(12) DEMANDE INTERNATIONALE PUBLIÉE EN VERTU DU TRAITÉ DE COOPÉRATION EN MATIÈRE DE BREVETS (PCT)

(19) Organisation Mondiale de la Propriété Intellectuelle
Bureau international

(43) Date de la publication internationale
02 janvier 2020 (02.01.2020)

WIPO | PCT

(10) Numéro de publication internationale
WO 2020/002783 A1

(51) Classification internationale des brevets :
G06F 15/80 (2006.01) *G06F 9/38* (2018.01)

(21) Numéro de la demande internationale :
PCT/FR2019/051156

(22) Date de dépôt international :
21 mai 2019 (21.05.2019)

(25) Langue de dépôt : français

(26) Langue de publication : français

(30) Données relatives à la priorité :
18 56000 29 juin 2018 (29.06.2018) FR

(71) Déposant : **VSORA** [FR/FR] ; 13-15 rue Jeanne Braconnier, 92360 MEUDON LA FORET (FR).

(72) Inventeurs : **MAALEJ, Khaled** ; 217 rue du Faubourg Saint Honoré, 75008 PARIS (FR). **NGUYEN, Trung-Dung** ; 7 rue Eric Tabarly, 91300 MASSY (FR). **SCHMITT, Julien** ; 4 rue de la Pérouse, 91300 MASSY (FR). **BERNARD, Pierre-Emmanuel** ; 1 rue joseph Fouriaux, 92160 ANTONY (FR).

(74) Mandataire : **PLASSERAUD IP** ; 66 rue de la Chaussée d'Antin, 75440 PARIS CEDEX 09 (FR).

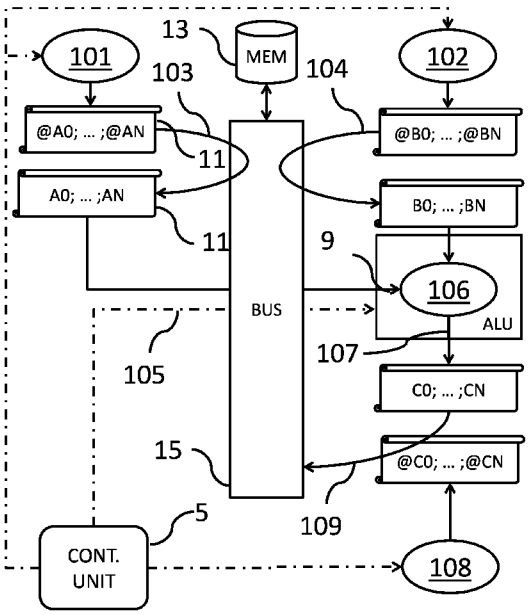
(81) États désignés *(sauf indication contraire, pour tout titre de protection nationale disponible)* : AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

WO 2020/002783 A1

(54) **Title:** ASYNCHRONOUS PROCESSOR ARCHITECTURE

(54) **Titre :** ARCHITECTURE DE PROCESSEUR ASYNCHRONE

Fig. 5



(57) **Abstract:** The invention concerns a data processing method comprising: - a control unit, at least one ALU (9), a set of registers (11), a memory (13) and a memory interface (15). The method comprises: a) obtaining (101, 102) the memory addresses of the operands; b) reading (103, 104) the operands in the memory (13); c) transmitting (105) an instruction to execute calculations to the ALU (9) without an addressing instruction; d) executing all the elementary operations (106) by the ALU (9) receiving as input each of the operands from the registers (11); e) storing (107) the data forming results of the processing operation on the registers (11); f) obtaining (108) a memory address for each of the data items forming a result of the processing operation; g) writing (109) the results in the memory (13), for storage and via the memory interface (15), by means of the obtained memory addresses.

(57) **Abrégé :** Un procédé de traitement de données comprenant : - une unité de commande, au moins une ALU (9), un jeu de registres (11), une mémoire (13) et une interface mémoire (15). Le procédé comprend : a) obtenir (101, 102) les adresses mémoire des opérandes; b) lire (103, 104) en mémoire (13) les opérandes; c) transmettre (105) une instruction d'exécution de calculs à destination de l'ALU (9) sans instruction d'adressage; d) exécuter l'ensemble des opérations élémentaires (106) par l'ALU (9) recevant en entrées chacun des opérandes depuis les registres (11); e) stocker (107) les données formant résultats du traitement sur les registres (11); f) obtenir (108) une adresses mémoire pour chacune des données formant résultat du traitement; g) écrire (109) en mémoire (13) les résultats, pour stockage et via l'interface mémoire (15), au moyen des adresses mémoire obtenues.

[Suite sur la page suivante]

(84) États désignés *(sauf indication contraire, pour tout titre de protection régionale disponible)* : ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), eurasien (AM, AZ, BY, KG, KZ, RU, TJ, TM), européen (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Publiée:

— *avec rapport de recherche internationale (Art. 21(3))*

Architecture de processeur asynchrone

L'invention relève du domaine des processeurs et de leur architecture fonctionnelle.

Classiquement, un dispositif informatique comprend un ensemble d'un ou plusieurs processeurs. Chaque processeur comprend une ou plusieurs unités de traitement, ou PU pour « *Processing Units* ». Chaque PU comprend un ou plusieurs organes de calcul appelés unités arithmétiques et logiques, ou ALU pour « *Arithmetic-Logic Unit* ». Pour disposer d'un dispositif informatique performant, c'est-à-dire rapide pour effectuer des opérations informatiques, il est classique de prévoir un nombre élevé d'ALU. Ainsi, les ALU peuvent traiter des opérations en parallèle, c'est-à-dire en même temps. L'unité de temps est alors un cycle de calcul. Il est donc commun de quantifier la puissance de calcul du dispositif informatique en nombre d'opérations qu'il est capable d'effectuer par cycle de calcul.

Néanmoins, une part importante de la puissance de calcul d'un dispositif informatique est consommée pour gérer les accès mémoire. Un tel dispositif comprend un ensemble mémoire, comprenant lui-même un ou plusieurs organes mémoire, chacun disposant d'un nombre fixe d'emplacements mémoire sur lesquels des données informatiques peuvent être stockées durablement. Au cours des traitements informatiques, les ALU reçoivent en entrée des données issues des organes mémoires et fournissent en sortie des données qui sont, à leur tour, stockées sur les organes mémoires. On comprend alors que, outre le nombre d'ALU, le nombre d'organes mémoire est un autre critère déterminant de la puissance de calcul du dispositif.

Le routage des données entre les ALU et les organes mémoire, dans les deux sens, est assuré par un bus du dispositif. Le terme « bus » est utilisé ici dans son sens général de système (ou interface) de transfert de données incluant du matériel (circuit d'interface) et les protocoles régissant les échanges. Le bus transmet les données elles-mêmes, des adresses et des signaux de commande. Chaque bus présente, lui aussi, des limites matérielles

et logicielles de sorte que le routage des données est limité. Notamment, le bus présente un nombre limité de ports côté organe mémoire et un nombre limité de ports côté ALUs. Ainsi, au cours d'un cycle de calcul, un emplacement mémoire est accessible via le bus dans un unique sens (en « lecture » ou en « écriture »). En outre, au cours d'un cycle de calcul, un emplacement mémoire est accessible pour une unique ALU.

Entre le bus et les ALUs, un dispositif informatique comprend généralement un jeu de registres et d'organes mémoires locaux, qui peuvent être vus comme des mémoires distinctes des organes mémoires précités. Pour faciliter la compréhension, on distingue ici les « registres », destinés à stocker des données en tant que telles, et les « organes mémoires locaux », destinés à stocker des adresses mémoire. À chaque registre sont attribuées les ALUs d'une PU. Une PU se voit attribuée plusieurs registres. La capacité de stockage des registres est très limitée en comparaison des organes mémoire mais leur contenu est directement accessible aux ALUs.

Pour effectuer les calculs, chaque ALU doit généralement, dans un premier temps, obtenir les données d'entrée du calcul, typiquement les deux opérandes d'un calcul élémentaire. Une opération de « lecture » de l'emplacement mémoire correspondant via le bus pour importer chacun des deux opérandes sur un registre est donc mise en œuvre. Puis, l'ALU effectue l'opération de calcul en elle-même à partir des données d'un registre et en exportant le résultat sous forme d'une donnée sur un registre. Enfin, une opération d' « écriture » est mise en œuvre pour enregistrer, dans un emplacement mémoire, le résultat du calcul. Au cours d'une telle opération d'écriture, le résultat stocké sur le registre est enregistré dans un emplacement mémoire via le bus. Chacune des opérations consomme *a priori* un ou plusieurs cycles de calcul.

Dans les dispositifs informatiques connus, il est usuel d'essayer de faire exécuter plusieurs opérations (ou plusieurs instructions) au cours d'un même cycle de calcul, afin de réduire le nombre total de cycles de calcul et donc d'augmenter l'efficacité. On parle alors de « chaînes de traitement » ou « pipelines » parallèles. Cependant, il existe souvent de nombreuses

dépendances des opérations les unes par rapport aux autres. Par exemple, il est impossible d'effectuer un calcul élémentaire tant que les opérandes n'ont pas été lus et qu'ils ne sont pas accessibles sur un registre pour l'ALU. Mettre en œuvre des chaînes de traitement implique donc de vérifier la dépendance des opérations (des instructions) les unes par rapport aux autres, ce qui est complexe et donc coûteux.

Usuellement, plusieurs opérations indépendantes sont mises en œuvre au cours d'un même cycle de calcul. Généralement, pour une ALU donnée et au cours d'un même cycle de calcul, il est possible d'effectuer une opération de calcul et une opération de lecture ou d'écriture. En revanche, pour une ALU donnée et au cours d'un même cycle de calcul, il est impossible d'effectuer à la fois une opération de lecture et une opération d'écriture (dans le cas d'organes mémoire mono-port). D'autre part, les accès mémoire (le bus) ne permettent pas d'effectuer, au cours d'un même cycle de calcul et pour un emplacement mémoire donné, des opérations de lecture ou d'écriture pour deux ALUs distinctes l'une de l'autre.

Par conséquent, il est connu d'effectuer un calcul élémentaire et d'écrire le résultat obtenu en mémoire au cours d'un même cycle de calcul. L'économie en termes de cycles de calculs (ou de ressources informatiques) reste faible.

L'invention vient améliorer la situation.

Il est proposé un procédé de traitement de données, décomposable en un ensemble d'opérations élémentaires à effectuer, mis en œuvre par un dispositif informatique, ledit dispositif comprenant :

- une unité de commande ;
- au moins une unité arithmétique et logique ;
- un jeu de registres aptes à alimenter en données formant opérande des entrées de ladite première unité arithmétique et logique et aptes à être alimentés en données issues des sorties de ladite unité arithmétique et logique ;
- une mémoire ;
- une interface mémoire par l'intermédiaire de laquelle des données sont

transmises et routées entre les registres et la mémoire.

Le procédé comprend :

a) obtenir les adresses mémoire de chacune des données absentes des registres et formant opérande pour une au moins desdites opérations élémentaires à effectuer et ;

b) lire en mémoire, pour chargement dans les registres via l'interface mémoire, chacune desdites données au moyen des adresses mémoire obtenues ;

c) transmettre une instruction d'exécution de calculs depuis l'unité de commande à destination de ladite première unité arithmétique et logique, ladite instruction étant dépourvue d'instruction d'adressage ;

d) à réception de ladite instruction d'exécution de calculs, et dès que les opérandes correspondants sont disponibles sur les registres, exécuter l'ensemble desdites opérations élémentaires par ladite première unité arithmétique et logique recevant en entrées chacun des opérandes depuis les registres ;

e) stocker les données formant résultats du traitement sur les registres en sorties de ladite première unité arithmétique et logique ;

f) obtenir une adresses mémoire pour chacune des données formant résultat du traitement ;

g) écrire en mémoire chacune des données formant résultat du traitement issues des registres, pour stockage et via l'interface mémoire, au moyen des adresses mémoire obtenues.

Un tel procédé permet, en dissociant dans le temps les tâches relatives au traitement des adresses mémoires et les tâches de calcul, de dispenser l'ALU effectuant des calculs d'effectuer en outre des opérations d'adressage qui nécessiterait de stopper les opérations de calcul. Ce faisant, le traitement dans son ensemble devient à la fois asynchrone et auto adaptable : les calculs élémentaires sont initiés (par une instruction transmise à une ALU) seulement une fois que les adresses mémoires ont été mises à jour dans les organes mémoires locaux. En dissociant les deux types d'opérations (mise à jour des adresses mémoire dans les organes mémoires locaux d'une part et calcul d'autre part), le temps de traitement peut être réduit. Autrement dit, pour une quantité de ressources fixée, la somme du temps nécessaire pour mettre à jour les adresse mémoire dans les organes mémoires locaux au cours d'un premier

processus, puis pour effectuer les calculs au cours d'un second processus, est inférieur au temps nécessaire à la même quantité de ressources pour effectuer l'ensemble du traitement au cours d'un unique processus (avec des accès mise à jour d'adresses mémoire dans les organes mémoires locaux à la volée). Le gain de temps est particulièrement important dans le cas de traitements itératifs qui peuvent typiquement être effectués au moyen de boucles informatiques.

Selon un autre aspect, il est proposé un dispositif informatique de traitement de données, ledit traitement étant décomposable en un ensemble d'opérations élémentaires à effectuer. Le dispositif comprend :

- une unité de commande ;
- au moins une première unité arithmétique et logique parmi une pluralité ;
- un jeu de registres aptes à alimenter en données formant opérande des entrées desdites unités arithmétiques et logiques et aptes à être alimentés en données issues des sorties desdites unités arithmétiques et logiques ;
- une mémoire ;
- une interface mémoire par l'intermédiaire de laquelle des données sont transmises et routées entre les registres et la mémoire. Le dispositif informatique est configuré pour :

a) obtenir les adresses mémoire de chacune des données absentes des registres et formant opérande pour une au moins desdites opérations élémentaires à effectuer et ;

b) lire en mémoire, pour chargement dans les registres via l'interface mémoire, chacune desdites données au moyen des adresses mémoire obtenues ;

c) transmettre une instruction d'exécution de calculs depuis l'unité de commande à destination de ladite première unité arithmétique et logique, ladite instruction étant dépourvue d'instruction d'adressage ;

d) à réception de ladite instruction d'exécution de calculs, et dès que les opérandes sont disponibles sur les registres, exécuter l'ensemble desdites opérations élémentaires par ladite première unité arithmétique et logique recevant en entrées chacun des opérandes depuis les registres ;

e) stocker les données formant résultats du traitement sur les registres en sorties de ladite première unité arithmétique et logique ;

f) obtenir une adresses mémoire pour chacune des données formant résultat du traitement ;

g) écrire en mémoire chacune des données formant résultat du traitement issues des registres, pour stockage et via l'interface mémoire, au moyen des adresses mémoire obtenues.

Selon un autre aspect, il est proposé un jeu d'instructions machines pour la mise en œuvre d'un procédé tel que défini dans les présentes lorsque ce programme est exécuté par un processeur. Selon un autre aspect, il est proposé un programme informatique, notamment de compilation, comportant des instructions pour la mise en œuvre de tout ou partie d'un procédé tel que défini dans les présentes lorsque ce programme est exécuté par un processeur. Selon un autre aspect, il est proposé un support d'enregistrement non transitoire, lisible par un ordinateur, sur lequel est enregistré un tel programme.

Les caractéristiques suivantes peuvent, optionnellement, être mises en œuvre. Elles peuvent être mises en œuvre indépendamment les unes des autres ou en combinaison les unes avec les autres :

- La première unité arithmétique et logique exécute l'ensemble des calculs élémentaires du traitement au cours de cycles de calculs consécutifs, aucun accès mémoire n'étant effectué par ladite première unité arithmétique et logique durant lesdits cycles de calculs. Cela permet de dispenser la première unité arithmétique et logique de toute opération d'accès mémoire durant les calculs élémentaires et donc d'accélérer la mise en œuvre desdits calculs.

- L'une au moins parmi les étapes suivantes comprend une boucle itératives :

a) obtenir les adresses mémoire de chacune des données absentes des registres et formant opérande pour une au moins desdites opérations élémentaires à effectuer et ;

d) à réception de ladite instruction d'exécution de calculs, exécuter l'ensemble desdites opérations élémentaires par ladite première unité arithmétique et logique recevant en entrées chacun des opérandes depuis les registres ;

f) obtenir une adresses mémoire pour chacune des données formant résultat du traitement.

Cela permet de mettre en œuvre des processus de calculs particulièrement rapides car répétitifs.

- Le dispositif comprend en outre au moins une unité arithmétique et logique supplémentaire et distincte de la première unité arithmétique et logique exécutant l'ensemble desdites opérations élémentaires. L'unité arithmétique et logique supplémentaire met en œuvre :

a) l'obtention des adresses mémoire de chacune des données absentes des registres et formant opérande pour une au moins desdites opérations élémentaires à effectuer ; et

b) la lecture en mémoire, pour chargement dans les registres via l'interface mémoire, de chacune desdites données au moyen des adresses mémoire obtenues.

Cela permet de répartir les fonctions fixement pour chaque ALU et donc d'améliorer leur efficacité respective.

En parallèle, la demanderesse décrit aussi une approche dans laquelle, à chaque opération de lecture, le nombre de données lues est supérieur au nombre de données strictement nécessaire pour la mise en œuvre du prochain calcul. Une telle approche pourrait, par opposition, être appelée « accès mémoire prévisionnel ». Il est alors possible qu'une donnée parmi les données lues soit utilisée pour un calcul futur, autre que le calcul mis en œuvre immédiatement après la lecture. Dans de tels cas, les données nécessaires ont été obtenues au cours d'une unique opération d'accès mémoire (avec augmentation de la bande passante de la mémoire) tandis que l'approche usuelle aurait nécessité au moins deux accès mémoire distincts. Une telle approche a donc pour effet, au moins dans certains cas, de réduire la consommation de cycles de calcul pour les accès mémoire et permet donc d'améliorer l'efficacité du dispositif. Sur le long terme (plusieurs cycles de calculs consécutifs) le nombre d'accès mémoire (en lecture et/ou en écriture) est réduit.

Cette approche n'exclut pas des pertes : certaines des données lues et mémorisées sur un registre peuvent être perdues (écrasées par d'autres données mémorisées ensuite sur le même registre) avant même d'avoir été utilisées dans un calcul. Néanmoins, sur un grand nombre de calculs et de cycles de calcul, la demanderesse a constaté une amélioration des performances, y compris en l'absence de sélection des ensembles de données

lues. Autrement dit, même en l'absence de sélection des données lues (ou sélection aléatoire) cette approche permet statistiquement d'améliorer l'efficacité du dispositif informatique par rapport à l'approche habituelle.

D'autres caractéristiques, détails et avantages de l'invention apparaîtront à la lecture de la description détaillée ci-après, et à l'analyse des dessins annexés, sur lesquels :

- la figure 1 montre une architecture d'un dispositif informatique selon l'invention ;
- la figure 2 est une représentation partielle d'une architecture d'un dispositif informatique selon l'invention ;
- la figure 3 représente un exemple d'accès mémoire ;
- la figure 4 est une variante de l'exemple de la figure 3 ;
- la figure 5 représente schématiquement une architecture de fonctionnement selon l'invention ; et
- la figure 6 représente une décomposition chronologique d'un exemple de fonctionnement selon l'invention.

La figure 1 montre un exemple d'un dispositif 1 informatique. Le dispositif 1 comprend un ensemble d'un ou plusieurs processeurs 3, parfois appelé unités centrales de traitement ou CPU pour « Central Processing Units ». L'ensemble de processeur(s) 3 comprend au moins une unité de commande 5 et au moins une unité de traitement 7, ou PU 7 pour « Processing Unit ». Chaque PU 7 comprend un ou plusieurs organes de calcul appelés unités arithmétiques et logiques 9, ou ALU 9 pour « Arithmetic-Logic Unit ». Dans l'exemple décrit ici, chaque PU 7 comprend en outre un jeu de registres 11. Le dispositif 1 comprend au moins une mémoire 13 apte à interagir avec l'ensemble de processeur(s) 3. À cet effet, le dispositif 1 comprend en outre une interface mémoire 15, ou « Bus ».

Dans le présent contexte, il est considéré que les organes mémoires sont mono-port, c'est-à-dire que les opérations de lecture et d'écriture sont mis en œuvre au cours de cycles différents, par opposition aux mémoires dites « double-port » (plus coûteuses en terme de surface et nécessitant des bus de commande dédoublés plus larges pour l'écriture et la lecture). En variante, les

solutions techniques proposées peuvent être mises en œuvre avec des mémoires dites « double-port ». Dans de tels modes de réalisation, des lectures et écritures peuvent être mises en œuvre au cours d'un même cycle de calcul.

En figure 1, trois PU 7 sont représentées : PU 1, PU X et PU N. Seule la structure de PU X est représentée en détail afin de simplifier la figure 1. Néanmoins, les structures des PUs sont analogues les unes aux autres. Dans des variantes, le nombre de PUs est différent. Le dispositif 1 peut comprendre une unique PU, deux PUs ou plus de trois PUs.

Dans l'exemple décrit ici, la PU X comprend quatre ALUs : ALU X.0, ALU X.1, ALU X.2 et ALU X.3. Dans des variantes, les PUs peuvent comprendre un nombre d'ALUs différent les unes des autres et/ou différent de quatre, y compris une unique ALU. Chaque PU comprend un jeu de registres 11, ici au moins un registre 11 attribué à chaque ALU. Dans l'exemple décrit ici, la PU X comprend un unique registre 11 par ALU, soit quatre registres référencés REG X.0, REG X.1, REG X.2 et REG X.3 et attribués respectivement à ALU X.0, ALU X.1, ALU X.2 et ALU X.3. Dans des variantes, chaque ALU se voit attribué une pluralité de registres 11.

Chaque registre 11 est apte à alimenter en données de type opérande les entrées desdites ALUs 9 et est apte à être alimenté en données issues des sorties desdites ALU 9. Chaque registre 11 est, en outre, apte à stocker des données issues de la mémoire 13 obtenues par l'intermédiaire du bus 15 par une opération dite de « lecture ». Chaque registre 11 est, en outre, apte à transmettre des données stockées, à destination de la mémoire 13 et par l'intermédiaire du bus 15, par une opération dite d' « écriture ». Les opérations de lecture et d'écriture sont gérées par le pilotage des accès mémoire depuis l'unité de commande 5.

L'unité de commande 5 impose à chaque ALU 9 la manière d'effectuer des calculs élémentaires, notamment leur ordre, et attribut à chaque ALU 9 les opérations à exécuter. Dans l'exemple décrit ici, l'unité de commande 5 est configurée pour piloter les ALUs 9 selon une microarchitecture en chaine de traitement de sorte que les ALUs 9 effectuent des calculs en parallèle les unes

des autres. Par exemple, le dispositif 1 présente une architecture à Unique flux d'instructions et multiple flux de données, dites SIMD pour « Single Instructions Multiple Data », et/ou une architecture à multiples flux d'instructions et multiples flux de données, dites MIMD pour « Multiple Instructions Multiple Data ». D'autre part, l'unité de commande 5 est en outre agencée pour piloter les accès mémoire par l'intermédiaire de l'interface mémoire 15 et notamment, ici, les opérations de lecture et d'écriture. Les deux types de pilotage (calcul et accès mémoire) sont représentés, en figure 1, par des flèches en traits discontinus.

Il est maintenant fait référence à la figure 2, sur laquelle une unique ALU Y est représentée. Les transmissions de données sont représentées par des flèches à traits pleins. La transmission des données étant réalisée de proche en proche, on comprend que la figure 2 ne représente pas nécessairement un instant t avec des transmissions de données simultanées. Au contraire, pour qu'une donnée soit transmise d'un registre 11 vers une ALU 9, il est par exemple nécessaire que ladite donnée soit préalablement transmise audit registre 11 depuis la mémoire 13, ici via l'interface mémoire 15 (ou Bus).

Dans l'exemple de la figure 2, trois registres 11, référencés respectivement REG Y.0, REG Y.1 et REG Y.2, sont attribués une ALU référencée ALU Y. Chaque ALU 9 présente au moins trois ports, à savoir deux entrées et une sortie. Pour chaque opération, au moins deux opérandes sont reçus, respectivement par la première et la seconde entrée. Le résultat du calcul est émis via la sortie. Dans l'exemple représenté en figure 2, les opérandes reçus en entrée proviennent respectivement du registre REG Y.0 et du registre REG Y.2. Le résultat du calcul est inscrit au registre REG Y.1. Une fois inscrite au registre REG Y.1, le résultat (sous forme d'une donnée) est écrite en mémoire 13, via l'interface mémoire 15. Dans des variantes, au moins une ALU peut présenter plus de deux entrées et recevoir plus de deux opérandes pour un calcul.

Chaque ALU 9 peut effectuer :

- des opérations arithmétiques sur des données entières (addition, soustraction, multiplication, division, etc.) ;

- des opérations arithmétiques sur des données flottantes (addition, soustraction, multiplication, division, inversion, racine carrée, logarithmes, trigonométrie, etc.) ;

- des opérations logiques (compléments à deux, « ET », « OU », « OU exclusif », etc.).

Les ALUs 9 n'échangent pas directement de données entre elles. Par exemple, si le résultat d'un premier calcul effectué par une première ALU constitue un opérande pour un deuxième calcul à effectuer par une deuxième ALU, alors le résultat du premier calcul doit au moins être inscrit dans un registre 11 avant d'être utilisable par une ALU 9.

Dans des modes de réalisation, les données inscrites sur un registre 11 sont en outre systématiquement écrites en mémoire 13 (via l'interface mémoire 15), même si ladite donnée est obtenue seulement pour servir d'opérande et pas en tant que résultat d'un processus de traitement dans son ensemble.

Dans des modes de réalisation, les données obtenues pour servir d'opérande et ayant une pertinence courte (résultat intermédiaire sans intérêt à l'issu du traitement dans son ensemble) ne sont pas systématiquement écrites en mémoire 13 et peuvent être stockées seulement de manière temporaire sur un registre 11. Par exemple, si le résultat d'un premier calcul effectué par une première ALU constitue un opérande pour un deuxième calcul à effectuer par une deuxième ALU, alors le résultat du premier calcul doit être inscrit dans un registre 11. Puis, ladite donnée est transmise à la deuxième ALU en tant qu'opérande directement depuis le registre 11. On comprend alors que l'attribution d'un registre 11 à une ALU 9 peut évoluer au fil du temps et notamment d'un cycle de calcul à un autre. Cette attribution peut notamment prendre la forme de données d'adressage qui permettent de localiser, à chaque instant, l'emplacement d'une donnée, que ce soit sur un registre 11 ou en un emplacement de la mémoire 15.

Dans la suite, le fonctionnement du dispositif 1 est décrit pour un traitement appliqué à des données informatiques, le traitement étant composé d'un ensemble d'opérations, dont des calculs effectués en parallèle par une

pluralité d'ALUs 9 au cours d'une période de temps constituée d'une séquence de cycles de calcul. On dit alors que les ALUs 9 fonctionnent selon une microarchitecture en chaine de traitement. Néanmoins, le traitement mis en œuvre par le dispositif 1 et dont il est question ici peut, lui-même, constituer une partie (ou un sous-ensemble) d'un processus informatique plus global. Un tel processus plus global peut comprendre, dans d'autres parties ou sous-ensembles, des calculs effectués de manière non parallèle par une pluralité d'ALUs, par exemple selon un fonctionnement en série, ou en cascade.

Les architectures de fonctionnement (parallèle ou en série) peuvent être constantes ou dynamiques, par exemple imposées (pilotées) par l'unité de commande 5. Les variations d'architecture peuvent par exemple être fonction des données à traiter et des instructions courantes reçues en entrée du dispositif 1. Une telle adaptation dynamique des architectures peut être mise en œuvre dès le stade de la compilation, en adaptant les instructions machines générées par le compilateur en fonction du type de données à traiter et des instructions lorsque le type de données à traiter et les instructions peuvent être déduites du code source. Une telle adaptation peut aussi être mise en œuvre seulement au niveau du dispositif 1, ou d'un processeur, lorsqu'il exécute un code machine classique et qu'il est programmé pour mettre en œuvre un ensemble d'instructions de configuration fonction des données à traiter et des instructions courantes reçues.

L'interface de mémoire 15, ou « bus », transmet et route les données entre les ALU 9 et la mémoire 15, dans les deux sens. L'interface de mémoire 15 est pilotée par l'unité de commande 5. Ainsi, l'unité de commande 5 pilote l'accès à la mémoire 13 du dispositif 1 par l'intermédiaire de l'interface mémoire 15.

L'unité de commande 5 pilote de manière coordonnée les opérations (calculs) mises en œuvre par les ALU 9 et les accès mémoire. Le pilotage de l'unité de commande 5 comprend la mise en œuvre d'une succession d'opérations décomposées en cycles de calcul. Le pilotage comprend la génération d'un premier cycle i et d'un deuxième cycle ii. Chronologiquement, le premier cycle i est antérieur au deuxième cycle ii. Comme cela sera décrit plus

en détails dans les exemples ci-après, le deuxième cycle ii peut être immédiatement ultérieur au premier cycle i, ou bien le premier cycle i et le deuxième cycle ii peuvent être chronologiquement espacés l'un de l'autre, par exemple avec des cycles intermédiaires.

Le premier cycle i comprend :
- la mise en œuvre d'un premier calcul par au moins une ALU 9 ; et
- le téléchargement, depuis la mémoire 13 sur au moins un registre 11, d'un premier jeu de données.

Le deuxième cycle ii comprend la mise en œuvre d'un deuxième calcul par au moins une ALU 9. Le deuxième calcul peut être mis en œuvre par la même ALU 9 que le premier calcul ou par une ALU 9 distincte. Une partie au moins du premier jeu de données téléchargé au cours du premier cycle i forme un opérande pour le deuxième calcul.

Il est maintenant fait référence à la figure 3. Des données, ou blocs de données, sont référencées respectivement A0 à A15 et sont stockées dans la mémoire 13. Dans l'exemple, il est considéré que les données A0 à A15 sont regroupées par quatre de la manière suivante :
- un jeu de données référencé AA0_3 constitué des données A0, A1, A2 et A3 ;
- un jeu de données référencé AA4_7 constitué des données A4, A5, A6 et A7 ;
- un jeu de données référencé AA8_11 constitué des données A8, A9, A10 et A11 ; et
- un jeu de données référencé AA12_15 constitué des données A12, A13, A14 et A15.

En variante, les données peuvent être regroupées différemment, notamment par groupe (ou « bloc », ou « slot ») de deux, trois ou plus de quatre. Un jeu de données peut être vu comme un groupe de données accessibles sur la mémoire 13 par l'intermédiaire d'un unique port de l'interface mémoire 15 au cours d'une unique opération de lecture. De même, les données d'un jeu de données peuvent être inscrites en mémoire 13 par l'intermédiaire d'un unique port de l'interface mémoire 15 au cours d'une unique opération d'écriture.

Ainsi, au cours d'un premier cycle i, au moins un jeu de données AA0_3, AA4_7, AA8_11 et/ou AA12_15 est téléchargé sur au moins un registre 11. Dans l'exemple de la figure, chacun des jeux de données AA0_3, AA4_7, AA8_11 et/ou AA12_15 est téléchargé sur un registre 11 respectif, soit quatre registres 11 distincts les uns des autres. Chacun des registres 11 est attribué au moins temporairement à une ALU 9 respective, ici référencée respectivement ALU 0, ALU 1 ALU 2 et ALU 3. Au cours de ce même cycle i, les ALUs 9 peuvent avoir mis en œuvre un calcul.

Au cours d'un deuxième cycle ii, chaque ALU 9 met en œuvre un calcul pour lequel l'une au moins des données stockées sur le registre 11 correspondant forme un opérande. Par exemple, l'ALU 0 met en œuvre un calcul dont l'un des opérandes est A0. A1, A2 et A3 peuvent être inutilisés au cours du deuxième cycle ii.

De manière générale, télécharger des données depuis la mémoire 13 jusqu'à un registre 11 est moins consommateur de temps de calcul que de mettre en œuvre des calculs par des ALUs 9. Ainsi, on peut généralement considérer qu'une opération d'accès mémoire (ici une lecture) consomme un unique cycle de calcul, tandis que la mise en œuvre d'un calcul par une ALU 9 consomme un cycle de calcul ou une succession de plusieurs cycles de calcul, par exemple quatre.

Dans l'exemple de la figure 3, il existe une pluralité de registres 11 attribués à chaque ALU 9, représentés par des groupe de registres 11 référencés REG A, REG B et REG C. Les données téléchargées depuis la mémoire 13 sur les registres 11 correspondent aux groupes REG A et REG B. Le groupe REG C est ici destiné à stocker des données obtenues par des calculs mis en œuvre par les ALUs 9 (au cours d'une opération d'écriture).

Les registres 11 des groupes REG B et REG C peuvent ainsi contenir des jeux de données référencés de manière analogues à ceux de REG A :

- le groupe REG B comprend quatre registres 11 sur lesquels sont respectivement stockés un jeu de données BB0_3 constitué de données B0 à

B3, un jeu de données BB4_7 constitué de données B4 à B7, un jeu de données BB8_11 constitué de données B8 à B11 et un jeu de données BB12_15 constitué de données B12 à B15 ;

- le groupe REG C comprend quatre registres 11 sur lesquels sont respectivement stockés un jeu de données CC0_3 constitué de données C0 à C3, un jeu de données CC4_7 constitué de données C4 à C7, un jeu de données CC8_11 constitué de données C8 à C11 et un jeu de données CC12_15 constitué de données C12 à C15.

Dans l'exemple de la figure 3, les données AN et BN constituent les opérandes d'un calcul mis en œuvre par une ALU 9 tandis que la donnée CN constitue le résultat, avec « N » un entier compris entre 0 et 15. Par exemple, dans le cas d'une addition, CN = AN + BN. Dans un tel exemple, le traitement des données mis en œuvre par le dispositif 1 correspond à 16 opérations. Les 16 opérations sont indépendantes les unes des autres au sens qu'aucun des résultats des 16 opérations n'est nécessaire pour mettre en œuvre l'une des 15 autres opérations.

La mise en œuvre du traitement (les 16 opérations) peut donc, par exemple, se décomposer de la manière suivante, en 18 cycles.

Exemple 1 :

- cycle #0 : lecture de AA0_3 ;
- cycle #1 : lecture de BB0_3 ;

- cycle #2 : calcul de C0 (du jeu CC0_3) et lecture de AA4_7 (formant par exemple un cycle i) ;
- cycle #3 : calcul de C1 (du jeu CC0_3) et lecture de BB4_7 (formant par exemple un cycle i) ;
- cycle #4 : calcul de C2 (du jeu CC0_3) ;
- cycle #5 : calcul de C3 (du jeu CC0_3) et écriture de CC0_3 ;

- cycle #6 : calcul de C4 (du jeu CC4_7) et lecture de AA8_11 (formant par exemple un cycle ii) ;
- cycle #7 : calcul de C5 (du jeu CC4_7) et lecture de BB8_11 (formant par exemple un cycle ii) ;

- cycle #8 : calcul de C6 (du jeu CC4_7) (formant par exemple un cycle ii) ;
- cycle #9 : calcul de C7 (du jeu CC4_7) et écriture de CC4_7 (formant par exemple un cycle ii) ;

- cycle #10 : calcul de C8 (du jeu CC8_11) et lecture de AA12_15 ;
- cycle #11 : calcul de C9 (du jeu CC8_11) et lecture de BB12_15 ;
- cycle #12 : calcul de C10 (du jeu CC8_11) ;
- cycle #13 : calcul de C11 (du jeu CC8_11) et écriture de CC8_11 ;

- cycle #14 : calcul de C12 (du jeu CC12_15) ;
- cycle #15 : calcul de C13 (du jeu CC12_15) ;
- cycle #16 : calcul de C14 (du jeu CC12_15) ;
- cycle #17 : calcul de C15 (du jeu CC12_15) et écriture de CC12_15.

On comprend alors que, à l'exception des cycles #0 et #1 initiaux, les accès mémoire (lectures et écritures) sont mis en œuvre en parallèle des calculs, sans consommer de cycle de calcul supplémentaire. Lire des jeux de (plusieurs) données, ou blocs de données, plutôt que de lire une unique donnée, permet de terminer l'importation des données depuis la mémoire 13 sur les registres avant mêmes que lesdites données ne deviennent nécessaires, en tant qu'opérande, pour un calcul.

Dans l'exemple du cycle #2 ci-dessus, si seule la donnée immédiatement nécessaire (A0) avait été lue plutôt que de lire le jeu AA0_3 = {A0 ; A1 ; A2 ; A3}, alors il aurait été nécessaire de mettre en œuvre, ultérieurement, trois opérations de lecture supplémentaires pour obtenir A1, A2 et A3.

Pour mieux comprendre, et par comparaison, on reproduit ci-après la mise en œuvre d'un traitement dans lequel une unique donnée est lue à chaque fois plutôt qu'un jeu de (plusieurs) données. On constate que 48 cycles sont nécessaires.

Exemple 0 :

- cycle #0 : lecture de A0 ;
- cycle #1 : lecture de B0 ;

- cycle #2 : calcul de C0 et écriture de C0 ;
- cycle #3 : lecture de A1 ;
- cycle #4 : lecture de B1 ;
- cycle #5 : calcul de C1 écriture de C1 ;
- …
- cycle #45 : lecture de A15 ;
- cycle #46 : lecture de B15 ;
- cycle #47 : calcul de C15 et écriture de C15.

Dans l'exemple 1 (18 cycles), on remarque que les deux premiers cycles #0 et #1 constituent des cycles d'initialisation. Le nombre I de cycles d'initialisation correspond au nombre d'opérandes par calcul. Ensuite un motif de quatre cycles successifs est répété quatre fois. Par exemple, les cycles #2 à #5 forment ensemble un motif. Le nombre de cycles par motif correspond au nombre D de données par jeu de données tandis que le nombre de motifs correspond au nombre E de jeu de données à traiter. Le nombre total de cycles peut donc s'exprimer de la manière suivante : I + D*E.

Atteindre une bonne performance équivaut à réduire au minimum le nombre total de cycles. Dans les conditions considérées, c'est-à-dire 16 opérations élémentaires et indépendantes pouvant être chacune mise en œuvre sur un cycle, le nombre de cycles optimum semblent donc être égal à celui nombre d'opérations élémentaires (16) auquel s'ajoute la phase d'initialisation (2 cycles), soit un total de 18 cycles.

Dans une variante, on considère que le nombre de données accessibles (en lecture ou en écriture) en un unique cycle (le nombre D de données par jeu de données) est égal à trois (et non plus quatre), par exemple à cause de limitations matérielles. Alors, la succession des cycles peut, par exemple, être décomposée de la manière suivante :

- une phase d'initialisation de 2 cycles ; puis
- 5 motifs de 3 cycles pour un total de 15 calculs élémentaires sur les 16 à effectuer ; puis
- un ultime cycle pour calculer et enregistrer le résultat du dernier calcul élémentaire.

Exemple 2 :

- cycle #0 : lecture de AA0_2={A0 ; A1 ; A2} ;
- cycle #1 : lecture de BB0_2={B0 ; B1 ; B2} ;

- cycle #2 : calcul de C0 (du jeu CC0_2={C0 ; C1 ; C2}) et lecture de AA3_5 (formant par exemple un cycle i) ;
- cycle #3 : calcul de C1 (du jeu CC0_2) et lecture de BB3_5 (formant par exemple un cycle i) ;
- cycle #4 : calcul de C2 (du jeu CC0_2) et écriture de CC0_2 ;

- cycle #5 : calcul de C3 (du jeu CC3_5) et lecture de AA6_8 (formant par exemple un cycle ii) ;
- cycle #6 : calcul de C4 (du jeu CC3_5) et lecture de BB6_8 (formant par exemple un cycle ii) ;
- cycle #7 : calcul de C5 (du jeu CC3_5) et écriture de CC3_5 (formant par exemple un cycle ii) ;

- cycle #8 : calcul de C6 (du jeu CC6_8) et lecture de AA9_11 ;
- cycle #9 : calcul de C7 (du jeu CC6_8) et lecture de BB9_11 ;
- cycle #10 : calcul de C8 (du jeu CC6_8) et écriture de CC6_8 ;

- cycle #11 : calcul de C9 (du jeu CC9_11) et lecture de AA12_14 ;
- cycle #12 : calcul de C10 (du jeu CC9_11) et lecture de BB12_14 ;
- cycle #13 : calcul de C11 (du jeu CC9_11) et écriture de CC9_11 ;

- cycle #14 : calcul de C12 (du jeu CC12_14) et lecture de A15 (formant par exemple un cycle i) ;
- cycle #15 : calcul de C13 (du jeu CC12_14) et lecture de B15 (formant par exemple un cycle i) ;
- cycle #16 : calcul de C14 (du jeu CC12_14) et écriture de CC12_14 ;

- cycle #17 : calcul de C15 (donnée isolée) et écriture de C15 (formant par exemple un cycle ii).

Dans l'exemple 2, on constate que chaque cycle inclut une opération d'accès mémoire (en lecture ou en écriture). On comprend donc que, si le nombre D de données accessibles en un unique cycle est strictement inférieur à trois, alors des cycles supplémentaires seront nécessaires pour effectuer des accès mémoire. L'optimum de 18 cycles pour 16 opérations élémentaires ne sera donc plus atteint. Pour autant, même si l'optimum n'est pas atteint, le nombre de cycles reste significativement inférieur au nombre de cycles nécessaire dans l'exemple 0. Un mode de réalisation dans lequel les jeux de données comprennent deux données présentent une amélioration par rapport à l'existant.

Dans l'exemple 1, si les cycles #2 et/ou #3 correspondent par exemple à un cycle i tel que défini ci-avant, alors chacun des cycles #6, #7, #8 et #9 correspond à un cycle ii. Bien entendu, cela est transposable de motif en motif. Dans l'exemple 2, si les cycles #2 et/ou #3 correspondent par exemple à un cycle i tel que défini ci-avant, alors chacun des cycles #5, #6 et #7 correspond à un cycle ii. Bien entendu, cela est transposable de motif en motif.

Dans les exemples décrits jusqu'ici, notamment les exemples 1 et 2, le faible nombre total de cycles est atteint notamment parce qu'un maximum d'opérations d'accès mémoire est mis en œuvre par jeu de (plusieurs) données plutôt qu'à l'unité et en parallèle d'opérations de calculs. Ainsi, pour certaines parties du processus (pour toutes les parties dans les exemples optimisés), la lecture de l'ensemble des opérandes nécessaires peut être achevée avant même que l'opération de calcul élémentaire précédente ne soit terminée. De préférence, il est préservé de la puissance de calcul pour effectuer un calcul et enregistrer (opération d'écriture) le résultat dudit calcul en un cycle de calcul commun (cycle #5 de l'exemple 1 par exemple).

Dans les exemples, la lecture des données-opérandes en avance est mise en œuvre tout au long du processus (répété d'un motif à l'autre). Les opérandes nécessaires aux calculs effectués au cours d'un motif sont systématiquement obtenus (lus) au cours du motif chronologiquement antérieur. On notera que, dans des modes de réalisation dégradés, la lecture en avance est mise en œuvre seulement partiellement (pour deux motifs successifs seulement). Un tel

mode dégradé par rapport aux exemples ci-avant présente de meilleurs résultats que les méthodes existantes.

Dans les exemples décrits jusqu'ici, il a été admis que les données étaient lues avant de servir d'opérandes. Dans des modes de réalisation, les données lues en avance le sont aléatoirement, ou du moins indépendamment des calculs à réaliser dans le futur. Ainsi, certaines au moins des données lues en avance parmi les jeux de données correspondent effectivement à des opérandes pour des calculs ultérieurs tandis que d'autres données lues ne sont pas des opérandes pour des calculs ultérieurs. Par exemple, certaines au moins des données lues peuvent être ultérieurement effacées des registres 11 sans avoir été utilisées par les ALUs 9, typiquement écrasées par d'autres données enregistrées ultérieurement sur les registres 11. Certaines données sont donc lues inutilement (et enregistrées inutilement sur les registres 11). Néanmoins, il suffit que certaines au moins des données parmi les jeux de données lues soient effectivement des opérandes pour qu'une économie en cycle de calcul se produise, et donc que la situation soit améliorée par rapport à l'existant. Aussi, en fonction du nombre de données à traiter et du nombre de cycles, il est probable (au sens mathématique du terme), que certaines au moins des données pré-lues puissent effectivement être utilisées en tant qu'opérande dans un calcul effectué par une ALU 9 dans un cycle suivant.

Dans des modes de réalisation, les données lues en avance sont présélectionnées, et dépendent des calculs à réaliser. Cela permet d'améliorer la pertinence des données pré-lues. En effet, dans les exemples à 16 calculs élémentaires ci-avant, chacun des 16 calculs élémentaires nécessite en entrée une paire d'opérandes, respectivement A0 et B0 ; A1 et B1 ; … ; A15 et B15. Si les données sont lues aléatoirement, alors les deux premiers cycles pourraient correspondre à la lecture de AA0_3 et BB4_7. Dans un tel cas, aucune paire complète d'opérande n'est disponible sur les registres 11 à l'issue des deux premiers cycles. Par conséquent, les ALUs 9 ne peuvent mettre en œuvre aucun calcul élémentaire au cycle suivant. Un ou plusieurs cycles supplémentaires seraient donc nécessairement consommés pour de l'accès mémoire avant que les calculs élémentaires ne puissent débuter, ce qui augmente le nombre total de cycles et est donc nuisible pour l'efficacité.

Compter sur le hasard et les probabilités pour que les données obtenues en lecture soient les plus pertinentes possibles suffit à améliorer l'existant mais n'est pas pleinement satisfaisant. La situation peut encore être améliorée.

La mise en œuvre d'un algorithme de prélecture (ou « prefetch algorithm ») permet d'obtenir, dès que possible, tous les opérandes du prochain calcul à effectuer. Dans l'exemple ci-avant, lire AA0_3 et BB0_3 au cours des deux premiers cycles permet, par exemple, de rendre disponibles, sur les registres 11, l'ensemble des opérandes nécessaires à la mise en œuvre de 4 premiers calculs élémentaires.

Un tel algorithme reçoit en paramètres d'entrée des données d'informations relatives aux calculs à réaliser ultérieurement par les ALUs 9, et en particulier relatives aux opérandes nécessaires. Un tel algorithme permet, en sortie, de sélectionner les données lues (par jeu) en prévision des calculs futurs à effectuer. Un tel algorithme est, par exemple, mis en œuvre par l'unité de commande 5 lors du pilotage des accès mémoire.

Selon une première approche, l'algorithme impose une organisation des données dès leur enregistrement dans la mémoire 13. Par exemple, les données que l'on souhaite voir former ensemble un jeu de données sont juxtaposées et/ou ordonnancées de sorte que l'ensemble du jeu de données peut être appelé par une requête unique. Par exemple, si les adresses des données A0, A1, A2 et A3 sont référencées respectivement @A0 @A1, @A2, @A3, alors l'interface mémoire 15 peut être configurée pour, en réponse à une requête de lecture sur @A0, lire en outre automatiquement les données aux trois adresses suivantes @A1, @A2 et @A3.

Selon une deuxième approche, l'algorithme de prélecture fourni en sortie des requêtes en accès mémoire adaptées en fonction des calculs à réaliser ultérieurement par les ALUs 9, et en particulier relatives aux opérandes nécessaires. Dans les exemples précédents, l'algorithme identifie par exemple que les données à lire en priorité sont celles de AA0_3 et BB0_3 pour rendre possible, dès le cycle suivant, les calculs élémentaires ayant pour résultat

CC0_3, soit le calcul de C0 avec les opérandes A0 et B0, le calcul de C1 avec les opérandes A1 et B1, le calcul de C2 avec les opérandes A2 et B2 et le calcul de C3 avec les opérandes A3 et B3. L'algorithme fournit donc, en sortie, des requêtes d'accès mémoire construites pour générer la lecture de AA0_3 et BB0_3.

Les deux approches peuvent, optionnellement, être combinées l'une avec l'autre : l'algorithme identifie les données à lire et l'unité de commande 5 en déduit des requêtes d'accès mémoire à l'interface mémoire 15 pour obtenir lesdites données, les requêtes étant adaptées en fonction des caractéristiques (structure et protocole) de l'interface mémoire 15.

Dans les exemples précédents, notamment les exemples 1 et 2 ci-avant, le nombre d'ALUs affectées aux calculs élémentaires n'est pas défini. Une unique ALU 9 peut effectuer l'ensemble des calculs élémentaires, cycle par cycle. Les calculs élémentaires à effectuer peuvent aussi être répartis sur une pluralité d'ALUs 9 d'une PU, par exemple quatre. Dans de tels cas, coordonner la répartition des calculs sur les ALUs avec la manière de regrouper les données à lire à chaque opération de lecture peut permettre d'améliorer encore l'efficacité. Deux approches se distinguent.

Dans une première approche, les données lues en une opération forment opérandes dans des calculs mis en œuvre par une seule et même ALU 9. Par exemple, les groupes AA0_3 et BB0_3 de données A0, A1, A2, A3, B0, B1, B2 et B3 sont lus les premiers et une première ALU est chargée du calcul de CC0_3 (C0, C1, C2 et C3). Les groupes AA4_7 (A4, A5, A6, A7) et BB4_7 (B4, B5, B6 et B7) sont lus ensuite et une seconde ALU est chargée du calcul de CC4_7 (C4, C5, C6 et C7). On comprend alors que la première ALU va pouvoir commencer à mettre en œuvre les calculs avant que la seconde ALU puisse faire de même car les opérandes nécessaires aux calculs de la première ALU seront disponibles sur les registres 11 avant que les opérandes nécessaires aux calculs de la seconde ALU ne le soient. Les ALUs 9 d'une PU fonctionnent alors de manière parallèle et asynchrone.

Dans une deuxième approche, les données lues en une opération forment opérandes dans des calculs chacun mis en œuvre par différentes ALUs 9, par exemple quatre. Par exemple, deux groupes de données incluant respectivement A0, A4, A8 et A12 ; B0, B4, B8 et B12 sont lus les premiers. Une première ALU est chargée du calcul de C0, une seconde ALU est chargée du calcul de C4, une troisième ALU est chargée du calcul de C8 et une quatrième ALU est chargée du calcul de C12. On comprend alors que les quatre ALUs vont pouvoir commencer à mettre en œuvre leur calcul respectif de manière sensiblement simultanée, car les opérandes nécessaires seront disponibles sur les registres 11 en même temps car téléchargés en une opération commune. Les ALUs 9 d'une PU fonctionnent de manière parallèle et synchronisées. En fonction, des types de calculs à effectuer, de l'accessibilité des données en mémoire et des ressources disponibles, l'une ou l'autre des deux approches peut être préférée. Les deux approches peuvent également être combinées : les ALUs peuvent être organisés en sous-groupes, les ALUs d'un sous-groupe fonctionnant de manière synchronisées et les sous-groupes fonctionnant de manière asynchrone les uns par rapport aux autres.

Pour imposer un fonctionnant synchronisé, asynchrone ou mixte des ALUs, le regroupement des données à lire par opération de lecture doit être sélectionné en correspondance avec la répartition des affectations des opérations de calcul à diverses ALUs.

Dans les exemples précédents, les calculs élémentaires sont indépendants les uns des autres. L'ordre dans lequel ils sont effectués n'a donc *a priori* pas d'importance. Dans des applications pour lesquels certains au moins des calculs sont dépendants les uns des autres, l'ordonnancement des calculs peut être spécifique. Une telle situation se présente typiquement dans le contexte de calculs récursifs. Dans de tels cas, l'algorithme peut être configuré pour identifier les données à acquérir (lire) en priorité. Par exemple, si :

- le résultat C1 est obtenu par un calcul dont l'un des opérandes est C0, C0 étant lui-même obtenu à partir des opérandes A0 et B0,
- le résultat C5 est obtenu par un calcul dont l'un des opérandes est C4, C4 étant lui-même obtenu à partir des opérandes A4 et B4,
- le résultat C9 est obtenu par un calcul dont l'un des opérandes est C8, C8

étant lui-même obtenu à partir des opérandes A8 et B8, et
- le résultat C13 est obtenu par un calcul dont l'un des opérandes est C12, C12 étant lui-même obtenu à partir des opérandes A12 et B12,
alors l'algorithme peut être configuré pour lire, lors des deux premiers cycles #0 et #1 d'initialisation, les jeux de données définis comme suit :
- {A0 ; A4 ; A8 ; A12}, et
- {B0 ; B4 ; B8 ; B12}.

Le jeu de données ainsi défini est représenté en figure 4. De manière imagée, on peut dire que les données sont regroupées « en ligne » dans le mode de réalisation représenté en figure 3 et regroupées « en colonne » dans le mode de réalisation représenté en figure 4. Ainsi, la mise en œuvre de l'algorithme permet de lire et de rendre disponibles sur les registres 11, les opérandes utiles pour les calculs élémentaires prioritaires. Autrement dit, la mise en œuvre de l'algorithme permet d'augmenter la pertinence à court terme des données lues par rapport à une lecture aléatoire.

Les exemples d'unités de traitement et de procédés décrits ci-avant, seulement à titre d'exemple ne sauraient être considérés comme limitatifs, d'autres variantes pourront être envisagées par l'homme de l'art dans le cadre de la protection recherchée. Les exemples peuvent aussi prendre la forme :
- d'un jeu d'instructions machines implémentable dans un processeur pour obtenir un tel dispositif informatique,
- d'un processeur ou un ensemble de processeurs,
- l'implémentation d'un tel jeu d'instructions machines sur un processeur,
- le procédé de gestion d'architecture de processeur mis en œuvre par le processeur,
- le programme informatique comprenant le jeu d'instructions machines correspondant, ainsi que
- le support d'enregistrement sur lequel est enregistré informatiquement un tel jeu d'instructions machines.

Il est maintenant fait référence à la figure 5. Il y est représenté un exemple d'architecture de fonctionnement d'un dispositif 1 dans laquelle les opérations d'accès mémoire et de traitement des adressages sont traitées distinctement

des opérations de calcul élémentaire. Une telle architecture peut prendre la forme d'un procédé informatique. Elle peut optionnellement être combinée avec les modes de réalisation décrits ci-avant. Les références numériques communes avec celles des figures précédentes désignent des éléments analogues, notamment une unité de contrôle 5, une ALU 9, des registres 11, une mémoire 13 et une interface mémoire 15, ou « bus ».

Afin de faciliter la compréhension, les même conventions de nommage sont utilisées : on considère une opération élémentaire, par exemple une addition, dans laquelle AX et BX sont des données formant opérandes pour obtenir une donnée formant résultat CX, avec X un entier compris entre 0 et N, N+1 étant le nombre d'opérations élémentaires à effectuer au cours d'un traitement. L'ensemble des N+1 opérations forme le traitement de données dans son ensemble. En outre, les adresses mémoire de chacune des données sont référencées par leur nom précédé du caractère « @ » (arobase). Par exemple, l'adresse de la donnée A0 est notée « @A0 ».

Pour chaque addition (chaque valeur de X), un jeu d'instructions peut être mis en œuvre par le dispositif informatique 1. Un exemple d'un tel jeu d'instructions est donné en fin de description sous la forme d'un pseudocode informatique. Habituellement, de telles instructions sont appliquées les unes à la suite des autres au cours d'un processus commun mis en œuvre par une ALU 9. Dans les modes de réalisation ci-après, les instructions relatives aux accès mémoire et les instructions relatives aux opérations de calcul élémentaire sont traitées par des processus distincts les uns des autres.

Dans un mode de réalisation d'un procédé informatique conforme à la figure 5, on peut décomposer le procédé en étapes référencées respectivement 101 à 109.

Au cours des étapes 101 et 102, les adresses mémoire @A0 à @AN, respectivement @B0 à @BN, de chacune des données formant opérande pour une au moins des opérations élémentaires à effectuer sont obtenues. On entend par « obtenues » qu'à l'issue des opérations 101 et 102, un ou plusieurs organes mémoires locaux stockent les adresses de toutes les données formant

opérande. De tels accès mémoire sont par exemple déclenchés par la réception d'instructions depuis l'unité de commande 5. Dans certains cas, certaines au moins desdites adresses sont déjà stockées dans les organes mémoires locaux. Aucun accès mémoire n'est donc nécessaire à ce stade pour obtenir lesdites adresses préalablement installées sur les organes mémoires locaux.

Dans l'exemple décrit ici, on distingue l'étape 101 concernant les premiers opérandes « A » de l'addition de l'étape 102 concernant les seconds opérandes « B » de l'addition. Distinguer les deux opérandes permet ensuite de mettre en œuvre des boucles itératives (au sens informatique) propres à chacun des deux opérandes, et éventuellement différentes l'une de l'autre.

En variante, en particulier lorsque les deux opérandes préexistent au début du procédé, les étapes 101 et 102 peuvent être mises en œuvre au moins en partie en parallèle l'une de l'autre, indépendamment l'une de l'autre.

Au cours des étapes 103 et 104, chacune desdites données obtenues, respectivement A0 à AN et B0 à BN, est lue en mémoire 13, pour chargement dans les registres 11, via l'interface mémoire 15. De telles lectures sont rendues possibles grâce aux adresses obtenues aux étapes 101 et 102. Dans l'exemple décrit ici, l'étape 103 concerne les premiers opérandes « A » tandis que l'étape 104 concerne les seconds opérandes « B ».

Au cours de l'étape 105, une instruction d'exécution de calculs est transmise depuis l'unité de commande 5 à destination d'une ALU 9. L'instruction d'exécution est construite de manière à déclencher la mise en œuvre des calculs élémentaires du traitement par l'ALU 9. L'instruction est, ici, dépourvue d'instruction d'adressage. Par dépourvue d'instruction d'adressage, on entend ici que, contrairement à ce qui se fait habituellement, l'instruction d'exécution de calculs transmise par l'unité de commande 5 n'est pas incluse dans un jeu d'instructions général combinant à la fois des instructions d'adressage et des instructions d'exécution de calculs. Ainsi, à réception des instructions, l'ALU 9 est en mesure d'appliquer immédiatement les instructions en effectuant les calculs élémentaires sans qu'il soit nécessaire d'appliquer préalablement des instructions de configuration de l'interface mémoire 15 et

donc sans qu'il soit nécessaire, non plus, de vérifier une quelconque dépendance mutuelle des diverses instructions reçues. De manière imagée, l'ALU 9 se comporte alors comme une ressource informatique mettant en œuvre les calculs (étape 106 décrite ci-après) indépendamment d'une éventuelle complexité d'interdépendance entre les différentes instructions. En conditionnant la transmission de l'instruction de calcul (étape 105) à l'exécution préalable des opérations d'adressage (étapes 103 et 104), la disponibilité des données formant opérandes sur les registres 11 est assurée. En pratique, les registres 11 se comportent comme des mémoires tampons de type premier-entré-premier-sorti (ou FIFO pour « First In, First Out »). Les registres 11 se remplissant et se vident en respectant l'ordre d'arrivée des données, ici les opérandes AN (étape 103) et BN (étape 104). L'étape 106 est exécutée si les registres 11 sont non vides : les registres 11 sont dépilés des opérandes AN et BN. En variante, les registres 11 ne fonctionnent pas en mode FIFO. Dans ce cas, des données peuvent y être stockées plus durablement sans risquer d'être effacées et peuvent être réutilisées ultérieurement si besoin.

Au cours de l'étape 106, à réception de l'instruction d'exécution de calculs, l'ALU 9 exécute l'ensemble des opérations élémentaires correspondantes, dès que les opérandes sont disponibles sur les registres 11. L'étape 106 inclut donc la réception en entrées de l'ALU 9 de chacun des opérandes depuis les registres 11. Sous réserve que les opérations d'adressage 103, 104 aient été préalablement et correctement effectuées, l'étape 106 peut être dépourvue d'accès mémoire (en lecture).

Au cours de l'étape 107, les données formant résultats du traitement sont stockées sur les registres 11 en sorties de l'ALU 9. On mentionne ici uniquement les résultats du traitement et non les résultats de chacun des calculs élémentaires. En effet, dans le cas où certains des résultats des calculs élémentaires sont utilisés en tant qu'opérandes pour d'autres calculs élémentaires au cours de l'étape 107, de tels résultats (intermédiaires) peuvent devenir inutiles à l'issu du traitement. Dans ces cas, les résultats intermédiaires peuvent être supprimés des registres 11 à l'issue de l'étape 107 (par exemple écrasés par d'autres données, mode FIFO). En variante, l'ensemble des

données formant résultats des opérations élémentaires sont stockées sur les registres 11 à l'issue de l'étape 107 (mode différent du FIFO).

Au cours de l'étape 108, une adresses mémoire @CX pour chacune des données CX formant résultat du traitement est obtenue. Une telle opération d'adressage permet de déterminer l'emplacement mémoire sur lequel sera stockée chacune des données formant résultat. Dans l'exemple décrit ici, l'étape 108 est mise en œuvre après l'étape 107 d'inscription des résultats dans les registres 11 et avant l'écriture des résultats en mémoire 13 (étape 109 décrite ci-après). En variante, l'étape 107 peut être mise en œuvre plus précocement au cours du procédé, notamment avant l'étape 106. En effet, notamment lorsque la forme (par exemple la taille) des données de résultat est connue à l'avance, il est possible d'adresser les données de résultat avant même leur calcul. L'obtention des adresses mémoire @CX peut inclure la transmission d'instructions d'adressage depuis l'unité de commande 5.

Au cours de l'étape 109, chacune des données formant résultat du traitement est écrite en mémoire 13 à partir des registres 11, pour stockage et via l'interface mémoire 15, au moyen des adresses mémoire obtenues à l'étape 108.

En figure 6, des exemples de mises en œuvre des étapes 101, 102, 106 et 108 sont données sous forme de pseudocode informatique. De tels exemples, non limitatifs, représentent des opérations sous forme de boucles informatiques. L'utilisation de boucles est particulièrement avantageux pour limiter le nombre de cycles de calculs nécessaires, et donc pour améliorer l'efficacité, lorsque les opérations à mettre en œuvre sont sensiblement analogues les unes aux autres (par exemple toutes des additions) et que seules les données d'entrée varient. Dans de tels cas, les instructions de calcul transmises à l'étape 105 peuvent prendre la forme d'une boucle réitérée pour chaque opération.

L'ordonnancement chronologique des étapes 101, 102, 106, 108 est représenté par la flèche « t » en figure 6. Un tel ordonnancement constitue un exemple non limitatif. La figure 6 illustre que, contrairement à l'usage dans le

domaine technique, l'adressage des données d'entrée (opérandes) est mis en œuvre distinctement des calculs eux-mêmes. Autrement dit, l'adressage et le calcul sont traités comme deux processus distincts l'un de l'autre plutôt que d'être traités indistinctement, à la volée, à réception d'instructions générales. En particulier, l'étape 106 d'exécution des opérations élémentaires peut débuter alors même que tous les opérandes n'ont pas encore été téléchargés depuis la mémoire 13 sur les registres 11. Typiquement, les premiers cycles de l'étape 106 peuvent débuter dès que les premiers opérandes correspondant sont disponibles sur les registres 11 pour les calculs. Cette mise en œuvre en cascade des opérations confère au système son caractère asynchrone.

Dans des modes de réalisation, l'ALU 9 exécute (étape 106) l'ensemble des calculs élémentaires du traitement au cours de cycles de calculs consécutifs. Aucun accès mémoire n'est effectué par l'ALU 9 durant ces cycles de calculs. Ainsi, la mise en œuvre des calculs peut être particulièrement rapide. L'ALU 9, durant de tels cycles, est dispensée d'opération d'accès mémoire. En outre, du point de vue de l'ALU 9 effectuant les calculs, l'obtention des opérandes est similaire à un appel en mémoire 13 mais l'obtention des opérandes est plus rapide et indépendantes de l'interface mémoire 15 car les opérandes sont en pratiques lues directement dans les registres 11. Les accès mémoire sont mis en œuvre par une autre ALU (distincte de celle effectuant les calculs). Au moins au cours d'un processus, chaque ALU 9 a une fonction fixe : soit la mise en œuvre des calculs, soit la mise en œuvre des accès mémoire. Cette attribution de fonction fixe à chaque ALU 9 peut être modifiée à l'issue de la mise en œuvre du processus pour conférer de la souplesse au dispositif informatique. Néanmoins, cela implique souvent d'adapter en conséquence le parcours d'adressage. Dans les modes de réalisation préférés, la fonction de chaque ALU 9 est donc figée d'un processus à l'autre : les ALUs sont spécialisées.

Les procédés et les variantes décrits ci-avant peuvent prendre la forme d'un dispositif informatique, incluant un processeur ou un ensemble de processeurs, agencé pour mettre en œuvre un tel procédé.

L'invention ne se limite pas aux exemples de procédés et de dispositifs décrits ci-avant, seulement à titre d'exemple, mais elle englobe toutes les variantes que pourra envisager l'homme de l'art dans le cadre de la protection recherchée. L'invention concerne aussi un jeu d'instructions machines implémentable dans un processeur pour obtenir un tel dispositif informatique, tel qu'un processeur ou un ensemble de processeurs, l'implémentation d'un tel jeu d'instructions machines sur un processeur, le procédé de gestion d'architecture de processeur mis en œuvre par le processeur, le programme informatique comprenant le jeu d'instructions machines correspondant, ainsi que le support d'enregistrement sur lequel est enregistré informatiquement un tel jeu d'instructions machines.

En pages suivantes de la description du document original, sont donnés des exemples de mis en œuvre sous forme de pseudocode informatique.

Exemple sous forme de pseudocode informatique d'un traitement à effectuer sous la forme de dix additions :

```
Int A[10], B[10], C[10]
for (i=0; i<10; i++)
{
C[i] = A[i] + B[i];
}
```

Exemple sous forme de pseudocode informatique d'instructions usuelles pour effectuer un traitement composé de dix additions :

```
@A, @B, @C
Addr0 = @A
Addr1 = @B
Addr2 = @C
LOOP:
Load Addr0 → reg0
Load Addr1 → reg1
reg2 = reg0 + reg1
Store addr2 → reg2
Addr0 = Addr0 + 1
Addr1 = Addr1 + 1
Addr2 = Addr2 + 1
GOTO LOOP (10x)
```

Exemple sous forme de pseudocode informatique d'instructions pour effectuer un traitement composé de dix additions selon un mode de réalisation dans lequel les instructions d'adressage et les instructions de calcul sont distinguées les unes des autres :

```
//étape 101//
Addr = @A
LOOP
Load Addr
Addr = addr + 1
GOTO LOOP (10x)
```

```
//étape 102//
Addr = @B
LOOP
Load Addr
Addr = addr + 1
GOTO LOOP (10x)

//étape 106//
LOOP
c = a + b
GOTO LOOP (10x)

//étape 108//
Addr = @C
LOOP
Load Addr
Addr = addr + 1
GOTO LOOP (10x)
```

Revendications

1. Procédé de traitement de données, décomposable en un ensemble d'opérations élémentaires à effectuer, mis en œuvre par un dispositif (1) informatique, ledit dispositif (1) comprenant :

- une unité de commande (5) ;
- au moins une unité arithmétique et logique (9) ;
- un jeu de registres (11) aptes à alimenter en données formant opérande des entrées de ladite première unité arithmétique et logique (9) et aptes à être alimentés en données issues des sorties de ladite unité arithmétique et logique (9) ;
- une mémoire (13) ;
- une interface mémoire (15) par l'intermédiaire de laquelle des données (A0, A15) sont transmises et routées entre les registres (11) et la mémoire (13) ;

ledit procédé comprenant :

a) obtenir (101, 102) les adresses mémoire (@A0, @A15) de chacune des données absentes des registres (11) et formant opérande pour une au moins desdites opérations élémentaires à effectuer et ;

b) lire (103, 104) en mémoire (13), pour chargement dans les registres (11) via l'interface mémoire (15), chacune desdites données (A0, A15) au moyen des adresses mémoire (@A0, @A15) obtenues ;

c) transmettre (105) une instruction d'exécution de calculs depuis l'unité de commande (5) à destination de ladite première unité arithmétique et logique (9), ladite instruction étant dépourvue d'instruction d'adressage ;

d) à réception de ladite instruction d'exécution de calculs, et dès que les opérandes correspondants sont disponibles sur les registres (11), exécuter l'ensemble desdites opérations élémentaires (106) par ladite première unité arithmétique et logique (9) recevant en entrées chacun des opérandes depuis les registres (11) ;

e) stocker (107) les données (C0, C15) formant résultats du traitement sur les registres (11) en sorties de ladite première unité arithmétique et logique (9) ;

f) obtenir (108) une adresses mémoire (@C0, @C15) pour chacune des

données formant résultat du traitement ;

g) écrire (109) en mémoire (13) chacune des données (C0, C15) formant résultat du traitement issues des registres (11), pour stockage et via l'interface mémoire (15), au moyen des adresses mémoire (@C0, @C15) obtenues.

2. Procédé selon la revendication 1, dans lequel ladite première unité arithmétique et logique (9) exécute l'ensemble des calculs élémentaires du traitement au cours de cycles de calculs consécutifs, aucun accès mémoire n'étant effectué par ladite première unité arithmétique et logique (9) durant lesdits cycles de calculs.

3. Procédé selon l'une des revendications précédentes, dans lequel l'une au moins parmi les étapes suivantes comprend une boucle itératives :

a) obtenir (101, 102) les adresses mémoire (@A0, @A15) de chacune des données absentes des registres (11) et formant opérande pour une au moins desdites opérations élémentaires à effectuer et ;

d) à réception de ladite instruction d'exécution de calculs, exécuter l'ensemble desdites opérations élémentaires (106) par ladite première unité arithmétique et logique (9) recevant en entrées chacun des opérandes depuis les registres (11) ;

f) obtenir (108) une adresses mémoire (@C0, @C15) pour chacune des données formant résultat du traitement.

4. Procédé selon l'une des revendications précédentes, dans lequel le dispositif (1) comprend en outre au moins une unité arithmétique et logique supplémentaire et distincte de la première unité arithmétique et logique (9) exécutant l'ensemble desdites opérations élémentaires (106), l'unité arithmétique et logique supplémentaire mettant en œuvre :

a) l'obtention (101, 102) des adresses mémoire (@A0, @A15) de chacune des données absentes des registres (11) et formant opérande pour une au moins desdites opérations élémentaires à effectuer ; et

b) la lecture (103, 104) en mémoire (13), pour chargement dans les registres

(11) via l'interface mémoire (15), de chacune desdites données (A0, A15) au moyen des adresses mémoire (@A0, @A15) obtenues.

5. Dispositif (1) informatique de traitement de données, ledit traitement étant décomposable en un ensemble d'opérations élémentaires à effectuer, ledit dispositif (1) comprenant :

- une unité de commande (5) ;
- au moins une première unité arithmétique et logique (9) parmi une pluralité ;
- un jeu de registres (11) aptes à alimenter en données formant opérande des entrées desdites unités arithmétiques et logiques (9, 10) et aptes à être alimentés en données issues des sorties desdites unités arithmétiques et logiques (9) ;
- une mémoire (13) ;
- une interface mémoire (15) par l'intermédiaire de laquelle des données (A0, A15) sont transmises et routées entre les registres (11) et la mémoire (13) ;

ledit dispositif (1) informatique étant configuré pour :

a) obtenir (101, 102) les adresses mémoire de chacune des données absentes des registres (11) et formant opérande pour une au moins desdites opérations élémentaires à effectuer et ;

b) lire (103, 104) en mémoire (13), pour chargement dans les registres (11) via l'interface mémoire (15), chacune desdites données au moyen des adresses mémoire obtenues ;

c) transmettre (105) une instruction d'exécution de calculs depuis l'unité de commande (5) à destination de ladite première unité arithmétique et logique (9), ladite instruction étant dépourvue d'instruction d'adressage ;

d) à réception de ladite instruction d'exécution de calculs, et dès que les opérandes sont disponibles sur les registres (11), exécuter l'ensemble desdites opérations élémentaires (106) par ladite première unité arithmétique et logique (9) recevant en entrées chacun des opérandes depuis les registres (11) ;

e) stocker (107) les données formant résultats du traitement sur les registres (11) en sorties de ladite première unité arithmétique et logique (9) ;

f) obtenir (108) une adresses mémoire pour chacune des données formant

résultat du traitement ;

g) écrire (109) en mémoire (13) chacune des données formant résultat du traitement issues des registres (11), pour stockage et via l'interface mémoire (15), au moyen des adresses mémoire obtenues.

6. Jeu d'instructions machines pour la mise en œuvre du procédé selon l'une des revendications 1 à 4 lorsque ce programme est exécuté par un dispositif (1) informatique incluant au moins un processeur.

7. Programme informatique comportant des instructions pour la mise en œuvre du procédé selon l'une des revendications 1 à 4 lorsque ce programme est exécuté par un dispositif (1) informatique incluant au moins un processeur.

8. Support d'enregistrement non transitoire lisible par un ordinateur sur lequel est enregistré un programme pour la mise en œuvre du procédé selon l'une des revendications 1 à 4 lorsque ce programme est exécuté par un processeur.

Fig. 1

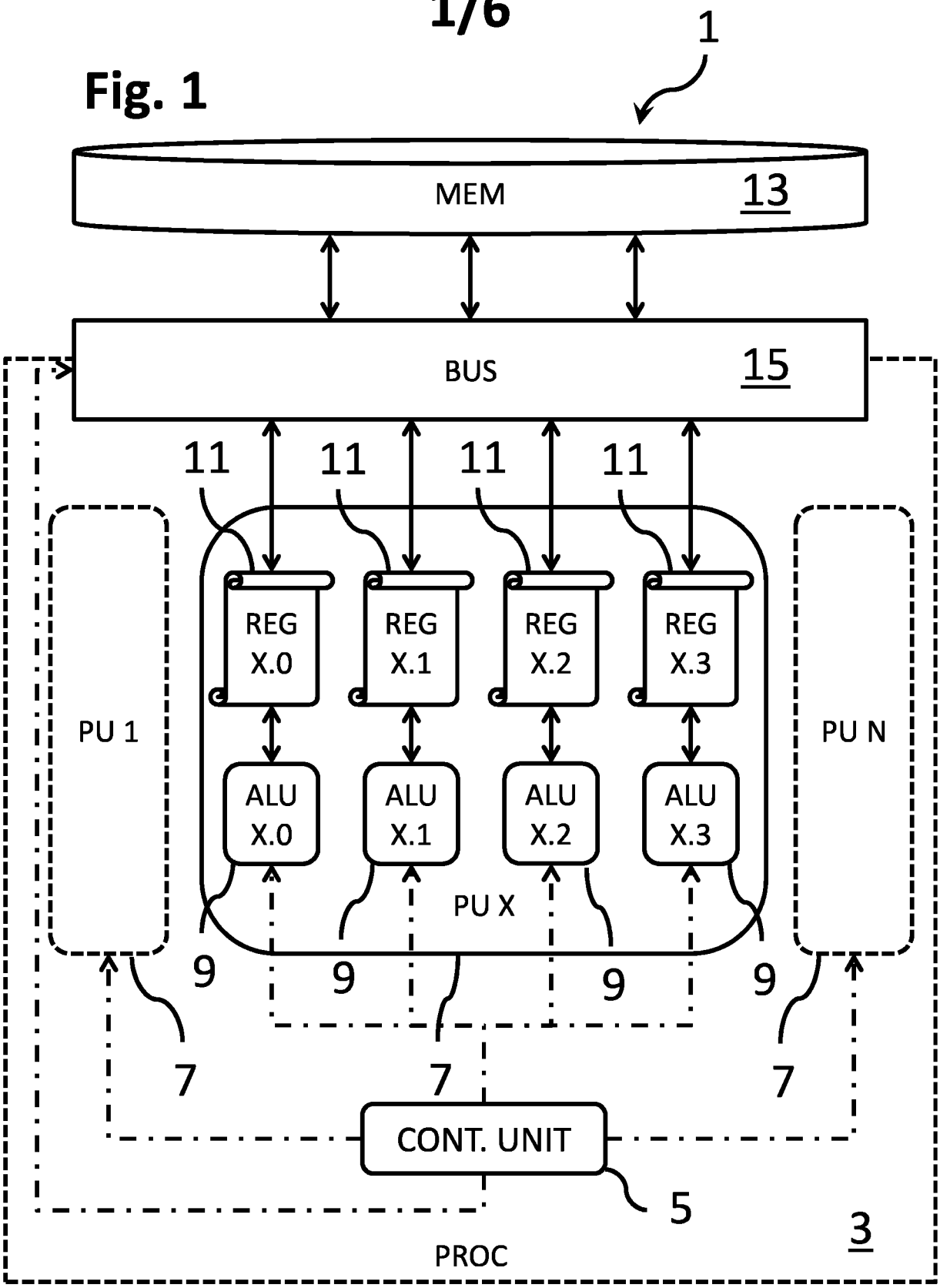

1
MEM
13
BUS
15
11
11
11
11
REG X.0
REG X.1
REG X.2
REG X.3
ALU X.0
ALU X.1
ALU X.2
ALU X.3
PU 1
PU X
PU N
9
9
9
9
7
7
7
CONT. UNIT
5
3
PROC

Fig. 2

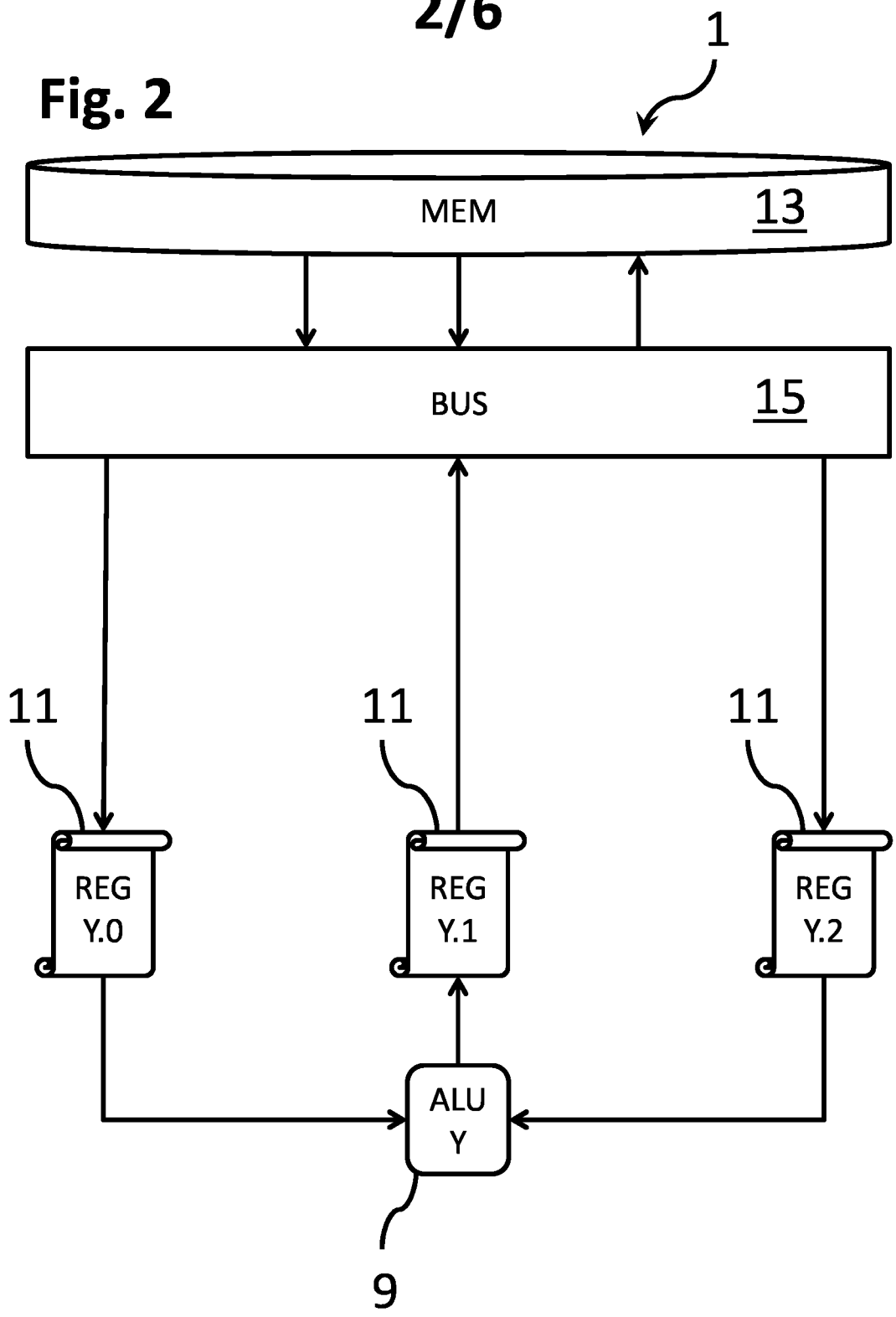


Fig. 3

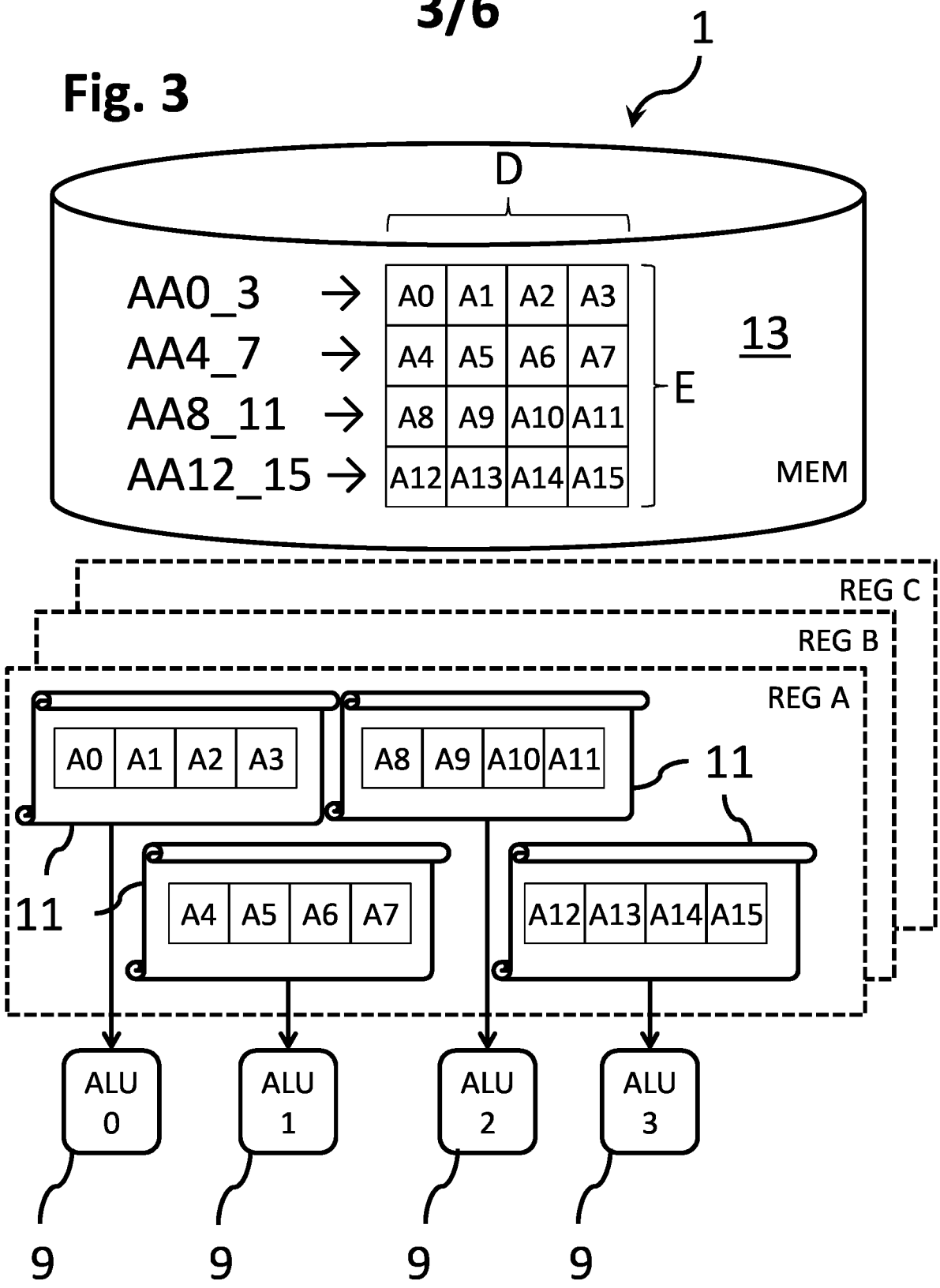

1
D
AA0_3
AA4_7
AA8_11
AA12_15
A0 A1 A2 A3
A4 A5 A6 A7
A8 A9 A10 A11
A12 A13 A14 A15
E
13
MEM
REG C
REG B
REG A
11
11
ALU 0
ALU 1
ALU 2
ALU 3
9
9
9
9

Fig. 4

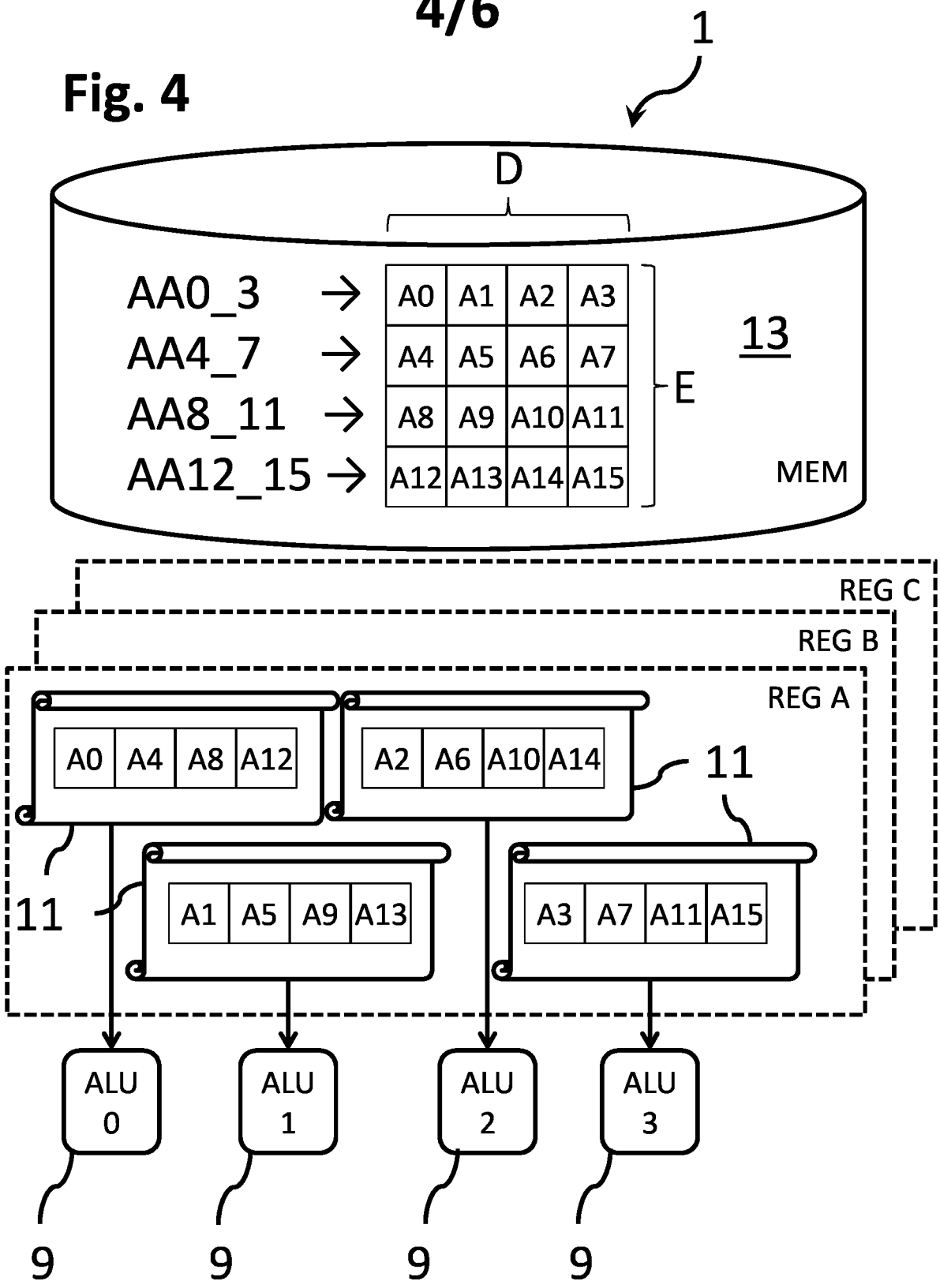

1
D
AA0_3
AA4_7
AA8_11
AA12_15
A0 A1 A2 A3
A4 A5 A6 A7
A8 A9 A10 A11
A12 A13 A14 A15
E
13
MEM
REG C
REG B
REG A
A0 A4 A8 A12
A2 A6 A10 A14
11
11
A1 A5 A9 A13
A3 A7 A11 A15
ALU 0
ALU 1
ALU 2
ALU 3
9
9
9
9

Fig. 5

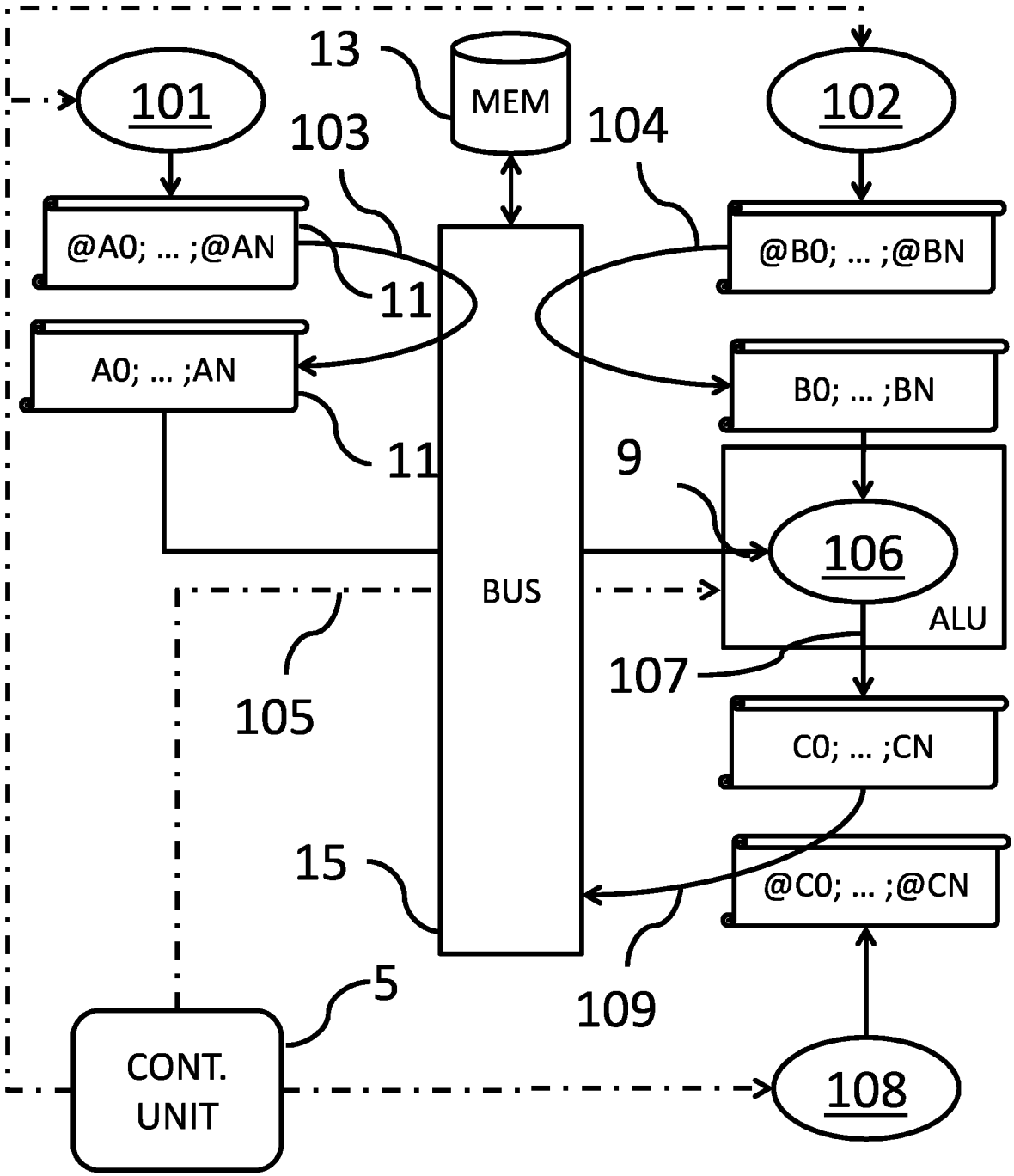
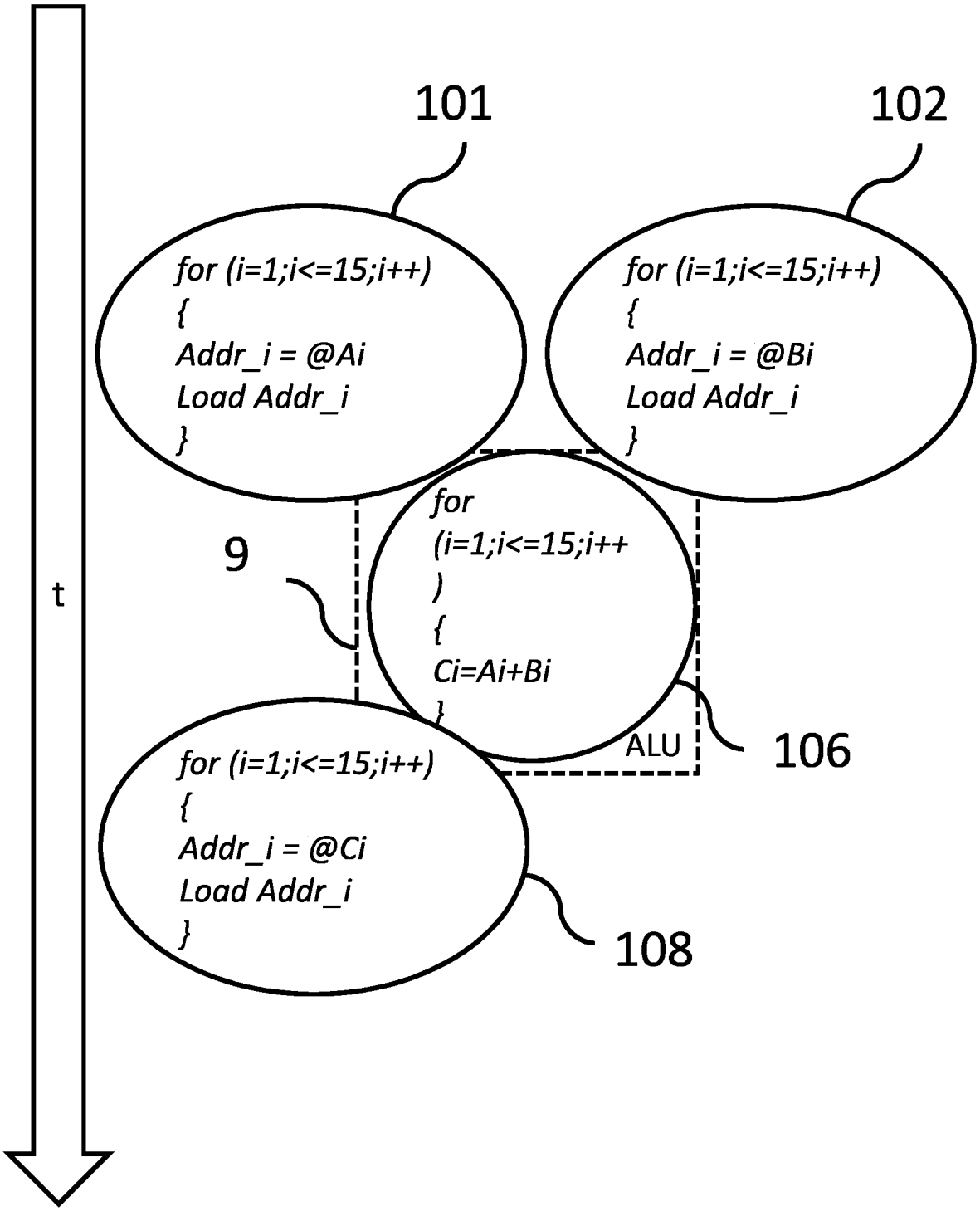

13
MEM
101
102
103
104
@A0; ... ;@AN
@B0; ... ;@BN
11
A0; ... ;AN
B0; ... ;BN
11
9
106
BUS
ALU
105
107
C0; ... ;CN
15
@C0; ... ;@CN
109
5
CONT.
UNIT
108

Fig. 6


t
101
102
for (i=1;i<=15;i++)
{
Addr_i = @Ai
Load Addr_i
}
for (i=1;i<=15;i++)
{
Addr_i = @Bi
Load Addr_i
}
9
for
(i=1;i<=15;i++
)
{
Ci=Ai+Bi
}
ALU
106
for (i=1;i<=15;i++)
{
Addr_i = @Ci
Load Addr_i
}
108

INTERNATIONAL SEARCH REPORT

International application No.
PCT/FR2019/051156

A. CLASSIFICATION OF SUBJECT MATTER

G06F 15/80(2006.01)i; ***G06F 9/38***(2018.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5513366 A (AGARWAL RAMESH C [US] ET AL) 30 April 1996 (1996-04-30) column 5, line 34 - line 35 column 6, line 12 - line 15 column 6, line 54 column 7, line 55 - line 61 column 8, line 3 - line 8 column 8, line 13 - line 14 column 8, line 33 - line 37 column 8, line 45 - line 47 column 8, line 49 - line 51 column 8, line 52 - line 53 column 9, line 51 - line 56 column 9, line 57 - line 65 column 9, line 58 - line 64 column 9, line 66 - column 10, line 6 figures 4,5	1-8

☑ Further documents are listed in the continuation of Box C. ☑ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
23 September 2019	**02 October 2019**

Name and mailing address of the ISA/EP	Authorized officer
European Patent Office **p.b. 5818, Patentlaan 2, 2280 HV Rijswijk** **Netherlands** Telephone No. **(+31-70)340-2040** Facsimile No. **(+31-70)340-3016**	**Gratia, Romain** Telephone No.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	GB 2407179 A (CLEARSPEED TECHNOLOGY LTD [GB]; CLEARSPEED SOLUTIONS LTD [GB] ET AL.) 20 April 2005 (2005-04-20) page 9, line 18 page 10, line 2 - line 4 page 14, line 2 - line 7 page 14, line 13 - line 14 page 15, line 18 - line 20 page 19, line 14 - line 17 page 20, line 10 - line 12 figure 2	1,5-8
X	EP 2144158 A1 (NEC CORP [JP]) 13 January 2010 (2010-01-13) paragraphs [0044], [0045], [0046], [0055], [0065], [0072], [0080]	1,5-8

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/FR2019/051156

Patent document cited in search report	Publication date (day/month/year)	Patent family member(s)	Publication date (day/month/year)
US 5513366 A	30 April 1996	NONE	
GB 2407179 A	20 April 2005	NONE	
EP 2144158 A1	13 January 2010	EP 2144158 A1	13 January 2010
		JP 4232838 B2	04 March 2009
		JP 2008250471 A	16 October 2008
		US 2010174891 A1	08 July 2010
		WO 2008123361 A1	16 October 2008

RAPPORT DE RECHERCHE INTERNATIONALE

Demande internationale n°
PCT/FR2019/051156

A. CLASSEMENT DE L'OBJET DE LA DEMANDE
INV. G06F15/80 G06F9/38
ADD.

Selon la classification internationale des brevets (CIB) ou à la fois selon la classification nationale et la CIB

B. DOMAINES SUR LESQUELS LA RECHERCHE A PORTE

Documentation minimale consultée (système de classification suivi des symboles de classement)
G06F

Documentation consultée autre que la documentation minimale dans la mesure où ces documents relèvent des domaines sur lesquels a porté la recherche

Base de données électronique consultée au cours de la recherche internationale (nom de la base de données, et si cela est réalisable, termes de recherche utilisés)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERES COMME PERTINENTS

Catégorie*	Identification des documents cités, avec, le cas échéant, l'indication des passages pertinents	no. des revendications visées
X	US 5 513 366 A (AGARWAL RAMESH C [US] ET AL) 30 avril 1996 (1996-04-30) colonne 5, ligne 34 - ligne 35 colonne 6, ligne 12 - ligne 15 colonne 6, ligne 54 colonne 7, ligne 55 - ligne 61 colonne 8, ligne 3 - ligne 8 colonne 8, ligne 13 - ligne 14 colonne 8, ligne 33 - ligne 37 colonne 8, ligne 45 - ligne 47 colonne 8, ligne 49 - ligne 51 colonne 8, ligne 52 - ligne 53 colonne 9, ligne 51 - ligne 56 colonne 9, ligne 57 - ligne 65 colonne 9, ligne 58 - ligne 64 colonne 9, ligne 66 - colonne 10, ligne 6 figures 4,5	1-8

-/--

[X] Voir la suite du cadre C pour la fin de la liste des documents

[X] Les documents de familles de brevets sont indiqués en annexe

* Catégories spéciales de documents cités:

"A" document définissant l'état général de la technique, non considéré comme particulièrement pertinent

"E" document antérieur, mais publié à la date de dépôt international ou après cette date

"L" document pouvant jeter un doute sur une revendication de priorité ou cité pour déterminer la date de publication d'une autre citation ou pour une raison spéciale (telle qu'indiquée)

"O" document se référant à une divulgation orale, à un usage, à une exposition ou tous autres moyens

"P" document publié avant la date de dépôt international, mais postérieurement à la date de priorité revendiquée

"T" document ultérieur publié après la date de dépôt international ou la date de priorité et n'appartenant pas à l'état de la technique pertinent, mais cité pour comprendre le principe ou la théorie constituant la base de l'invention

"X" document particulièrement pertinent; l'invention revendiquée ne peut être considérée comme nouvelle ou comme impliquant une activité inventive par rapport au document considéré isolément

"Y" document particulièrement pertinent; l'invention revendiquée ne peut être considérée comme impliquant une activité inventive lorsque le document est associé à un ou plusieurs autres documents de même nature, cette combinaison étant évidente pour une personne du métier

"&" document qui fait partie de la même famille de brevets

Date à laquelle la recherche internationale a été effectivement achevée	Date d'expédition du présent rapport de recherche internationale
23 septembre 2019	02/10/2019

Nom et adresse postale de l'administration chargée de la recherche internationale	Fonctionnaire autorisé
Office Européen des Brevets, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Gratia, Romain

2

C(suite). DOCUMENTS CONSIDERES COMME PERTINENTS

Catégorie*	Identification des documents cités, avec, le cas échéant, l'indication des passages pertinents	no. des revendications visées
X	GB 2 407 179 A (CLEARSPEED TECHNOLOGY LTD [GB]; CLEARSPEED SOLUTIONS LTD [GB] ET AL.) 20 avril 2005 (2005-04-20) page 9, ligne 18 page 10, ligne 2 - ligne 4 page 14, ligne 2 - ligne 7 page 14, ligne 13 - ligne 14 page 15, ligne 18 - ligne 20 page 19, ligne 14 - ligne 17 page 20, ligne 10 - ligne 12 figure 2	1,5-8
X	EP 2 144 158 A1 (NEC CORP [JP]) 13 janvier 2010 (2010-01-13) alinéas [0044], [0045], [0046], [0055], [0065], [0072], [0080]	1,5-8

RAPPORT DE RECHERCHE INTERNATIONALE

Renseignements relatifs aux membres de familles de brevets

Demande internationale n°
PCT/FR2019/051156

Document brevet cité au rapport de recherche	Date de publication	Membre(s) de la famille de brevet(s)	Date de publication
US 5513366 A	30-04-1996	AUCUN	
GB 2407179 A	20-04-2005	AUCUN	
EP 2144158 A1	13-01-2010	EP 2144158 A1	13-01-2010
		JP 4232838 B2	04-03-2009
		JP 2008250471 A	16-10-2008
		US 2010174891 A1	08-07-2010
		WO 2008123361 A1	16-10-2008