d'un système parallèle pour le graphique.

Design of a parallel system for computer graphics.

Mohammed REDJMI

1 - ARCHITECTURE GLOBALE DU SYSTÈME.

Un réseau de 16 processeurs parallèles et micro-programmables se partage les traitements d'informations stockées dans une mémoire d'images de 4 × 64 K pixels (ou octets). Chaque processeur accède à cette mémoire par une fenêtre, qui lui est propre, de 8 bits (soit un pixel), une image est ainsi décomposée en tranches de 16 pixels traités en parallèles.

Le pilotagede ce réseau ainsi que l'adressage de la mémoire d'images et la gestion des entrées-sorties sont confiés à un 17e processeur ; le processeur maître Pm.

Des outils cablés de communication et de synchronisation ainsi qu'une horloge commune permettent les coopérations entre les différents éléments du système ainsi que les cohérences temporelles et de traitement.

Un 18e processeur assure ; à la demande, la visualisation des activités du système en affichant l'activité des 17 processeurs selon divers modes de trace tous à l'échelle de la micro-instruction. C'est un outil indispensable à la mise au point de microprogrammes parallèles.

Le système convient à différentes activités dans le domaine de l'informatique graphique, notamment en synthèse et en manipulations d'images en temps réel. Une logique complémentaire permet l'accès direct en mémoire d'images et l'affichage d'un plan de celle-ci sur un moniteur d'affichage.

Enfin l'ensemble est supporté par un système hôte basé autour d'un proceseur 8085 (cartes multibus) assurant la gestion de fichier, la compilation, la production et le chargement de microprogrammes ainsi que la gestion des entrées-sorties via un clavier alphanumérique et un écran.

2 - PROCESSEUR ÉLÉMENTAIRE.

Les nécessités de fonctionnement évoquées précédemment ont conduit au choix du microprocesseur microprogrammable SIGNETICS 8 x 300 dont certaines caractéristiques sont présentées fig. 2.

Ce microprocesseur exécute en 250 ns une micro-instruction codée sur 16 bits issue d'une mémoire de 8 K mots. La micro-instruction est décomposée de la façon suivante :

- 3 bits (poids forts) : code opération
- 1 à 4 champs de micro-commandes pour représenter des constantes, des registres internes ainsi que le contrôle de rotation ou masques sur les données.

Dans la configuration actuelle chaque processeur dispose de sa propre mémoire locale de microprogrammes ceci permet d'avoir un fonctionnement en SIMD en dupliquant les mêmes micro-programmes dans les différents processeurs ou MIMD en y inscrivant des microprogrammes différents, les processeurs peuvent dans ce cas participer à l'adressage de la mémoire d'images grâce à des déplacements codés sur 8 bits.

3 - LES COMMUNICATIONS.

La coopération entre les différents processeurs exige des outils rapides de communication. Deux processeurs P_i et P_j peuvent communiquer s'ils disposent de ressources partagées, celles-ci pouvant être une mémoire, un registre ou une ligne. Ainsi plusieurs voies de communications ont été prévues pour les besoins du fonctionnement :

- diffusion d'une information depuis le processus maître vers un ou plusieurs processeurs du réseau
- Scrutation par P_m d'un processeur quelconque du réseau.

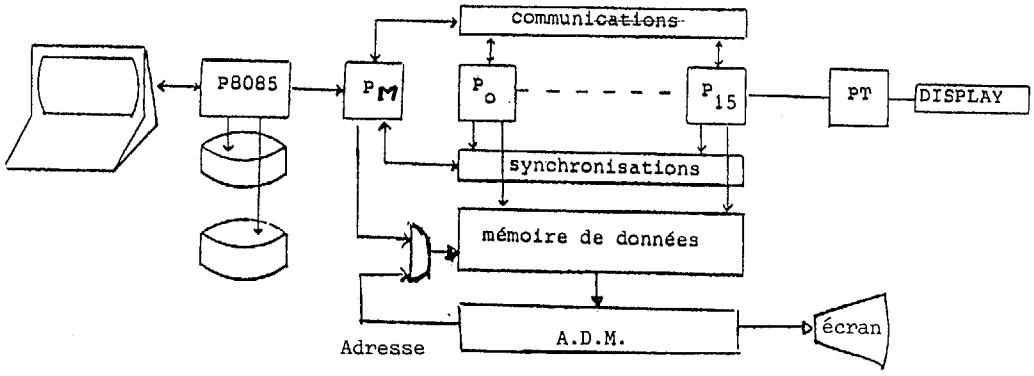
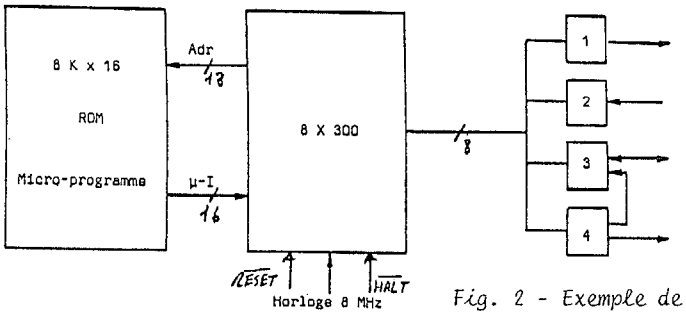


Fig. 1 - Architecture générale.

- Routage des informations entre un processeur P_i et ses deux voisins P_{i-1} et P_{i+1} (bidirectionnel).

Toutes ces communications se font par l'intermédiaire de registres adressables compatibles avec le 8 x 300, l'accès étant contrôlé par microprogrammes.



Registres externes : (1) : sortie
(2) : entrée
(3) : bidirectionnel
(4) : sortie (2 bits sélectionnent le sens de (3)
Reset : initialisation
Halt : arrêt.

Fig. 2 - Exemple de configuration.

4 - LA SYNCHRONISATION.

La synchronisation entre des sites différents est indispensable lorsque ces sites doivent concourrir à l'exécution de tâches coopérantes. Plusieurs notions différentes sont attachées à cette fonction, en effet des processeurs peuvent se synchroniser pour l'accès à une ressource critique par exemple ou pour signaler leur arrivée en un point donné de leur évolution ...

Dans notre cas, les fonctions de synchronisation doivent permettre le démarrage de tâches synchrones pouvant être de longueurs différentes.

Les processeurs peuvent exploiter les voies de communication et établir des protocoles de synchronisation gérés par micro-programmes. Cette solution ne peut être retenue car elle présente un overhead important ainsi que le risque d'encombrer le réseau de communication.

Des mécanismes cablés de synchronisation fonctionnent au niveau des micro-programmes selon le principe des rendez-vous. Ce fonctionnement est rendu possible étant donnée l'architecture même de la machine : sites identiques, toutes les micro-instructions sont de même durée (250 ns), horloge commune...

Principe des rendez-vous :

- Chaque processeur P_i dispose d'un indicateur S_i d'arrivée au rendez-vous.
- A l'initialisation S_i = FAUX.
- P_i arrive au rendez-vous et met S_i := VRAI.

La réalisation du rendez-vous se fait lorsque

$$C = \bigwedge_{i=0}^{n} S_i = \text{VRAI}.$$

- Chaque processeur qui arrive au rendez-vous exécute la primitive :

ATTENDRE : tant que C = FAUX aller à ATTENDRE

Ce test se fait par une seule micro-instruction (NZT).

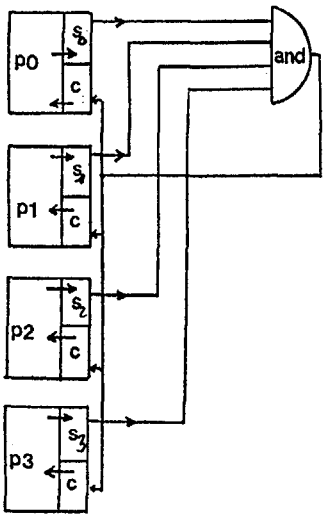


Fig. 3 - Illustration au principe du rendez-vous étendu à quatre processeurs.

4.1 - LA SYNCHRONISATION PAR GROUPE.

Ce mode répond aux exigences de synchronisation les plus fréquentes du système ; synchronisation entre 2 processeurs voisins (pour le routage d'une information par exemple), entre l'ensemble des processeurs etc...

Un groupe correspond à un ensemble figé de processeurs. La configuration actuelle autorise les 6 groupes suivants :

G1 : (0-1), (2-3), (4-5), (6-7), (8-9), (A-B), (C-D), (E-F)
G2 : (1-2), (3-4), (5-6), (7-8), (9-A), (B-C), (D-E), (F-0)
G3 : (0-1-2-3), (4-5-6-7), (8-9-A-B), (C-D-E-F)
G4 : (0-1-2-3-4-5-6-7), (8-9-A-B-C-D-E-F)
G5 : 16 processeurs de traitement
G6 : 16 processeurs avec le maître
G0 et G7 servent à l'initialisation.

Matériellement chaque processeur dispose d'un registre (3 bits) pour signaler le numéro de groupe dans lequel il veut s'insérer le signal de synchronisation lui est adressé par une logique cablée dans une bascule C.

4.2 - LA SYNCHRONISATION PAR NOM.

Cette version permet la formation dynamique des groupes de synchronisation. Au lieu d'afficher le numéro d'un groupe de synchronisation, un processeur affiche un nom, le circuit logique identifie les noms qui se forment. Les processeurs ayant affichés le même nom se synchronisent ensemble selon le principe du rendez-vous.

4.3 - CONCLUSION.

Il existe un risque d'interblocage certain dès qu'un processeur concerné par un rendez-vous ne respecte pas les règles de synchronisation ou tombe en panne. Ces primitives ne doivent pas être maniées par le programmeur mais restent au niveau de la microprogrammation.

5 - LE LOGICIEL.

Le processeur 8085 tourne sous le système d'exploitation ISIS II et permet dans la configuration actuelle :

- la programmation en assembleur 8085
- La programmation en PLM 80.
- La gestion de fichiers (gestion de 2 unités de disquette, liaison parallèle avec une mémoire de masse)
- L'édition de texte.
- La production de micro-codes 8 × 300 (grâce à un cross-assembleur 8 × 300).
- Des fonctions diverses de développement et de mise au point de programmes.

En plus l'extensibilité de ce processeur hôte est assurée ; ce système étant à base de cartes multibus.

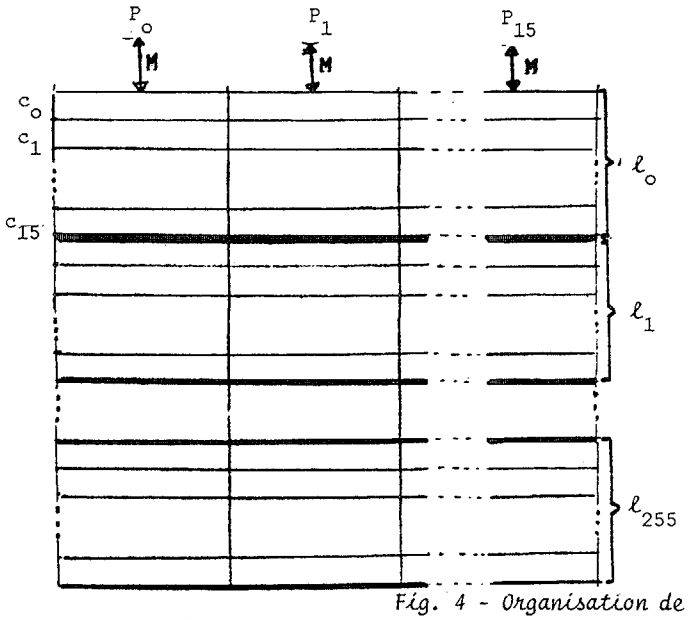
Une liaison parallèle permet l'interfaçage avec le processeur maître, les communications se font selon un principe producteur-consommateur.

Grâce à ces fonctions, nous avons développé un noyau de logiciel de base permettant :

- le chargement et l'exécution de micro-programmes.
- Une aide à la production de micro-programmes parallèles par l'intermédiaire d'un jeu de macro-commandes ; facilitant ainsi l'exploitation du système multiprocesseur.

6 - LES MACRO-COMMANDES.

Une image correspond à 256 × 256 pixels et occupe une page de la mémoire d'images. Les 16 processeurs traitent 16 pixels en parallèle. Une macro-commande est une séquence de micro-instructions réclamant un certain concours à l'ensemble des P_i et exploitant la mémoire d'images.



ℓ_i : numéro de ligne ($o \leq i \leq 255$).

c_j : numéro de colonne ($o \leq j \leq 15$).

M : accès mémoire.

Fig. 4 - Organisation de la mémoire d'images.

L'adresse mémoire correspond au triplet $<\ell,c,i>$ où ℓ désigne un numéro de ligne, c un numéro de colonne et i un numéro d'image ($o \leq i \leq 3$).

Dans un fonctionnement detype SIMD, l'adressage est confié au processeur maître. Les processeurs du réseau se partagent le traitement. Une macro-commande se présente sous la forme de 2 ou 3 octets (le bit 7 indique le nombre d'octets).

OCTET 1 : type d'opération (possibilité d'avoir jusqu'à 256 macro-commandes différentes).
OCTET 2 : Adressage (cet octet concerne uniquement P_m).
OCTET 3 : Paramètre quelconque (couleur, complément sur l'adressage...).

Exemples de macro-commandes :

a) LDI i_j, co: chargement de l'image numéro i avec la couleur Co

```
début :
    i:= i_i
    Pour ℓ := o à 255
        faire
            Pour c := o à 15
                faire
                    ∮ M ← co
                fait
        fait
    Retour
fin
```

b) MOVE i_j, i_k : $i_j \leftarrow i_k$

```
début
    Pour ℓ := o à 255
        faire
            Pour c := o à 15
                faire
                    i := i_k
                    ∮ R ← M
                    i := i_j
                    ∮ M ← R
                fait
        fait
        Retour
fin
```

CO R : registre de travail

ℓ, i, c paramètres définissant l'adresse.
∮x : la séquence x est exécutée par les 16 processeurs de traitement en mode SIMD. P_m exécute le reste de la procédure (adressage).
P_m assure aussi le contrôle de lecture/écriture en mémoire d'images.

7 - PRINCIPE DU NOYAU.

Le noyau se compose de 4 modules permettant :

- l'initialisation :

Initialisation des variables de synchronisation de communication et de contrôle de la mémoire d'images.

- L'analyse :

Réception d'octets issus du 8085, diffusion vers les P_i et identification des types de macro-commandes.

- La synchronisation :

Démarrage de traitements synchrones.

- Le lancement :

Branchement aux séquences spécifiques correspondant au type d'opération de la macro-commande.

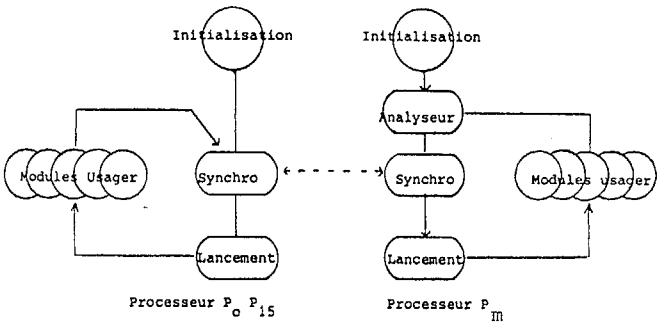


Fig. 5 - Principe du noyau.

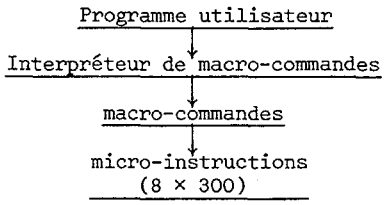
8 - LES PROGRAMMES DE L'UTILISATEUR.

Un programme utilisateur correspond à la combinaison des macro-commandes, deux modules permettent les exécutions des macro-programmes :

- le premier exécute les macro-commandes au fur et à mesure de leur entrée au clavier permettant ainsi la mise au point des programmes.

- Le second permet la constitution d'une séquence de macro-commandes, son exécution ne se faisant que sur demande de l'utilisateur.

Un niveau de programmation plus élevé est à l'étude, il s'agit là de l'écriture d'un interpréteur de macro-commandes permettant d'offrir des structures de contrôle plus élaborées ainsi que la possibilité de paramétrage.



9 - CONCLUSION.

L'architecture de la machine rend son application possible dans plusieurs domaines de traitements associatifs tels que la synthèse et l'animation d'images, la reconnaissance de formes... Le logiciel doit être, évidemment, spécifique à chaque application.

BIBLIOGRAPHIE.

[1] M.J. FLYNN, *Some computer organization and their effectiveness*, I.E.E.E Transactions on computer, sept. 72, pp. 59-76.

[2] L. LAMPORT, *The synchronization of independant processes*, Acta Informatica, vol. 7.

[3] V. CORDONNIER, *Un modèle associatif de synchronisation dans un système multiprocesseurs*, I.A.S. I.A.S.T.E.D. Symposium, Lile, mars 1983.

[4] V. CORDONNIER, *Noyau pour l'écriture de microprogrammes parallèles*, Publication interne ERA, février 1983.

[5] G. CONCALVES, Y. MERIADEC, *Architecture du processeur trace sur M.A.P*, Mémoire d'ingénieur ISEN Lille 1981.

[6] M. CULOTTI, *Langage d'animation d'images*, Mémoire de D.E.A, Lille 1983.

[7] M. REDJIMI, *Architecture d'un multiprocesseur associatif parallèle*, Mémoire d'ingénieur du C.E.R.I., Alger 1981.

[8] Notice SIGNETICS 8 × 300.

[9] B.R. RAU, C.D. GLASER, M.L. PICARD, *Efficient code generation for horizontal architetures : compiler techniques and architectural support.* The 9th Annual Symposium on Computer Architecture vol 10, n°3, april 1982.