SEPTIEME COLLOQUE SUR LE TRAITEMENT DU
SIGNAL ET SES APPLICATIONS

NICE du 28 MAI au 2 JUIN 1979

PROGRAMMES DE CONCEPTION DES MICROSYSTEMES NUMERIQUES A HAUTES PERFORMANCES (*)

M. GHERBI

S I N T R A - 26, rue Malakoff - ASNIERES 92/600/Institut de programmation PARIS VI

RESUME

La conception de systèmes numériques pour le traitement du signal pose quelques problèmes particuliers :

- les débits des échantillons en entrée sont en général importants,
- les délais (retard entre l'entrée et la sortie) n'ont que peu d'importance,
- les algorithmes à implémenter ont un caractère répétitif et arithmétique,
- recherche de performances élevées à partir d'une technologie donnée et d'un minimum de ressources.

Nous présentons un ensemble de programmes de CAO qui permet d'aider l'Ingénieur logicien dans la conception des architectures spécialisées dans "l'implémentation" d'expressions arithmétiques complexes ou de récurrences linéaires telles qu'elles apparaissent en traitement du signal (FFT, filtres, ...).

Ces programmes permettent :

1) - d'optimiser le chemin des données : nombre d'opérateurs (multiplicateurs, additionneurs, ...), capacité des mémoires "pipe-line", ..., en fonction des contraintes (débit, délai, technologie des composants, ...),
2) - d'en déduire le graphe de contrôle qui pourra être implanté sur PROM ou PLA.

Les diverses étapes de la CAO sont explicitées dans l'article :

1) - Parallèlisation de l'algorithme.
2) - Affectation des opérateurs en fonction des contraintes.
3) - Calcul de la capacité des mémoires "pipe-line".
4) - Séquencement des opérations élémentaires.

SUMMARY

The conception of numerical systems in signal processing engender some particular problems :

- flows of input samples are generally important,
- delays (delay between input and output) have little importance,
- the algorithms to implement are repetitive and arithmetical,
- research of high performance from a given technology and a minimal resources.

We present a set of programs CAO to help a software engineer to realize a specialized architectural for implementation of complet or recurrent arithmetical expressions which appear in signal processing (FTT, filter)...)

These programs allow :

1) - optimizing the data flow : number of operators (multiplicator, UAL,...), capacity of memories "pipe-line" according to constraints (flows, delays, technology of components),
2) - deduction of control graph which will be implemented in PROM or PLA.

The CAO steps are explained in this paper :

1) - decomposition of algorithm in parallel steps,
2) - assignement of operator according to constraints,
3) - processing of capacity memories "pipe-line,
4) - timing of elementary operations.

(*) Contrat DRET

PROGRAMMES DE CONCEPTION DES MICROSYSTEMES NUMERIQUES
A HAUTES PERFORMANCES

CONCEPTION DE SYSTEMES NUMERIQUES A HAUTE PERFORMANCE

I - PRESENTATION

Le traitement du signal regroupe un "ensemble de théories et de techniques" dont les algorithmes (FFT, filtres transversaux, normalisation, traitement d'images ...) présentent les caractéristiques suivantes :

- Traitement en temps réel d'informations digitales à débit constant.
- Rapidité du traitement.
- Caractère répétitif marqué.
- Grande variété d'algorithmes.

Jusqu'à présent, la spécificité de chaque problème particulier a conduit à la conception de circuits câblés spécifiques pour chaque traitement particulier. Néanmoins, si les circuits câblés satisfont au critère de rapidité, ils trouvent leurs limites dans :

- la rigidité, l'absence de banalisation ...

Le but de cette étude est de présenter un outil de conception d'opérateurs spécialisés qui remplaceront les circuits câblés par la microprogrammation qui offre une plus grande souplesse et des ratios de performance (nbre d'opérations/encombrement ou coût) meilleurs ou équivalents. Ceci nous amène à concevoir une machine adaptée à une famille d'algorithmes où chaque algorithme sera réalisé par un opérateur spécialisé.

II - METHODOLOGIE

Le schéma de la figure n° 1 représente les différentes phases de traitement pour la conception d'opérateurs spécialisés.

Nous allons voir en détail les principales étapes.

II - 1 - Introduction et analyse du chemin de données [1]

L'architecture (mémoires, opérateurs, bus, interconnexions ...) est introduite et analysée dans cette partie pour déterminer la construction du graphe de commande du chemin de données. (Ensemble des commandes exécutables en parallèles, ou champ de microprogrammation).

II - 2 - Introduction et analyse de l'algorithme.

Un algorithme qui a été programmé en langage Fortran peut faire apparaître plusieurs types d'équations (figure n° 2). Des études faites pour paralléliser des expressions arithmétiques [2] nous ont aidé à présenter quelques algorithmes de transformation de programmes séquentiels en programmes parallèles selon les types d'équations qui les composent (figure n° 3). L'application de ces algorithmes de transformations aux expressions de la figure 2 est illustrée par la figure 4. (Modélisation par Réseau de Pétri où les places représentent les données et les transitions représentent les opérations [3]).

$y = (U + 5) * (D - E) - F) * (G/(H + K) - M)$

a) équation simple : type 1

$x = (A + B + C) - ((F/K) * H)$

$y = (A + B) - (H * C)$

$z = (F/K) + (A - C)$

b) équations à expressions communes : type 2

$x = A * B + C/D$

$y = X - E$

$z = X + Y + C$

c) équations dépendantes : type 3

$$y = \sum_{i=1}^{n} xi$$

d) équation récurrente : type 4

Figure n° 2 : Types d'équation.

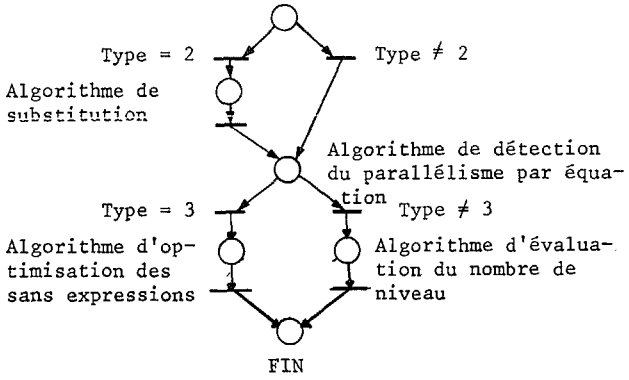


Figure n° 3 : Algorithme général de transformation de programmes séquentiels en programmes parallèles (modélisé en Réseau de Pétri).

Après avoir vu la partie traitement de l'algorithme qui consistait à déterminer le parallélisme maximal au niveau des expressions, nous allons étudier la conception du graphe de contrôle.

II - 3 - Conception du graphe de contrôle

Les niveaux de la conception du graphe de contrôle sont représentés par le schéma de la figure n° 6. Ils seront détaillés et illustrés à travers l'exemple de l'opération papillon de la FFT (4) :

$x = ao + (a1c - b1s)$
$y = ao - (a1c - b1s)$
$z = b1 + (a1s + b1c)$
$w = b1 - (a1s + b1c)$

dont la modélisation en RdP après transformation est représentée par la figure n° 5.

niveau 1 : Ce niveau consiste à récupérer le graphe de commande du chemin de données élaboré lors de l'étape "Introduction et analyse du chemin de données" et de déterminer le nombre d'opérateurs (UAL, multiplieurs) et leurs temps d'exécution.

Pour notre exemple, nous prendrons un multiplieur de 200 ns (4 unités) et un additionneur de 50 ns (1 unité).

niveau 2 : La figure n° 5 fait apparaître l'ensemble des opérations exécutables en parallèles. Au niveau 1, quatre multiplications sont possibles, au niveau 2 et au niveau 3 on peut effectuer en parallèles 2 additions/soust. et 4 additions/soust. respectivement.

Ce degré de parallélisme offert par l'algorithme et les transformations effectuées, doit être adapté à celui qu'offre l'architecture existante. Dans notre cas, il y a impossibilité d'effectuer en parallèles les 4 multiplications, car on ne dispose que d'un seul multiplieur. L'algorithme "d'affectation des opérateurs" appliqué à ce niveau aboutit au RdP transformé de la figure n° 7.

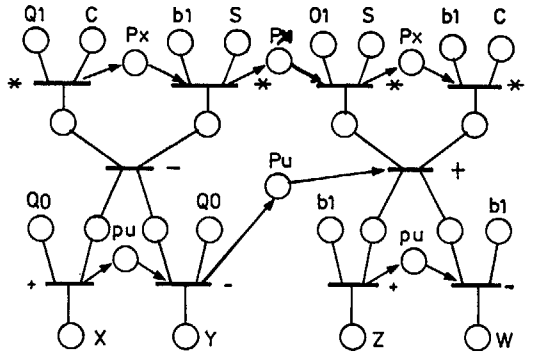


Figure n° 7 : Affectation d'opérateurs.

Les places marqués P^* et P^u établissent l'ordre d'utilisation du multiplieur et de l'UAL respectivement.

Niveau 3 : Pour les algorithmes qui opèrent sur des données arrivant par flot, la contrainte principale est le débit d'entrée/sortie et le temps de traitement. Par conséquent, les opérateurs d'adressage (processeurs d'adressage) et les moyens de communications (bus) doivent être pris en compte à ce niveau après avoir été décrits lors de "l'introduction et l'analyse du chemin de données". Ceci nous amène à compléter la modélisation du RdP de la figure n° 7 en incluant les entrées/sorties des données, et en introduisant le parallélisme dû à la structure pipe-line.

Pour notre exemple, nous utilisons 2 processeurs d'adressage dont le temps d'adressage est de 4 unités, 2 bus bidirectionnels et un temps de traitement de 1 µs.

Niveau 4 : Ce niveau temporise le RdP [5] obtenu à partir du niveau 3. Chaque transition (opération) sera tirée (exécutée) à un temps calculé lors de l'étape précédente. Le RdP temporisé obtenu (figure n° 8) sera fourni au concepteur ainsi que diverses indications sur les composants du chemin de données (goulots d'étrangement) au cas où le temps de traitement calculé est supérieur au temps de traitement défini ; le concepteur modifiera en conséquence le chemin des données (niveau 5, figure n° 6).

Niveau 5 : Le schéma de la figure n° 8 est une évaluation du temps de traitement minimal. Il apparaît dans cette figure que toute tentative de réduction du temps de traitement passe par une modification du chemin de données au niveau de nombre des processeurs d'adressage.

Niveau 6 : L'utilisation d'une structure pipe-line (6) entraîne un certain accroissement du nombre de mémoires, ceci étant dû au fait qu'à chaque instant, le traitement s'effectue sur des données différentes (traitement de 3 opérations papillons en parallèles).

Niveau 7 : Le coût des mémoires étant important, on optimisera à ce niveau le nombre de mémoires engendrées au niveau 6. Il est à noter que l'utilisation de ces mémoires se fait à l'aide d'instructions d'adressage, ce qui a pour effet d'augmenter le temps de traitement calculé lors du niveau 4. Si cette augmentation aboutit à un temps trop grand, on introduira des tampons (niveau 8) pour résoudre les problèmes d'accès simultanés aux mémoires.

Niveau 8 : L'ajout de mémoires et éventuellement de tampons, fait apparaître des instructions élémentaires (adressage, transfert) en amont et en aval des opérations de multiplication, d'addition et de soustraction (Figure n° 9).

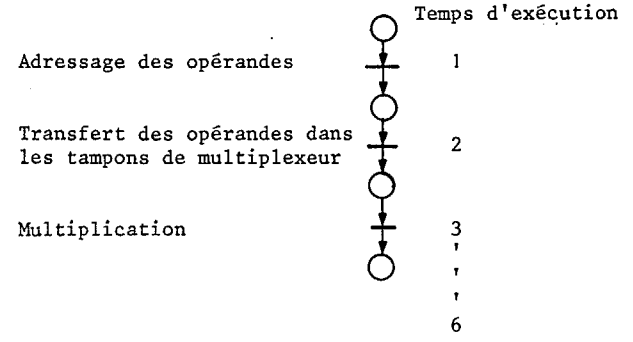


Figure n° 9 : Décomposition de l'opération *.

On aboutit finalement au Rdp de la figure n° 10. Chaque niveau correspond à un ensemble d'opérations à exécuter en parallèles à chaque coup d'horloge. Le diagramme de la figure 11 illustre le temps d'occupation des ressources du chemin de données. Le Rdp séquentiel de la figure 13 est implémenté sur FPLA (7). La séquentialité de ce Rdp est dû au fait qu'il n'y a pas de test à effectuer dans l'algorithme.

Définition des opérations associées aux transitions (figure 10).

Symboles utilisés :

M^u : mémoires de l'UAL (16 registres)
M^m : mémoires de multiplieur (16 registres)
BUS PA1 : bus connecté sur processeur d'adressage 1
BUS PA2 : bus connecté sur processeur d'adressage 2
MV1 : mémoire vive de stockage de données
MV2 : mémoire vive de stockage de données
T1, T2 : tampons entrées du multiplieur
TPS : tampon sortie du multiplieur

T1 : adressage mémoire vive. Transfert donnée sur BUS PA1
T1 (1) : adressage mémoire locale (M^u) - rangt. donnée de BUS PA1 dans M^u
T2 : adressage MV1 - Transfert donnée sur BUS PA1
T2 (1) : ad. mémoire locale ou ML (M^m) - rangt. donnée de BUS PA1 dans M^m
T3 : ad. MV1 - Transfert sur BUS PA1
T3 (1) : ad. ML (M^m) - rangt. donnée dans M^m
T3 (2) : ad. ML (M^u) - sortie sur BUS PA1 du résultat
T17 : ad. MV3 pour rangt. donnée dans MV1
T17 (1) : ad. ML (M^u) - sortie sur BUS PA1
T19 : ad. MV1 pour rangt. donnée dans MV1

T19 (1) : ad. M^u - sortie résultat
T6 : ad. MV2 - transfert donnée sur BUS PA2
T6 (1) : ad. (M^u) - rangt. donnée de BUS PA1 dans M^u
T4 : ad. MV2 - transfert sur BUS PA1
T4 (1) : ad. ML (M^m) - rangt. donnée de BUS PA1 dans M^m
T5 : ad. MV2 - transfert dans BUS PA2
T5 (1) : ad. ML (M^m) - rangt. donnée dans M^m
T18 : ad. MV2 pour rangt. résultat dans MV2
T18 (1) : ad. ML (M^u) - sortie sur BUS PA2
T20 : ad. MV2 pour rangt. résultat dans MV2
T20 (1) : ad. M^u - sortie résultat sur BUS PA2
T7 (1) : ad. M^m (opérandes)
T7 (2) : transfert opérandes dans tampons T1 T2
T7 (3) : multiplication : résultat dans TP3
T7 (4) : ad. M^u
T7 (5) : transfert de TP3 dans M^u
T8, T9 et T 10 : idem
T11 : adressage M^u
addition/soust. résultat dans M^u
T12, T13, T14, T15 et T16 : idem
T21 : sortie sur BUS PA1 (résultat)
T22 : sortie sur BUS PA2 (résultat).

CONCLUSION

Nous venons de présenter les différentes étapes du système de "conception de systèmes numériques à hautes performances".

Ce système présente une grande souplesse à deux niveaux.

- L'opérateur construit devient une opération manipulable par le programmeur (faisant partie du champ de commande de la micro-instruction).

- Les opérateurs sont micro-programmés et donc facilement modifiables.

D'autre part, toute modification de l'architecture peut être répercutée sur l'opérateur (mode de construction semi-interactif).

Enfin, tous les niveaux décrits sont automatisables (FORTRAN) et faciles à mettre en oeuvre.

BIBLIOGRAPHIE

1. R. Valette : "Sur la description, l'Analyse et la validation des systèmes de commande parallèles", thèse d'état, univ. Paul Sabatier, Toulouse, novembre 1976.
2. Baer, T.L. and D.P. Bovet, "Compilation of arithmetique expressions for parallel computations", IFIPS congress 68, august 1968, pp. B4 - B10.
3. G. GIRAULT. Réseaux de Petri et synchronisation de processus. Publication Institut de Programmation n° 78-02-1978.
4. Cooley, J.W. and J.W. Tukey, "An Algorithm for the Machine Calculation of Complex Fourex Series", Math. of Comput., Vol 19, April 1965, pp. 297 - 301
5. Romchandoni, C. "Analysis of asynchronons concurent systems by timed Petri nets". Ph. D thesis, Dept. Electrical Engineering - MIT, Cambridge, Mass. 1974
6. M.J. Gonzales, C.V. Ramamoorthy. Recognition and representation of parallel processable streams in computer programs. Parallel Processor Systems, technologies and Applications. L.C. HOBBS ed. Spartan books 1970.
7. C.L. KWAN, P. Le Beux, C. Michel. The design of structured digital systems controlled by PLA. Euromicro - Amsterdam 1977.

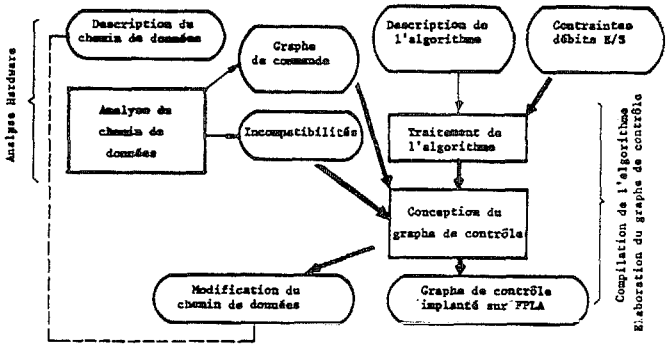


Figure 1 - Conception d'opérateurs spécialisés

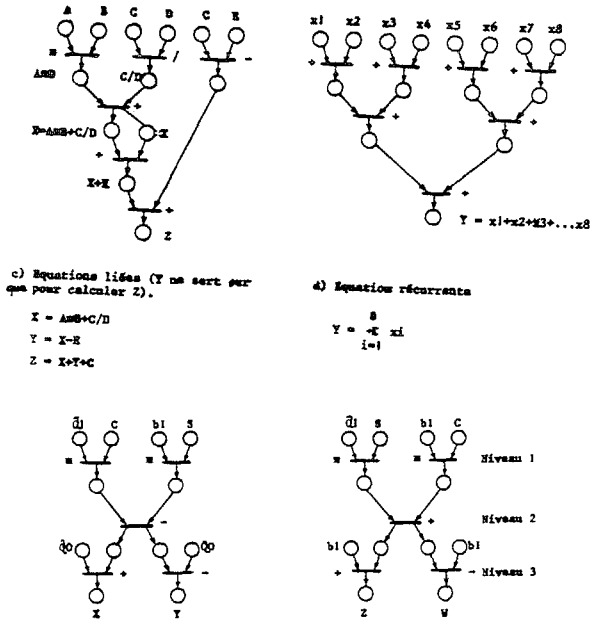


Figure 5 - Opération Papillon

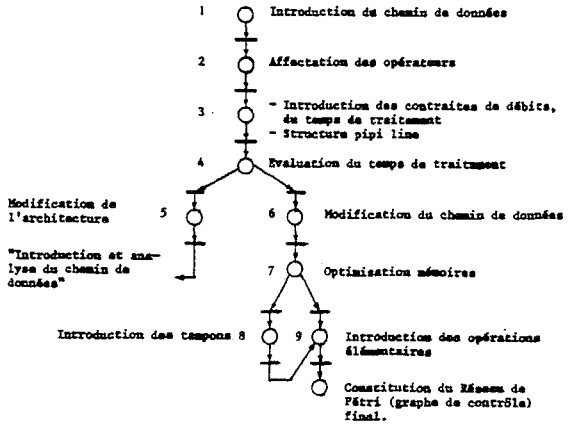
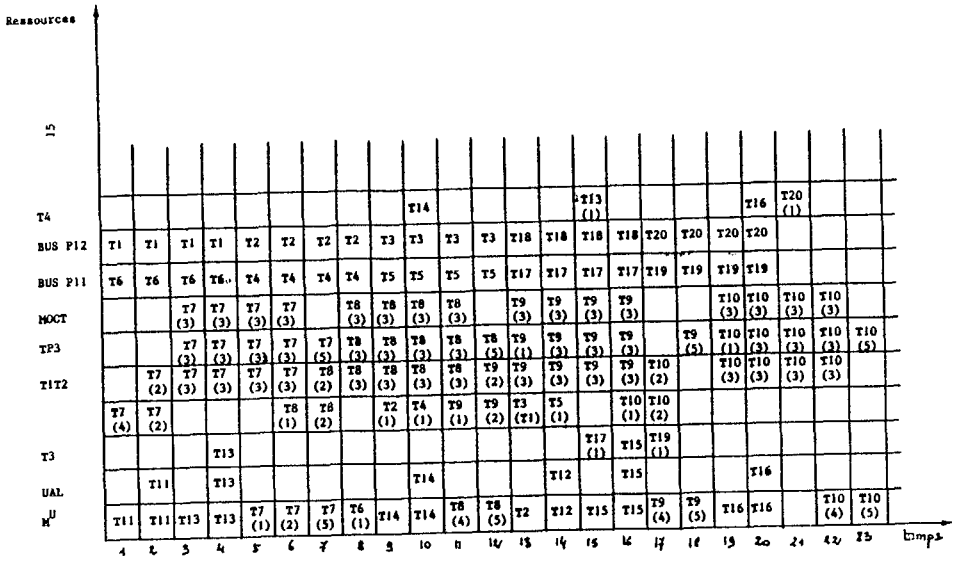


Figure 6 - Etapes de conception du graphe de contrôle

PROGRAMMES DE CONCEPTION DES MICROSYSTEMES NUMERIQUES
A HAUTES PERFORMANCES



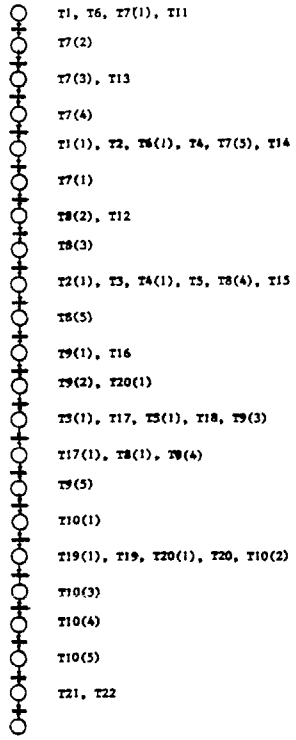
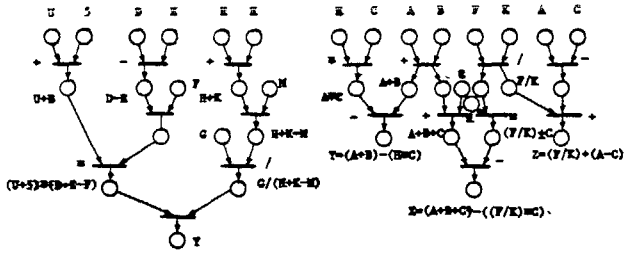


Figure 13 - RdP implémentable sur FPLA



a) Y = ((U+5)*(D-E)-F)*(G/((H+K)-M)) b) X = (A+B+C)-((F/K)*E)
Y = (A+B)-(E*C)
Z = (F/K)+(A-C)

Figure 3 - Algorithme général de transformation de programme séquentiels en programmes parallèles (modélisé en Réseau de Pétri)

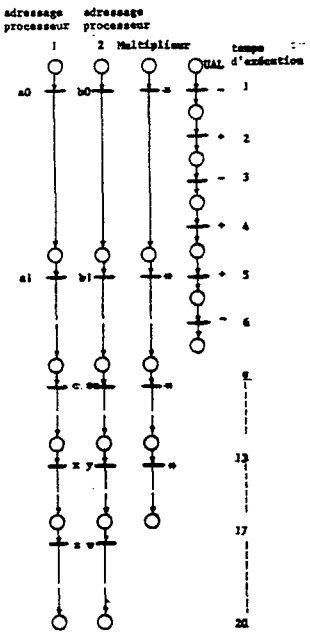


Figure n° 8 - Séquencement des opérations

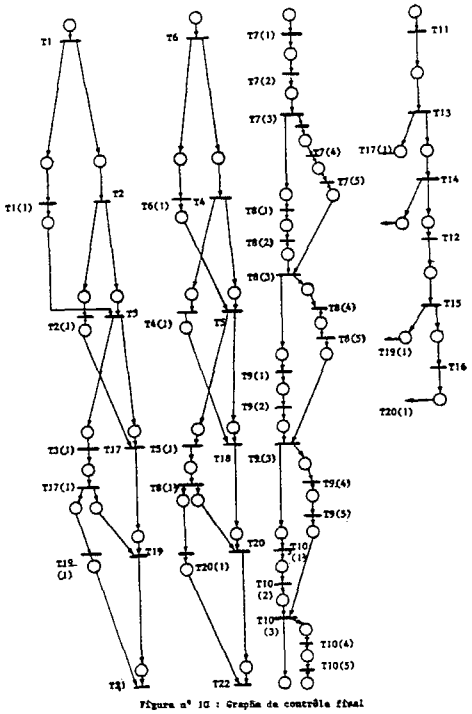


Figure n° 10 : Graphe de contrôle final