



(22) Date de dépôt/Filing Date: 2001/05/22

(41) Mise à la disp. pub./Open to Public Insp.: 2001/11/23

(30) Priorité/Priority: 2000/05/23 (00 06 569) FR

(51) Cl.Int.⁷/Int.Cl.⁷ G06F 15/00, G06F 15/76, G06F 11/30

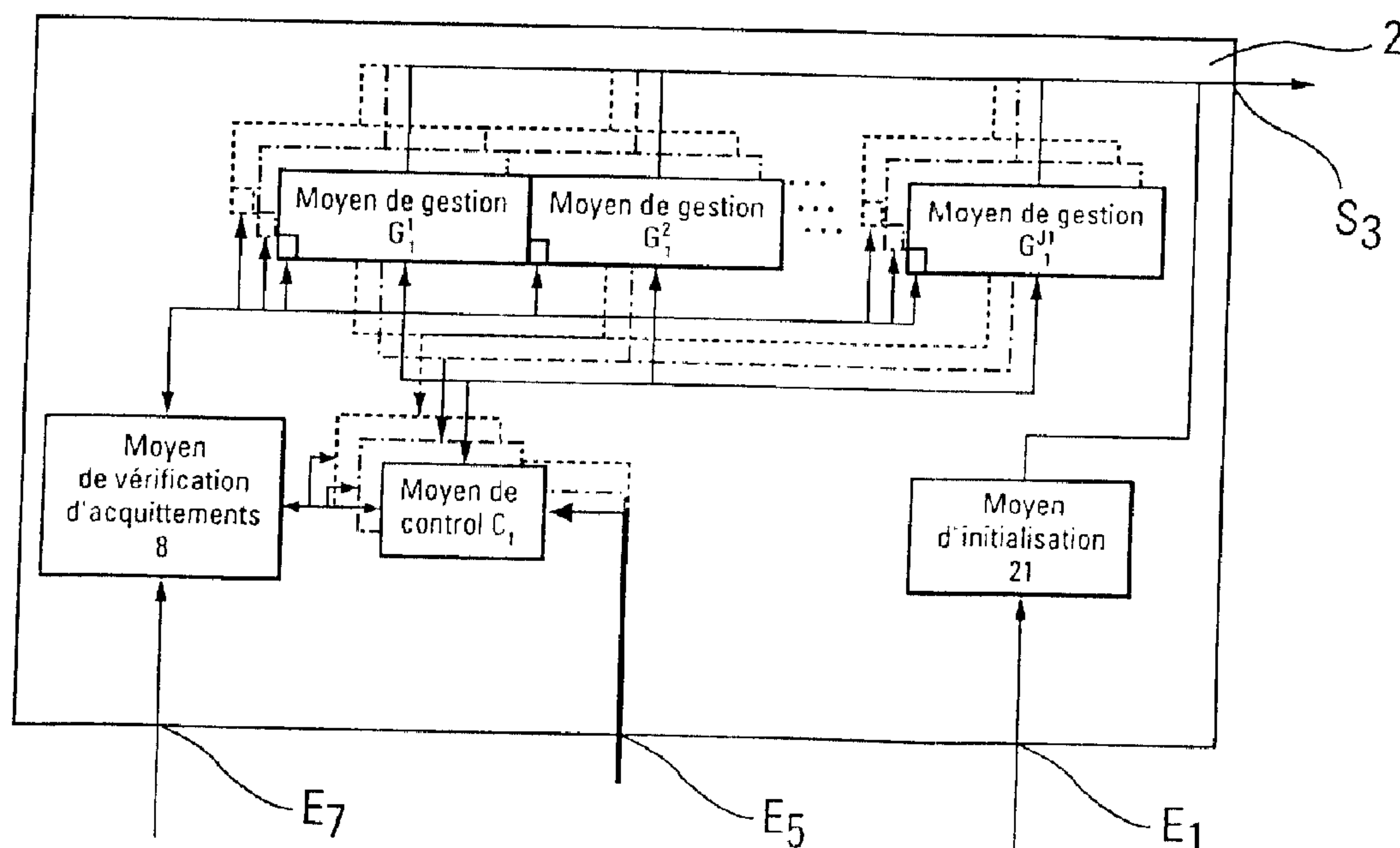
(71) Demandeur/Applicant:
THOMSON-CSF, FR

(72) Inventeurs/Inventors:
BONNOT, PHILIPPE, FR;
LENORMAND, ERIC, FR

(74) Agent: ROBIC

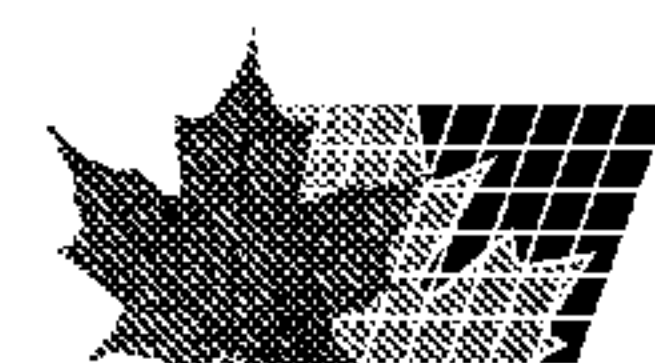
(54) Titre : SYSTEME ET METHODE DE GESTION D'UNE ARCHITECTURE MULTI-RESSOURCES

(54) Title: MULTI-RESOURCE ARCHITECTURE MANAGEMENT SYSTEM AND METHOD



(57) Abrégé/Abstract:

L'invention concerne le domaine des circuits intégrés programmables conçus pour des applications à forte puissance de calcul. Elle propose un système peu complexe et de faible volume permettant de mettre en oeuvre un schéma de planification statique de gestion temps réel des ressources capable de prendre en compte l'acquittement d'une tâche. Ce système, permettant la gestion d'une architecture multi-ressources, dans laquelle plusieurs tâches peuvent être exécutées simultanément par différentes ressources, comporte ° au moins un moyen de gestion G_j^i l'exécution d'une partie j ($0 \leq j$) d'un ensemble ou sous-ensemble i ($0 \leq i$) d'une ou plusieurs tâches, ° au moins un moyen de contrôle C_i du ou des moyens de gestion G_j^i de l'exécution dudit ensemble ou sous-ensemble i de tâches, ° au moins un moyen 8 de vérification de l'acquittement d'une ou plusieurs tâches données, chaque moyen de vérification d'acquittement 8 étant associé à tout ou partie des moyens de gestion G_j^i /ou tout ou partie des moyens de contrôle C_i et/ou tout ou partie des ressources 6, et/ou ° un moyen d'initialisation 21 permettant d'initialiser tout ou partie des moyens précédents.



ABREGE

L'invention concerne le domaine des circuits intégrés programmables conçus pour des applications à forte puissance de calcul. Elle propose un système peu complexe et de faible volume permettant de mettre en œuvre un schéma de planification statique de gestion temps réel des ressources capable de prendre en compte l'acquittement d'une tâche.

Ce système, permettant la gestion d'une architecture multi-ressources, dans laquelle plusieurs tâches peuvent être exécutées simultanément par différentes ressources, comporte :

- au moins un moyen de gestion G_j^i de l'exécution d'une partie j ($0 \leq j$) d'un ensemble ou sous-ensemble i ($0 \leq i$) d'une ou plusieurs tâches,
- au moins un moyen de contrôle C_i du ou des moyens de gestion G_j^i de l'exécution dudit ensemble ou sous-ensemble i de tâches,
- au moins un moyen δ de vérification de l'acquittement d'une ou plusieurs tâches données, chaque moyen de vérification d'acquittement δ étant associé à tout ou partie des moyens de gestion G_j^i et/ou tout ou partie des moyens de contrôle C_i et/ou tout ou partie des ressources ϕ , et/ou
- un moyen d'initialisation τ_1 permettant d'initialiser tout ou partie des moyens précédents.

SYSTÈME ET MÉTHODE DE GESTION D'UNE
ARCHITECTURE MULTI-RESSOURCES

L'invention concerne le domaine des circuits intégrés programmables de calcul numérique conçus pour des applications à forte puissance de calcul, par exemple: des applications caractérisées par des algorithmes à base de boucles imbriquées dont le déroulement ne dépend pas ou peu des données à traiter. Ce type d'application se rencontre notamment dans les premiers étages de traitement numérique du signal.

10 Les circuits considérés contiennent plusieurs ressources physiques capables d'exécuter un microprogramme mettant en œuvre une fonctionnalité particulière telle que, par exemple, un calcul (exécution d'opérations arithmétiques microprogrammées sur un ensemble de données) ou un transfert de données (acheminement d'un ensemble de données d'une zone de stockage à une autre zone). Ces ressources physiques fonctionnent en parallèle, exécutant chacune un microprogramme de manière séquentielle ou éventuellement en boucle. Deux modes de gestion de l'exécution d'un programme par ces ressources physiques existent.

20 Soit un programme unique est déroulé de façon commune. Une seule instruction, contenant un champ (microinstruction) pour chacune des ressources physiques, est active à chaque cycle élémentaire. C'est le cas des architectures VLIW (abréviation anglo-saxonne de Very Long Instruction Word, traduit en français par mot d'instruction très long).

Soit les microprogrammes sont spécifiques aux ressources physiques, ce qui permet aux ressources d'exécuter à leur rythme et, en particulier, de répéter certaines séquences de manière indépendante.

30 Le second mode est celui choisi par la plupart des processeurs de traitement du signal (DSP, abréviation anglo-saxonne de Digital Signal Processing) car il laisse une plus grande souplesse à l'architecture. La ressource principale est la partie opérative (celle effectuant les opérations arithmétiques). Elle fonctionne à son propre rythme en parallèle avec les ressources de transfert de données, appelés contrôleurs de DMA (abréviation anglo-saxonne de Dynamic Memory Access, traduit en français par "accès dynamique à la mémoire"). De plus, les ressources sont supposées téléchargeables.

Comme le montre la figure 1, chaque ressource est capable de stocker, au départ, un jeu de microcommandes ou microinstructions propre et suffisant à l'exécution de sa tâche. Une intervention externe n'est nécessaire que pour fournir ces microcommandes et pour en initialiser
5 l'exécution. Chaque ressource fonctionne dès lors de manière indépendante par rapport aux autres ressources. Elle peut signaler en retour la fin d'exécution de la tâche pour laquelle elle a été configurée. L'unité de stockage de microinstructions dans chaque ressource peut être vue comme une mémoire cache permettant de réduire le débit moyen d'entrée des
10 microcommandes qui viennent de l'extérieur du circuit (difficulté classique des circuits à forte puissance de calcul –bande passante du programme-).

Afin de permettre l'exécution en parallèle de tâches par plusieurs ressources, un dispositif de gestion de l'exécution des tâches par les ressources suivant un schéma de planification statique (scheduling en terme
15 anglo-saxon) est mis en œuvre. La gestion de l'exécution par les ressources des microprogrammes se fait par séquençement à un ou deux niveaux.

La solution classique, employée en particulier dans les microprocesseurs de traitement du signal (DSP), consiste à inclure des directives d'ordonnancement dans la partie opérative du programme. Le
20 séquençement se fait, alors, seulement à un niveau. Les microprogrammes sont spécifiques à chaque ressource, ce qui leur laisse la liberté d'être exécuter à leur rythme (en particulier, de répéter certaines séquences indépendamment de leurs voisins).

Comme le montre la figure 1, le microséquençeur propre à une
25 ressource donnée déroule, alors, le microprogramme de cette ressource de façon indépendante par rapport aux autres ressources. La plupart des instructions sont destinées à la ressource de calcul. Certaines servent à paramétrer les ressources de transfert. Le principal inconvénient du séquençement à un niveau est la gestion simultanée par les
30 microséquençeurs des différentes ressources de multiples activités produisant chacune des événements (fin d'une itération, fin de boucle...) désynchronisés avec l'activité des autres. D'où, la nécessité de pouvoir interrompre une activité pour en lancer une autre ou prendre en compte l'acquittement d'une tâche, ce qui donne lieu à une logique relativement

complexe (gestion d'interruptions, sauvegarde de contexte...) induisant un coût de conception élevé.

Les applications envisagées par l'invention sont simples, elles laissent donc espérer des réalisations de circuit à hautes performances plus simple. C'est pourquoi, deux niveaux de séquençement distincts sont introduits :

- un niveau de séquençement rapide pour chacune des ressources, ne gérant pas d'événements asynchrones ;
- un niveau de séquençement global contrôlant le téléchargement du microprogramme dans chacune des ressources et leur lancement. L'utilisation de ce deuxième niveau permet une réduction du besoin en débit de microprogramme car le téléchargement peut s'effectuer en parallèle sans ralentir l'activité des ressources). Ce niveau de séquençement global peut-être plus ou moins élaboré.

Une élaboration consiste à dérouler un programme en forçant les instants de lancement des tâches sans prendre en compte les acquittements, il s'agit du lancement direct à des instants prédéterminés. Cette élaboration est plus simple et plus réactive, mais elle ne s'accommode d'aucune variation du temps d'exécution des tâches, et devient problématique dans le cas du multimode.

Une autre élaboration consiste à utiliser une structure de type cœur de processeur, mais il s'agit d'une approche volumineuse et moins réactive.

La présente invention permet de pallier ou, pour le moins, de réduire ces inconvénients en proposant un système peu complexe et de faible volume permettant de mettre en œuvre un schéma de planification statique de gestion temps réel des ressources capable de prendre en compte l'acquittement d'une tâche.

A cet effet, l'invention a pour objet un système permettant la gestion d'une architecture multi-ressources dans laquelle plusieurs ressources peuvent exécuter simultanément différentes tâches, caractérisé en ce qu'il comporte :

- au moins un moyen de gestion de l'exécution d'une partie d'un ensemble ou sous-ensemble d'une ou plusieurs tâches,

- au moins un moyen de contrôle du ou des moyens de gestion de l'exécution dudit ensemble ou sous-ensemble de tâche(s),
- au moins un moyen de vérification de l'acquittement d'une ou plusieurs tâches données, chaque moyen de vérification d'acquittement étant associé à tout ou partie des moyens de gestion et/ou tout ou partie des moyens de contrôle et/ou tout ou partie des ressources,
- et/ou un moyen d'initialisation permettant d'initialiser tout ou partie des moyens précédents.

10

L'invention concerne un circuit intégré programmable caractérisé en ce qu'il comporte une mémoire programme unique comprenant :

- une entrée recevant les signaux d'adresse émis par ledit système et
- une sortie transmettant les tâches à exécuter audit système et aux ressources dudit circuit.

15

Le système de gestion d'une architecture multi-ressources proposé par la présente invention, nommé ordonnanceur par la suite, utilise une méthode de gestion d'une architecture multi-ressources dans laquelle plusieurs ressources peuvent exécuter simultanément différentes tâches caractérisée en ce qu'elle comporte au moins les étapes suivantes:

20

- la gestion de l'exécution d'au moins une partie d'un ensemble ou sous-ensemble d'une ou plusieurs tâches,
- le contrôle de la gestion de chaque partie d'au moins un ensemble ou sous-ensemble d'une ou plusieurs tâches,
- la vérification de l'acquittement de l'exécution d'au moins une tâche lors de la gestion et/ou du contrôle et/ou de l'exécution par les ressources,

25

l'initialisation de tout ou partie des étapes précédentes.

30

Les caractéristiques et avantages de l'invention apparaîtront plus clairement à la lecture de la description, faite à titre d'exemple, et des figures s'y rapportant qui représentent :

- Figure 1, un schéma de base d'une architecture multi-ressources selon l'état de l'art,

- Figure 2, un exemple de schéma de base d'une architecture multi-ressources à deux niveaux de séquençement selon l'invention,
- Figure 3, un exemple de schéma de principe de l'ordonnanceur selon l'invention,
- 5 - Figure 4, un exemple d'architecture de l'ordonnanceur selon l'invention,
- Figure 5, un exemple de réalisation détaillé d'une partie de l'ordonnanceur de la Figure 4,
- Figure 6, un exemple du principe de fonctionnement d'un ordonnanceur selon l'invention,
- 10 - Figure 7, un exemple de fonctionnement détaillé de l'ordonnanceur lors de l'exécution d'une directive d'ordonnancement,
- Figure 8, un exemple de fonctionnement détaillé de l'ordonnanceur lors du passage à l'itération suivante.

15 La figure 2 donne un exemple de schéma de base d'une architecture multi-ressources selon l'invention. L'ordre d'initialisation 1 induit l'émission par l'ordonnanceur 2 d'un signal 3 contenant une adresse vers une mémoire programme unique 4. Le code stocké dans la mémoire de programme 4 est formé à partir :

- 20
- de séquences de code spécifiques aux ressources auxquelles elles sont destinées,
 - de directives d'ordonnancement qui reflètent le schéma de planification des activités et destinées à l'ordonnanceur 2.

25 La tâche 5 se trouvant à l'adresse 3 est, donc, exécutée soit par l'une des ressources 6 s'il s'agit d'une instruction de traitement, soit par l'ordonnanceur 2 s'il s'agit d'une directive d'ordonnancement. Lorsqu'une ressource 6 a terminé l'exécution de l'instruction, elle en signale l'acquittement 7 à l'ordonnanceur 2.

30 La figure 3 présente un exemple de schéma de principe de l'ordonnanceur 2 selon l'invention. L'ordre d'initialisation 1 est reçu via l'entrée E_1 par le moyen d'initialisation 21 qui initialise l'ensemble des moyens de l'ordonnanceur 2 (non représenté sur la figure) et émet la première adresse 3 sur la sortie adresse S_3 . La tâche à exécuter 5 lue à cette adresse 3 est reçue par les ressources et l'ordonnanceur 2 sur son

entrée E_5 et transmise aux moyens de contrôle C_i des moyens de gestion G_i^j de l'exécution d'une ou des parties j d'un ensemble ou sous-ensemble de tâche(s) i . Les tâches 5 contiennent, par exemple, soit l'indication de leur destinataire, soit l'indication de type (calcul, transfert, ordonnancement). Lors de l'exécution d'une partie j d'un ensemble ou sous-ensemble de tâche(s) i , le moyen de gestion G_i^j émet une adresse 3 sur la sortie S_3 . Un moyen de vérification d'acquittement 8 vérifie si l'acquittement attendu par le moyen de gestion G_i^j , c'est à dire la fin d'exécution d'une ou plusieurs tâches attendue par le moyen de gestion G_i^j , est l'acquittement signalé par l'une des ressources 6 ou l'un des moyens de gestion G_i^j ou l'un des moyens de contrôle C_i .

Vu d'une ressources particulière 6, celle-ci télécharge les microinstructions qui lui parviennent, et déclenche leur exécution dès que l'ordre de lancement lui parvient. Ces tâches 5 (microinstructions et ordre de lancement) sont lus séquentiellement dans la mémoire programme 4 à partir de l'adresse transmise par l'ordonnanceur 2. Le rôle de l'ordonnanceur 2 est de gérer la suite d'adresses correspondante le bon nombre de fois, s'il s'agit d'une boucle, et aux moments opportuns. Les directives d'ordonnancement lui permettant d'initialiser et de gérer en parallèle autant de pointeurs d'adresses programme qu'il y a de partie j dans l'ensemble ou sous-ensemble de tâches i .

Prenons l'exemple de la gestion d'une architecture multi-ressources par un ordonnanceur 2 pour l'exécution d'un ensemble i de tâches formant une boucle i de programme. Un moyen de gestion G_i^j d'une partie j d'un ensemble ou sous-ensemble i de tâches est alors un moyen de gestion de l'exécution de la partie ou étage j d'une boucle principale ou imbriquée i de programme, moyen de gestion nommé par la suite COUCHE G_i^j . Le moyen de contrôle C_i des COUCHES G_i^j est nommé par la suite NOYAU C_i . A chacun des NOYAUX C_i correspond un niveau i de boucle différent. Il y a donc plusieurs NOYAUX C_i en niveaux hiérarchiques pour les boucles imbriquées.

La figure 4 propose un exemple d'architecture de l'ordonnanceur 2 appliqué à la gestion de l'exécution d'une boucle i de programme. La figure 5 donne un exemple de réalisation détaillé d'un NOYAU C_i , des COUCHES G_i^j associées de l'architecture proposée par la figure 4.

Pour la gestion de l'exécution du nombre d'itération N_i de la boucle i conformément aux paramètres d'initialisation de la boucle i , le NOYAU C_i comporte un compteur d'itération it_i qui est initialisé au nombre d'itération de la boucle i à effectuer $it_i = N_i$ et décrémente $it_i = it_i - 1$ à chaque nouvelle
 5 itération de la boucle i . Lorsque le compteur it_i atteint la valeur 0 et dès que l'arbitre 27^+ d'accès au bus d'acquittement 27 autorise l'accès, l'émetteur d'acquittement em_i émet un signal d'acquittement pour signifier la fin du traitement de la boucle i .

L'autorisation d'accès au bus d'acquittement 27 délivré par l'arbitre
 10 27^+ donne droit à l'utilisation du bus d'acquittement 27 pendant un cycle pour émettre un signal d'acquittement 7. Dans le cas de plusieurs demandes simultanées, dues au fait que chaque ressource 6 et chaque NOYAU C_i ayant terminée sa tâche veut envoyer un signal d'acquittement 7, l'autorisation de l'arbitre 27^+ est fonction d'un ordre de priorité qui peut être
 15 fixe.

L'ensemble des COUCHES G_i^j associées à un NOYAU C_i constitue une chaîne de registre à décalage. Le rôle du NOYAU C_i est de gérer une boucle i , de contrôler le pipeline associé (la chaîne formée par les COUCHES G_i^j branchées en cascade) et de synchroniser les COUCHES G_i^j
 20 entre elles. Pour cela, chaque NOYAU C_i comporte un mécanisme d'avance pipeline c_i . L'ensemble NOYAU C_i et COUCHES G_i^j associées est appelé SEQPAR.

Les informations que comporte chaque registre de cette chaîne sont un indicateur d'état e , un compteur d'adresse programme ad , un
 25 indicateur de l'état d'avancement du pipeline a et un indicateur de dimension (non représenté sur les figures).

L'ordonnanceur 2 comporte, alors, un ensemble de registres d'activité a_i^j qui indique l'état d'avancement de l'exécution dans les étages j de la boucle i . Seules les COUCHES G_i^j validées par le nombre d'étages J_i
 30 de la boucle i peuvent être activées.

L'ordonnanceur 2 comporte, aussi, un ensemble de compteurs d'adresse ad_i^j incrémentables qui pilotent chacun l'exécution d'un étage j de la boucle i . Chaque COUCHE G_i^j gère une suite d'adresse $A[j]_i$ permettant ainsi aux étages j de la boucle i de s'exécuter en parallèle. Au sein d'une
 35 COUCHE G_i^j , le déroulement du programme s'effectue par incrémentation

du compteur d'adresse $ad_i^j = ad_i^j + 1$. L'incrémentation du compteur d'adresse $ad_i^j = ad_i^j + 1$ s'effectue lors de l'émission du signal 3 contenant l'adresse $A[j]_i$ du compteur ad_i^j sur le bus d'adresse programme 23. L'adresse de base $A0_i$ vient initialiser le compteur d'adresse $ad_i^0 = A0_i$ de la première COUCHE G_i^0 ,
 5 les compteurs d'adresse ad_i^j des autres COUCHES G_i^j sont initialisés par l'adresse $A[j-1]_i$ contenue dans le compteur d'adresse ad_i^{j-1} de la COUCHE précédente G_i^{j-1} au début d'une nouvelle itération.

L'indicateur d'état e_i^j indique si la COUCHE G_i^j est en repos, en émission, en attente ou en "fin" de traitement. Lorsqu'une COUCHE G_i^j est
 10 active et son registre d'état e_i^j est en émission, l'arbitre 23^+ du bus adresse 23 note une demande d'accès à ce bus 23 pour cette COUCHE G_i^j .

L'autorisation d'accès au bus d'adresse 23 délivré par l'arbitre 23^+ donne droit à l'utilisation du bus d'adresse 23 tant que le demandeur maintient sa demande. Dans le cas de plusieurs demandes simultanées,
 15 dues au fait que chaque COUCHE G_i^j peut vouloir émettre un signal d'adresse 3, l'autorisation de l'arbitre 23^+ est fonction d'une ordre de priorité qui peut être fixe. La COUCHE G_i^j ayant obtenue l'autorisation d'accès est dite en phase d'émission.

(Mode d'observation) Lorsqu'une COUCHE G_i^j est active et son
 20 registre d'état e_i^j est en attente, le détecteur d'acquittement d_i^j de cette COUCHE G_i^j observe les signaux d'acquittements 7 émis sur le bus d'acquittement 27 par les émetteurs d'acquittements em_i des NOYAUX C_i de l'ordonnanceur 2 et les ressources. Lorsque le signal d'acquittement 7 reçu correspond au code d'acquittement attendu, le registre d'état e_i^j de la
 25 COUCHE G_i^j sort de l'état d'attente.

(Mode de mémorisation) Lorsqu'un signal d'acquittement 7 est reçu par le détecteur d'acquittement d_i^j de la COUCHE G_i^j alors que son registre d'état e_i^j n'est pas en attente, le détecteur d'acquittement d_i^j mémorise le signal d'acquittement 7 reçu pour le prendre en compte lorsque
 30 le registre d'état e_i^j se mettra en attente.

Quand toutes les COUCHES G_i^j d'un NOYAU C_i ont leurs registres d'état e_i^j en "fin" de traitement, le mécanisme avance pipeline c_i déclenche le décalage des registres de la chaîne associée à ce NOYAU C_i selon l'horloge 29 fournie par ce NOYAU C_i et décrémente le compteur
 35 d'itération it_i de ce NOYAU C_i . Le contenu d'un registre est, donc, passé à la

COUCHE suivante G_i^{j+1} lorsque le pipeline i (la chaîne de registre à décalage associée à un NOYAU C_i) avance.

Il est ainsi possible de gérer l'exécution de la boucle i de programme en faisant suivre les indications d'adressage des données
 5 (contenues dans le registre indicateur de dimension non représenté) et du programme $ad_i^{j+1} = ad_i^j$ tout au long de cette chaîne. Ces indications peuvent aussi, de cette manière, être spécifiques à une récurrence donnée (programmation multimode). De ce fait, les codes liés à une itération sont forcément séquentiels. Une fois l'adresse de base $A0_i$ adressée,
 10 l'ordonnanceur 2 contrôle toutes les exécutions sans intervention externe.

Puisqu'il ne comporte que des registres et compteurs chaînés, l'ordonnanceur 2 proposé utilise un faible volume de matériel. De plus, puisqu'il ne comporte qu'un bus de tâche 25, le débit des tâches 5 nécessaire pour l'ordonnancement est peu élevé.

15 Afin d'initialiser l'exécution de la boucle i de programme, le moyen d'initialisation 21 reçoit l'ordre 1 d'initialiser les moyens de l'ordonnanceur 2 comme décrit dans l'étape (S1) du principe de fonctionnement de la figure 6, principe donné à titre d'exemple. Puis ce moyen d'initialisation 21 émet sur le bus d'adresse 23 un signal 3 contenant la première adresse issue d'un
 20 registre préalablement chargé (étape S2). L'ordonnanceur 2 reçoit la première tâche 5 lue à cette première adresse, qui comporte une directive d'ordonnancement c'est-à-dire une instruction destinée à l'ordonnanceur (étape S3 et S4). L'ordonnanceur 2 exécute alors cette directive (S5).

25 Dans notre exemple de réalisation, les directives d'ordonnancement qui reflètent le schéma de planification des activités sont au nombre de quatre :

- l'instruction d'initialisation de boucle: elle configure un des NOYAUX C_i car elle initialise le registre d'activité a_i^0 en mode actif et comporte les paramètres d'initialisation de la boucle i (adresse de base $A0_i$ et/ou
 30 nombre d'itérations N_i et/ou nombre d'étages J_i de la boucle et/ou le niveau d'imbrication i de la boucle, et/ou le ou les codes d'acquiescement de la boucle i et/ou des étages j de la boucle i ...),
- l'instruction d'attente: elle identifie l'attente de la fin d'exécution d'une tâche et comporte, par exemple, le code d'acquiescement attendu. Elle
 35 permet donc d'attendre qu'une tâche se termine et que la ressource 6

exécutant cette tâche soit libre avant d'exécuter la tâche suivante d'un même étage j . Elle peut aussi servir à programmer une attente entre tâches de deux étages j et j' différents d'une même boucle i ou de deux boucles imbriquées i et i' . Une attente de ce type se justifie, par exemple, par une dépendance de données entre les tâches ou encore par l'utilisation d'une ressource 6 commune.

- l'instruction frontière entre deux étages j et $j+1$: elle peut comporter le code d'acquittement de l'étage j , par exemple, et
- l'instruction fin de boucle: elle indique la fin de l'exécution du dernier étage J_i-1 d'une boucle i .

Les directives décrites ci-dessus à titre d'exemple comportent des paramètres associés et peuvent donner lieu à plusieurs mots consécutifs de la mémoire de programme 4. Une variante consisterait à utiliser pour ces 3 derniers types de directives un champ inséré dans les instructions d'initialisation et les instructions de traitement.

Les instructions de traitement comportent les paramètres nécessaire à la tâche (paramètres de configuration ou microinstructions de microprogramme). En effet, des informations doivent être fournies à la ressource 6 avant l'exécution. Ces informations peuvent être des paramètres (nombre d'itérations d'une boucle, paramètres d'adresse...) ou un microprogramme que la ressource 6 utilise éventuellement pour commander l'exécution. La programmation, dans ce dernier cas, est hiérarchique car le programme fait appel à des tâches qui utilisent elles-mêmes un microprogramme.

La figure 7 présente un exemple de principe de fonctionnement de l'ordonnanceur 2 lors de l'exécution de ces directives d'ordonnancement.

(DEBUT) La directive lue à la première adresse est une instruction d'initialisation de la boucle 0. L'exécution de cette instruction induit la sélection d'un NOYAU C_0 de l'ordonnanceur 2 (étape S53d). Le registre d'activité a_0^0 se met, alors, en mode actif et tous les autres des registres d'activité a_0^j ($j \neq 0$) en mode inactif (étape S54d). Et, le mécanisme avance pipeline c_0 du NOYAU C_0 reçoit les paramètres d'initialisation de la boucle 0 (étape 55d). Ce mécanisme avance pipeline c_0 initialise le compteur d'itération au nombre d'itération à effectuer par la boucle 0 ($it_0=N_0$) et met au repos les registres d'état e_0^j des COUCHES G_0^j associées au NOYAU C_0 .

Puis il transmet l'adresse de base au compteur d'adresse de la COUCHE G_0^0 ($ad_0^0 = A0_0$) (étape S56d), ce qui induit le passage du registre d'état e_0^0 de cette COUCHE G_0^0 en émission et le passage du détecteur d'acquiescement d_0^0 de cette COUCHE G_0^0 en mode de mémorisation (étape S57d).

- 5 (EMISSION) La COUCHE G_0^0 est alors prête à émettre sur le bus d'adresse 23 l'adresse $A0_0$ contenue dans son compteur d'adresse ad_0^0 (étape S8e) dès qu'elle en sera autorisée par l'arbitre 23⁺ d'accès à ce bus 23 (étape S7e). L'instruction contenue à l'adresse émise $A0_0$ est lue et exécutée par la ressource 6 ou l'ordonnanceur 2 auquel elle est destinée. Le
10 compteur d'adresse ad_0^0 s'incrémente $ad_0^0 = ad_0^0 + 1$ (étape S9e). Tant que l'instruction lue n'est pas une directive d'ordonnancement (étape S3-S4), le registre d'état e_0^0 reste en émission (étape S6), la COUCHE G_0^0 conserve son accès au bus 23, une nouvelle adresse $A[0]_0$ contenu dans le compteur d'adresse ad_0^0 est émise (étape S8e) et le compteur d'adresse ad_0^0
15 incrémentée (étape S9e).

- (ATTENTE) Si l'instruction lue est la directive d'ordonnancement "instruction d'attente" (étapes S3 à S5), le registre d'état e_0^0 de la COUCHE G_0^0 passe en attente et son détecteur d'acquiescement d_0^0 en mode d'observation. Dès la réception du code d'acquiescement attendu par la
20 COUCHE G_0^0 du NOYAU C_0 , la COUCHE G_0^0 passe en état d'émission, ce qui indique qu'elle demande à nouveau l'accès au bus d'adresse programme 23. Le processus se déroule alors de la même façon que lors de la première demande d'accès au bus d'adresse 23 par la COUCHE G_0^0 .

- (FRONTIERE) Si l'instruction lue est la directive d'ordonnancement "instruction frontière entre deux étages" (étapes S3 à S5),
25 le registre d'état e_0^0 de la COUCHE G_0^0 passe en "fin" de traitement.

- (FIN DE TRAITEMENT) La COUCHE G_0^0 étant la seule active (soit parce que la boucle ne comporte qu'un seul étage 0, soit parce que les autres COUCHES G_0^j ($j \neq 0$) ne sont pas activées), le mécanisme d'avance
30 pipeline c_0 gère, alors, le passage à l'itération suivante (étape S8f). Pour cela, il décrémente le compteur d'itération $it_0 = it_0 - 1$ (étape S81f) et met au repos les registres d'état e_0^j des COUCHES G_0^j associées au NOYAU C_0 . Puis il décale les registres d'activités $a_0^j = a_0^{j-1}$ ($j \geq 1$) et, pour les COUCHE G_0^{j-1} actives, les compteurs d'adresse $ad_0^j = ad_0^{j-1}$ ($j \geq 1$) (étape S82f). Enfin, si

- le compteur d'itération it_0 a atteint la valeur 0, le registre d'activité a_0^0 de la première COUCHE G_0^0 du NOYAU C_0 est inactivé, ce qui signifie que cette COUCHE G_0^0 est dans une phase fin partielle de la boucle (étape S83f-S84f). Sinon, le registre d'activité a_0^0 de la première COUCHE G_0^0 du NOYAU
- 5 C_0 est activé (étape S83f-S84f) et le mécanisme d'avance pipeline c_0 transmet l'adresse de base $A0_0$ au compteur d'adresse ad_0^0 de la première COUCHE G_0^0 (étape S85f). Les registres d'état e_0^j des COUCHES G_0^j active passe en émission et les détecteurs d'acquittements d_0^j de ces COUCHES G_0^j en mode de mémorisation (étape S86f).
- 10 Si la boucle 0 ne comporte qu'un seul étage 0 et que la COUCHE G_0^0 est en phase de fin partielle de la boucle, c'est à dire inactive et en état de fin traitement dans notre exemple, le NOYAU C_0 entre dans une phase de fin de boucle comportant l'émission du signal 7 d'acquittement de la boucle 0 par l'émetteur d'acquittement em_0 sur le bus d'acquittement 27 après accord
- 15 de l'arbitre 27^+ (étape S9f-S10f).
- Si la boucle 0 comporte plusieurs étages j , la deuxième COUCHE G_0^1 est alors active et le compteur d'adresse ad_0^1 de cette COUCHE G_0^1 contient la dernière adresse $A[0]_0$ calculée par le compteur d'adresse ad_0^0 de la première COUCHE G_0^0 . Si la boucle comporte plusieurs itérations, la
- 20 première COUCHE G_0^0 est toujours active et le compteur d'adresse ad_0^0 de cette COUCHE G_0^0 contient l'adresse de base $A0_0$.
- (EMISSION) Les deux COUCHES G_0^0 et G_0^1 sont en état d'émission, c'est à dire demandeur d'accès au bus d'adresse 23. L'arbitre 23^+ autorise, par exemple, prioritairement la deuxième COUCHE G_0^1 (ordre de
- 25 priorité allant dans ce cas de la dernière à la première COUCHE d'un NOYAU C_i). Le compteur d'adresse ad_0^1 de cette COUCHE G_0^1 émet alors son adresse $A[1]_0$ et s'incrémente $ad_0^1 = ad_0^1 + 1$ (étape S7e à S9e). L'instruction lue à l'adresse émise $A[1]_0$ est alors la première instruction du deuxième étage de la boucle 0 destinée à une des ressources 6 ou à
- 30 l'ordonnanceur 2.
- Tant que l'instruction lue n'est pas une directive d'ordonnancement (étape S3-S4), le registre d'état e_0^1 reste en émission (étape S6), la COUCHE G_0^1 conserve, donc, son accès au bus 23, une

nouvelle adresse $A[1]_0$ est émise (étape S8e) et le compteur d'adresse ad_0^1 incrémentée (étape S9e).

(ATTENTE) Si l'instruction lue est la directive d'ordonnancement "instruction d'attente" (étapes S3 à S5), le registre d'état e_0^1 de la COUCHE G_0^1 passe en attente et son détecteur d'acquittement d_0^1 en mode d'observation (étape S51a). L'arbitre 22 accorde alors l'autorisation d'accès au bus d'adresse 23 à la première COUCHE G_0^0 qui était toujours demandeur. Dès la réception du code d'acquittement attendu par la deuxième COUCHE G_0^1 du NOYAU C_0 , la COUCHE G_0^1 passe en état d'émission (étape S7a-S8a). Elle demande donc, à nouveau, l'accès au bus d'adresse 23. Le processus se déroule alors de la même façon que pour la première itération.

(FRONTIERE) Si l'instruction lue est la directive d'ordonnancement "instruction frontière entre deux étages" (étapes S3 à S5), le registre d'état e_0^j ($j = 0,1$) de la COUCHE G_0^j ($j = 0,1$) en phase d'émission, c'est à dire en état d'émission et ayant l'autorisation d'accès au bus d'adresse 23, prend la valeur "fin" de traitement (étape S51f).

(FIN DE TRAITEMENT) Lorsque tous les registres d'état e_0^j ($j = 0,1$) des COUCHES actives G_0^j ($j = 0,1$) sont en "fin" de traitement, le mécanisme d'avance pipeline c_0 peut passer à l'itération suivante (étape S7f-S8f). Pour cela, il décrémente le compteur d'itération $it_0 = it_0 - 1$ (étape S81f), décale les registres d'activités $a_0^j = a_0^{j-1}$ ($j \geq 1$) et, pour les COUCHE G_0^{j-1} actives, les compteurs d'adresse $ad_0^j = ad_0^{j-1}$ ($j \geq 1$) (étape S82f).

Si toutes les COUCHES G_0^j ($j = 0,1$) sont en phase de fin partielle, c'est à dire que le compteur d'itération it_0 a atteint la valeur 0 et toutes les COUCHES G_0^j ($j = 0,1$) sont inactives, l'exécution de la boucle 0 est terminée. Le NOYAU C_0 est alors dans une phase de fin de boucle qui comporte l'émission du signal d'acquittement de la boucle 0 sur le bus d'acquittement 27 par l'émetteur d'acquittement em_0 après autorisation de l'arbitre 27⁺.

Nous avons volontairement limité l'exemple précédent à le contrôle par un NOYAU C_i de deux COUCHES G_i^j ($j = 0,1$) mais il est entendu qu'un NOYAU C_i peut contrôler sur le même schéma que celui donné dans cet exemple une seule COUCHE ou plus de deux COUCHES G_i^j ($j \in \mathbb{N}$).

(DEBUT) Lorsqu'un programme contient des boucles imbriquées, l'instruction lue à l'adresse émise $A[j]_i$ par une COUCHE G_i^j peut être une

instruction d'initialisation de boucle (étape S4-S5). Dans ce cas, ladite COUCHE G_i en phase d'émission, c'est à dire en état d'émission et ayant accès au bus d'adresse programme, passe en état d'attente et son détecteur d'acquiescement d_i en mode d'observation du signal d'acquiescement du

5 nouveau NOYAU C_i ($i \neq 0$) (étape 51d-53d).

Ce nouveau NOYAU C_i peut être sélectionné parmi ceux disponible, par exemple, en fonction des données contenues dans le champ NUMSEQ de l'instruction d'initialisation de la boucle i . L'instruction d'initialisation de boucle fonctionnera de la même façon pour la boucle
10 imbriquée i que pour la boucle principale 0. Le nouveau NOYAU C_i pourra ainsi mettre en œuvre plusieurs COUCHES G_i et fonctionner de la même façon que le NOYAU C_0 contrôlant la gestion de l'exécution de la boucle principale 0.

Les fonctionnalités de l'ordonnanceur 2 selon l'invention sont multiples, il peut donc, par exemple, gérer l'exécution d'une suite de tâches:

- *en multimode*. En effet, l'ordonnanceur 2 peut fonctionner en multimode car le moyen d'initialisation 21 peut permettre d'insérer une itération d'une boucle i d'un programme différent de celui de la boucle i qu'il était en train de gérer car le fonctionnement multimode comporte
20 comme sont nom l'indique plusieurs modes. Pour cela, il suffit, par exemple, que le moyen d'initialisation fournisse une adresse $A0_0$ différente correspondant à une boucle i au lieu de la boucle i (cf l'étape S85f). Un mode correspond à une chaîne de traitement fixée à l'avance qui est appliquée sur un ensemble de données pendant un
25 intervalle de temps appelé récurrence. Les récurrences successives peuvent supporter des modes différents, par exemple :

	Récurrence N :	Récurrence N+1 :
	{	{
30	Boucle 0	Boucle 2
	Boucle 1	}
	}	

- *par anticipation*. La fonction d'anticipation de l'ordonnanceur 2 permet
35 d'anticiper la programmation des ressources 6, c'est à dire éviter

d'attendre la fin de l'exécution 7 d'une tâche 5 dans une ressource 6' pour programmer la tâche suivante pour la ressource suivante 6". Cela permet, en effet, alors qu'une tâche 5 vient d'être lancée, de programmer la ressource 6" pour la tâche suivante. La ressource 6" doit pour cela accepter deux zones de mémoire pour son programme. Pour programmer la première tâche pour chaque COUCHE G_i^j lors de l'itération précédente, la chaîne de registre d'activité peut, par exemple, être doublée et tâche définie.

- 5

10

15

20

25

30

• *en tenant compte de l'itération pour accéder aux données.* Un compteur d'itération multidimensionnel peut permettre d'associer à chaque itération des accès à des tableaux de données multidimensionnels. C'est un exemple d'utilisation d'un compteur d'itération it_i pour calculer une adresse mémoire permettant d'accéder à des données différentes en fonction des itérations de la boucle. La valeur du compteur multidimensionnel $it_i[j]$ peut alors être passée de COUCHE G_i^j en COUCHE G_i^{j+1} grâce à un registre à décalage de la même que les registres d'activités a_i^j et les compteurs d'adresse ad_i^j sont décalés pour que la valeur corresponde bien à l'avancement de chaque COUCHE G_i^j de la chaîne dans la boucle i .

• *en associant un buffer de données à un NOYAU C_i .* Le compteur d'itération it_i de chaque NOYAU C_i permet d'avoir une gestion de buffer de données associé au pipeline (à la chaîne composée par le NOYAU C_i et les COUCHES G_i^j associées), ce qui signifie que la réservation d'une zone de mémoire pour chaque itération de la boucle i gérée par l'ordonnanceur 2 est possible. Lorsque l'itération est terminée, la zone mémoire peut être libérée pour une nouvelle itération soit en calculant la valeur du compteur d'itération it_i modulo le nombre de COUCHES G_i^j utilisées, soit en utilisant un compteur tournant au même rythme que le compteur d'itération it_i du NOYAU C_i , dont la valeur est passée de COUCHE G_i^j en COUCHE G_i^{j+1} par un registre à décalage.

La liste des fonctionnalités présentées ci-dessus n'est pas limitatives.

Une première variante de l'ordonnanceur 2 réside dans le fait que les COUCHES G ne soient pas liées physiquement à un NOYAU C_i

particulier. N'importe quel NOYAU C_i peut utiliser n'importe quelle COUCHE G en multiplexant les informations échangées entre un NOYAU C_i et les COUCHES G .

5 Une seconde variante de l'ordonnanceur 2 consiste à regrouper les registres contenus dans les COUCHES G_j^i sur un banc de registres généralisé accessible par toutes les COUCHES G_j^i . Cette solution nécessite que tous les registres d'activité a_j^i et tous les registres d'état ad_j^i des COUCHES G_j^i associées à un NOYAU C_i puissent être lus à chaque cycle.

10 Une troisième variante consiste en ce que les directives d'ordonnancement peuvent soit faire l'objet d'une instruction au même titre que toutes les instructions programmant les ressources 6 ou bien faire seulement l'objet de champs d'instruction associés aux autres instructions.

REVENDEICATIONS

- 1 Système (2) permettant la gestion d'une architecture multi-ressources dans laquelle plusieurs ressources (6) peuvent exécuter simultanément différentes tâches, caractérisé en ce qu'il comporte :
 - au moins un moyen de gestion (G_i^j) de l'exécution d'une partie (j , $0 \leq j$) d'un ensemble ou sous-ensemble (i , $0 \leq i$) d'une ou plusieurs tâches,
 - au moins un moyen de contrôle (C_i) du ou des moyens de gestion (G_i^j) de l'exécution dudit ensemble ou sous-ensemble (i) de tâche(s),
 - au moins un moyen (8) de vérification de l'acquittement d'une ou plusieurs tâches données, chaque moyen de vérification d'acquittement (8) étant associé à tout ou partie des moyens de gestion (G_i^j) et/ou tout ou partie des moyens de contrôle (C_i) et/ou tout ou partie des ressources (6),
 - et/ou un moyen d'initialisation (21) permettant d'initialiser tout ou partie des moyens précédents.
- 2 Système (2) selon la revendication précédente caractérisé en ce que chaque moyen de vérification d'acquittement (8) comporte:
 - au moins un émetteur d'acquittement (em) associé à tout ou partie des moyens de gestion (G_i^j) et/ou tout ou partie des moyens de contrôle (C_i), chaque émetteur (em) émettant un signal (7) indiquant la fin de l'exécution, l'acquittement, d'une ou plusieurs tâches données,
 - au moins un détecteur d'acquittement (d) associé directement ou indirectement à tout ou partie des moyens de gestion (G_i^j), chaque détecteur (d) recevant les signaux d'acquittements (7) de tous les émetteurs d'acquittement (em) et/ou de toutes les ressources (6).
- 3 Système (2) selon la revendication 2 caractérisé en ce que chaque moyen de vérification d'acquittement (8) comporte :
 - au moins un émetteur d'acquittement (em_i) associé au moyen de contrôle (C_i), chaque émetteur (em_i) émettant un signal (7) indiquant la fin de l'exécution d'une ou plusieurs tâches données dont la gestion de l'exécution est contrôlée par ledit moyen de contrôle (C_i),
 - au moins un détecteur d'acquittement (d_i^j) associé directement au moyen de gestion (G_i^j), chaque détecteur (d_i^j) recevant les signaux

d'acquittements (7) de tous les émetteurs d'acquittements (em_i) et/ou de toutes les ressources (6).

4 Système (2) selon l'une des revendications 2 ou 3 caractérisé en ce que chaque moyen de vérification d'acquittement (8) comporte:

- 5 • un bus d'acquittement (27) unique transmettant tous les signaux d'acquittement (7) émis par chaque émetteur d'acquittement (em) ou (em_i) et/ou chaque ressource (6) à tous les détecteurs d'acquittement (d) ou (d_i^j), et
- un moyen d'arbitrage (27^+) de l'accès audit bus.

10 5 Système (2) selon l'une des revendications précédentes caractérisé en ce qu'il comporte :

- une ou plusieurs sorties (S_3) permettant d'émettre l'adresse (3) de la tâche à exécuter (5) comportant une liaison avec le moyen d'initialisation (21) et/ou les moyens de gestion (G_i^j) et/ou les moyens de contrôle (C_i),
- 15 • une ou plusieurs entrées (E_5) permettant de recevoir la tâche à exécuter (5) comportant une liaison avec tous les moyens de contrôle (C_i), et/ou
- une ou plusieurs entrées (E_1) permettant de recevoir l'ordre d'initialisation (1) et comportant une liaison avec le moyen d'initialisation (21), et/ou
- 20 • une ou plusieurs entrées (E_7) permettant de transmettre les signaux d'acquittement (7) des ressources (6) aux moyens de vérification d'acquittement (8) associés aux moyens de contrôle (C_i) et/ou aux moyens de gestion (G_i^j).

25 6 Système (2) selon la revendication précédente caractérisé en ce qu'il comporte :

- une seule entrée (E_1) permettant de recevoir l'ordre d'initialisation (1) comportant une liaison avec le moyen d'initialisation (21),
- 30 • une seule sortie (S_3) permettant d'émettre l'adresse (3) de la tâche à exécuter (5) comportant une liaison avec le moyen d'initialisation (21) et les moyens de gestion (G_i^j),
- une seule entrée (E_5) permettant de recevoir la tâche à exécuter (5) comportant une liaison avec les moyens de contrôle (C_i), et

- une seule entrée (E_7) permettant de recevoir les signaux d'acquittement (7) des ressources (6) comportant une liaison avec les détecteurs d'acquittement (d_i^j).
- 5 7 Système (2) selon la revendication précédente caractérisé en ce qu'il comporte :
- un bus d'adresse (23) unique reliant le moyen d'initialisation (21) et les moyens de gestion (G_i^j) à la sortie d'adresse (S_3), et
 - un moyen d'arbitrage (23^+) de l'accès audit bus.
- 10 8 Système (2) selon l'une des revendications précédentes caractérisé en ce que, lorsque le moyen d'initialisation (21) reçoit un ordre d'initialisation (1), ledit moyen d'initialisation (21) :
- émet le premier signal d'adresse (3), adresse à laquelle se trouve une instruction d'initialisation,
 - active les premiers moyens de gestion (G_i^0) de chaque moyen de
- 15 contrôle (C_i) et
- inactive les autres moyens de gestion (G_i^j , $j \neq 0$).
- 9 Système (2) selon l'une des revendications précédentes caractérisé en ce que :
- l'adresse ($A[j]_i$) contenue dans un moyen de gestion (G_i^j) est émise si
- 20 ledit moyen de gestion (G_i^j) est en phase d'émission, c'est à dire en état d'émission avec autorisation d'émettre, et en ce que
- cette émission induit l'incrémentation de ladite adresse ($A[j]_i = A[j]_i + 1$) contenue dans ledit moyen de gestion (G_i^j).
- 25 10 Système (2) selon la revendication précédente caractérisé en ce qu'un moyen de gestion (G_i^j) en état d'émission passe dans la phase d'émission de cet état lorsque ledit moyen d'arbitrage (23^+) de l'accès au bus d'adresse (23) autorise ledit moyen de gestion (G_i^j) à émettre sur ce bus (23).
- 30 11 Système (2) selon l'une des revendications précédentes caractérisé en ce que, lorsque la tâche à exécuter (5) reçue par les moyens de contrôle (C_i) comporte une instruction d'initialisation d'un ensemble ou sous-ensemble (i) de tâches :
- l'un des moyens de contrôle (C_i) parmi ceux disponibles reçoit les paramètres d'initialisation contenus dans cette instruction

- d'initialisation comportant le nombre de parties (J_i) dudit ensemble ou sous-ensemble (i) de tâches,
- le premier moyen de gestion (G_i^0) associé audit moyen de contrôle (C_i) reçoit l'adresse de base ($A0_i$),
 - 5 • ledit premier moyen de gestion (G_i^0) passe en état d'émission, et
 - le moyen de vérification d'acquiescement (8) associé audit moyen de gestion (G_i^0) se met en mode de mémorisation des acquiescements.
- 12 Système (2) selon la revendication précédente caractérisé en ce que, lorsque la tâche à exécuter (5) reçue par les moyens de contrôle
- 10 comporte une instruction d'initialisation d'un sous-ensemble (i') de tâche(s), et qu'un moyen de gestion (G_i^j) de l'exécution d'une partie (j) d'un ensemble ou sous-ensemble (i , $i \neq i'$) de tâche(s) est en phase d'émission, le moyen de gestion G_i^j en état d'émission passe dans un état d'attente.
- 15 13 Système (2) selon l'une des revendications précédentes caractérisé en ce que, lorsque la tâche à exécuter (5) reçue par les moyens de contrôle (C_i) comporte une instruction de mise en attente le moyen de gestion (G_i^j) en phase d'émission passe dans un état d'attente.
- 14 Système (2) selon l'une des revendications 12 ou 13 caractérisé en ce
- 20 que,
- lorsqu'un moyen de gestion (G_i^j) passe en état d'attente, le moyen de vérification d'acquiescement (8) associé audit moyen de gestion G_i^j se met en mode d'observation de l'acquiescement attendu, c'est à dire l'acquiescement du sous-ensemble (i') de tâche(s) indiquée par
 - 25 l'instruction d'initialisation ou de l'acquiescement de la ou des tâches indiquées par l'instruction d'attente, sachant qu'un moyen de vérification d'acquiescement (8) en mode d'observation effectue les opérations:
 - d'observation des acquiescements et
 - 30 – de détection de l'acquiescement attendu parmi ceux mémorisés lors du mode de mémorisation et ceux observés, et en ce que
 - lorsque le moyen de vérification d'acquiescement (8) associé audit moyen de gestion G_i^j détecte l'acquiescement attendu,
 - ledit moyen de gestion (G_i^j) passe dans un état d'émission,

- le moyen de vérification d'acquiescement (8) associé audit moyen de gestion (G_i^j) se met en mode de mémorisation des acquiescements.

- 15 Système (2) selon l'une des revendications précédentes caractérisé en ce que, lorsque la tâche à exécuter (5) reçue par les moyens de contrôle (C_i) comporte une instruction indiquant la frontière entre deux parties (j) et (j+1), le moyen de gestion (G_i^j) en état d'émission passe dans un état de fin de traitement.
- 16 Système (2) selon la revendication précédente caractérisé en ce que, si un ensemble ou sous-ensemble (i) de tâche est une boucle (i) de programme et si tous les moyens de gestion ou COUCHES (G_i^j) actifs d'un moyen de contrôle ou NOYAU (C_i) sont dans un état de fin de traitement:
- ledit NOYAU (C_i) passe de la gestion d'une itération (it_i) à la gestion de l'itération suivante (it_{i+1}) de la boucle (i) ,
 - 15 • chaque COUCHE (G_i^j) dudit NOYAU (C_i) prend au moins une partie des paramètres de la COUCHE précédente (G_i^{j-1}) comportant, si ladite COUCHE (G_i^j) n'est pas en état fin de traitement de l'itération (N_i). de ladite boucle (i), l'adresse ($A[j]_i=A[j-1]_i$).
 - la première COUCHE (G_i^0) reçoit l'adresse de basse ($A0_i$) si ladite première COUCHE (G_i^0) n'est pas en état fin de traitement de l'itération (N_i). de ladite boucle (i),
 - 20 • chaque COUCHE (G_i^j) ayant reçu une adresse $A[j]_i$ passe dans un état d'émission,
 - et les moyens de vérification d'acquiescement (8) associés aux COUCHES (G_i^j) en état d'émission se mettent en mode de mémorisation des acquiescements,
 - 25
- 17 Système (2) selon l'une des revendications 15 ou 16 caractérisé en ce que, lorsque l'ensemble des moyens de gestions (G_i^j) sont en fin de traitement et, dans le cas d'une boucle (i), en état fin de traitement de l'itération (N_i) de ladite boucle (i), le moyen de contrôle (C_i) indique l'acquiescement de l'exécution de l'ensemble ou sous-ensemble (i).
- 30
- 18 Circuit intégré programmable selon l'une des revendications précédentes caractérisé en ce qu'il comporte une mémoire programme unique (4) comprenant :

- une entrée recevant les signaux d'adresse (3) émis par ledit système (2) et
 - une sortie transmettant les tâches à exécuter (5) audit système (2) et aux ressources (6) dudit circuit.
- 5 19 Méthode de gestion d'une architecture multi-ressources dans laquelle plusieurs ressources (6) peuvent exécuter simultanément différentes tâches caractérisée en ce qu'elle comporte au moins les étapes suivantes:
- 10 • la gestion de l'exécution d'au moins une partie (j , $0 \leq j$) d'un ensemble ou sous-ensemble (i , $0 \leq i$) d'une ou plusieurs tâches,
 - le contrôle de la gestion de chaque partie (j) d'au moins un ensemble ou sous-ensemble (i) d'une ou plusieurs tâches,
 - la vérification de l'acquittement de l'exécution d'au moins une tâche lors de la gestion et/ou du contrôle et/ou de l'exécution par les
 - 15 ressources (6),
 - l'initialisation de tout ou partie des étapes précédentes.
- 20 Méthode selon la revendication précédente caractérisée en ce que la vérification d'acquittement comporte:
- 20 • l'émission d'un signal (7) indiquant la fin de l'exécution (l'acquittement) d'une ou plusieurs tâches données tâche lors de la gestion et/ou du contrôle, et
 - la détection du signal d'acquittement attendu lors de la gestion comprenant la réception des signaux d'acquittements (7) émis lors de la gestion et/ou du contrôle et/ou de l'exécution.
- 25 21 Méthode selon l'une des revendications 19 ou 20 caractérisée en ce que, l'initialisation comporte au moins l'émission de la première adresse à laquelle se trouve une instruction d'initialisation.
- 22 Méthode selon l'une des revendication 19 à 21 caractérisée en ce que la gestion de l'exécution d'une partie (j) d'un ensemble ou sous-ensemble (i)
- 30 en phase d'émission, c'est à dire en état d'émission avec autorisation d'émettre, comporte :
- l'émission de l'adresse $A[j]_i$ à laquelle se trouve la tâche à exécuter (5),
 - le passage simultané de l'adresse $A[j]_i$ à l'adresse $A[j]_{i+1}$.
- 35 23 Méthode selon l'une des revendications 19 à 22 caractérisée en ce que :

- le contrôle de la gestion de l'exécution des parties (j) d'un ensemble ou sous-ensemble (i) de tâche(s) comporte:
 - l'exécution d'une instruction d'initialisation dudit ensemble ou sous-ensemble (i) de tâche(s),
 - 5 – la réception, lors de ladite exécution, des paramètres d'initialisation dudit ensemble ou sous-ensemble (i) de tâche(s) contenus dans cette instruction d'initialisation, lesdits paramètres comportant le nombre de parties (J_i) dudit ensemble ou sous-ensemble (i) de tâche(s) et l'adresse de base ($A0_i$),
 - 10 – la transmission de ladite adresse de base $A0_i$ reçue, et en ce que
 - la gestion de l'exécution de la première partie dudit ensemble ou sous-ensemble (i) de tâche(s) comporte:
 - la réception de ladite adresse de base $A0_i$ transmise lors du contrôle de la gestion dudit ensemble ou sous-ensemble (i) de tâche(s),
 - 15 – le passage à l'état d'émission déclenché par ladite réception,
 - la vérification d'acquiescement comporte le passage en mode de mémorisation des acquiescements déclenché par ledit passage en état d'émission.
- 20 24 Méthode selon la revendication précédente caractérisée en ce que, lorsque l'instruction d'initialisation d'un sous-ensemble (i') est exécutée lors du contrôle de gestion de l'exécution d'un sous-ensemble (i') et que la gestion de l'exécution d'une partie (j) d'un ensemble ou sous-ensemble ($i, i \neq i'$) est en phase d'émission:
 - 25 • le contrôle de gestion de l'exécution de la dite partie (j) dudit ensemble ou sous-ensemble (i) comporte l'exécution de ladite instruction d'initialisation du sous-ensemble (i'), et
 - la gestion de ladite partie (j) de l'ensemble ou sous-ensemble (i) comporte le passage à l'état d'attente déclenché par l'exécution, lors
 - 30 dudit contrôle de la gestion de l'exécution de l'ensemble ou sous-ensemble (i), de ladite instruction d'initialisation du sous-ensemble (i') dont l'acquiescement est attendu.
- 25 Méthode selon l'une des revendications 19 à 24 caractérisée en ce que :
 - le contrôle de la gestion de l'exécution d'un ensemble ou sous-ensemble (i) comporte l'exécution d'une instruction de mise en attente
 - 35

si la gestion de l'exécution d'une partie (j) dudit ensemble ou sous-ensemble (i) est en phase d'émission, et en ce que

- la gestion de l'exécution de ladite partie (j) dudit ensemble ou sous-ensemble (i) en phase d'émission comporte le passage à l'état d'attente déclenché par ladite exécution indiquant la ou les tâches dont l'acquiescement est attendu.

26 Méthode selon l'une des revendications 24 ou 25 caractérisée en ce que:

- la vérification d'acquiescement comporte le passage en mode d'observation déclenché par le passage en état d'attente de la gestion de l'exécution d'une partie (j) d'un ensemble ou sous-ensemble (i) de tâche(s); le passage en mode d'observation comportant au moins:
 - l'observation des acquiescements et
 - la détection de l'acquiescement attendu parmi ceux mémorisés lors du mode de mémorisation et ceux observés, en ce que
- la gestion de l'exécution d'une partie (j) d'un ensemble ou sous-ensemble (i) de tâche(s) en état d'attente comporte le passage à l'état d'émission déclenché par ladite détection de l'acquiescement attendu, et en ce que
- la vérification d'acquiescement comporte le passage en mode de mémorisation des acquiescements déclenché par ledit passage en état d'émission.

27 Méthode selon l'une des revendications 19 à 26 caractérisée en ce que :

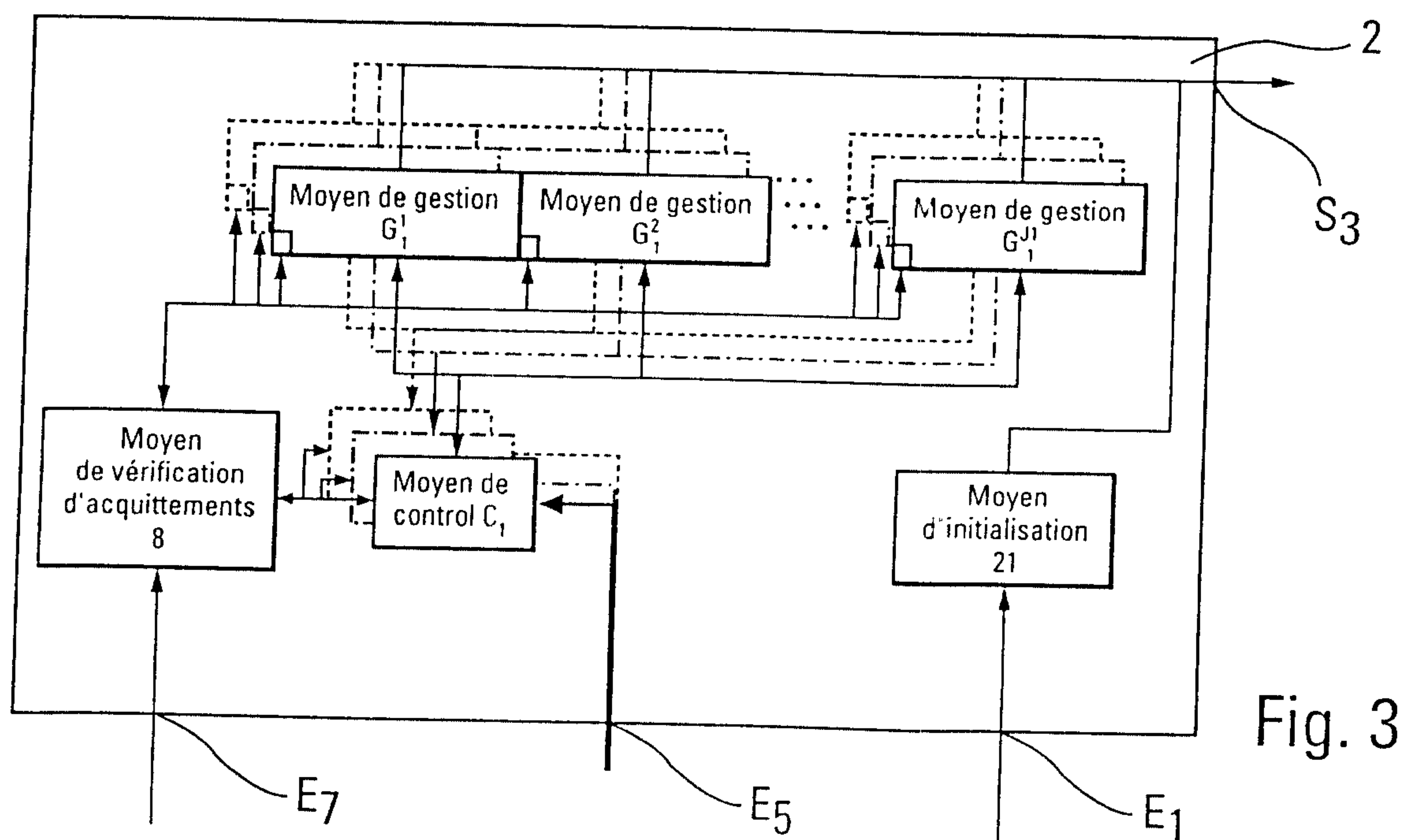
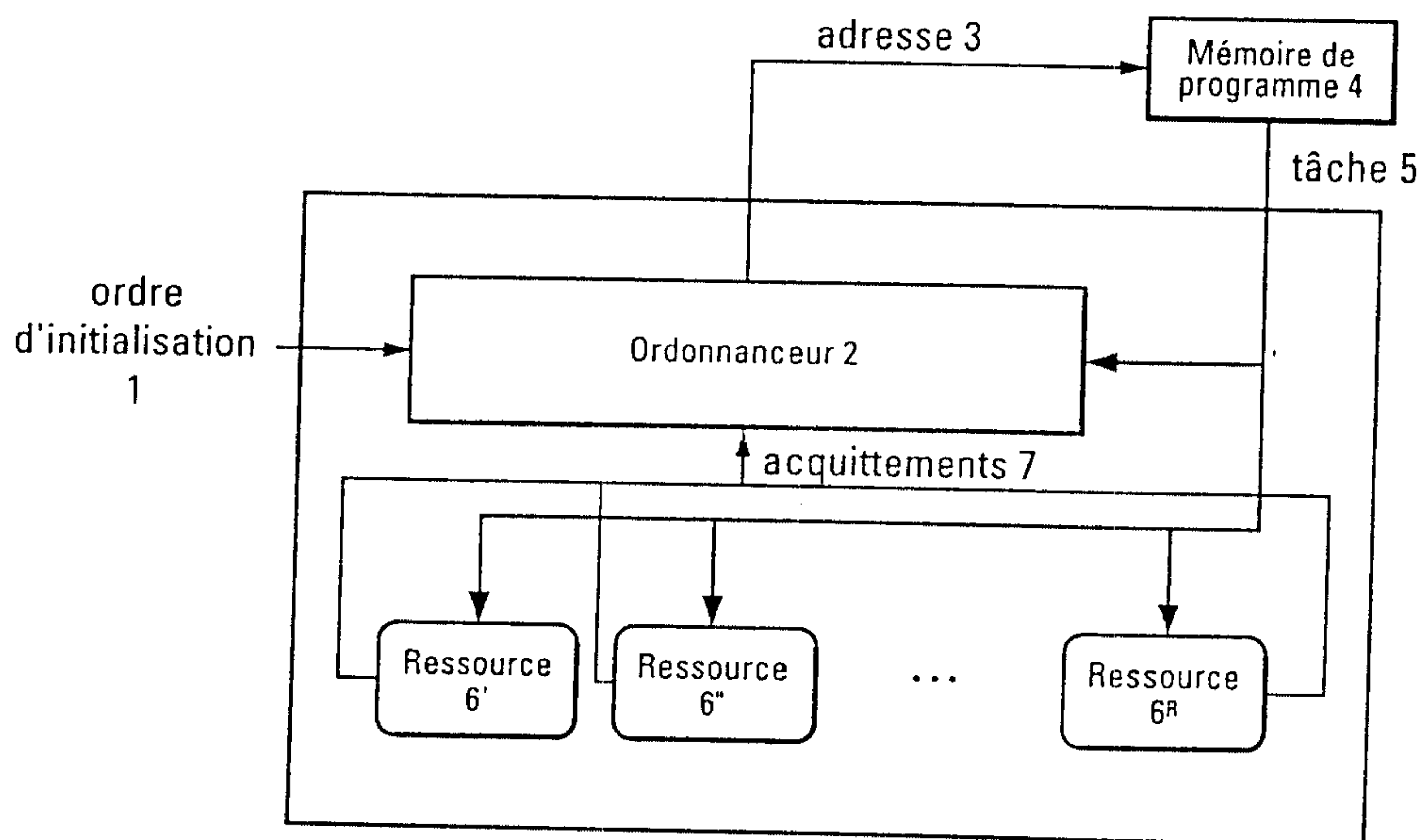
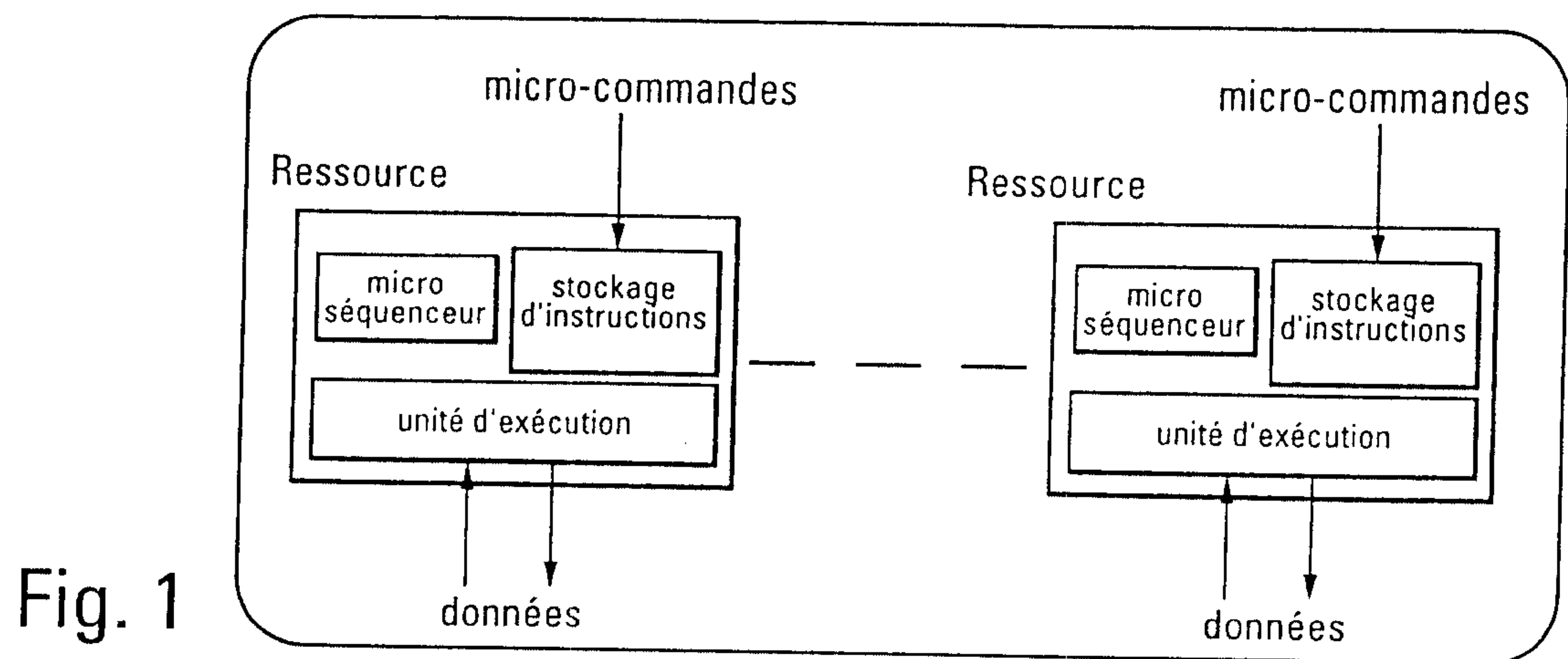
- si la gestion de l'exécution d'une partie (j) d'un ensemble ou sous-ensemble (i) est en phase d'émission, le contrôle de la gestion de l'exécution dudit ensemble ou sous-ensemble (i) de tâche(s) comporte l'exécution d'une instruction indiquant la frontière entre deux parties (j) et (j+1), et en ce que
- la gestion de l'exécution de ladite partie (j) dudit ensemble ou sous-ensemble (i) comporte le passage à l'état fin de traitement déclenché par ladite exécution de l'instruction indiquant frontière entre ladite partie (j) et la partie suivante (j+1).

28 Méthode selon la revendication précédente caractérisée en ce que, si l'ensemble ou sous-ensemble (i) de tâche est une boucle de programme et si la gestion de l'exécution de toutes les parties ou étages (j) de la boucle (i) sont en état fin de traitement :

- 5 • le contrôle de ladite gestion de l'exécution de la boucle (i) comporte:

 - le passage à l'itération suivante $(it_i)=(it_i)+1$,
 - la transmission de l'adresse $(A[j]_i)$ obtenue lors de la gestion de l'exécution d'un étage (j) de ladite boucle (i), si la gestion de l'exécution de l'étage suivant (j+1) de ladite boucle (i) n'est pas en état fin de traitement de l'itération (N_i) . de ladite boucle (i).
 - la transmission de l'adresse de base $(A0_i)$, si la gestion de l'exécution du premier étage de ladite boucle (i) n'est pas en état fin de traitement de l'itération (N_i) . de ladite boucle (i), et en ce que:
 - 10 • la gestion d'un étage (j+1) de ladite boucle (i) comporte:

 - la réception, le cas échéant, de ladite adresse transmise: adresse de base $(A0_i)$ pour le premier étage et adresse $(A[j+1]_i)=(A[j]_i)$ provenant de la gestion de l'exécution de l'étage précédent (j) de ladite boucle (i), et
 - le passage en état d'émission déclenché par ladite réception,
 - 15 • la vérification d'acquittement comporte le passage en mode de mémorisation des acquittements déclenché par ledit passage en état d'émission.
- 29 Méthode selon l'une des revendications 27 ou 28 caractérisée en ce que,
- 20 le contrôle de la gestion de l'exécution d'un ensemble ou sous-ensemble (i) de tâche(s) comporte le signalement de l'acquittement dudit ensemble ou sous-ensemble (i) de tâche(s) dès que la gestion de chaque partie (j) dudit ensemble ou sous-ensemble (i) de tâche(s) est en en état fin de traitement et, dans le cas d'une boucle (i), en état fin de traitement de
- 25 l'itération (N_i) de ladite boucle (i).



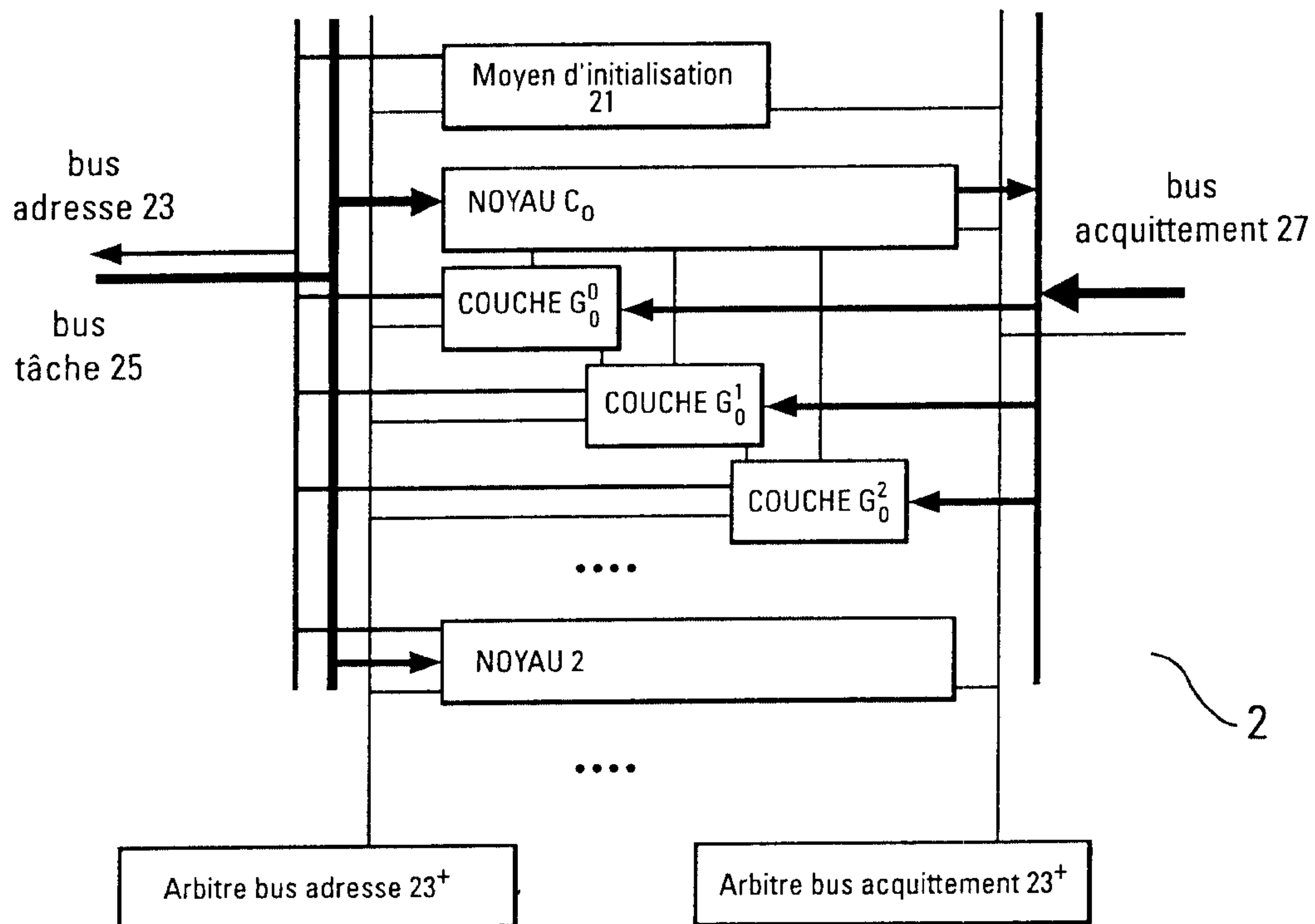


Fig. 4

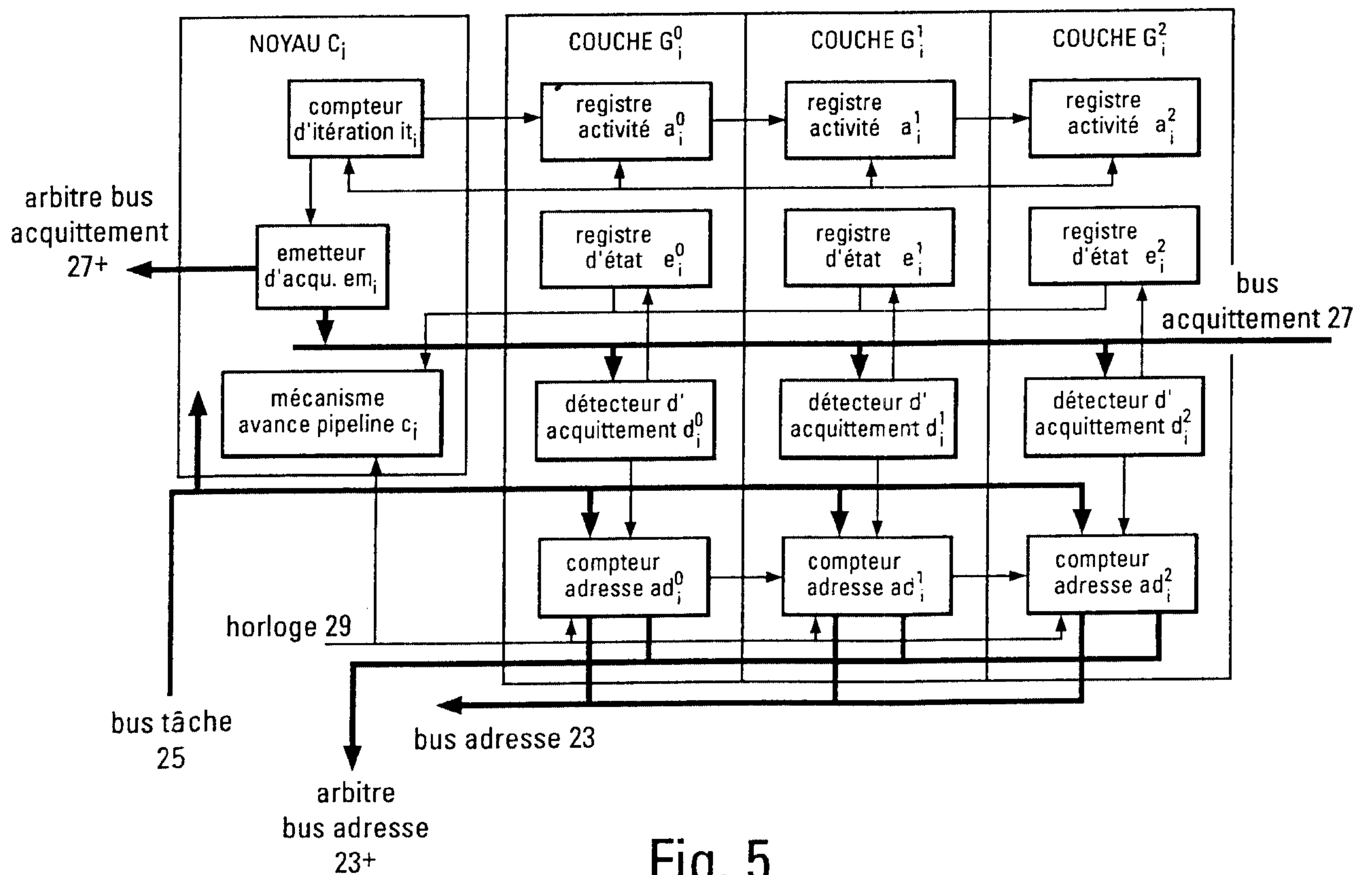


Fig. 5

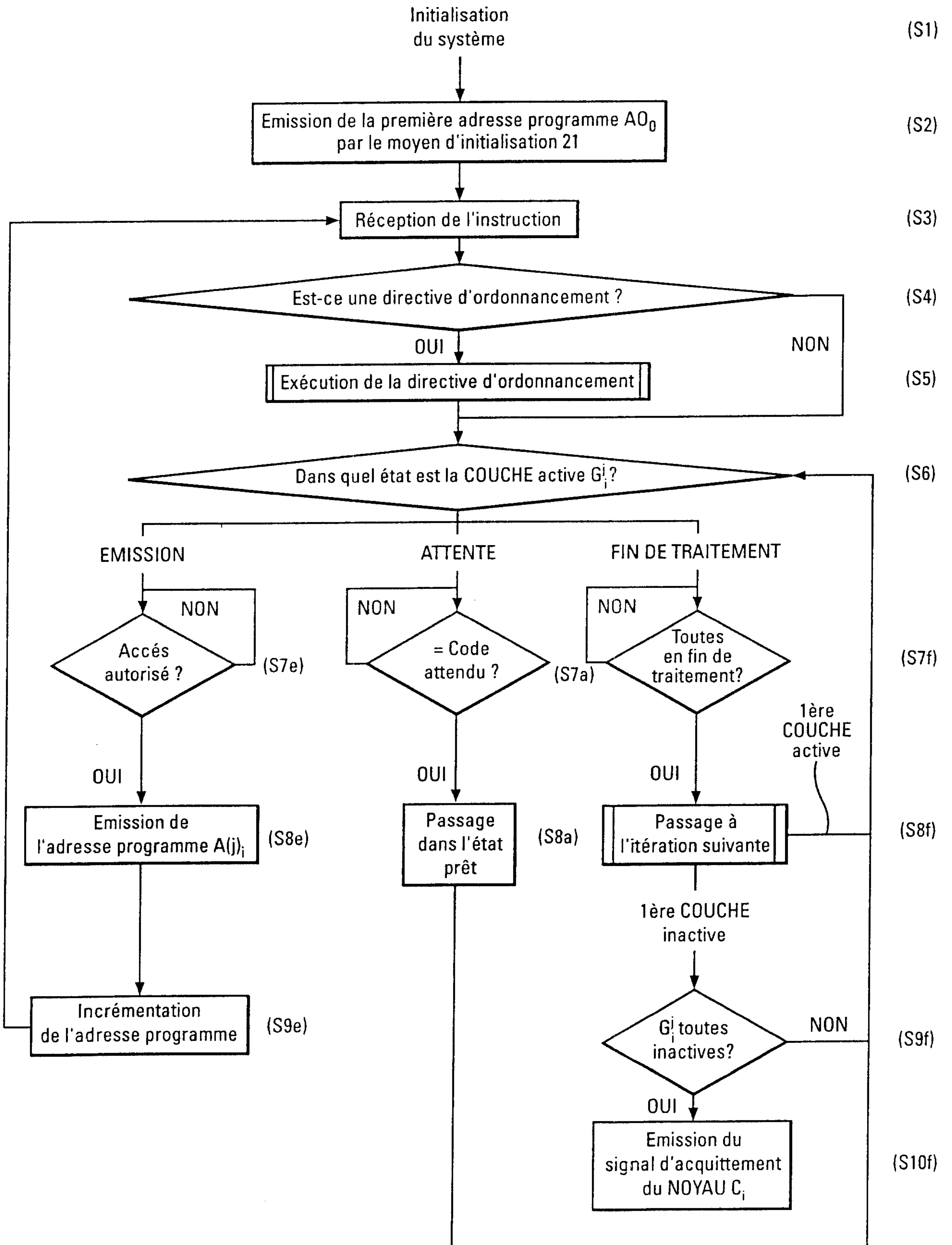


Fig. 6

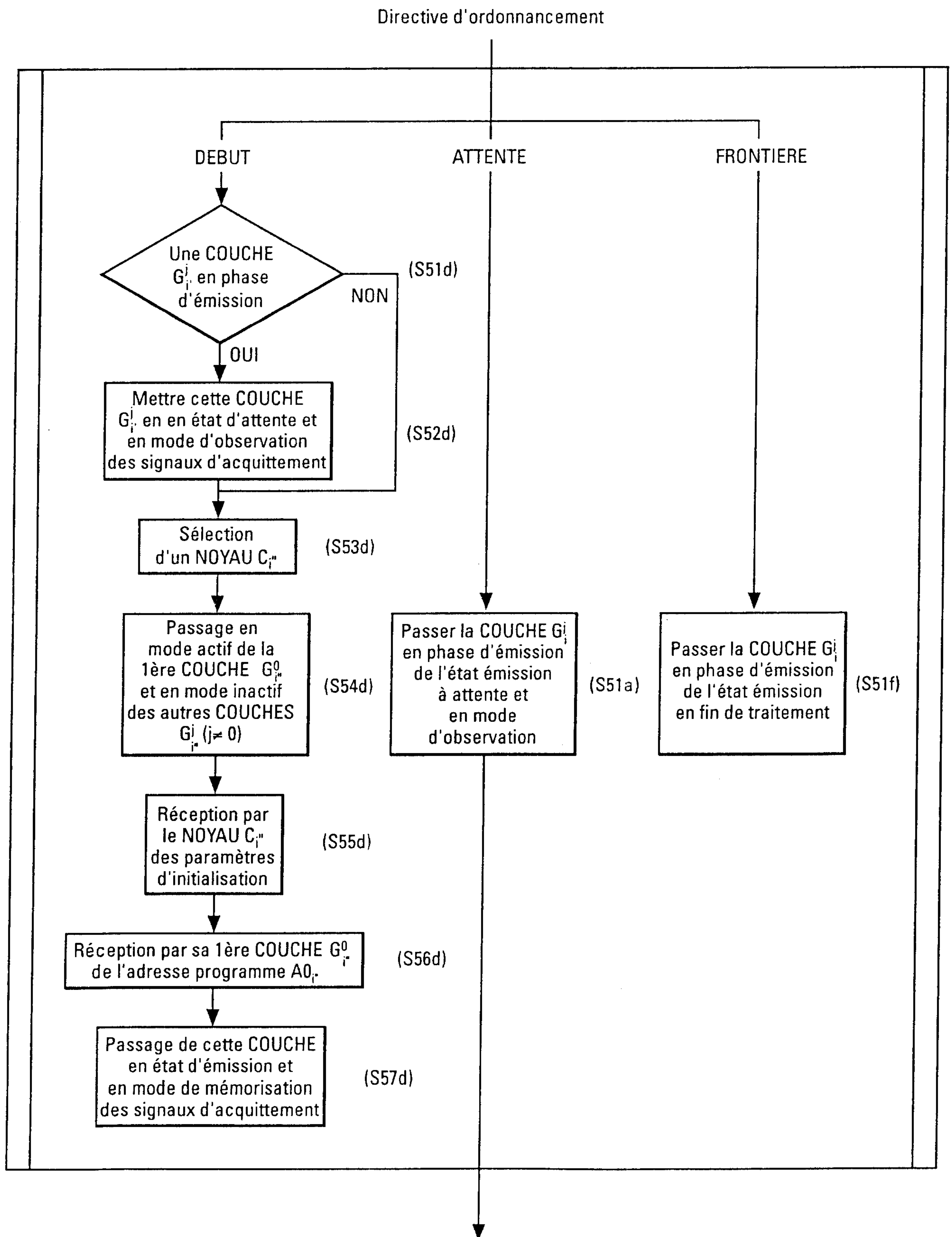


Fig. 7

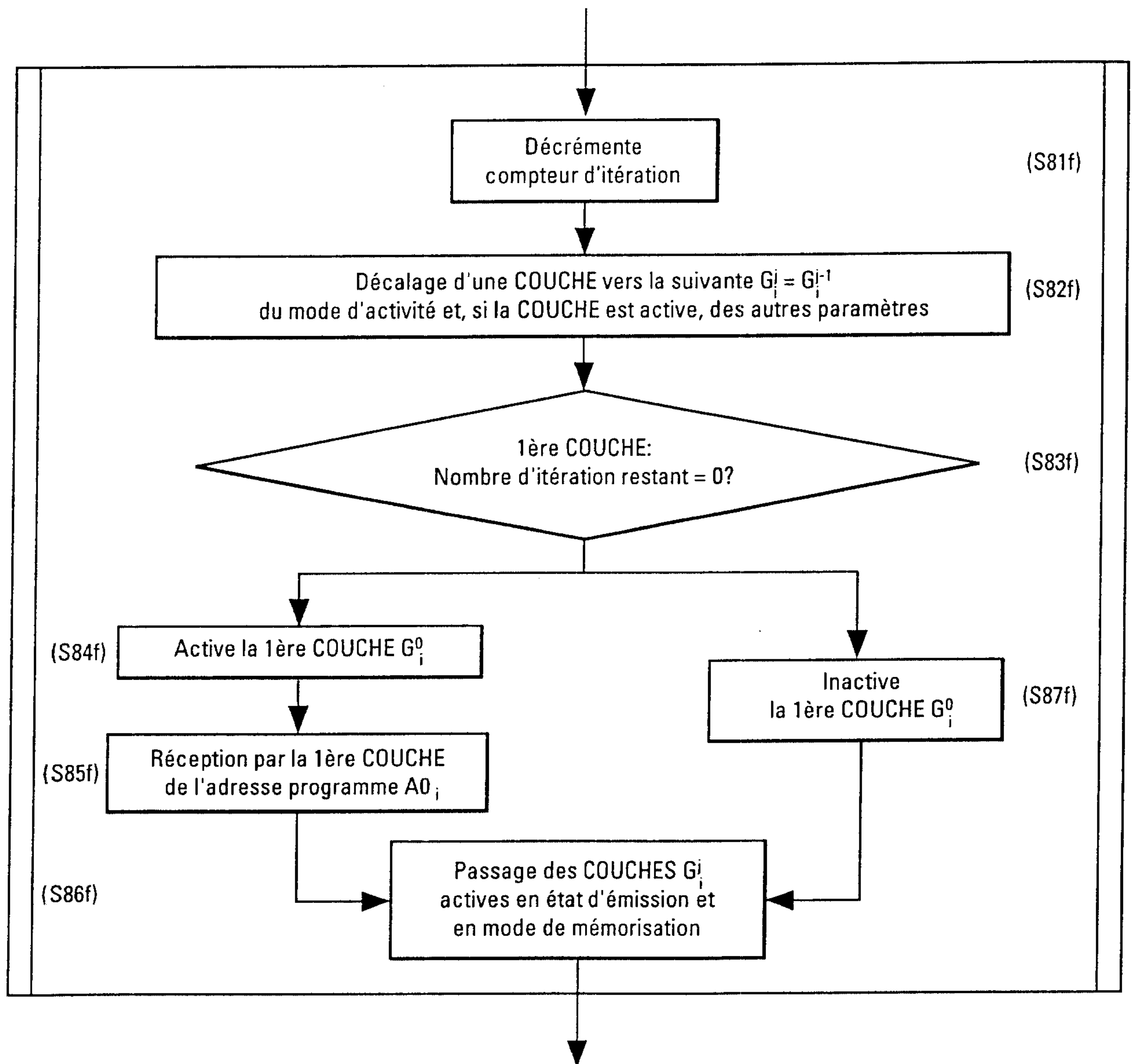


Fig. 8

