(19) RÉPUBLIQUE FRANÇAISE

INSTITUT NATIONAL
DE LA PROPRIÉTÉ INDUSTRIELLE

PARIS

(11) N° de publication : **2 643 166**
(à n'utiliser que pour les commandes de reproduction)

(21) N° d'enregistrement national : **90 01451**

(51) Int Cl[5] : G 06 F 12/02.

(12) DEMANDE DE BREVET D'INVENTION A1

(22) **Date de dépôt** : 8 février 1990.

(30) **Priorité** : US, 10 février 1989, n° 309.429.

(43) **Date de la mise à disposition du public de la demande** : BOPI « Brevets » n° 33 du 17 août 1990.

(60) **Références à d'autres documents nationaux apparentés** :

(71) **Demandeur(s)** : *INTEL CORPORATION.* — US.

(72) **Inventeur(s)** : Leslie D. Kohn.

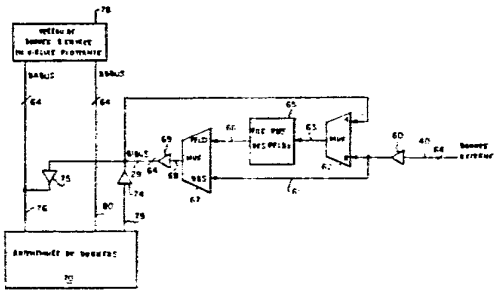
(73) **Titulaire(s)** :

(74) **Mandataire(s)** : Cabinet Regimbeau, Martin, Schrimpf, Warcoin et Ahner.

(54) Microprocesseur à architecture pipeline comprenant une instruction d'accès à des données en mémoire externe, et procédé d'accès à ces données.

(57) L'invention concerne un microprocesseur à architecture en pipeline, comprenant une antémémoire de données 70 et une unité de traitement avec un verrou de données 78, et pourvu d'une instruction permettant d'accéder à des données provenant d'une mémoire externe par l'intermédiaire d'un bus externe 40.

Selon l'invention, cette instruction met en œuvre des moyens de mémoire 65, pour accumuler des données; des premiers moyens de circuit, pour délivrer aux moyens de mémoire des données en provenance de la mémoire externe en cas de survenance d'une non-concordance d'antémémoire; des seconds moyens de circuit, pour délivrer au verrou de données les données accumulées dans les moyens de mémoire, ces seconds moyens de circuit reliant également l'antémémoire de données aux premiers moyens de circuit de manière à retourner aux moyens de mémoire la donnée résidant dans l'antémémoire, en cas de concordance d'antémémoire; et des moyens de contrôle de bus, reliés aux premiers et aux seconds moyens de circuit, aux moyens de mémoire et à l'antémémoire de données, pour contrôler le flux en pipeline de données allant de la mémoire externe au verrou de données en cas de non-concordance d'antémémoire, et pour contrôler le retour des données de l'antémémoire aux moyens mémoire en cas de concordance d'antémémoire.



FR 2 643 166 - A1

La présente invention concerne le domaine des microprocesseurs à semiconducteurs, en particulier les processeurs de ce type qui sont capables d'accéder à des données dans une mémoire externe.

Plus précisément, la présente invention se rapporte à une instruction de chargement à virgule flottante en pipeline qui peut être mise en oeuvre à l'intérieur de l'unité de contrôle du bus d'un microprocesseur. Le microprocesseur utilisé dans le cas de la présente invention est le microprocesseur Intel 860™, souvent appelé processeur N10™ (Intel est une marque déposée de Intel Corporation).

Le processeur N10 est un processeur 32/64 bits à virgule flottante compatible IEEE, avec un processeur de nombres entiers RISC 32 bits et un processeur graphique tridimensionnel 64 bits. Du fait qu'il utilise un processeur numérique optimisé à la fois pour les opérations vectorielles et pour les opérations scalaires, il constitue, dans l'industrie, le premier processeur vectoriel intégré à hautes performances incorporant plus d'un million de transistors et procurant des performances de l'ordre de la moitié de celle d'un *Cray I*, le tout sur une seule puce.

Tous les processeurs possèdent une certaine forme d'instruction de chargement permettant d'accéder à des informations soit depuis une mémoire externe soit depuis une antémémoire de données interne. L'accès aux données mémorisées à l'extérieur est généralement effectué par l'intermédiaire d'un bus de données externes contrôlé par la logique interne du processeur. La raison d'être d'une antémémoire de données est la possibilité d'obtenir un accès performant à des informations souvent utilisées, ce qui permet d'accélérer la vitesse du traitement. Dans les processeurs qui utilisent une antémémoire de données, les instructions de chargement normales agiront plus efficacement si l'information de donnée réside dans l'antémémoire incorporée à la puce. En d'autres termes, si la donnée ne se trouve pas dans l'antémémoire, on sera pénalisé en ce qui concerne les performances lorsque l'on voudra accéder à cette donnée.

Habituellement, lorsque l'on référence une donnée externe

au moyen d'une instruction de chargement normal, cette donnée est mémorisée dans l'antémémoire. La raison en est que, dans des conditions normales, il est très probable que la donnée que l'on vient juste de référencer soit à nouveau référencée à bref délai. On minimise la pénalisation liée à l'accès à la donnée en mettant dans l'antémémoire de données internes les informations auxquelles on accède le plus fréquemment, et en réservant la mémoire externe pour les informations rarement référencées ou réutilisées. C'est le principe de localisation qui fait que l'antémémoire de données est un instrument utile, du fait que les programmes ont tendance à référencer certaines données de façon répétée à bref délai.

Il se présente cependant un problème lorsque l'on fait traiter à un processeur des structures de données de très grande dimension ou, en tout état de cause, des structures de données qui sont beaucoup plus grandes que ce que l'antémémoire de données peut normalement contenir. Comme illustration de la difficulté qui peut survenir, on peut citer le cas d'un processeur auquel on demande souvent d'exécuter diverses opérations à virgule flottante, par exemple des inversions matricielles, des multiplications, etc., qui nécessitent la manipulation d'immenses matrices de données. Dans les processeurs de l'art antérieur, lorsque la donnée ne se trouve pas dans l'antémémoire de données incorporée à la puce, le processeur doit geler l'exécution et demander l'accès à la mémoire externe. Pendant la durée de ce gel de l'exécution, on empêche le processeur de délivrer une quelconque nouvelle adresse à la mémoire. En d'autres termes, le processeur doit attendre que les données nécessaires à la première opération arrivent de la mémoire externe avant de poursuivre l'exécution de ses opérations. De ce fait, ce type d'accès à une mémoire externe peut prendre jusqu'à six cycles d'horloge, ou même plus. On introduit ainsi un retard important, qui joue sur la vitesse de traitement du système lorsque la taille des structures de données impliquées dans le traitement requiert des accès fréquents à la mémoire externe.

Un autre problème lié à la gestion de structures de données de grandes dimensions apparaît lorsque l'on amène au processeur la donnée à laquelle on a accédé à l'extérieur. Lorsque la donnée externe est délivrée au processeur, elle est inscrite dans l'antémémoire et remplace généralement une donnée qui y résidait antérieurement. Cependant, on se rappelera que certaines données externes sont constituées d'opérations rarement référencées (ce cas est extrêmement fréquent avec des structures de données de grandes dimensions), c'est-à-dire que l'on n'envisage pas de les réutiliser, alors que les données qu'elles ont remplacé dans l'antémémoire étaient des informations qui allaient très probablement être référencées à bref délai. Ainsi, le processeur élimine des données qui doivent être réutilisées en faveur de données qui, selon toute probabilité, ne seront référencées qu'une seule fois. Il en résulte que l'on perd un temps inutile à rappeler les données de l'antémémoire qui ont été substituées. Ce temps d'accès accru est une autre raison pour laquelle les processeurs de l'art antérieur fonctionnent à une vitesse bien inférieure à celle que permet la présente invention.

Comme on le verra, la présente invention prévoit une structure en pipeline qui est capable d'effectuer des opérations mémoire à une vitesse très supérieure (pratiquement, à une vitesse correspondant à la largeur de bande maximale du bus) sans aucun retard pour attendre que le processeur produise l'adresse suivante. Par utilisation de cette structure en pipeline, le processeur associé à la présente invention peut continuer à délivrer des adresses sans avoir à attendre l'arrivée des données en provenance de la mémoire externe. Par rapport aux processeurs de l'art antérieur, cette possibilité procure un avantage certain au microprocesseur que l'on va décrire ici.

Pour arriver à ceci, la présente invention prévoit une instruction de chargement à virgule flottante en pipeline permettant d'accéder rapidement à une donnée mémorisée en mémoire externe. Cette instruction logicielle de chargement à virgule flottante en pipeline, que l'on appelera par

commodité "instruction PFLoad" (*Pipelined Floating-point Load* : chargement à virgule flottante en pipeline) ou "instruction PFLD", peut être utilisée par un programmeur pour accéder à une donnée mémorisée soit dans l'antémémoire de données incorporée à la puce soit dans une mémoire système externe. L'instruction est optimisée pour tenir compte du cas où la donnée ne réside pas déjà dans l'antémémoire de données interne du processeur. On appellera cette situation "non-concordance d'antémémoire" ou, formulée d'une autre manière, "non-concordance PFLoad". Le cas contraire, où la donnée que l'on doit charger est déjà mémorisée dans l'antémémoire de données, et que l'on appellera "concordance d'antémémoire", est également gérée par la présente invention.

En outre, l'instruction PFLoad de la présente invention ne remplacera pas la donnée résidant déjà dans l'antémémoire de données mais, au contraire, dirigera la nouvelle donnée à laquelle on a accédé vers un emplacement de mémorisation situé à l'intérieur de l'unité à virgule flottante du processeur. L'instruction PFLoad sera décrite dans le cadre de sa mise en oeuvre normale dans l'unité de contrôle du bus du processeur N10.

On va décrire ci-dessous une instruction de microprocesseur spécialisée optimisée pour l'accès à des données à virgule flottante mémorisées dans un système de mémoire externe. L'invention utilise l'architecture en pipeline du microprocesseur pour retrouver de façon efficace la donnée externe qui ne sera très probablement pas réutilisée à bref délai, en transférant la donnée directement vers un verrou de données à virgule flottante. Les données fréquement référencées, qui résident dans l'antémémoire de données incorporée à la puce, ne seront pas dérangées par ces opérations.

L'instruction de chargement à virgule flottante en pipeline de la présente invention comporte une pile mémoire du type "premier entré-premier sorti", ci-après désignée "mémoire PEPS", permettant de cumuler les données auxquelles on a accédé. Des premiers moyens de circuit coopèrent avec la mémoire PEPS pour délivrer à la mémoire PEPS les données

depuis la mémoire externe, et il est prévu des seconds moyens de circuit pour transférer au verrou de données à virgule flottante les données de la mémoire PEPS. Les seconds moyens de circuit assurent également l'application en retour des données en provenance de l'antémémoire de données vers la mémoire PEPS dans le cas d'une concordance d'antémémoire (c'est-à-dire dans le cas où la donnée référencée réside effectivement dans l'antémémoire de données). Enfin, il est prévu des moyens de contrôle de bus, reliés aux premiers et aux seconds moyens de circuit, pour contrôler le flux en pipeline des données depuis le système de mémoire externe jusqu'à l'unité à virgule flottante.

L'invention a également pour objet un procédé pour accéder, dans un tel processeur comprenant une unité de traitement, une antémémoire de données et une liaison à un système de mémoire externe, à une donnée se trouvant dans cette mémoire externe, comportant les étapes suivantes : adressage de la donnée dans la mémoire externe ; inscription de cette donnée, depuis la mémoire externe, dans une mémoire du type premier entré-premier sorti ; lecture de cette donnée, depuis cette mémoire du type premier entré-premier sorti, vers un bus ; et transfert de cette donnée sur ce bus vers un verrou de données situé à l'intérieur de l'unité de traitement.

La présente invention permet de disposer ainsi d'une instruction opérant en pipeline qui soit capable d'effectuer des opérations mémoire à une vitesse nettement plus élevée que dans le cas des processeurs de l'art antérieur. De la sorte, la présente invention permet de continuer à délivrer des adresses sans devoir attendre l'arrivée effective des données depuis la mémoire externe.

La présente invention permet également d'accéder ainsi à des données externes sans remplacer des données déjà résidentes dans l'antémémoire de données incorporées à la puce. La présente invention permet donc de minimiser le temps total d'accès, ce qui permet d'accroître la vitesse de fonctionnement.

La présente invention permet également ainsi de disposer

d'un moyen efficace et rapide pour gérer des opérations arithmétiques impliquant des structures de données de très grande dimension.

◊

On comprendra mieux la présente invention à la lecture de la description détaillée ci-dessous et des dessins annexés du mode de réalisation préféré de la présente invention. Cette description et ces dessins ne doivent cependant pas être considérés comme limitant la présente invention à un mode de réalisation particulier, et ils ne sont donnés qu'à des fins explicatives et illustratives.

La figure 1 est un schéma logique éclaté du processeur associé à l'instruction de chargement à virgule flottante en pipeline de la présente invention.

La figure 2 montre le mode de réalisation actuellement préféré de l'instruction de chargement à virgule flottante en pipeline de la présente invention.

La figure 3 illustre une série d'instructions de chargement à virgule flottante en pipeline qui ont été émises pour l'exécution d'une opération typique impliquant une matrice vectorielle mémorisée en mémoire externe et un vecteur interne mémorisé dans l'antémémoire de données incorporée à la puce. On a indiqué les adresses associées à chaque instruction de chargement à virgule flottante en pipeline, avec leurs emplacements de mémorisation en virgule flottante associés.

La figure 4 illustre une variante du mode de réalisation de la présente invention.

La figure 5A illustre un mode de réalisation de la mémoire du type premier entré-premier sorti de chargement à virgule flottante en pipeline faisant partie de la présente invention.

La figure 5B illustre le mode de réalisation actuellement préféré de la mémoire du type premier entré-premier sorti faisant partie de la présente invention.

La figure 6 est un organigramme montrant les opérations

de contrôle assurées par la logique de contrôle du bus associée à l'instruction PFLoad de la présente invention.

◊

On va décrire un microprocesseur comprenant une instruction de chargement à virgule flottante en pipeline permettant de charger des données depuis une mémoire externe. Dans la description qui va suivre, on donnera de nombreux détails particuliers, par exemple des tailles en nombre de bits, etc., afin de permettre une compréhension parfaite de la présente invention. L'homme du métier comprendra bien évidement que ces détails particuliers n'ont pas besoin d'être nécessairement repris pour mettre en oeuvre la présente invention. Inversement, des structures et des circuits bien connus n'ont pas été décrits en détail afin de ne pas alourdir inutilement la description de la présente invention.

Si l'on se réfère à la figure 1, on y a représenté un schéma éclaté du processeur M10. Le processeur M10 est logiquement et physiquement divisé en huit unités distinctes.

L'unité arithmétique 35 comporte une unité d'additionneur à virgule flottante qui permet de réaliser des additions en simple précision et en double précision conformément à la norme IEEE, et un bloc multiplieur à virgule flottante, qui permet, de la même façon, de réaliser des multiplications en simple précision et en double précision conformément à la norme IEEE. L'additionneur et le multiplieur sont tous deux constitués d'unités qui peuvent fonctionner en pipeline à trois étages. En cours de fonctionnement, les opérandes d'entrée représentés par les mnémoniques "src1" et "src2" sont appliqués à l'unité arithmétique 35 par les lignes 42 et 43, respectivement. Les mnémoniques "src1" et "src2" (ainsi que le mnémonique "dest") indiquent l'un des trente-deux registres à virgule flottante situés à l'intérieur du processeur M10. Le résultat en sortie de l'unité arithmétique 35 apparaît sur la ligne 49, qui est reliée au bus de destination 41 à 64 bits. Les bus 41 à 43 ont chacun une

largeur de 64 bits et ils sont reliés à l'unité de contrôle à virgule flottante 33. Les opérandes source à virgule flottante "src1" et "src2" sont également appliqués à l'unité graphique 34.

L'unité graphique 34 est un processeur graphique 64 bits permettant d'effectuer à grande vitesse des calculs de pixels et de graphiques tridimensionnels pour divers algorithmes d'ombrage et pour la vérification du tampon de profondeur en vue de l'élimination des surfaces cachées. Le processeur graphique est capable de regrouper les résultats de plusieurs opérations pour délivrer sur la ligne 45 un résultat vers le bus de destination 41.

L'antémémoire d'instruction 20, également appelé "unité I", délivre des instructions sur des entiers à 32 bits et sur des nombres à virgule flottante à 32 bits, à la fois au noyau RISC 31 et à l'unité de contrôle à virgule flottante 33 sur les lignes 24 et 25, respectivement. L'antémémoire d'instruction est une mémoire associative bidirectionnelle de 4 K octets, en blocs de 32 octets. Les instructions sont délivrées aux bus respectifs par l'antémémoire 20 à chaque cycle d'horloge. L'unité de gestion de la mémoire 21 exécute la traduction de l'adresse virtuelle en adresse physique pour les accès aux données et aux instructions, effectue le contrôle des violations d'accès et compare les adresses physiques du cycle en cours à celles du cycle précédent pour produire, sur un même cycle d'horloge, le signal le plus proche qui suit. La traduction est effectuée au moyen d'un tampon de traduction par transcodage, qui est une antémémoire associative à 64 entrées. L'unité de gestion de la mémoire 21 reçoit ses données d'entrée par les bus 26 et 27 de 32 bits, en sortie du noyau RISC 31.

Le noyau RISC 31, également désigné "noyau d'exécution" ou "unité E", exécute toutes les opérations sur les entiers de 32 bits, et celles de chargement/mémorisation. C'est le contrôleur central du processeur M10. Il possède une batterie de registres à trois ports, avec des registres à 32 bits, un réseau de contournement, un décaleur, une unité

arithmétique et logique et les pointeurs d'instruction. L'unité de noyau d'exécution 31 va chercher aussi bien les instructions relatives aux entiers que les instructions à virgule flottante. Il contient la batterie des registres d'entiers et décode et exécute les opérations de chargement, de mémorisation, de traitement des entiers, de traitement des valeurs binaires et de contrôle-transfert.

L'unité d'antémémoire de données 22 assure les fonctions d'antémémoire de données et d'alignement des octets pour les opérations de lecture/écriture et les processeurs N10. L'échange des données de l'antémémoire est exécuté entre l'antémémoire de données 22 et l'unité de contrôle à virgule flottante 33 par l'intermédiaire du bus 37. L'antémémoire de données 22 peut également recevoir des données externes par l'intermédiaire de l'unité de contrôle du bus 30 sur un bus à 64 bits 29.

L'unité de contrôle du bus 30, également appellée "unité B", est le contrôleur de cycle de bus qui assure l'interfaçage entre le bus externe et la puce interne. Il reçoit des requêtes de cycle de bus de l'unité E, effectue les accès à l'antémémoire de données et à l'antémémoire d'instruction, gère les cas de non-concordance d'antémémoire, avec contrôle du gel et remplacement de la ligne d'antémémoire, contrôle la traduction par le tampon de traduction par transcodage et le traitement des non-concordances et des défauts, et assure l'interfaçage avec le bus externe. L'instruction de chargement à virgule flottante en pipeline qui fait l'objet de la présente invention est mise en oeuvre à l'intérieur de l'unité de contrôle du bus 30 du processeur N10. Le contrôleur de bus possède une architecture en pipeline qui permet d'avoir jusqu'à trois cycles de bus simultanément en cours.

Dans l'architecture de nombreux microprocesseurs modernes, on utilise le fonctionnement en pipeline pour accroître la vitesse à laquelle on peut introduire ou exécuter des opérations. L'architecture en pipeline traite chaque opération sous forme d'une série d'opérations primitives (appelées étages), qui peuvent être exécutées en parallèle.

Dans le processeur N10, le nombre d'étages du pipeline peut aller de un à trois. Une opération en pipeline avec un pipeline à trois étages mémorise le résultat de l'opération située trois opérations en arrière. Une opération en pipeline avec un pipeline à deux étages mémorise le résultat de l'opération située deux opérations en arrière. Une opération en pipeline avec un pipeline à un étage mémorise le résultat de l'opération précédente. Le processeur N10 permet d'avoir un fonctionnement en pipeline du cycle de bus sur trois niveaux sur le bus externe, de sorte que l'on peut avoir trois cycles de bus qui aient été émis avant que la donnée correspondant au premier cycle n'ait été retournée. En outre, le chemin interne des adresses dans l'unité B est organisé en pipeline, de telle sorte que l'on exécute la traduction d'adresse du cycle suivant parallèlement au cycle de bus externe en cours.

Comme on l'a indiqué plus haut, on utilise l'instruction PFLoad pour accéder, dans la mémoire externe, aux données rarement utilisées, c'est-à-dire aux données que l'on ne prévoit pas de réutiliser à bref délai. Dès lors, la logique de contrôle PFLoad, qui se trouve à l'intérieur de l'unité de contrôle du bus, est optimisée pour tenir compte du cas d'une non-concordance d'antémémoire. Une "non-concordance d'antémémoire" correspond à la situation dans laquelle la donnée demandée par PFLoad n'est pas une donnée résideant déjà dans l'unité de l'antémémoire de données.

La figure 2 illustre un mode de réalisation préféré de mise en oeuvre de l'instruction de la présente invention. L'instruction PFLoad retourne une donnée, depuis la mémoire externe, sur un bus 40 à 64 bits. La donnée retournée est la donnée qui avait été adressée trois instructions en arrière, conformément au fonctionnement en pipeline du processeur N10. Le tampon à trois états logiques 60 est utilisé pour appliquer la donnée externe sur l'entrée "zéro" du multiplexeur 62, ainsi que sur l'entrée "DBS" du multiplexeur 67. Le tampon 60 est relié aux multiplexeurs 62 et 67 par l'intermédiaire de la ligne 61. Le tampon 60, ainsi que les étages de pilotage 69, 74 et 75, sont constitués d'étages de

pilotage classiques en logique à trois états. De la même façon, dans le mode de réalisation préféré les multiplexeurs 62 et 67 sont formés de circuits logiques classiques. Le contrôle des étages de pilotage à trois états 60, 69, 74 et 75, ainsi que des multiplexeurs 62 et 67, est assuré par la logique de contrôle du bus située à l'intérieur de l'unité B. Cette logique de contrôle a été synthétisée par ordinateur à partir d'une description en langage fonctionnel à haut niveau de l'unité de contrôle du bus. On donnera plus bas un organigramme des opérations de contrôle exécutées par l'unité B.

La mise en oeuvre de l'instruction PFLoad nécessite également une mémoire 65, ci-après désignée "mémoire PEPS PFLoad" ou "mémoire PEPS", reliée à la sortie du multiplexeur 62 par la ligne 63. La mémoire PEPS 65 produit une donnée à sa sortie 66, donnée qui est appliquée à l'entrée PFLD du multiplexeur 67. Dans le mode de réalisation actuellement préféré, la mémoire PEPS 65 a une taille de 64 bits, avec une profondeur de trois mots. La mémoire PEPS PFLoad 65 contient les données sur 64 bits référencées par les trois instructions PFLoad antérieures. Lors de l'initialisation du processeur, ou après une réinitialisation, le contenu de la mémoire PEPS est indéterminé. En conséquence, pour les trois premières instructions PFLoad, ce sont des données inutilisables ou des résidus qui seront retournés (les programmeurs devront alors ignorer les données retournées par les trois premières instructions PFLoad qui auront été délivrées).

Le multiplexeur 67 possède une entrée permettant de recevoir des données en sortie du sommet de la mémoire PEPS 65 et une autre entrée permettant de recevoir des données directement de la mémoire externe. La donnée externe est directement appliquée au multiplexeur 67 via la ligne 61. La mémoire PEPS 65 est reliée au multiplexeur 67 par la ligne 66. La sortie du multiplexeur 67 apparaît sur la ligne 68, qui est reliée au tampon à trois états 69. Le tampon de bus à trois états 69, quant à lui, est relié à un bus interne à 64 bits, désigné BIBUS, qui a été illustré figure 2 par la

ligne 29. Comme son nom l'indique, ce bus est bidirectionnel, c'est-à-dire qu'il peut délivree des données à l'unité à virgule flottante et, aussi bien, retourner au multiplexeur 62 des données en provenance de l'antémémoire. Le bus BIBUS 29 est relié, par l'intermédiaire du tampon 75, à un bus DABUS 76, qui a également une largeur de 64 bits. Le bus DABUS 76 relie le verrou de données d'entrée à virgule flottante 78 à l'antémémoire de données 70. Comme indiqué précédemment, le bus BIBUS 29 est bidirectionnel afin de permettre, dans le cas d'une concordance d'antémémoire, le retour au multiplexeur 62 des données résidant déjà dans l'antémémoire de données 70. Les données se trouvant dans l'antémémoire sont appliquées par la ligne 79 au bus BIBUS, par l'intermédiaire du tampon 74.

Lorsque le processeur N10 a besoin d'accéder à une donnée, cette donnée peut se trouver soit dans un système mémoire externe soit à l'intérieur de l'antémémoire de données interne. Lorsque la donnée est extérieure au processeur N10, l'instruction PFLoad agit de manière à charger dans la mémoire PEPS PFLoad 65 la donnée depuis la mémoire externe, en utilisant le bus externe 40 et par l'intermédiaire du multiplexeur 62. Après que trois instructions PFLoad successives (correspondant à des étages successifs de la mémoire PEPS) aient été émises, la donnée initiale est produite sur la ligne 66, puis ensuite délivrées sur le bus BIBUS 29. Une fois la mémoire PEPS pleine, chaque cycle PFLoad normal transfère la donnée du dessus de la mémoire PEPS au bus BIBUS 29, puis au verrou de données d'entrée à virgule flottante 78, via le bus DEABUS 76. Ainsi, chaque cycle PFLoad normal déclenche à l'intérieur du processeur un cycle de lecture correspondant.

Lorsqu'une nouvelle donnée arrive du bus de donnée externe, elle est normalement dirigée vers la base de la mémoire PEPS 65. Cependant, si la donnée adressée réside déjà dans l'antémémoire de données, on aura un cas de "concordance PFLoad" (synonyme de "concordance d'antémémoire") et on produira une exception. Dans le cas d'une concordance PFLoad, la logique de contrôle du bus de l'unité

B va attendre que tous les cycles de bus en cours soit achevés. Ensuite, elle transfèrera la donnée demandée de l'antémémoire de données 70 à la base de la mémoire PEPS 65 en suivant le chemin de données constitué par la ligne 79, le tampon 74, le bus BIBUS 29 (relié à l'entrée "1" du multiplexeur 62) et la ligne 63.

Le contrôleur doit attendre que tous les cycles de bus en cours soit achevés pour transférer la donnée de concordance PFLoad de l'antémémoire de données 70 à la base de la mémoire PEPS 65 et ce, pour deux raisons.

En premier lieu, il se peut qu'il y ait un cycle PFLoad en cours. Toute tentative d'inscription de la donnée de concordance PFLoad dans la mémoire PEPS avant que tous les cycles PFLoad en cours n'aient été achevés va détruire l'ordre des données se trouvant dans la mémoire PEPS.

En second lieu, le bus BIBUS 29 peut être déjà occupé par des données que l'on est en train de transférer vers l'unité à virgule flottante au moyen d'une instruction PFLoad se trouvant trois instructions en arrière. Il en résultera un conflit de bus si la donnée de concordance PFLoad en provenance de l'antémémoire est transféréé immédiatement sur le bus BIBUS. Pour empêcher l'apparition d'un conflit de bus, la logique de contrôle de bus de l'unité B met hors service le tampon 74 pendant un cycle d'horloge supplémentaire avant de remettre la donnée dans la mémoire PEPS 65. Après que l'unité B ait attendu que tous les cycles en cours soient achevés, la donnée de concordance PFLoad est placée à la base de la mémoire PEPS, et le traitement PFLoad s'exécute de façon habituelle.

Il est possible au processeur de fonctionner à une cadence plus rapide que le bus externe ou que la mémoire. Par exemple, si la vitesse du processeur est supérieure à celle du matériel externe, il se peut que la mémoire PEPS se vide. Il faut que la mémoire PEPS ait un nombre suffisant d'étages pour conserver le nombre maximum d'adresses délivrées par le processeur, et qu'elle soit également compatible avec l'architecture en pipeline du processeur. On notera que c'est pour cette raison que l'instruction PFLoad que

l'on décrit ici opère de façon dynamique.

Si l'on se réfère maintenant à la figure 5B, on y a illustré la mémoire PEPS PFLoad 65 constituée, dans le mode de réalisation actuellement préféré, de trois verrous ou registres par bit, d'un compteur de position de lecture 87 et d'un compteur de position d'écriture 86. Le compteur de position de lecture est relié au décodeur 88 qui, quant à lui, est relié au multiplexeur 3-vers-1 95 par les lignes 94 et 93. Ces lignes sont reliées aux lignes de sélection du multiplexeur 95 de telle sorte que, lors d'une opération PFLoad, le compteur de lecture 87 sélectionne le registre dans lequel le cycle PFLoad suivant lira la donnée.

Le compteur de position d'écriture 86 est relié au décodeur 85 qui, quant à lui, est relié aux registres A, B et C, respectivement par les lignes 92, 91 et 90. Les lignes 90 à 92 permettent de verrouiller dans le registre approprié les données apparaissant sur la ligne 63. En fonctionnement, le compteur de position d'écriture 86 pointe sur le verrou dans lequel le cycle PFLoad suivant inscrira la donnée retournée. Le compteur de position de lecture et le compteur de position d'écriture sont tous deux constitués de compteurs bouclés classiques 0-vers-2.

Le compteur de lecture est incrémenté lorsque l'instruction PFLoad se trouve à l'étape de réécriture ou se trouve en attente, le bus BIBUS n'est pas actif et la mémoire PEPS PFLoad n'est pas vide. Si la mémoire PEPS est vide, le compteur de lecture est incrémenté, bien que la donnée ne soit pas effectivement verrouillée dans la mémoire PEPS mais directement dirigée vers le verrou de données d'entrée à virgule flottante. Le compteur d'écriture est incrémenté lorsque le cycle de bus externe actuel est un PFLoad, ou lorsque l'on est en train d'inscrire, depuis l'antémémoire de données, une donnée PFLoad à la base de la mémoire PEPS dans le cas d'une concordance PFLoad. Les deux bits du compteur de position d'écriture sont utilisés pour produire les signaux d'écriture de la mémoire PEPS qui verrouillent la donnée dans la mémoire PEPS.

On a illustré figure 5A une variante du mode de

réalisation de la mémoire PEPS 65. La figure 5A montre une mémoire PEPS dans laquelle les données passent en série du verrou 1 au verrou 2, puis ensuite au verrou 3 avant d'être délivré en sortie sur la ligne 66. Bien que la mémoire PEPS de la figure 5A permettent de préserver convenablement la nature en pipeline des instructions PFLoad, elle n'a pas la même souplesse que la mémoire PEPS illustrée figure 5B. Par exemple, dans la mémoire PEPS de la figure 5B on pourrait inscrire des données dans le verrou 87 en un cycle d'horloge et les lire au cycle suivant sans avoir à traverser les autres registres. On peut ainsi rendre disponible certaines données de façon anticipée, si nécessaire.

Comme expliqué précédemment, le processeur N10 peut émettre jusqu'à trois cycles de bus simultanément en cours. Si le processeur délivre des cycles de bus à une cadence supérieure à celle de la mémoire externe, la mémoire PEPS PFLoad peut venir à se trouver vidée. Ceci implique que les trois cycles en cours soient des cycles PFLoad. Si un quatrième cycle PFLoad se trouve à l'étape de réécriture et qu'il n'y a pas de donnée disponible à charger dans le verrou de données 78, l'unité de contrôle du bus délivre un ordre de gel. Lorsque l'on demande un cycle PFLoad mais que la mémoire PEPS est vide, on doit geler le noyau jusqu'à ce que la donnée PFLoad soit disponible sur le bus DABUS (un PFLoad est en cours lorsque la donnée provenant de la mémoire PEPS ne peut pas être transférée au verrou de données 78 du fait que la mémoire PEPS PFLoad est vide, que le bus BIBUS est occupé ou que l'on a une non-concordance du tampon de traduction par transcodage).

On remarquera que, lorsque l'antémémoire de données effectue une lecture normale, l'unité d'antémémoire de données 70 pilote à la fois le bus DABUS 76 et le bus DBBUS 80, de sorte qu'il devient impossible de délivrer des données externes au verrou au cours du même cycle d'horloge. Cependant, si l'on utilise une instruction PFLoad, l'antémémoire de données 70 est contrôlée par la logique de l'unité B de manière à arrêter les circuits internes de pilotage de l'antémémoire (qui sont reliés au bus DABUS 76).

En arrêtant les circuits de pilotage de l'antémémoire reliés au bus DABUS 76, on peut délivrer des données externes au verrou de données 78 sans créer de situation de conflit de bus.

Une fois que la donnée PFLoad est devenue disponible sur le bus DABUS 76, on met fin au gel. La donnée externe est alors pilotée par le tampon 75 sur le bus DABUS, depuis le bus BIBUS 29. La donnée délivrée sur le bus BIBUS peut provenir soit de la mémoire PEPS 65 soit directement, par la ligne 61, du système de mémoire externe. Le multiplexeur 67 ne sélectionne la ligne 61 comme source de la donnée que lorsque la mémoire PEPS 65 est vide. Le fait que la mémoire PEPS 65 soit vide ou non dépend de la vitesse de la mémoire externe, de la fréquence à laquelle le programmeur émet des instructions PFLoad, etc. Si la séquence PFLoad est interrompue, alors la donnée va continuer à être délivrée à partir la mémoire externe aussi longtemps que l'on aura des ordres PFLoad en cours émis sur le bus externe 40. Les données externes référencées par les instructions PFLoad antérieures vont alors s'accumuler dans la mémoire PEPS 65 jusqu'à ce que le programmeur commence à émettre à nouveau des instructions PFLoad.

Si le programmeur émet des ordres PFLoad successifs tels que la mémoire PEPS de données 65 vienne à se vider avant qu'aucun cycle PFLoad en cours n'ait été achevé, les nouvelles données arrivant sur le bus de données externes 40 seront appliquées sur la ligne 61 par le multiplexeur 67, pour application directe au bus BIBUS 29. Ensuite, les données appliquées sur le bus BIBUS 29 seront envoyées par le tampon 75 sur le bus DABUS 76 afin de pouvoir être finalement mémorisées dans le verrou de données d'entrée à virgule flottante 78. Le verrou d'entrée de données fait effectivement partie de l'unité à virgule flottante du processeur N10 (le bus DBBUS n'est utilisé que lorsque l'on traite des charges sur 128 bits. Comme l'instruction PFLoad n'opère que sur des charges de 64 bits ou de 32 bits, le bus DBBUS ne concerne pas directement l'instruction PFLoad de la présente invention).

Lorsqu'une situation de concordance d'antémémoire apparaît, c'est-à-dire lorsque la donnée demandée réside effectivement dans l'antémémoire de données et non dans la mémoire externe, il se passe alors ce qui suit. Tout d'abord, le mot de donnée actuel doit être retourné soit de la mémoire PEPS PFLoad 65 soit du bus de données externe 40. A cet effet, la logique de contrôle de l'unité de contrôle du bus met hors service le tampon 75 et la donnée est retournée en suivant le trajet du bus BIBUS qui aboutit à l'entrée "1" du multiplexeur 62. Le multiplexeur 62, qui est également contrôlé par l'unité de contrôle du bus, retourne alors la donnée à la base de la mémoire PEPS 65, via la ligne 63.

Dans le cas où l'on a émis un certain nombre d'ordres PFLoad en cours dont aucun n'a encore retourné de donnée de la mémoire externe, et où une concordance d'antémémoire apparaît, la séquence des évènements est différente. Dans ce cas, le processeur ne peut, dans l'immédiat, rien faire avec le mot de données résidant actuellement dans l'antémémoire de données. L'unité de contrôle du bus doit tout d'abord attendre que toutes les instructions PFLoad précédemment emises aient retourné les données de la mémoire externe. Le premier mot de données retourné est transféré dans le verrou d'entrée de données 78, soit de la mémoire PEPS 65 soit directement du bus externe, tandis que les deux mots restants sont inscrits dans la mémoire PEPS 65. Finalement, le mot de donnée mémorisé dans l'antémémoire de données 70 est transféré dans la mémoire PEPS 65 en tant que dernier mot de cette mémoire PEPS. On aura ainsi une pénalisation temporelle notable dans le cas d'une concordance d'antémémoire, car le programmeur devra attendre que tous les cycles PFLoad en cours aient été achevés avant de transférer la donnée de l'antémémoire à la mémoire PEPS. Des explications qui précèdent, on voit que l'instruction PFLoad est optimisée pour le cas de la non-concordance d'antémémoire plutôt que pour le cas de la concordance d'antémémoire.

Sur la figure 4, on a représenté une variante du mode de

réalisation de la présente invention. Le circuit de la figure 4 est identique à celui de la figure 2, à l'exception du fait que la ligne 79, le tampon 74 et le trajet de retour du bus BIBUS aboutissant à l'entrée "1" du multiplexeur 62 ont été supprimés. A la place, on a prévu un bus 81. Le bus 81 relie directement le tampon interne 82 de l'antémémoire 70 à l'entrée "1" du multiplexeur 62. Le premier bénéfice procuré par cette variante du mode de réalisation est que l'utilisateur n'a pas besoin d'attendre que le tampon 69 se mette hors service avant d'inscrire la donnée dans la mémoire PEPS 65. Du fait que l'on a inclus un bus dédié au cas d'une concordance d'antémémoire, le programmeur n'a, également, pas besoin d'attendre, avant de poursuivre, que toutes les données externes provenant d'instructions PFLoad en attente aient été retournées. On n'a pas non plus de pénalisation en termes de cycles d'horloges supplémentaires, car la donnée est délivrée directement de l'antémémoire de données 70 à la mémoire PEPS 65 lorsque survient un cas de concordance d'antémémoire.

En revanche, le mode de réalisation préférentiel de la figure 2 présente l'avantage d'une surface de silicium moindre, car on n'a besoin que d'un seul bus tandis que, dans la variante de réalisation de la figure 4, on a besoin du bus supplémentaire 81.

La figure 6 est un organigramme illustrant les opérations de contrôle effectuées par la logique de contrôle du bus associée à l'instruction PFLoad de la présente invention. Cet organigramme résume la description donnée ci-dessus de la logique de contrôle du bus mise en oeuvre lors de l'exécution de l'instruction PFLoad. Sur la figure 6, l'organigramme de l'instruction PFLoad commence au bloc 100, qui correspond à une requête PFLoad. Une fois la requête émise, la logique de contrôle du bus et l'unité B doivent déterminer si la mémoire PEPS 65 est vide ou si le bus BIBUS 29 est occupé. Ceci a été illustré par le test 101. Si l'une de ces deux conditions est vérifiée, le contrôleur de bus attendra jusqu'à ce que la condition change. S'il y a une donnée présente dans la mémoire PEPS et si le bus BIBUS

n'est pas occcupé, le contrôleur passe alors au bloc 102, où la donnée résidant dans la mémoire PEPS est transférée vers le verrou de données à virgule flottante 78. Au test 103, le contrôleur de bus examine si l'on a eu ou non une concordance PFLoad, c'est-à-dire, en d'autres termes, si l'on a eu une concordance d'antémémoire. Si la donnée réside en mémoire externe et ne se trouve pas dans l'antémémoire de données 70, le contrôleur de bus va alors émettre un cycle de bus PFLoad, comme illustré au bloc 104. Le traitement continue de façon normale jusqu'à atteindre la fin de l'instruction.

Il peut également se trouver qu'une concordance PFLoad apparaisse, auquel cas la logique de contrôle du bus doit examiner si l'on a un cycle PFLoad en cours ou si le bus BIBUS est occupé ou non. Ceci est illustré sur la figure 6 par le bloc de test 105. Si l'on a des cycles en cours, ou si le bus BIBUS est occupé, la logique de contrôle du bus doit attendre jusqu'à ce que tous les cycles de bus en cours soient achevés et que le bus BIBUS soit disponible pour transférer la donnée. Lorsque ceci arrive, la donnée est alors transférée de l'antémémoire de données vers la base de la mémoire PEPS, comme illustré par le bloc 106. Le transfert peut avoir lieu de la manière décrite à propos de la figure 2, la donnée suivant la ligne 79, via le tampon 74, pour être retourné par le bus BIBUS à l'entrée "1" du multiplexeur 62. Dans la variante de réalisation, la donnée provenant de l'antémémoire de données 70 passe directement, le long de la ligne 81, à l'entrée "1" du multiplexeur 62. Ici encore, l'opération suit son cours normal jusqu'à atteindre la fin de l'instruction.

Afin de mieux comprendre le fonctionnement et les avantages de la présente invention, on va maintenant se référer à l'exemple d'opération donné figure 3. L'exemple d'opération donné par l'équation de la figure 3 est le suivant :

$$k.V_1 + V_2 \rightarrow V_2$$

où k est une constante prédéterminée donnée, V_1 est l'un

de 1000 vecteurs à 1000 éléments différents mémorisés dans la mémoire externe et V_2 est un vecteur mémorisé de façon interne dans l'unité d'antémémoire de données 70. Pour ce calcul particulier, le processeur doit réutiliser le vecteur V_2 un millier de fois (une fois pour chacun des différents éléments de chaque vecteur V_1), tandis que les divers éléments de V_1 ne sont utilisés qu'une seule fois. En utilisant l'instruction PFLoad de la présente invention, le processeur charge les divers éléments du vecteur V_2 dans le registre à virgule flottante sans réécrire sur les données V_2 résidant dans l'antémémoire de données.

La figure 3 montre une première instruction PFLoad, référencée $PFLD_1$, délivrée à l'adresse de V_{11} afin de charger le premier élément du premier vecteur V_1 à l'emplacement de mémorisation à virgule flottante F_2. L'ordre $PFLD_1$ a besoin de deux cycles d'horloge pour commencer. Sans attendre que la donnée soit retournée de la mémoire externe, on émet une seconde instruction PFLoad, référencée $PFLD_2$, à l'adresse de l'élément de vecteur V12, que l'on dirige vers l'emplacement de mémorisation à virgule flottante F_3. D'autres ordres PFLoad sont lancés pour les éléments vectoriels V_{13}, V_{14}, V_{15}, etc. On peut demander un nouvel ordre PFLoad tous les deux cycles d'horloge, car le bus externe permet d'émettre une nouvelle adresse tous les deux cycles d'horloge.

Lorsque le quatrième ordre PFLoad, c'est-à-dire l'ordre $PFLD_4$, est émis, la mémoire externe commence à retourner la donnée de la première instruction PFLoad. Cette donnée retournée correspond en fait à l'instruction PFLoad située trois instructions en arrière, de sorte que le registre destinataire spécifié par la quatrième instruction PFLoad est le registre F_2. L'arrivée de la donnée pour l'ordre PFLoad situé trois instructions en arrière est illustré par les traits interrompus et les flèches allant, en diagonale, de l'emplacement de mémorisation à virgule flottante à l'adresse de l'ordre PFLoad. Le programmeur peut continuer à spécifier des ordres PFLoad tous les deux cycles d'horloge, utilisant ainsi la pleine capacité de largeur de bande du bus externe. En pratique, on a un total de six cycles

d'horloge de latence entre l'instant où le programmeur spécifie l'adresse de la donnée et l'instant où cette donnée est retournée. Il faut donc au total six cycles d'horloge au système mémoire pour délivrer la donnée, même si un nouveau cycle de bus commence tous les deux cycles d'horloge. Ceci correspond à un triplement de la vitesse à laquelle on peut traiter les cycles. Sans le concept en pipeline, le programmeur serait contraint d'émettre un nouveau cycle tous les six cycles d'horloge au lieu de l'émettre tous les deux cycles d'horloge.

On comprendra que, bien évidemment, la description ci-dessus suppose que chaque instruction PFLoad corresponde à une non-concordance de l'antémémoire de données. S'il s'agissait d'un chargement à virgule flottante normal avec un processeur de l'art antérieur, dès lors que le programmeur aurait tenté d'exécuter une seconde instruction PFLoad, l'unité de contrôle du bus aurait répondu que l'on avait un accès en cours et aurait gelé la totalité des six cycles d'horloge en attente du retour des charges du bus en cours. Ainsi, avec un processeur de l'art antérieur, le programmeur ne peut émettre de nouvelle adresse que tous les six cycles d'horloge, au lieu de les émettre tous les deux cycles d'horloge. Le concept en pipeline associé à l'instruction PFLoad de la présente invention permet ainsi à l'utilisateur d'émettre des adresses à une cadence supérieure, en dépit du fait que la donnée ne réside pas dans l'antémémoire incorporée à la puce. On peut ainsi traiter et accéder efficacement à des structures de données de très grandes dimensions.

La présente invention permet de mémoriser le vecteur V_2 sur la même puce, tandis que l'on conserve l'élément V_1 dans la mémoire externe, de sorte que l'utilisateur peut référencer les éléments de V_1 au moyen d'une instruction PFLoad et référencer V_2 au moyen d'une instruction normale de chargement. Dans un microprocesseur typique de l'art antérieur sans instruction PFLoad, chaque fois que l'utilisateur référence V_1, les éléments de données sont amenés dans l'antémémoire, faisant ainsi disparaître les éléments

du vecteur V_2. Ceci produit un retard supplémentaire la fois suivante où l'on doit charger l'élément de V_2.

REVENDICATIONS

1. Un microprocesseur à architecture en pipeline, comprenant une antémémoire de données (22,70), une unité de traitement et une unité de contrôle de bus qui contrôle l'accès à une mémoire externe par un bus externe (40), et pourvu d'une instruction permettant d'effectuer en pipeline le chargement d'une donnée dans le microprocesseur,

caractérisé en ce que cette instruction met en oeuvre :

- des premiers moyens de circuit (30,62,65,67), pour mémoriser une donnée en pipeline lorsqu'elle est délivrée depuis la mémoire externe,

- des moyens formant bus (29), reliés aux premiers moyens de circuit, pour transférer à un verrou de données (78) la donnée mémorisée dans ces premiers moyens de circuit, et

- des moyens de contrôle (30), pour contrôler les premiers moyens de circuit et l'antémémoire de données de manière à pouvoir transférer en pipeline la donnée depuis la mémoire externe jusqu'au verrou de données situé à l'intérieur du microprocesseur.

2. Le microprocesseur de la revendication 1, dans lequel la donnée externe est transférée directement au verrou de données, sans être placée dans l'antémémoire.

3. Le microprocesseur de la revendication 2, dans lequel ladite instruction peut émettre à la pleine largeur de bande du bus les adresses de données supplémentaires associées à la mémoire externe lorsque la donnée à laquelle on accède ne se trouve pas dans l'antémémoire de données, permettant ainsi d'accroître la vitesse d'accès à cette mémoire externe.

4. Le microprocesseur de la revendication 3, dans lequel l'antémémoire de données (70) est reliée aux moyens formant bus (29) par un tampon (74) tel que la donnée résidant dans l'antémémoire soit retournée aux premiers moyens de circuit en cas de concordance d'antémémoire, ce tampon étant également contrôlé par les moyens de contrôle.

5. Le microprocesseur de la revendication 4, dans lequel les premiers moyens de circuit comprennent :

- un premier multiplexeur (62), comportant une entrée reliée au bus externe (40), une autre entrée reliée aux moyens formant bus (29) et une sortie,

- une mémoire du type premier entré-premier sorti (65), avec une entrée et une sortie, cette entrée étant reliée à la sortie du premier multiplexeur, et

- un second multiplexeur (67), avec une entrée (PFLD) reliée à l'entrée de la mémoire du type premier entré-premier sorti, une autre entrée (DBS) reliée au bus externe (40) et une sortie, la sortie de ce second multiplexeur étant reliée aux moyens formant bus (29) par l'intermédiaire d'un autre tampon,

le premier multiplexeur délivrant des données à la mémoire du type premier entré-premier sorti en provenance soit de la mémoire externe, dans le cas d'une non-concordance d'antémémoire, soit de l'antémémoire de données, dans le cas d'une concordance d'antémémoire, et

le second multiplexeur délivrant les données aux moyens formant bus depuis la mémoire du type premier entré-premier sorti, ou bien directement depuis le bus externe.

6. Le microprocesseur de la revendication 5, dans lequel l'antémémoire de données (70) est reliée au verrou de données (78) par un bus de données (76), lesdits moyens formant bus (29) étant reliés à ce bus de données par l'intermédiaire d'un troisième tampon (75), ce troisième tampon et l'antémémoire de données étant contrôlés par les moyens de contrôle de manière à refuser à l'antémémoire de données l'accès au bus de données lorsque les moyens formant bus sont en train de transférer des données au verrou de données, de manière à éviter des conflits de bus.

7. Le microprocesseur de la revendication 6, dans lequel chacun des tampons (69, 74, 75) est un tampon à trois états logiques.

8. Un microprocesseur à architecture en pipeline, comprenant une antémémoire de données (22,70) et une unité de traitement avec un verrou de données (78), et pourvu d'une instruction permettant d'accéder à des données provenant d'une mémoire externe par l'intermédiaire d'un bus externe (40),

caractérisé en ce que cette instruction met en oeuvre :

- des moyens de mémoire (65), pour accumuler des données,
- des premiers moyens de circuit (60,62), pour délivrer aux moyens de mémoire des données en provenance de la mémoire externe en cas de survenance d'une non-concordance d'antémémoire,
- des seconds moyens de circuit (67,69,74,75), pour délivrer au verrou de données les données accumulées dans les moyens de mémoire, ces seconds moyens de circuit reliant également l'antémémoire de données aux premiers moyens de circuit de manière à retourner aux moyens de mémoire la donnée résidant dans l'antémémoire, en cas de concordance d'antémémoire, et
- des moyens de contrôle de bus (30), reliés aux premiers et aux seconds moyens de circuit, aux moyens de mémoire et à l'antémémoire de données, pour contrôler le flux en pipeline de données allant de la mémoire externe au verrou de données en cas de non-concordance d'antémémoire, et pour contrôler le retour des données de l'antémémoire aux moyens mémoire en cas de concordance d'antémémoire.

9. Le microprocesseur de la revendication 8, dans lequel les moyens de mémoire (65) comportent une mémoire du type premier entré-premier sorti.

10. Le microprocesseur de la revendication 9, dans lequel les premiers moyens de circuit comprennent un premier multiplexeur (62) et un premier tampon (60), ce premier tampon reliant le bus externe (40) à l'une des entrées du premier multiplexeur, et ce premier multiplexeur ayant son autre entrée reliée aux seconds moyens de circuit et sa sortie reliée à la mémoire du type premier entré-premier sorti (65).

11. Le microprocesseur de la revendication 10, dans lequel le premier tampon (60) relie également le bus externe (40) aux seconds moyens de circuit de manière à pouvoir transférer directement les données au verrou de données depuis la mémoire externe lorsque la mémoire du type premier entré-premier sorti est vide.

12. Le microprocesseur de la revendication 11, dans lequel les seconds moyens de circuit comprennent un second multiplexeur (67) comportant une entrée reliée de manière à recevoir en pipeline les données de la mémoire du type premier entré-premier sorti (65), une autre entrée reliée au premier tampon (60) et une sortie,

les seconds moyens de circuit comprenant en outre un second tampon (69) relié à la sortie du second multiplexeur et un bus bidirectionnel (29) permettant de délivrer des données au verrou de données (78) et de retourner les données de ce verrou de données à ladite autre entrée du premier multiplexeur.

13. Le microprocesseur de la revendication 12, dans lequel la mémoire du type premier entré-premier sorti (65) comporte :

- une pluralité de registres (A, B, C),
- des premiers moyens sélecteurs (85, 86), pour sélectionnner, parmi cette pluralité de registres, celui dans lequel on doit inscrire la donnée, et
- des seconds moyens sélecteurs (87, 88, 95), pour déterminer, parmi cette pluralité de registres, celui dans lequel on doit lire la donnée.

14. Le microprocesseur de la revendication 13, dans lequel ladite instruction peut émettre des adresses en direction de la mémoire externe en utilisant la pleine capacité de largeur de bande du bus externe.

15. Un processeur à architecture en pipeline, comprenant une antémémoire de données (70), une unité de traitement et

une unité de contrôle de bus (30) qui contrôle l'accès à une mémoire externe par un bus externe (40), et pourvu d'une instruction permettant d'effectuer en pipeline le chargement d'une donnée dans le microprocesseur,

caractérisé en ce que cette instruction met en oeuvre :

- un premier tampon (60), relié au bus externe, pour délivrer au processeur des données en provenance de la mémoire externe,

- des premiers moyens de circuit (30;62,65,67), pour mémoriser une donnée en pipeline lorsqu'elle est délivrée depuis la mémoire externe par ce tampon,

- un premier bus (29), relié aux premiers moyens de circuit, pour transférer à un verrou de données (78) la donnée mémorisée dans ces premiers moyens de circuit, et

- des moyens de contrôle (30), pour contrôler ledit premier tampon, les premiers moyens de circuit et l'antémémoire de données de manière à pouvoir transférer en pipeline la donnée depuis la mémoire externe jusqu'au verrou de données situé à l'intérieur du microprocesseur.

16. Le processeur de la revendication 15, comprenant en outre un second bus (81) reliant l'antémémoire de données (70) aux premiers moyens de circuit de telle sorte que la donnée résidant dans l'antémémoire soit retournée aux premiers moyens de circuit en cas de survenance d'une concordance d'antémémoire.

17. Le processeur de la revendication 16, dans lequel les premiers moyens de circuit comprennent :

- un premier multiplexeur (62), comportant une entrée reliée audit premier tampon (60), une autre entrée reliée audit second bus (81) et une sortie,

- une mémoire du type premier entré-premier sorti (65), avec une entrée reliée à la sortie du premier multiplexeur et une sortie, et

- un second multiplexeur (67), avec une entrée (PFLD) reliée à la sortie de la mémoire du type premier entré-premier sorti, une autre entrée (DBS) reliée audit premier

tampon (60) et une sortie, cette sortie du second multiplexeur étant reliée audit premier bus (29) par l'intermédiaire d'un troisième tampon (69),

le premier multiplexeur délivrant des données à la mémoire du type premier entré-premier sorti en provenance soit de la mémoire externe, dans le cas d'une non-concordance d'antémémoire, soit de l'antémémoire de données, dans le cas d'une concordance d'antémémoire, et

le second multiplexeur délivrant les données audit premier bus depuis la mémoire du type premier entré-premier sorti, ou bien directement depuis ledit premier tampon lorsque cette mémoire du type premier entré-premier sorti est vide.

18. Un procédé pour accéder, dans un processeur comprenant une unité de traitement, une antémémoire de données (70) et une liaison à un système de mémoire externe, à une donnée se trouvant dans cette mémoire externe,

caractérisé en ce qu'il comporte les étapes suivantes :

(a) adressage de la donnée dans la mémoire externe,

(b) inscription de cette donnée, depuis la mémoire externe, dans une mémoire du type premier entré-premier sorti (65),

(c) lecture de cette donnée, depuis cette mémoire du type premier entré-premier sorti, vers un bus (29), et

(d) transfert de cette donnée sur ce bus vers un verrou de données (78) situé à l'intérieur de l'unité de traitement.

19. Le procédé de la revendication 18, dans lequel les étapes (a) à (d) sont exécutées en pipeline.

20. Le procédé de la revendication 19, comprenant en outre l'étape consistant à retourner la donnée de l'antémémoire de données (70) à la mémoire du type premier entré-premier sorti (65) en cas de survenance d'une concordance d'antémémoire.

21. Le procédé de la revendication 20, comprenant en outre l'étape de transfert direct de la donnée de la mémoire externe au verrou de données (78) lorsque la mémoire du type premier entré-premier sorti (65) est vide.

22. Le procédé de la revendication 20, comprenant en outre l'étape consistant à attendre l'achèvement des cycles de bus en cours avant de retourner la donnée de l'antémémoire de données (70) à la mémoire du type premier entré-premier sorti (65), en cas de survenance d'une concordance d'antémémoire.

2643166

FIG_1

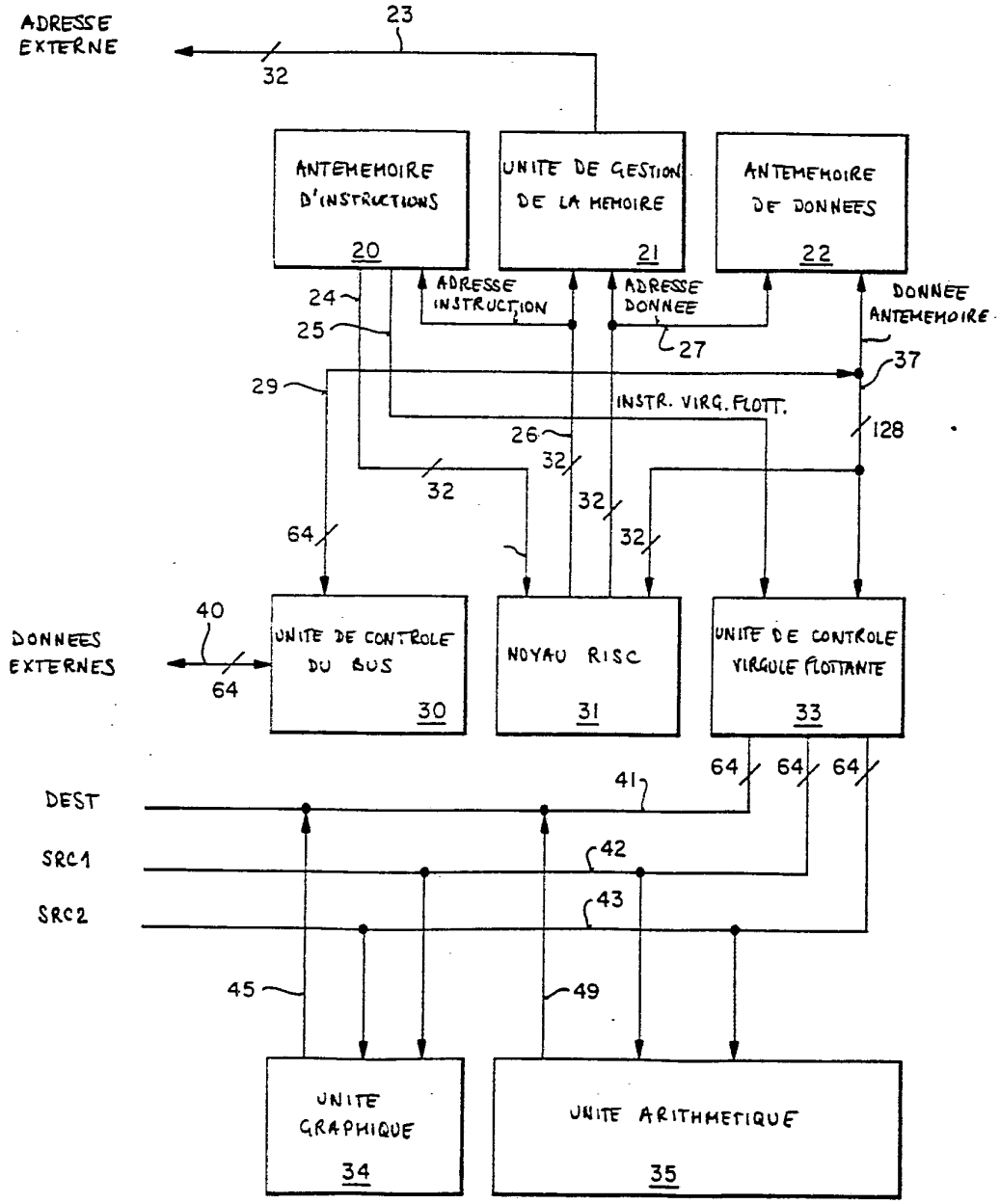


FIG_2

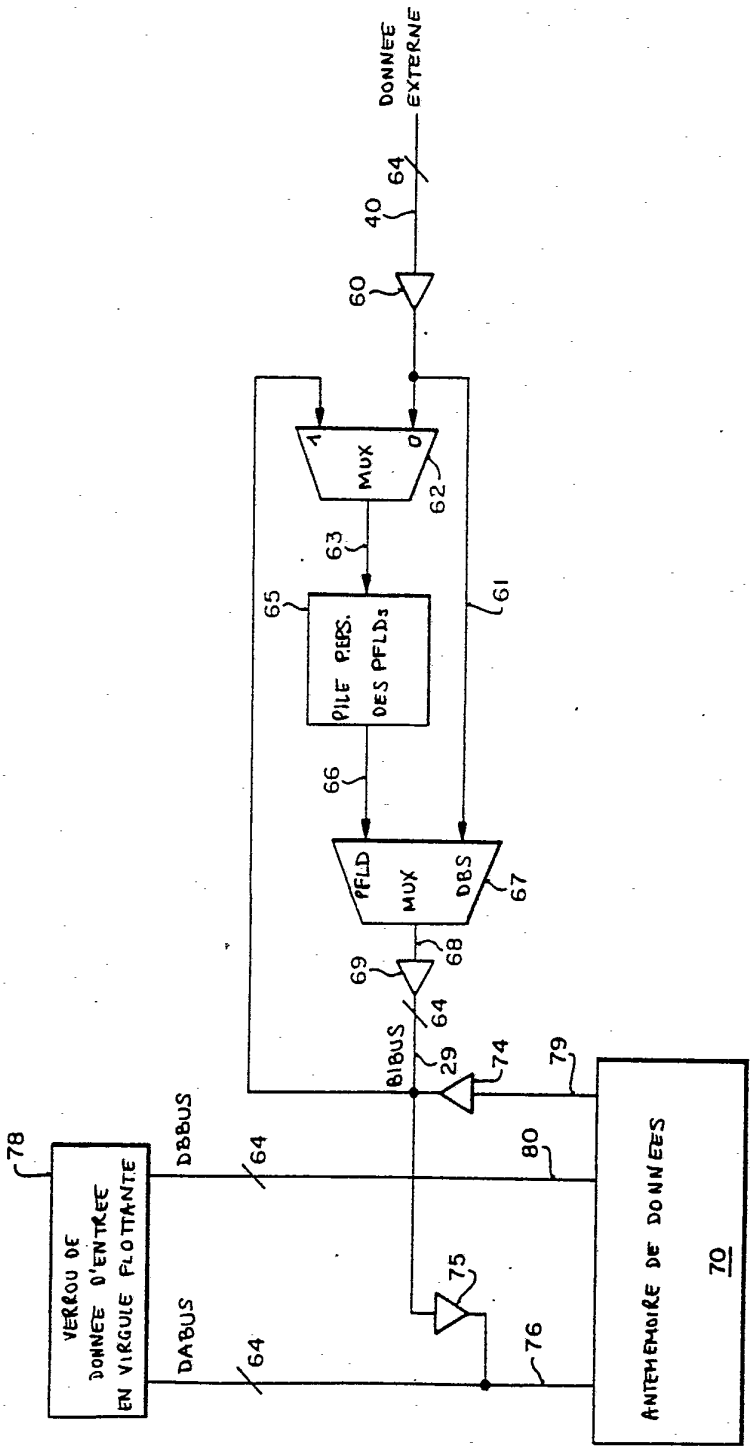

DONNEE EXTERNE
64
40
60
1
MUX
0
62
63
65
PILE PBS. DES PFLDs
61
66
PFLD
MUX
DBS
67
68
69
64
BIBUS
29
74
79
78
DBBUS
64
80
VERROU DE DONNEE D'ENTREE EN VIRGULE FLOTTANTE
ANTEMEMOIRE DE DONNEES
70
75
DABUS
64
76

FIG_3

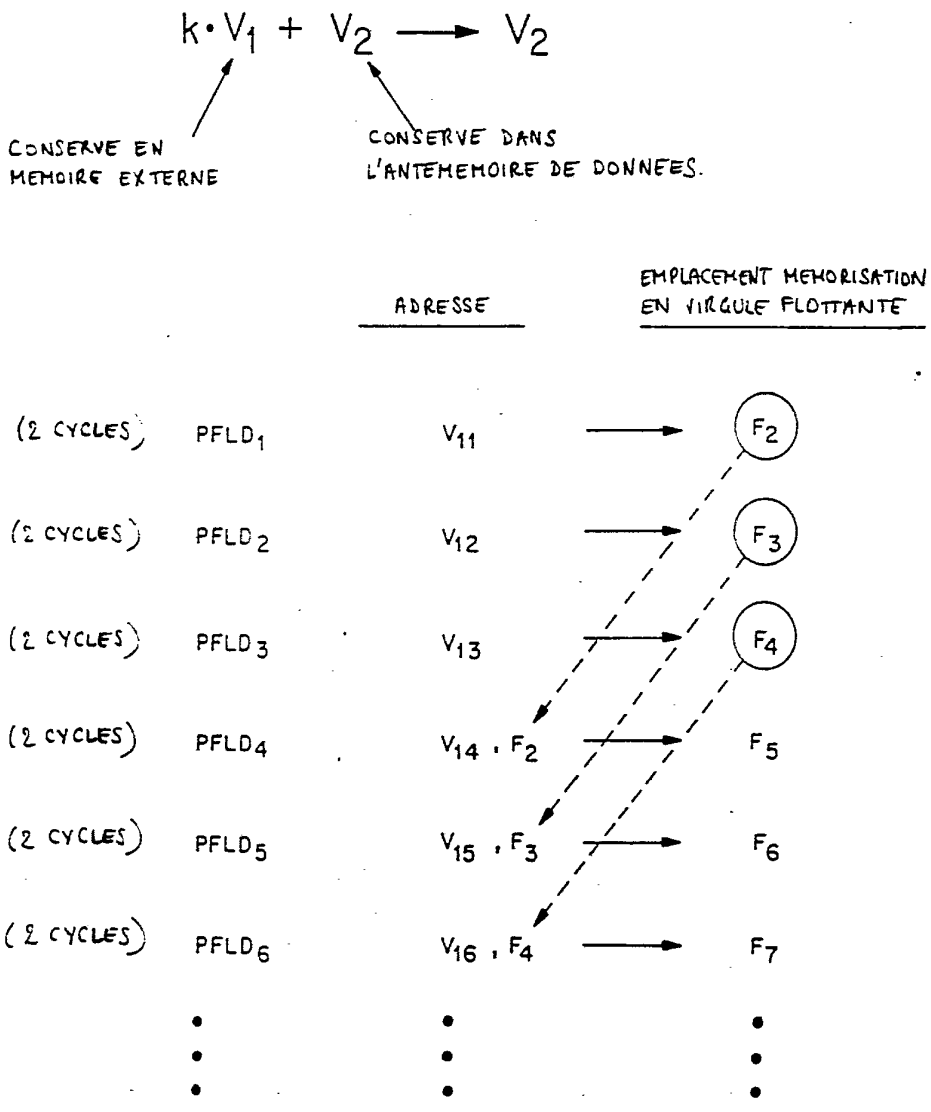

k·V1 + V2 → V2
CONSERVE EN MEMOIRE EXTERNE
CONSERVE DANS L'ANTEMEMOIRE DE DONNEES.
ADRESSE
EMPLACEMENT MEMORISATION EN VIRGULE FLOTTANTE
(2 CYCLES) PFLD1 V11 F2
(2 CYCLES) PFLD2 V12 F3
(2 CYCLES) PFLD3 V13 F4
(2 CYCLES) PFLD4 V14 , F2 F5
(2 CYCLES) PFLD5 V15 , F3 F6
(2 CYCLES) PFLD6 V16 , F4 F7

FIG — 4

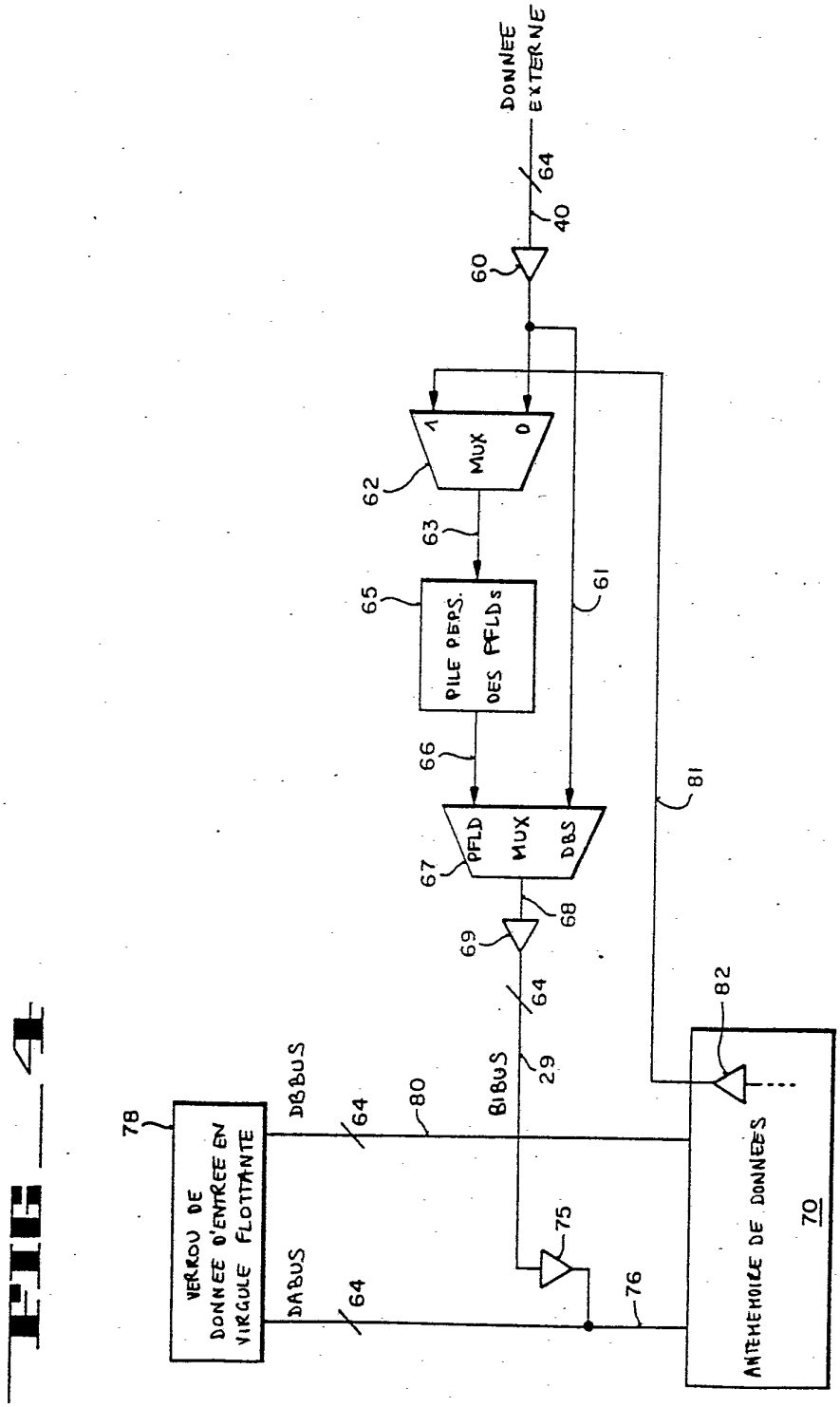

DONNEE EXTERNE
64
40
60
1
MUX
0
62
63
65
PILE PEPS. DES PFLDs
61
66
81
PFLD
MUX
DBS
67
68
69
64
29
82
78
VERROU DE DONNEE D'ENTREE EN VIRGULE FLOTTANTE
DBBUS
64
80
BIBUS
DABUS
64
75
76
ANTEMEMOIRE DE DONNEES
70

FIG 5A

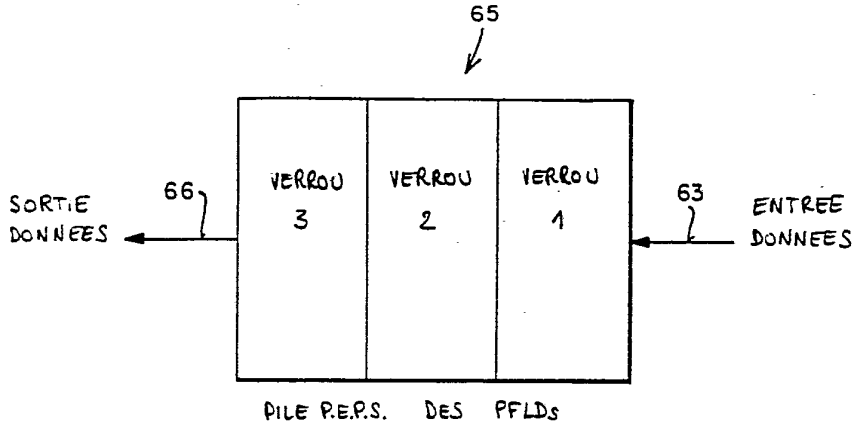

65
SORTIE DONNEES
66
VERROU 3
VERROU 2
VERROU 1
63
ENTREE DONNEES
PILE P.E.P.S. DES PFLDs

FIG 5B

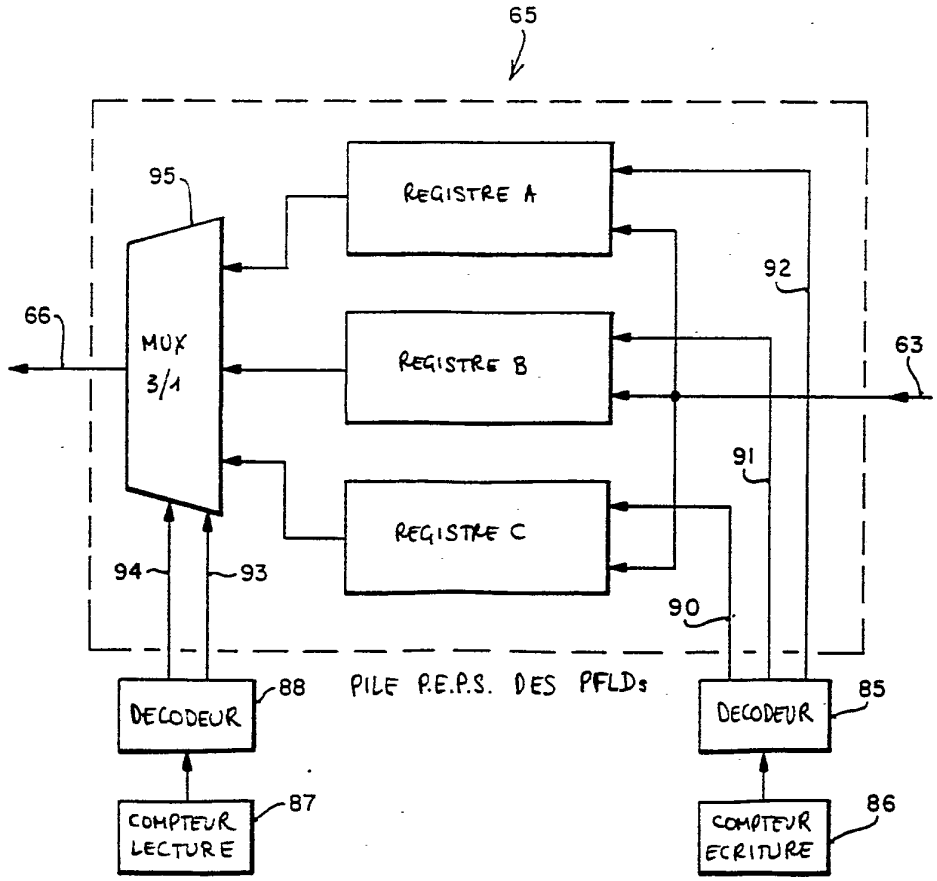

65
95
REGISTRE A
92
66
MUX 3/1
REGISTRE B
63
91
REGISTRE C
94
93
90
PILE P.E.P.S. DES PFLDs
DECODEUR
88
DECODEUR
85
COMPTEUR LECTURE
87
COMPTEUR ECRITURE
86

FIG_6

ORGANIGRAMME D'UNE INSTRUCTION PFLD