METHODOLOGIE DE DETERMINATION D'OPERATEURS SPECIFIQUES EXEMPLE D'APLLICATION DE TRAITEMENT D'IMAGE EN TEMPS REEL.

Patrick ABELLARD - Gilles NOLIBE*

Laboratoire d'Automatique et d'Informatique appliquées de TOULON - Université de TOULON et du VAR - 83 130 LA GARDE - FRANCE.

* Sofware Based Systems - Groupe de traitement de l'information opérationnelle (SBS - TRIO)
Immeuble GALAXIE B - Avenue de LATTRE de TASSIGNY - 83000 TOULON - FRANCE

RESUME

Dans des domaines d'application tels que le traitement numérique du signal en temps réel ou la commande dynamique de robots, les quantités de données nécessaires à traiter en un temps souvent réduit est fort conséquente. De ce fait, les contraintes liées au traitement en temps réel, sont difficilement satisfaites par l'utilisations de machines séquentielles classiques construites suivant les principes définis par Von NEUMANN; et imposent alors de définir des structures calculatoires susceptibles d'exploiter de façon optimale le parallélisme intrinsèque des tâches à effectuer. Aussi pour détecter ce parallélisme, une description algébrique des flux de données représentatifs du traitement algorithmique est présentée. Une liaison étroite avec les Réseaux de PETRI à Flux de Données a pu être mise en évidence. Enfin, nous présentons un opérateur de convolution rapide pour le traitement d'image, obtenu en utilisant cette méthodologie.

Mots clefs: Détection de parallélisme, Traitement en temps réel, Calculateurs spécialisés, Architectutre d'opérateurs de traitement, Réseaux de PETRI à Flux de Données, Convolution rapide.

SUMMARY

In many computation domains such as Real-time digital signal processing or dynamic robot control, the great quantity of data to be computed in a short fixed time, is essentially the main encountered problem. So the use of sequential machines according on Von NEUMANN's principles, does not allow to completly satisfy the real-time contraints. An algebraic description of data flows which are representative of the algorithm structure s, is presented. The optimal use of parallelism is strongly attached with the algorithm writing. A strongly relationship with Data Flow PETRI Nets have been proofed. Then we present a fast convolution operator for image processing.

Keywords: parallelism detection, Real-time processing, Parallel computing structures, Specific operators, Data Flow PETRI Nets, Fast convolution.

I. Introduction.

Dans de nombreux domaines d'applications tels que le traitement numérique du signal en temps réel, les problèmes sont caractérisés par la grande quantité de données à traiter. Ce type de problèmes nécessite d'effectuer de nombreuses opérations en un temps réduit. De ce fait, les contraintes de temps réel sont difficilement satisfaites en utilisant des machines séquentielles classiques construites suivant les principes définis par Von NEUMAN; et imposent donc de définir des structures calculatoires susceptibles d'exploiter de façon optimale le parallèlisme intrinsèque des calculs à effectuer. Cependant *la détection du parallélisme est une condition préalable à son utilisation.*

Le principe du parallélisme étant, non plus de faire les opérations les unes après les autres, mais d'effectuer à un instant donné "toutes" celles susceptibles d'être réalisées. Il faut donc pour celà définir des relations de dépendance entre les différentes étapes de l'algorithme. Aussi pour permettre sa détection, nous proposons d'élaborer une description algébrique des flux de données représentatifs du traitement algorithmique, développée par NOLIBE[1].

Depuis une dizaine d'années, de nombreuses propositions ont été faites quant à la recherche optimale du parallélisme. Nous pouvons notamment citer la description géométrique proposée par QUINTON[2] pour la définition des réseaux systoliques, qui forment une classe particulière de structures parallèles de calcul.

Pour celà, nous définissons les données transitant entre les différents étapes algorithmiques, comme des ensembles que l'on appellera **flux**. Les étapes algorithmiques seront quant à elles représentées par des fonctions que l'on appellera **transformateurs**. A partir de cette description, les contraintes spécifiques aux problèmes de traitement du signal en temps réel ont été introduites.

Nous proposons d'appliquer cette méthode d'obtention d'opérateurs de traitement en temps réel à la définition d'un opérateur de corrélation rapide.

II Représentation algébrique et définition du parallélisme.

Un algorithme est constitué d'un ensemble d'étapes de traitement auxquelles on associe des "tâches" ou "processus". Ces différentes étapes traitent un ensemble d'informations que nous appellerons "données". L'ensemble des données transitant entre différents processus, forment un ensemble que l'on nommera **flux**. Ces étapes sont également formées par des sous-flux et des sous-étapes représentatifs de "sous-algorithmes". Cette décomposition structurelle dépend du niveau d'abstraction considéré pour la description algorithmique. Nous résumons dans cet article la description algébrique proposée par NOLIBE [1] pour la description des flux et des transformateurs agissant sur ces flux.

Nous dirons qu'un algorithme est représenté par un processus $\mathcal{A}$ formé d'un ensemble d'étapes $\mathcal{E}_A$ qui est inclus dans l'ensemble $\mathcal{E}$ des étapes, et d'un ensemble de flux $\mathcal{F}_A$ qui est inclus dans l'ensemble des flux $\mathcal{F}$ (Figure 1). Nous allons alors faire intervenir l'ordre des traitements dans cette description. Lorsque l'ordonnancement des traitements fera apparaître une indépendance entre les flux et les étapes, nous pourrons alors déterminer le parallélisme intrinsèque du processus $\mathcal{A}$ représentatif d'un algorithme pour un niveau d'abstraction donné.

II.1. Flux de données.

Un flux de données f de l'ensemble $\mathcal{F}$, est représenté par une entité f qui aura une existence sur l'axe d'ordonnancement, représenté par l'ensemble des réels **R**, et aura une durée d'existence (positive ou nulle) durant laquelle il véhicule effectivement des données.

Nous représenterons donc un flux $f \in \mathcal{F}$ à l'aide de la relation fonctionnelle:

$$\mathcal{F} \longrightarrow \mathbf{F}(\mathbf{R}, \mathbf{R}^+) \qquad (1)$$
$$f \longrightarrow f(t,\mu)$$

où t est l'instant d'existence du flux suivant l'axe d'ordonnancement et μ la durée pendant laquelle la représentation de f transporte des données. $t \in \mathbf{R}$ et $\mu \in \mathbf{R}^+$.

II.2. Définition des flux.

La représentation $f(t,\mu)$ du flux f est donc un ensemble qui existe à un instant donné pendant une **durée finie** μ. Nous pouvons donc écrire:

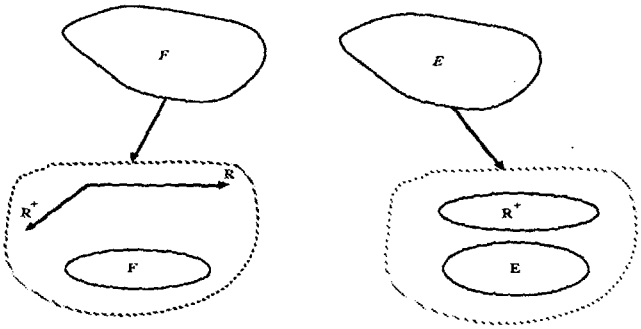

Figure 1. Représentation d'une étape algorithmique $\mathcal{A}$.

$$f(t,\mu) = \mathbf{f}(0,\mu) * \partial(\tau) \qquad (2)$$

tel que le support de $\mathbf{f}(0,\mu)$ soit borné et égal à $[0, \mu]$ et τ le temps dont il faut décaler $\mathbf{f}(0,\mu)$ pour obtenir $f(t,\mu)$. $\mathbf{f}(0,\mu)$ est appelé le modèle ou le patron du flux $f(t,\mu)$.

$$\mathbf{f}(0,\mu) \in \mathbf{F}.$$

De plus, l'application $\mathcal{L}$ de l'ensemble **F** vers l'ensemble des réels positifs $\boldsymbol{R}^+$ peut être définie tel que:

$$\mathcal{L}: \quad \mathbf{F} \longrightarrow \mathbf{R}^+ \qquad (3)$$
$$f(t,\mu) \longrightarrow \mathcal{L}(f(t,\mu)) = \mu.$$

Nous avons ainsi défini la durée d'un flux sur l'axe d'ordonncement.

II.3. Notion de séquentialité de flux.

On ne peut considérer l'entité de flux sans prendre en compte les éléments de transformation de ces flux. Aussi allons-nous définir des **transformateurs** qui représentent les étapes de transformations algorithmiques subies par les flux de données Avant d'introduire les transformateurs, nous allons définir la notion de séquentialité de flux, qui permettra de décrire les liaisons algorithmiques entre les différentes étapes d'un processus algorithmique.

Définition:

Soit $g(t,\mu_g)$ la représentation du flux g à l'instant t et de durée μ_g. Ce flux ne peut exister à cet instant que, si et seulement si, un flux $f(t,\mu_f)$, représentation des flux g à l'instant t et de durée μ_f, le précède dans la séquence. Ceci découle directement du principe de causalité dans l'ordonnancement des tâches.

Nous dirons alors que le flux $g(t,\mu_g)$ est séquentiel au flux $f(t,\mu_f)$. Par extension nous dirons que le flux f est séquentiel au flux g à l'instant t. Nous le noterons:

$$g(t,\mu_g) \text{ <------ } f(t,\mu_f) \qquad (5)$$

Le principe de causalité dans l'ordonnancement implique que, si deux flux $f(t,\mu_f)$ et $g(t,\mu_g)$ vérifient les relations:

$$f(t,\mu_f) = \mathbf{f}(0,\mu_f) * \partial(\tau_f) \text{ et } g(t,\mu_g) = \mathbf{g}(0,\mu_g) * \partial(\tau_g), \qquad (6)$$

alors g est séquentiel à f si et seulement si : $\tau_f \leq \tau_g$

Par extension on peut définir la relation "séquentiel ou égal à ", noté par " $\leq$--- ".

Ces deux relations sont des relations d'ordre [1], ce qui montre que dans l'ensemble des flux d'un algorithme, il existe un ordre suivant l'axe d'ordonnancement des traitements. Les flux sont donc des ensembles ordonnés suivant l'axe d'ordonnancement. Ils forment alors des "cortèges" [3].

II.4. Transformateurs de flux.

Nous appellerons **Transformateur**, toute fonction de l'ensemble des flux **F** dans lui-même transformant un flux f appelé "entrant", en un flux g appelé "sortant" (figure 2).

Soit T un transformateur agissant sur le flux à l'instant t:

$$T: g(t,\mu_g) \text{ <----- } f(t,\mu_f). \qquad (7)$$

Le transformateur est par conséquent une opération qui rend un flux g (sortant) **séquentiel à** un flux f (entrant).

Nous pourrons appliquer sur cet ensemble de transformateurs l'ensemble des règles de composition des applications.

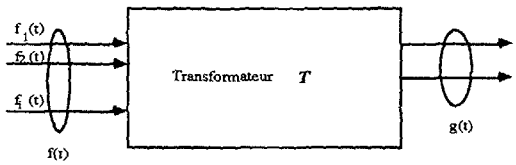

Figure 2. Transformateur de flux.

Propriété:

Un transformateur est une fonction causale. C'est à dire que le flux sortant $g(t,\mu_g)$ ne peut exister avant que le flux entrant $f(t,\mu_f)$ n'existe. Aussi d'après la définition de la séquentialité des flux:

- Soient deux flux $f(t,\mu_f)$ et $g(t,\mu_g)$ tels que:

$$f(t,\mu_f) = \mathbf{f}(0,\mu_f) * \partial(\tau_f) \text{ et } g(t,\mu_g) = \mathbf{g}(0,\mu_g) * \partial(\tau_g),$$

le transformateur T : g $\leq$--- f a un sens si et seulement si : $\tau_f \leq \tau_g$ (8)

Remarque:

Un transformateur est représentatif des étapes algorithmiques. L'ensemble **T** des transformateurs T est donc une représentation des étapes de traitement de l'ensemble $\mathcal{E}$.

II.5. Définition du parallélisme de traitement.

Nous dirons que deux transformateurs sont **parallèles**, si le flux entrant du premier transformateur n'est ni séquentiel ni égal au flux sortant du second transformateur **et** si le flux entrant du second transformateur n'est ni séquentiel ni égal au flux sortant du premier. C'est à dire que la proposition logique suivante est vraie:

$$T_1 : g_1 \leq\text{----- } f_1 \quad \text{et} \quad T_2 : g_2 \leq\text{----- } f_2 .$$

$$T_1 \,//\, T_2 \Leftrightarrow \overline{(f_1 \leq\text{---- } g_2)} \wedge \overline{(f_2 \leq\text{---- } g_1)} \qquad (9)$$

Cette condition est également connue sous le nom de "condition de BERNSTEIN" [4][5][6].

Nous noterons:

$$T_2 // T_1 \quad \text{avec } T_1 : g_1 \leq\text{----- } f_1 \quad \text{et} \quad T_2 : g_2 \leq\text{----- } f_2 . \qquad (10)$$

Par définition, nous dirons que deux transformateurs sont **séquentiels** si cette proposition booléenne est fausse. C'est à dire que l'un des transformateurs doit être appliqué avant l'autre. Nous noterons:

$$T_2 <\text{----} T_1 \quad \text{avec } T_1 : g_1 \leq\text{----- } f_1 \quad \text{et} \quad T_2 : g_2 \leq\text{----- } f_2,$$

lorsque $f_2 <\text{---- } g_1$ ou lorsque $f_2 = g_1$.

III. Introduction des contraintes de temps réel.

Nous nous intéressons dès lors à des applications de traitement en temps réel dans des problèmes d'identification et de filtrage. Aussi les critères propres à ce type d'applications, vont-ils être introduits dans ce modèle. Ces critères sont au nombre de trois:

- *Discrétisation temporelle des flux.*
- *Séquencement des calculs par rapport à des évènements extérieurs.*
- *Utilisation exclusive des données passées ou présentes .*

Généralement dans les algorithmes de traitement du signal en temps réel, on effectue un certain nombre de tâches à l'arrivée de chaque mesure. Ces tâches sont repétitives et sont effectuées pour chacun des échantillons reçus au cours du temps. Les relations de dépendance entre les différents transformateurs séquentiels d'un algorithme peuvent alors être définies (figure 3). Nous avons un espace de calcul défini d'une part par l'axe d'acquisition des mesures (axe t_m) et d'autre part par l'axe d'ordonnancement associé aux calculs pour une mesure k (axe t_c).

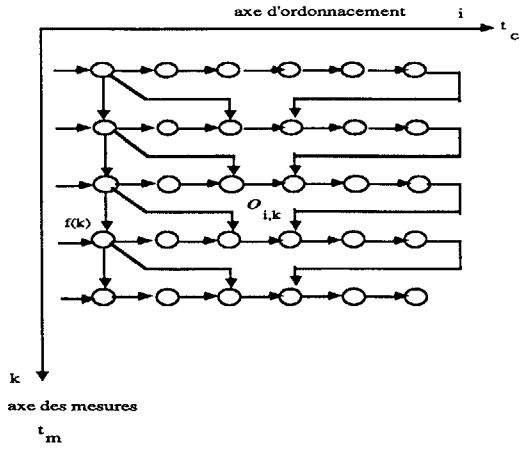

Figure 3. Exemple de relations de dépendance d'un algorithme temps réel.

Nous avons ainsi déterminé l'ensemble des relations de séquentialité et de parallélisme entre les différents transformateurs associés à un algorithme donné. Cependant l'axe associé effectivement au temps dans ce type d'algorithmes est l'axe t_m. Aussi le but recherché pour définir des opérateurs exploitant au mieux l'éventuel parallélisme des calculs, est de transformer l'axe d'ordonnancement sur un axe dépendant de l'axe d'acquisition des mesures. Cet axe représente en effet l'axe du "temps réel".

III.1. Introduction de l'ordonnancement des mesures.

Posons k l'indice associé à l'axe t_m, et i l'indice associé à l'axe t_c, qui représente la tâche de la structure algorithmique à l'instant de mesure k. Dans ce cas de figure les flux sont ordonnés suivant l'axe t_c et l'axe t_m représente "l'espace".

Posons également $O_{i,k}$ l'opérateur associé à l'exécution de la tâche i pour l'instant de mesure k tel que:

$O_{i,k}$: $g_{i,k} <\text{--- } f_{i,k-1}$ avec $O_{i,k}$ identique à $O_{i,k-1}$ (similitude des tâches pour chaque mesure).

Comme à la figure 3, et compte tenu des contraintes de temps réel, seules les données présentes ou passées interviennent dans les calculs. Aussi pourrons nous avoir dans ce type d'algorithmes:

$$O_{i,k} <\text{--- } O_{j,k-1} \quad \text{avec } l > 0 \text{ lorsque } j=i \text{ et } l \geq 0 \text{ lorsque } j < i. \qquad (11)$$

Si $j < i$ en posant $q = i - j$, on a suivant l'axe t_c, les relation entre les flux entrants et sortants:

$$f_{i,k} = \{g_{i-1,k}; \ldots ; g_{j,k-1}\} = \{g_{i-1,k}; \ldots ; g_{i-q,k-1}\} \qquad (12)$$

On obtient ainsi un retard de q-1 suivant l'axe t_c. Cela revient à inclure des transformateurs retards suivant l'axe t_c dans la description algorithmique.

Si $j \geq i$, en posant $q = j - i$, on obtient:

$$f_{i,k} = \{g_{i-1,k}; \ldots ; g_{i+q,k-1}\} \Longrightarrow O_{i,k} <\text{--- } O_{i+q,k-1}$$

avec également $O_{i+q,k-1} <\text{--- } O_{i,k-1}$.

Il faut donc attendre que les opérateurs $O_{i,k-1}$ à $O_{i+q,k-1}$ aient totalement fini de travailler avant d'entreprendre le calcul associé à l'opérateur $O_{i,k}$.

III.2. Cardinal d'interaction.

Compte tenu de ces contraintes, il serait utile de définir un critère permettant de détecter le parallélisme de ce type d'algorithme. Aussi définit-on le cardinal d'interaction de l'opérateur $O_{i,k}$ comme:

$$Q(O_{i,k}) = (j-i) \text{ lorsque } j \geq i. \qquad (13)$$

C'est un paramètre qui traduit le nombre maximal d'opérateurs que doit traverser un flux pour pouvoir intervenir sur l'opérateur $O_{i,k}$ depuis un opérateur $O_{i,k-1}$.

On définit le cardinal d'interaction de l'opérateur global associé à l'algorithme:

$$K_O = \max Q(O_{i,k}) \text{ pour l'ensemble des opérateurs } O_{i,k}. \qquad (14)$$

Si $K_O = N$ (N nombre d'opérateurs unitaires $O_{i,k}$ pour une branche associée à l'indice k), le problème est totalement séquentiel. En effet avant d'entreprendre les calculs relatifs à l'instant de mesure k+1 suivant, l'ensemble des calculs associés à l'instant de mesure k doivent être effectués. Les axes t_c et t_m sont alors colinéaires (figure 4).

Si $K_O < N$, alors le problème peut être parallélisé sous forme "pipe-line".

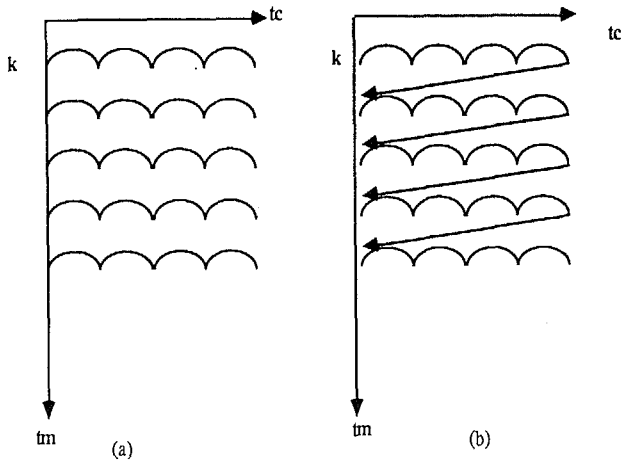

Figure 4. ordonnancement des calculs. Relations entre t_c et t_m. En (a): un algorithme totalement "pipe line". En (b): un algorithme totalement séquentiel.

Si on pose G_k, le transformateur fournissant la mesure f(k) à traiter à l'instant k, la séquentialité inhérente au problème de temps réel impose :

$$G_k \text{ <--- } G_{k-1} \text{ et } O_{0,k} \text{ <--- } G_k. \qquad (15)$$

Nous avons donc:

$$O_{0,k+1} \text{ <--- } G_{k+1} \text{ <--- } G_k \text{ ==> } O_{0,k+1} \text{ <--- } G_k . \qquad (16)$$

Compte tenu de cette contrainte et de la définition de K_O, nous pouvons écrire:

$$\min K_O = 1. \qquad (17)$$

Lorsque $K_O < N$, on a:

$$\forall O_{ik} \ , \ Q(O_{ik}) = p_{ik} \text{ avec } p_{ik} < n \qquad (18)$$

On peut alors définir un macro transformateur $H_{i,k}$ tel que:

$$H_{i,k} = O_{i,k-1} \circ \ldots \circ O_{i+p,k-1}. \qquad (19)$$

Compte tenu des relations 13, 14 et 17, nous obtenons le nouveau cardinal d'interaction global qui est égal à 1. Nous pouvons alors écrire qu'à l'instant k:

$$H_{i,k} \ // \ H_{i+1,k-1}.$$

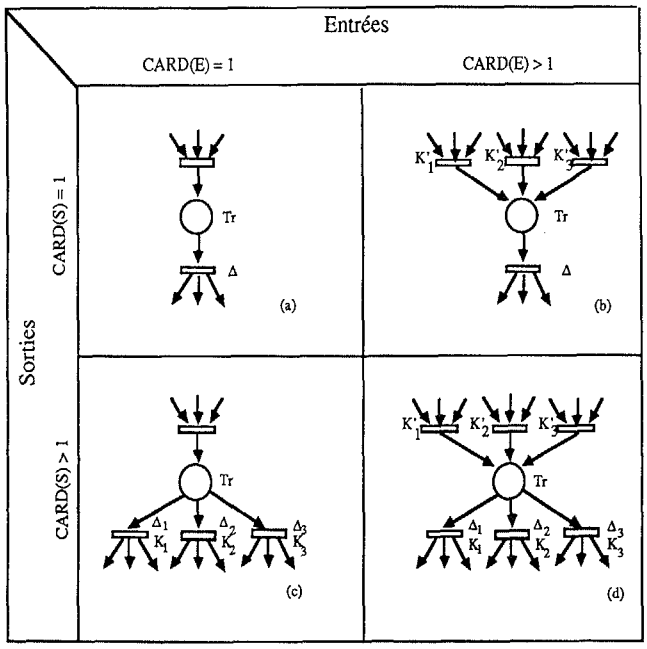

Figure 5. Représentation d'opérateurs par RdPFD en fonction de la cardinalité des flux.

IV. Analogie avec les réseaux de PETRI à Flux de Données.

Le tableau de la figure 5 permet d'tablir facilement un lien avec les réseaux de PETRI à Flux de Données (RdPFD) [5][6]. Dans ces réseaux, la place opérateur T_r représente le transformateur d'action sur le flux d'entrée tel qu'il a été défini préalablement. Si l'on associe à cette place une constante pour le temps d'exécution, nous obtenons alors un transformateur muni de l'opérateur retard, qui peut alors être inclus dans la transition Δ.

A cette fonction de temporisation qui définit le temps d'exécution de la tâche, il est possible d'asscier des conditions logiques de transition pouvant être liées ou non aux propriétés ou aux caractéristiques des flux.

Il est à noter [1], que le passage entre ces deux modèles est très aisé et que la description algébrique proposée permet une programmation immédiate à l'aide de langages concurrents tels que OCCAM.

V. Application.

La mise en oeuvre de cette méthode permet d'obtenir par exemple, un opérateur rapide de convolution bidimensionnel présentant la structure de la figure 6. Dans cet exemple, nous avons considéré que le filtre de convolution avait la propriété de séparabilité.

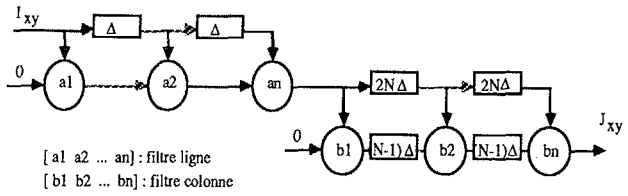

Figure 6. Exemple d'opérateur rapide de convolution 2D séparable.

VI. Conclusion.

Cette méthodologie permet, en utilisant les bons critères liés au traitement du signal en temps réel, doptimiser la quantité de ressources matérielle à mettre en oeuvre pour définir un opérateur dédié, ainsi que le temps d'exécution de l'algorithme. Un lien étroit a été mis en évidence entre un modèle algébrique de description des flux et les Réseaux de PETRI à Flux de données, ce qui permet d'envisager une programmation quasi-immédiate à l'aide de langages concurrents.

Bibliographie.

[1] G. NOLIBE - "Développement d'une méthodologie de détermination d'opérateurs de calcul spécifiques dans des problèmes d'estimation et d'identification en temps réel."
Thèse de Doctorat de l'Université de TOULON et du VAR 1988

[2] P. QUINTON -"The Systematic Design of Systolic Arrays."
Publication Interne IRISA n° 193 Avril 1983

[3] Y. KORCHOUNOV - "Fondements mathématiques de la Cybernétique." Editions MIR MOSCOU- 1975

[4] J. DUMAS - "Sur l'Implantation Optimale d'une Structure de Commande sur Calculateur Temps Réel."
Thèse d'Etat de l'Université du LANGUEDOC Montpellier - Juin 1979

[5] J.ALMHANA - "Modélisation par Réseaux de PETRI à Flux de Données. Applications à la Synthèse d'un Opérateur de RICCATI Rapide."
Thèse d'Etat de l'Université d'Aix-Marseille III
Juin 1983

[6] P. ABELLARD - "Contribution à l'Etude d'Extension des Réseaux de PETRI à Flux de Données pour la Télésymbiotique Assistée par Calculateur."
Thèse de Doctorat de l'Université de TOULON et du VAR
Juin 1988