(19) RÉPUBLIQUE FRANÇAISE

INSTITUT NATIONAL
DE LA PROPRIÉTÉ INDUSTRIELLE

PARIS

(11) **N° de publication :** **2 919 081**
(à n'utiliser que pour les commandes de reproduction)

(21) **N° d'enregistrement national :** **07 05152**

(51) Int Cl[8] : ***G 06 F 7/38*** *(2006.01)*

(12) DEMANDE DE BREVET D'INVENTION A1

(22) **Date de dépôt :** 17.07.07.

(30) **Priorité :**

(43) **Date de mise à la disposition du public de la demande :** 23.01.09 Bulletin 09/04.

(56) **Liste des documents cités dans le rapport de recherche préliminaire :** *Se reporter à la fin du présent fascicule*

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

(71) **Demandeur(s) :** *UNIVERSITE DE LA REUNION* — FR.

(72) **Inventeur(s) :** BURCKEL SERGE et GIOAN EMERIC.

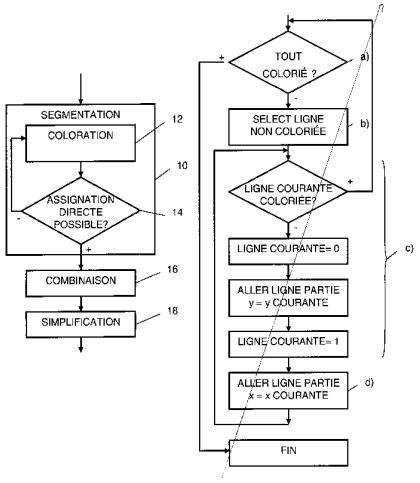
(73) **Titulaire(s) :**

(74) **Mandataire(s) :** CABINET PLASSERAUD.

(54) IMPLEMENTATION DE CALCULS POUR PROCESSEURS.

(57) Ce procédé d'implémentation d'une opération (E) sur un nombre quelconque K, strictement supérieur à 1, de mémoires élémentaires pouvant chacune prendre M valeurs, caractérisé en ce qu'il comprend les étapes suivantes:

- au moins une segmentation (10) de l'opération en une série de bijections;
- une combinaison (16) desdites bijections en fonction des mémoires élémentaires qu'elles affectent;
- une détermination (18) d'une séquence d'assignation des mémoires élémentaires à partir desdites bijections combinées pour effectuer le calcul de ladite opération (E).



FR 2 919 081 - A1

IMPLEMENTATION DE CALCULS POUR PROCESSEURS

La présente invention concerne les calculs réalisés par des processeurs.

De manière générale, les processeurs réalisent des calculs par des séquences d'opérations simples d'assignations sur des mémoires élémentaires, tels que des registres.

Ces mémoires élémentaires sont en nombre limité et les accès du processeur à ces mémoires sont plus rapides que les accès aux autres mémoires périphériques associées aux processeurs.

Afin d'optimiser les temps de calcul, il faut donc limiter le nombre d'accès aux mémoires périphériques et optimiser l'utilisation des mémoires élémentaires.

Plus précisément, les opérations de calcul sont souvent implémentées avec l'utilisation de mémoires élémentaires supplémentaires de temporisation.

Par implémentation, on entend la réalisation de la phase finale de l'élaboration d'un système qui permet aux éléments matériels ainsi qu'aux éléments logiciels d'entrer en fonction, c'est-à-dire la détermination de la séquence de contrôle dans un système informatique.

Par exemple, une opération simple telle que l'échange du contenu de deux registres notés A et B utilise en général un troisième mémoire C dans laquelle le contenu d'un des deux registres est mémorisé de manière temporaire.

Il existe également d'autres méthodes d'implémentation des opérations de calcul utilisant des opérations entre les registres afin d'éviter l'utilisation de mémoires élémentaires pour le stockage de valeurs tampons.

Par exemple, la fonction d'échange du contenu de deux registres A et B peut être réalisée à l'aide des trois étapes suivantes :

- le registre A prend la valeur de la somme des registres A et B ;
- le registre B prend ensuite la valeur du registre A diminuée de la valeur du registre B ; puis,

- le registre A prend la valeur du registre A diminuée de la valeur du registre B.

A l'issue de ces trois étapes, les valeurs des registres A et B sont échangées.

Il est démontré dans la publication « *Closed iterative calculus* » Theoritical Computer Science, n° 158 de 1996, pages 371 à 378 par Serge Burckel, qu'il est possible d'implémenter n'importe quelle opération sur un nombre quelconque K de registres sans utiliser de variables supplémentaires à l'aide de telles séquences d'opérations.

Si chacun de ces registres comporte N bits, tel que 32 bits par exemple, le nombre d'opérations nécessaires est de l'ordre de NxK^2 comme démontré dans la publication « *Quadratic Sequential Computations of Bouleon Mappings*, Theory of Computing Systems », n° 37 de 2004, pages 519 à 525 par Serge Burckel et Marianne Morillon.

Ces techniques requièrent donc un nombre élevé d'opérations.

Comme indiqué précédemment, il est intéressant d'optimiser l'utilisation des registres afin de réduire les temps de calcul des processeurs.

Ceci entraîne également des économies en énergie.

Un avantage de l'invention est de permettre une implémentation plus efficace des opérations, notamment sur des registres pour optimiser les temps de calcul et la consommation énergétique.

A cet effet, la présente invention a pour objet un <REV 1>

En conséquence, le nombre total d'opération est sensiblement réduit de sorte que le temps de calcul et la consommation sont également réduits.

Selon d'autres modes de réalisation :

<REV secondaires>

L'invention sera mieux comprise à la lumière de la description et des dessins, sur lesquels :

- la figure 1 est un organigramme général d'un mode de réalisation de l'invention ;
- la figure 2 représente le détail d'une opération de coloration ;

- la figure 3 représente un exemple numérique de l'opération de coloration de la figure 2 ;
- la figure 4 est un organigramme général d'un autre mode de réalisation de l'invention ; et
- la figure 5 est un organigramme général d'encore un autre mode de réalisation de l'invention.

Dans un premier mode de réalisation, l'invention est appliquée pour l'implémentation d'une opération bijective notée E sur un nombre K de registres binaires.

Une opération E est dite bijective, si et seulement si deux antécédents distincts, auxquels est appliquée cette opération, aboutissent forcément à deux images distinctes, c'est-à-dire que deux n-uplets différents n'auront pas la même image par E.

A titre préalable, il convient également de rappeler que, de manière générale, on appelle un n-uplet un vecteur comprenant n composantes, chacune sélectionnée dans l'ensemble des valeurs possibles. Par exemple, dans le cas binaire, les valeurs possibles sont 0 et 1 et un 3-uplet est un mot de trois bits.

Les registres peuvent être considérés chacun comme une mémoire élémentaire pouvant prendre un nombre M de valeurs. Dans le cas des registres binaires, ce nombre M est égal à 2 exposant le nombre N de bits du registre. Il est également possible de considérer un nombre M de valeurs égales à 0 ou 1 et de calculer bit à bit, chaque bit du registre étant considéré comme une mémoire élémentaire. Les séquences d'assignations successives portant sur des bits d'un même registre peuvent ensuite être regroupées en une seule assignation de ce registre.

En référence aux figures 1 à 3, on va maintenant décrire l'organigramme général du procédé de l'invention appliqué à une opération bijective E portant sur K mémoires élémentaires pouvant prendre M valeurs.

Le procédé comporte tout d'abord une étape 10 de segmentation de l'opération en une série de bijections.

Plus précisément, cette étape 10 de segmentation est une segmentation itérative, ou encore inductive, pour former M bijections à chaque itération jusqu'à ce que les fonctions bijectives obtenues soient directement calculables pas des assignations des mémoires élémentaires.

Ainsi, on considère formellement une table de M^K lignes pour la définition de E où M est le nombre de valeurs possibles. De manière générale on note : E(x_1,x_2,...,x_K)=(y_1,y_2,...,y_K).

Dans l'exemple, les mémoires élémentaires sont des bits dont les valeurs possibles sont S = {0,1} de sorte que M est égal à 2. Pour l'exemple numérique, K est sélectionné égal à 3. Une bijection E sur trois bits A, B, C est alors définie par une table de 2^3 lignes, soit huit lignes. On définit pour l'exemple la fonction E suivante :

E(ABC) = (ac OR AbC OR Bc; A; aB OR AC)

en notant NON(A) = a, NON(B)=b, NON (C) = c et A AND B = AB.

Cette fonction logique se définit par la table de huit lignes :

E(000) = 100 ;
E(001) = 000 ;
E(010) = 101 ;
E(011) = 001 ;
E(100) = 010 ;
E(100) = 010 ;
E(101) = 111 ;
E(110) = 110 ; et
E(111) = 011.

Au cours de l'étape 10, la bijection E est tout d'abord segmentée en M autres bijections. Au cours d'une étape 12, on applique une coloration des lignes définissant la fonction E, grâce à un théorème classique de Köning en théorie des graphes. Cette coloration est réalisée en répartissant les termes de la fonction E sur les arêtes d'un graphe bipartite. Il est ainsi possible de donner une couleur parmi M à chaque ligne de la table définissant la fonction E en considérant de manière générale que deux lignes ont une couleur différente si

les mêmes parties des vecteurs sources sont égales ou si les mêmes parties de vecteurs images sont égales.

Ainsi, en considérant :

$$E(a\_1,a\_2\ldots a\_K) = (y\_1,y\_2, \ldots y\_K) \text{ et}$$

$$E(b\_1\ldots b\_K) = (z\_1\ldots z\_K)$$

si les parties droites des vecteurs source (a_2…a_K) et (b_2…b_K) ou encore les parties droites des vecteurs images (y_2, …y_K) et (z_2, …z_K) sont égales, alors les couleurs des deux lignes sont différentes.

Dans le cas de l'ensemble binaire {0,1}, la coloration peut être obtenu algébriquement comme une application booléenne notée C dans l'ensemble de solution {0,1} vérifiant :

$C(0,x) \neq C(1,x)$ et $C(E(0,x)) \neq C(E(1,x))$ pour tout x appartenant à S^{K-1}. Dans ce cas, la couleur est donnée à la partie image des lignes.

Des méthodes existantes permettent d'obtenir de telles colorations selon la théorie des graphes.

Dans le cas particulier illustré précédemment, avec S = {0,1}, il est possible d'utiliser l'algorithme suivant pour effectuer le coloriage des lignes de la table,définissant l'opération E. Cet algorithme est illustré sur la figure 2 :

a) si toutes les lignes sont coloriées, la coloration est terminée ;

b) une ligne non coloriée est sélectionnée ;

c) si la ligne courante est coloriée, aller en a) sinon la colorier par 0. En considérant la partie droite de la ligne courante, c'est-à-dire la partie image, à l'exception du premier terme, aller à l'autre ligne ayant la même partie droite à l'exception du premier terme et la colorier par la valeur 1 ;

d) en considérant la partie gauche de la ligne courante, c'est-à-dire la partie antécédents, à l'exception du premier terme, aller à l'autre ligne dont la partie gauche est identique à l'exception du premier terme et se rendre à l'étape c).

L'application de ce procédé aboutit dans le cas particulier illustré au coloriage suivant illustré également en référence à la figure 3 :

E(000) = 100 : couleur 0 ;

E(001) = 000 : couleur 1 ;

E(010) = 101 : couleur 1 ;

E(011) = 001 : couleur 0 ;

E(100) = 010 : couleur 1 ;

E(101) = 111 : couleur 0 ;

E(110) = 110 : couleur 0 ; et

E(111) = 011 : couleur 1.

Si l'on considère chaque ensemble de lignes de même couleur en ignorant leurs premières composantes, on obtient M bijections notées E_0 à E_M-1, chacune sur l'ensemble M^{K-1}.

Ainsi dans l'exemple, les quatre lignes de couleur 0, en oubliant leur première composante respective, permettent d'obtenir la bijection E_0 définie par les lignes suivantes :

E_0(00) = 00 ;

E_0(11) = 01 ;

E_0(01) = 11 ; et

E_0(10) = 10.

Pour les lignes de couleur 1, on obtient donc la bijection E_1 selon laquelle :

E_1(01) = 00 ;

E_1(10) = 01 ;

E_1(00) = 10 ; et

E_1(11) = 11.

Cette étape 12 de segmentation par coloration est répétée jusqu'à ce que les fonctions bijectives obtenues soient directement transformables en des assignations des mémoires élémentaires, c'est-à-dire en des assignations des bits.

En fonction des modes de réalisation, cette étape peut donc être répétée jusqu'à ce que l'on obtienne des bijections portant à chaque fois sur une seule mémoire élémentaire ou alors jusqu'à ce que l'on obtienne des bijections compilables automatiquement, c'est-à-dire directement calculables en

assignations. Ceci est représenté en référence à la figure 3, sur laquelle la segmentation est poursuivie jusqu'à l'obtention d'assignations unitaires.

Un test 14 permet de déterminer si les fonctions obtenues sont calculables en des assignations, par exemple par comparaison avec une liste de fonctions connues.

A l'issue de l'étape 10 de segmentation, le procédé délivre donc, pour chaque bijection, un nombre M de programmes de calcul portant sur K-1 mémoires, c'est-à-dire M séquences d'assignations de ces K-1 mémoires.

De manière générale, on obtient ainsi pour chaque bijection E_i, pour i de 0 à M-1, une séquence d'assignation de la forme suivante :

R_2 : = E_i2 (R_2,...,R_K) ;

....

R_(K-1) : = E_i(K-1) (R_2,...,R_K) ;

R_K : = E_iK (R_2,...,R_K) ;

R_(K-1) : = E'_i(K-1)(R_2,...,R_K) ;

....

R_2 : = E'_i2 (R_2,...,R_K).

Par rapport à l'exemple précédent, on obtiendra les programmes suivants pour le calcul algébrique de E_0 :

B: = B+C ;

C: = C ; et

B: = B.

De même, le programme suivant est obtenu pour le calcul algébrique de E_1 :

B: = 1+B+C ;

C: = 1+B+C ; et

B: = B.

Le procédé comprend ensuite une étape 16 de combinaison de ces M calculs en un calcul de la bijection traitée de la forme suivante :

R_1 : = E_1 (R_1,...,R_K) ;

R_2 : = E_2 (R_1,...,R_K) ;

...

R_(K-1) : = E_(K-1) (R_1,...,R_K) ;

R_K : = E_K (R_1,...,R_K) ;

R_(K-1) : = E'_(K-1) (R_1,...,R_K) ;

....

R_2 : = E'_2 (R_1,...,R_K) ; et

R_1 : = E'_1 (R_1,...,R_K).

Pour le premier terme, on adopte la couleur de la ligne de sorte que :

E_1(x_1,...,x_K)=la couleur de la ligne de E(x_1,x_2,...,x_K)

Pour l'exemple :

E_1(000)=0 ;

E_1(001)=1 ;

E_1(010)=1 ;

E_1(011)=0 ;

E_1(100)=1 ;

E_1(101)=0 ;

E_1(110)=0 ; et

E_1(111)=1.

Ceci se traduit par l'assignation algébrique suivante : A := A+B+C.

Les autres termes sont définis de manière conditionnelle en fonction du terme précédent. Ainsi, E_2 est définie par les cas suivants :

E_2(x_1,...,x_K) = si x_1 = 0 alors E_02(x_2,..,x_K) ;

si x_1 = 1 alors E_12(x_2,..,x_K) ;

...

si x_1 = M-1 alors E_(M-1)2(x_2,..,x_K).

De la même manière, on définit les assignation E_3,...,E_K,E'_(K-1),...,E'_2.

Pour l'exemple on obtient les assignations conditionnelles suivantes :

B : = si A=0 alors (B+C) ;

si A=1 alors (1+B+C) ;

C : = si A=0 alors (C) ;

si A=1 alors (1+B+C) ;

B : = si A=0 alors (B) ; et

si A=1 alors (B).

Ces différentes assignations se traduisent en :

B := (1+A)*(B+C) + A*(1+B+C) ;

C := (1+A)*C+ A*(1+B+C) ; et

B := (1+A)*B+ A*B.

Il est alors possible de simplifier ces assignations lors d'une étape 18 pour obtenir les séquences suivantes :

B := A+B+C ;

C := A+C+A*B ; et

B := B.

Enfin, la dernière assignation f'_1 est déterminée de manière similaire à f_1 en utilisant la couleur de la ligne de sorte que :

f'_1(x_1,y_2,...,y_K) = y_1 pour la ligne de couleur x_1 de la forme :

E(...) = (y_1,y_2,...,y_K).

Dans l'exemple illustré précédemment :

f'_1(000)=1 car la couleur 0 a été donnée à E(000)=100 ;

f'_1(001)=0 car la couleur 0 a été donnée à E(011)=001 ;

f'_1(010)=1 car la couleur 0 a été donnée à E(110)=110 ;

f'_1(011)=1 car la couleur 0 a été donnée à E(101)=111 ;

f'_1(100)=0 car la couleur 1 a été donnée à E(001)=000 ;

f'_1(101)=1 car la couleur 1 a été donnée à E(010)=101 ;

f'_1(110)=0 car la couleur 1 a été donnée à E(100)=010 ; et

f'_1(111)=0 car la couleur 1 a été donnée à E(111)=011.

Ceci se traduit algébriquement par l'assignation

A := 1+A+C+B*C.

Il est alors possible après combinaison et simplification d'obtenir une expression complète du calcul de la bijection E :

A := A+B+C ;

B := A+B+C ;

C := A+C+A*B;

B := B ; et

A := 1+A+C+B*C.

Le procédé de l'invention permet donc d'exprimer l'opération E par une série de 2K-1 assignations qui sont implémentables aisément au niveau du microprocesseur.

Ces assignations sont réalisées dans un ordre croissant puis décroissant d'une première mémoire élémentaire à une dernière mémoire élémentaire puis dans l'ordre décroissant correspondant. Cependant, l'ordre initial des mémoires peut être choisi arbitrairement.

Ces assignations peuvent notamment être implémentées par des méthodes standard, notamment avec des portes logiques, telles que des portes NAND couramment utilisées dans les processeurs actuels.

Il est également à noter que l'expression des opérations peut être faite indifféremment de manière polynomiale, logique ou binaire.

Par ailleurs, le principe de l'invention peut également être étendu de manière plus générale à tout type d'opération et notamment à des opérations non bijectives.

Ainsi, en appelant E une opération quelconque sur K registres pouvant chacun prendre M valeurs, il est possible de calculer l'opération E par une séquence de 5K-4 assignations. La figure 4 représente un organigramme général du procédé de l'invention appliqué à une opération non bijective.

Le procédé débute par une étape 30 préalable de construction de deux bijections et d'une application simple.

L'étape 30 comporte d'abord une sous-étape 32 de détermination d'une bijection F qui permet d'attribuer à tous les K-uplets ayant une même image par E des éléments consécutifs par paquets dans l'ordre lexicographique.

Ensuite, une application simple P est déterminée lors d'une sous-étape 34. Cette application P transforme des éléments consécutifs en le numéro du paquet correspondant, ce numéro étant lui-même représenté en base M par un K-uplet.

Plus précisément, la bijection F et l'application P sont obtenues par l'algorithme suivant à l'aide de la fonction NEXT qui, à un K-uplet donné, associe la valeur du K-uplet suivant dans l'ordre lexicographique :

a) pour tous les K-uplets x, les images F(x) et P(x) sont indéfinies et R et T sont les K-uplets nuls : R=T=(0,0,0,...,0) ;

b) si F est partout définie alors l'algorithme est fini ;

c) pour un x tel que F(x) est indéfinie faire :

tant qu'il existe un x' tel que E(x') = E(x) et F(x') est indéfinie faire :

F(x') = T; P(T) = R; T = NEXT(T) ;

d) faire R = NEXT(R) et aller en b).

On reprend ci après un exemple numérique avec M = 2 et K = 3. On a NEXT(000) = 001; NEXT(001) = 010; NEXT(010)=011; NEXT(011) = 100 etc.

On considère pour l'exemple, une opération E définie sur trois bits ABC par la table suivante :

E(000)=101 ;

E(001)=000 ;

E(010)=101 ;

E(011)=001 ;

E(100)=000 ;

E(101)=101 ;

E(110)=001 ; et

E(111)=000.

Cette opération correspond à la définition logique suivante :

E(ABC)=(ac OR AbC ; FAUX ; ac OR aB OR Bc OR AbC).

L'application de l'algorithme défini précédemment permet d'obtenir la définition suivante des fonctions F et P :

1 : F(000) = 000 P(000)=000 ;

4 : F(001) = 011 P(011)=001 ;

2 : F(010) = 001 P(001)=000 ;

7 : F(011) = 110 P(110)=010 ;

5 : F(100) = 100 P(100)=001 ;

3 : F(101) = 010 P(010)=000 ;

8 : F(110) = 111 P(111)=010 ; et

6 : F(111) = 101 P(101)=001.

Le numéro en tête de ligne indique l'ordre dans lequel les lignes ont été considérées.

Pour calculer l'application P, on identifie successivement chaque K-uplet (x_1,...,x_K) à son image P(x_1,..,x_K)=(Y_1,...,Y_K) de la droite vers la

gauche. Il suffit de K étapes d'assignation pour calculer P.

Plus précisément, on définit K applications de M^K dans M : p_K,...,p_2,p_1 telles que pour tout K-uplet x et tout i de 1 à K :

si x=(x_1,..,x_i,...,x_K) et P(x)=(Y_1,...,Y_i,...,Y_K) alors

p_i(x_1,..,x_(i-1),x_i,Y_(i+1),..,Y_K)=Y_i.

Dans le cas de l'exemple numérique utilisé précédemment, on obtient p_3,p_2,p_1 de $\{0,1\}^3$ dans {0,1} selon le principe suivant :

comme P(000) = (000) alors p_3(000) = 0 p_2(000) = 0 p_1(000) = 0 ;

comme P(001) = (000) alors p_3(001) = 0 p_2(000) = 0 p_1(000) = 0 ;

comme P(010) = (000) alors p_3(010) = 0 p_2(010) = 0 p_1(000) = 0 ;

comme P(011) = (001) alors p_3(011) = 1 p_2(011) = 0 p_1(001) = 0 ;

comme P(100) = (001) alors p_3(100) = 1 p_2(101) = 0 p_1(101) = 0 ;

comme P(101) = (001) alors p_3(101) = 1 p_2(101) = 0 p_1(101) = 0 ;

comme P(110) = (010) alors p_3(110) = 0 p_2(110) = 1 p_1(110) = 0 ; et

comme P(111) = (010) alors p_3(111) = 0 p_2(110) = 1 p_1(110) = 0.

Les valeurs des p_i(x) non définies sont étendues arbitrairement. On peut les choisir de manière à rendre p_i la plus simplement calculable. Si on choisit ici p_2(111)=1 et p_1(111)=0, on obtient la séquence d'assignations suivante, exprimée sous forme logique :

C: = aBC OR Ab ;

B: = AB ; et

A: = FAUX.

Ou encore par la séquence d'assignations algébriques suivante :

C: = A+A*B+A*B*C+B*C ;

B: = A*B ; et

A: = 0.

Le procédé comporte ensuite une sous-étape 36 de détermination d'une autre bijection notée G qui transforme un numéro de paquet en l'image par E des K-uplets de départ identifiés par ce numéro de paquet.

La bijection G vérifie donc pour tout K-uplet, on a G(P(F(x)))=E(x).

Dans l'exemple G est définie par :

G(000) =101 car les éléments du paquet numéro 000 ont pour image 101 ;

G(001) =000 car les éléments du paquet numéro 001 ont pour image 000 ;

G(010) =001 car les éléments du paquet numéro 010 ont pour image 001 ;

G(011) =au choix par exemple 011 ;

G(100) =au choix par exemple 100 ;

G(101) =au choix par exemple 010 ;

G(110) =au choix par exemple 110 ; et

G(111) =au choix par exemple 111.

L'opération E est donc exprimée sous la forme de deux bijections F et G et d'une application P.

Da manière générale, on considère que la bijection F est déterminée à partir de l'opération à transformer afin d'attribuer une image différente à chaque antécédent ayant la même image par cette opération à transformer. L'application P fait le lien entre les antécédents et les images obtenues par la première bijection. La deuxième bijection G transforme les produits obtenus par la bijection F et l'application P pour former les termes de l'opération à transformer.

Le procédé consiste ensuite à appliquer la segmentation du cas bijectif explicitée précédemment tour à tour à chacune des bijections F et G.

Lors d'une étape 40, on transforme la bijection F en une séquence de 2K-1 assignations de sorte que tout K-uplet x est transformé en y=F(x) par une séquence d'assignations.

On obtient ainsi à l'issue de l'étape 40, une séquence de 2K-1 assignations pour calculer F :

R_K := f_K (R_1,...,R_K) ;

R_(K-1) := f_(K-1) (R_1,...,R_K) ;

...

R_2 := f_2 (R_1,...,R_K) ;

R_1 := f_1 (R_1,...,R_K) ;

R_2 := f'_2 (R_1,...,R_K) ;

...

R_(K-1) := f'_(K-1) (R_1,...,R_K) ; et

R_K :=f'_K (R_1,...,R_K).

Dans le cas illustré, le calcul est construit de la droite vers la gauche. C'est une construction similaire au cas explicité précédemment car il suffit de symétriser pour transformer les relations F(x_1,...,x_K)=(y_1,...,y_K) en F'(x_K,...,x_1)=(y_K,...,y_1) puis d'appliquer la méthode de calcul pour la bijection F'.

Dans le cas illustré, on obtient la fonction F' suivante :

F(000)=000 donc F'(000)=000 ;
F(001)=011 donc F'(100)=110 ;
F(010)=001 donc F'(010)=100 ;
F(011)=110 donc F'(110)=011 ;
F(100)=100 donc F'(001)=001 ;
F(101)=010 donc F'(101)=010 ;
F(110)=111 donc F'(011)=111 ; et
F(111)=101 donc F'(111)=101.

Cette fonction est transformée, à l'issue de l'étape 40 en la séquence de calcul suivante :

C :=A+B+C+A*B ;
B :=A+B+A*C ;
A :=A+B+B*C ;
B :=B+C+A*C ; et
C :=A+C+A*B.

Le procédé comporte ensuite une étape 42 de calcul de l'application de la fonction P de sorte que tout K-uplet y est remplacé par son image z=P(y) en K étapes d'assignations :

R_K :=p_K (R_1,...,R_K) ;
R_(K-1) :=p_(K-1) (R_1,...,R_K) ;
...
R_2 :=p_2 (R_1,...,R_K) ; et
R_1 :=p_1 (R_1,...,R_K).

Dans l'exemple, cela se traduit par les assignations suivantes :

C: = A+A*B+A*B*C+B*C ;
B: = A*B ; et

A: = 0.

Enfin, le procédé comporte une étape 44 de calcul de la bijection G selon la méthode explicitée précédemment de sorte que tout K-uplet z est transformé par la méthode des bijections en t= G(z) en 2K-1 étapes :

R_1 :=g_1 (R_1,...,R_K) ;
R_2 :=g_2 (R_1,...,R_K) ;
....
R_(K-1) :=g_(K-1) (R_1,...,R_K) ;
R_K :=g_K (R_1,...,R_K) ;
R_(K-1) :=g'_(K-1) (R_1,...,R_K) ;
....
R_2 :=g'_2 (R_1,...,R_K) ; et
R_1 :=g'_1 (R_1,...,R_K).

Cette fois-ci, la bijection décomposée de gauche à droite à l'inverse de la segmentation de la bijection F.

Pour l'exemple on avait choisi :

G(000)=101 ;
G(001)=000 ;
G(010)=001 ;
G(011)=011 ;
G(100)=100 ;
G(101)=010 ;
G(110)=110 ; et
G(111)=111.

Le procédé de l'invention aboutit à la séquence d'assignations suivante pour le calcul de G :

A :=A+B+B*C ;
B : = B+A*C ;
C : = 1+A+B+C ;
B : = B+A*C ; et
A : = A+B+C.

Les assignations sont ensuite combinées au cours d'une étape 46 afin d'obtenir 5K-2 assignations qui modifient successivement les registres de

numéros :

K,K-1,...,2,1,2,...K-1,K,K,K-1,...,2,1,1,2,...,K-1,K,K-1,...2,1.

Dans l'exemple, on obtient :

C :=A+B+C+A*B ;

B :=A+B+A*C ;

A :=A+B+B*C ;

B :=B+C+A*C ;

C :=A+C+A*B ;

C :=A+A*B+A*B*C+B*C ;

B :=A*B ;

A :=0 ;

A :=A+B+B*C ;

B :=B+A*C ;

C :=1+A+B+C ;

B :=B+A*C ; et

A :=A+B+C.

Le procédé comporte enfin une étape 48 de simplification des assignations successives portant sur le même registre. Par exemple, la séquence de deux assignations : A: = (B+C)*F et A: = A*D+B+4*A, se résume en une seule en remplaçant dans la seconde assignation les « A » par la valeur prise par « A » dans la première assignation. On obtient ainsi A: = [(B+C)*F]*D+B+4*[(B+C)*F].

De même, toujours à titre d'exemple, la séquence de deux assignations A: = (B+C)*F et A: = G+H se résume en une seule en substituant dans la seconde les A par la valeur prise par A dans la première. Toutefois, dans ce cas, la seconde assignation est indépendante de la valeur de A, de sorte que seule la seconde assignation est prise en compte pour obtenir : A: = G+H.

Dans l'exemple, les assignations suivantes :

C :=A+C+A*B ; et

C :=A+A*B+A*B*C+B*C,

se résument formellement en :

C :=A+A*B+A*B*[A+C+A*B]+B*[A+C+A*B]

c'est-à-dire

C : = A+A*B+B*C+A*B*C.

De même, la séquence d'assignations :

A : = 0 ; et

A : = A+B+B*C,

se résume en :

A :=[0]+B+B*C

c'est-à-dire

A :=B+B*C.

Le procédé permet donc d'obtenir pour le calcul de l'opération E, une séquence de 5K-4 assignations qui modifient successivement les registres de numéros :

K,K-1,...,2,1,2,...K-1,K,K-1,...,2,1,2,...,K-1,K,K-1,...2,1.

Le calcul de l'exemple s'exprime donc en 5K-4=11 étapes :

C :=A+B+C+A*B

B :=A+B+A*C

A :=A+B+B*C

B :=B+C+A*C

C :=A+C+A*B+B*C

B :=A*B

A :=B+B*C

B :=B+A*C

C :=1+A+B+C

B :=B+A*C

A :=A+B+C

Comme précédemment, toutes ces assignations peuvent être implémentées par des circuits électroniques avec des portes logiques et en particulier, des portes NAND uniquement comme c'est le cas actuellement dans les processeurs.

En notant (PQ) pour P NAND Q, le calcul de l'exemple par des portes NAND devient :

C :=(((1C)((1A)(1B)))((1B)(1(C(1A))))) ;

B :=((B(A(1C)))((1C)(1(A(1B))))) ;

A :=((A(B(1C)))((1C)(1(B(1A))))) ;

B :=((B(C(1A)))((1B)(1(C(1A))))) ;

C :=(((1B)(1(C(1A))))((1C)(1(A(1B))))) ;

B :=(1(AB)) ;

A :=(1(B(1C))) ;

B :=((B(AC))((1B)(1(AC)))) ;

C :=(((AB)((AC)(BC)))((1C)((AB)((1A)(1B))))) ;

B :=((B(AC))((1B)(1(AC)))) ; et

A :=(((BC)((B(1A))(C(1A))))((C(AB))((1C)(A(1B))))).

Le procédé de l'invention permet donc d'implémenter n'importe quelle opération sur K registres à l'aide 5K-4 assignations sur ces registres ou avantageusement, 2K-1 assignations si cette opération est bijective.

Le nombre d'opération est donc sensiblement réduit de sorte que le temps de calcul est limité et la consommation minimisée.

Par ailleurs, il est également possible d'optimiser le procédé général décrit précédemment afin de réduire la séquence d'assignations à 4K-3 assignations et éventuellement à 4K assignations, illustré en référence à la figure 5.

Dans ce mode de réalisation, le procédé débute par une étape d'ordonnancement 50 des ensembles de K-uplets ayant même image par E de façon à ce que, pour tout k, 0<k<K+1, et pour tout entier j positif ou nul, les ensembles consécutifs dont le numéro est compris entre j2^k et (j+1)2^k-1 aient un nombre total d'éléments multiple de 2^k.

Plus précisément, cette étape d'ordonnancement 50 débute par le calcul 52du nombre n de ensembles de K-uplets ayant même image par E.

Ensuite, l'étape 50 comporte une numération 54 des ensembles par une application N de S^K dans l'intervalle d'entiers [1..n]. La numération est suivie d'une mémorisation 56 des tailles des ensembles par une application T de [1..n] dans [1..2^K].

Dans l'exemple, l'étape d'ordonnancement 50 est mise en œuvre par l'algorithme suivant dans lequel les K-uplets sont considérés comme des entiers compris entre 0 et 2^K-1 :

a). Pour tous les K-uplets x, l'image N(x) est indéfinie.

Initialisation: n:=0

b). Si N est partout définie alors STOP.

c). Pour un x tel que N(x) est indéfinie faire :

n:=n+1; N(x):=n; T(n):=1;

Tant qu'il existe x' tel que N(x') indéfinie et E(x)=E(x') faire

N(x'):=n; T(n):=T(n)+1

On définit ensuite un algorithme qui permettra un regroupement par induction satisfaisant les propriétés voulues. Cet algorithme est mis en œuvre à partir de n qui est le nombre de ensembles et de T qui est la liste de tailles de ensembles. L'algorithme délivre une sortie m qui est le nombre de nouveaux ensembles et une sortie S qui est l'affectation des anciens ensembles dans les nouveaux.

Dans le cas présent, il s'agit d'un algorithme de l'ensemble [1..n] dans l'ensemble [1..m] avec M qui est la liste de tailles des nouveaux ensembles divisées par 2. Cet algorithme s'exprime de la manière suivante :

```
j:=1; parité:=faux;
Pour i variant de 1 à n faire
    si T[i] est impair alors faire
        si parite=faux alors faire
            M[i]:=j; S[j]:=T[i]; parite:=vrai;
        si parite=vrai alors faire
            M[i]:=j; S[j]:=(S[j]+T[i])/2; j:=j+1; parite:=faux;
Pour i variant de 1 à n faire
    si T[i] est pair alors faire
M[i]:=j; S[j]:=T[i]/2; j:=j+1;
m:=j-1;
```

Il est également possible de regrouper entre eux deux à deux les ensembles de cardinal pair. Toutefois, il peut rester à la fin un ensemble de cardinal pair isolé.

Ensuite, on associe à chaque K-uplet une liste (N_1,N_2,...,N_{l-1},N_l) de numéros de ensembles regroupés, en notant l le nombre de regroupements successifs effectués. Ceci se traduit par l'algorithme suivant :

```
T_1:=T; n_1:=n; N_1:=N;
i:=1;
```

tant que n>0 faire

Regroupement(Entrée: T_i,n_i; Sortie: M,T_{i+1},n_{i+1})

pour x variant de 0 à 2^K-1 faire N_{i+1}[x]:=M[N_i[x]];

i:=i+1;

l:=i-1;

On définit après un algorithme de comparaison colexicographique des l-uplets (N_1,N_2,...,N_{l-1},N_l) :

Fonction Inférieur (Entrée x,y: K-uplets): booléen

Sortie t: vrai si x<y pour l'ordre considéré ou si E(x)=E(y)

faux si x>y

i:=l

tant que t est indéfinie et i>0 faire

si N_i[x]<N_i[y] alors t=vrai

si N_i[x]>N_i[y] alors t=faux

si N_i[x]=N_i[y] alors i:=i-1

si i=0 alors t:=vrai

On définit enfin, lors d'une sous-étape 58, une bijection F et une application P. Ces bijection F et application P sont obtenues similairement au mode de réalisation précédent mais en respectant l'ordre de construction par l'algorithme suivant :

a). Pour tous les K-uplets x, les images F(x) et P(x) sont indéfinies.

Initialisation: R et T sont les K-uplets nuls : R=T=(0,0,0,...,0).

b). Si F est partout définie alors STOP.

c). Pour un x tel que F(x) est indéfinie

ET tel que pour tout y tel que F(y) est indéfinie on a Inférieur(x,y) faire :

Tant qu'il existe un x' tel que E(x')=E(x) et F(x') est indéfinie faire F(x')=T; P(T)=R; T=NEXT(T)

d). Faire R=NEXT(R) et aller en B.

Comme souhaité, l'ordre obtenu des ensembles de K-uplets ayant même image par E vérifie : pour tout k, 0<k<K+1, et pour tout entier j positif ou nul, les ensembles consécutifs dont le numéro est compris entre j2^k et

(j+1)2^k-1 ont un nombre total d'éléments multiple de 2^k.

A titre d'exemple, le procédé de l'invention va être appliqué à l'application E donnée par la table suivante :

E(000)=101
E(001)=000
E(010)=101
E(011)=001
E(100)=000
E(101)=101
E(110)=001
E(111)=000

En parcourant l'ensemble des K-uplets, par exemple, dans l'ordre naturel, le premier regroupement N=N_1 est le suivant :

N(000)=1
N(001)=2
N(010)=1
N(011)=3
N(100)=2
N(101)=1
N(110)=3
N(111)=2.

On en déduit le premier tableau de tailles :

T(1)=3
T(2)=3
T(3)=2.

Ce premier tableau est déjà ordonné avec les ensembles de tailles impaires d'abord. On poursuit alors avec le second regroupement :

N_2(000)=1
N_2(001)=1
N_2(010)=1
N_2(011)=2
N_2(100)=1
N_2(101)=1

N_2(110)=2

N_2(111)=1.

Ce second regroupement aboutit à l'ordre colexicographique des (N_1,N_2) suivant :

000,010,101<001,100,111<011,110.

Dans cet exemple on aboutit ainsi aux mêmes applications que précédemment. Ceci n'est toutefois pas le cas général.

1 : F(000)=000 P(000)=000

4 : F(001)=011 P(011)=001

2 : F(010)=001 P(001)=000

7 : F(011)=110 P(110)=010

5 : F(100)=100 P(100)=001

3 : F(101)=010 P(010)=000

8 : F(110)=111 P(111)=010

6 : F(111)=101 P(101)=001

Le procédé comporte ensuite une étape 60 de segmentation de la fonction F comme dans le mode de réalisation précédent. Toutefois, dans ce mode de réalisation les ensembles de K-uplets ayant même image par E sont envoyés sur des éléments consécutifs dans l'ordre calculé à l'étape 1. Par opposition, n'importe quel ordre pouvait être utilisé dans le mode de réalisation précédent. La première bijection est calculée ici aussi de droite à gauche.

Plus précisément, la bijection F est calculée par la méthode des bijections, exactement comme dans la méthode précédente, de la droite vers la gauche.

C'est à dire que l'on transforme tout K-uplet x en y=F(x) par une séquence d'opérations sur les registres

R_K :=f_K (R_1,...,R_K)

R_(K-1) :=f_(K-1) (R_1,...,R_K)

...

R_2 :=f_2 (R_1,...,R_K)

R_1 :=f_1 (R_1,...,R_K)

R_2 :=f'_2 (R_1,...,R_K)

...

R_(K-1) :=f'_(K-1) (R_1,...,R_K)

R_K :=f'_K (R_1,...,R_K)

Dans l'exemple on obtient encore :

C :=A+B+C+A*B

B :=A+B+A*C

A :=A+B+B*C

B :=B+C+A*C

C :=A+C+A*B

Le procédé se poursuit, comme dans le cas précédent par une étape 70 de segmentation de la fonction G. Toutefois, la seconde bijection est calculée cette fois de droite à gauche, comme la première, et dans le "calcul complet" on s'arrête au calcul en 5K-2 assignations, juste avant la dernière simplification.

Plus précisément, on définir ensuite une bijection G telle que pour tout K-uplet x on ait:

G(P(F(x)))=E(x)

Pour l'exemple on peut donc prendre de même :

G(000)=101 car les éléments du paquet numéro 000 ont pour image 101

G(001)=000 car les éléments du paquet numéro 001 ont pour image 000

G(010)=001 car les éléments du paquet numéro 010 ont pour image 001

G(011)=au choix par ex 011

G(100)=au choix par ex 100

G(101)=au choix par ex 010

G(110)=au choix par ex 110

G(111)=au choix par ex 111

Cette bijection est à nouveau calculée par la méthode des bijections, mais cette fois de la droite vers la gauche.

Il en résulte que l'on transforme tout K-uplet z en t=G(z) par une séquence d'opérations sur les registres. Cette séquence est exprimée ici :

R_K :=g_K (R_1,...,R_K)

R_(K-1) :=g_(K-1) (R_1,...,R_K)

```
...
R_2          :=g_2       (R_1,...,R_K)
R_1          :=g_1       (R_1,...,R_K)
R_2          :=g'_2 (R_1,...,R_K)
...
R_(K-1)  :=g'_(K-1)    (R_1,...,R_K)
R_K          :=g'_K(R_1,...,R_K)
```

Pour l'exemple on avait choisi

```
G(000)=101
G(001)=000
G(010)=001
G(011)=011
G(100)=100
G(101)=010
G(110)=110
G(111)=111
```

La méthode pour les bijections donne le calcul :

```
C    :=   A+C+A*B
B    :=   B
A    :=   1+A+B+C+B*C
B    :=   1+A+B+C+A*C
C    :=   1+B+C
```

Le procédé comporte ensuite une étape 80 de remplacement de la séquence des K premières assignations calculées à l'étape 70 appliquées aux paquets définissant une bijection G_1 en une séquence de K assignations appliquées au K-uplets permettant de programmer directement l'application composée de G_1 et P.

En comparaison avec le mode de réalisation précédent, on a ajouté une étape d'ordonnancement au débute et on a supprimé l'étape de calcul en K assignations de l'application P, que l'on a remplacé par une étape finale de calcul en K assignations de l'application composée de P et de la première séquence de K assignations.

Plus précisément, appelons G_1 la bijection définie par les K premières

assignations définissant G qui est composée de G_2 et G_1, c'est à dire que u=G_1(z) est défini par

R_K :=g_K (R_1,...,R_K)
R_(K-1) :=g_(K-1) (R_1,...,R_K)
...
R_2 :=g_2 (R_1,...,R_K)
R_1 :=g_1 (R_1,...,R_K)

où g_K,...,g_1 ont été calculées à l'étape précédente.

L'application qui à y associe u=G_1(P(y))=H(y) est alors définie directement par :

R_K :=h_K (R_1,...,R_K)
R_(K-1) :=h_(K-1) (R_1,...,R_K)
...
R_2 :=h_2 (R_1,...,R_K)
R_1 :=h_1 (R_1,...,R_K)

où les applications h_i sont définies par :

pour tout y=(y_1,..,y_i,...,y_K) avec P(y)=(Z_1,...,Z_i,...,Z_K) on a

h_i(y_1,..,y_(i-1),y_i,Z_(i+1),..,Z_K)=g_i(Z_1,...,Z_K)

Cette definition est partielle mais suffisante pour les besoins du procédé de l'invention.

Dans l'exemple, l'application G_1 est définie par les trois assignations :

C := A+C+A*B
B := B
A := 1+A+B+C+B*C

C'est à dire :

G_1(000)=100
G_1(001)=001
G_1(010)=010
G_1(011)=011
G_1(100)=101
G_1(101)=000
G_1(110)=110
G_1(111)=111.

D'où l'application H programmée par la modification successives des variables C, B, A respectivement par h_3, h_2, h_1 :

H(000)=100 c'est à dire h_3(000)=0, h_2(000)=0, h_1(000)=1
H(001)=100 c'est à dire h_3(001)=0, h_2(000)=0, h_1(000)=1
H(010)=100 c'est à dire h_3(010)=0, h_2(010)=0, h_1(000)=1
H(011)=001 c'est à dire h_3(011)=1, h_2(011)=0, h_1(001)=1
H(100)=001 c'est à dire h_3(100)=1, h_2(101)=0, h_1(101)=0
H(101)=001 c'est à dire h_3(101)=1, h_2(101)=0, h_1(101)=0
H(110)=010 c'est à dire h_3(110)=0, h_2(110)=1, h_1(110)=0
H(111)=010 c'est à dire h_3(111)=0, h_2(110)=1, h_1(110)=0

Ce qui peut s'exprimer algébriquement, par exemple, par les formules :

C :=A+A*B+B*C+A*B*C
B :=A*B
A :=1+A

Finalement, on a calculé l'application E comme composée de la bijection F en 2K-1 assignations, puis l'application G_1 composées avec P en K assignations, puis la bijection G_2 en K-1 assignations.

On obtient donc une méthode en 4K-2 étapes qui modifient successivement les registres de numéros :

K,K-1,...,2,1,2,...K-1,K,K,K-1,...,2,1,2,...,K-1,K

Pour l'exemple :

C :=A+B+C+A*B
B :=A+B+A*C
A :=A+B+B*C
B :=B+C+A*C
C :=A+C+A*B
C :=A+A*B+B*C+A*B*C
B :=A*B
A :=1+A
B :=1+A+B+C+A*C
C :=1+B+C

Comme précédemment, deux étapes successives sur un même registre K sont simplifiées en une seule opération lors d'une étape 90. On obtient donc

4K-3 opérations.

Dans l'exemple, les deux assignations :

C :=A+C+A*B

C :=A+A*B+B*C+A*B*C

peuvent être remplacées par l'assignation :

C :=A+A*B+B*C+A*B*C

On obtient donc finalement un procédé en 4K-3 étapes qui modifient successivement les registres de numéros :

K,K-1,...,2,1,2,...K-1,K,K-1,...,2,1,2,...,K-1,K

Dans l'exemple, l'application E de départ est programmée finalement par les 9 assignations :

C :=A+B+C+A*B

B :=A+B+A*C

A :=A+B+B*C

B :=B+C+A*C

C :=A+A*B+B*C+A*B*C

B :=A*B

A :=1+A

B :=1+A+B+C+A*C

C :=1+B+C

A titre complémentaire un autre exemple est décrit ici. On considère l'application E sur 3 bits définie par :

E(000)=101

E(001)=001

E(010=101

E(011)=000

E(100)=110

E(101)=101

E(110)=000

E(111)=110

Le premier regroupement N=N_1 donne, par exemple :

N(000)=3

N(001)=1

N(010=3

N(011)=2

N(100)=4

N(101)=3

N(110)=2

N(111)=4

Les tailles sont :

T(1)=1

T(2)=2

T(3)=3

T(4)=2

Elles doivent être réordonnées pour avoir un ordre de tailles 1,3,2,2 (par exemple)

d'où un second regroupement

N_2(1)=1

N_2(2)=2

N_2(3)=1

N_2(4)=3

et les tailles associées :

T_2(1)=2

T_2(2)=1

T_2(3)=1

Le troisième regroupement prévu par l'algorithme est inutile ici puisque les conditions voulues sont déjà satisfaites, c'est-à-dire que les deux paires consécutives ont un cardinal multiple de 4. Il est donc possible de s'arrêter à l=2.

Les K-uplets sont donc associés aux l-uplets suivants :

000 à (3,1)

001 à (1,1)

010 à (3,1)

011 à (2,2)

100 à (4,3)

101 à (3,1)

110 à (2,2)

111 à (4,3)

Ceci donne dans l'ordre colexicographique :

001<000,010,101<011,110<100,111

On obtient donc les applications F et et P suivante, en numérotant dans l'ordre de construction :

2: F(000)=001 P(001)=001

1: F(001)=000 P(000)=000

3: F(010)=010 P(010)=001

5: F(011)=100 P(100)=010

7: F(100)=110 P(110)=011

4: F(101)=011 P(011)=001

6: F(110)=101 P(101)=010

8: F(111)=111 P(111)=011

Ensuite, la bijection F s'obtient par la méthode de décomposition d'une bijection comme dans les modes de réalisation précédents. Pour simplifier la notation on l'écrit comme deux bijections successives F_1 puis F_2 composée de F_2 et F_1 et obtenues respectivement par les K premières assignations et les K-1 suivantes :

F_1(000)=000 F_2(000)=001

F_1(001)=001 F_2(001)=000

F_1(010)=010 F_2(010)=010

F_1(011)=111 F_2(111)=100

F_1(100)=100 F_2(100)=110

F_1(101)=011 F_2(011)=011

F_1(110)=110 F_2(110)=101

F_1(111)=101 F_2(101)=111

Par la suite, pour la bijection G on peut choisir par exemple :

G(000)=001

G(001)=101

G(010=000

G(011)=110

G(100)=010

G(101)=011
G(110)=100
G(111)=111

Cette bijection G se calcule par la méthode des bijections, comme deux bijections successives G_1 puis G_2 obtenues respectivement par les K premières assignations et les K-1 suivantes :

G_1(000)=000 G_2(000)=001
G_1(001)=101 G_2(101)=101
G_1(010)=011 G_2(011)=000
G_1(011)=110 G_2(110)=110
G_1(100)=010 G_2(010)=010
G_1(101)=001 G_2(001)=011
G_1(110)=100 G_2(100)=100
G_1(111)=111 G_2(111)=111

L'application H obtenue par P puis G_1 se calcule directement par K assignations des registres de K à 1:

H(000)=000 c'est à dire h_3(000)=0, h_2(000)=0, h_1(000)=0
H(001)=101 c'est à dire h_3(001)=1, h_2(001)=0, h_1(001)=1
H(010)=101 c'est à dire h_3(010)=1, h_2(011)=0, h_1(001)=1
H(011)=101 c'est à dire h_3(011)=1, h_2(011)=0, h_1(001)=1
H(100)=011 c'est à dire h_3(100)=1, h_2(101)=1, h_1(111)=0
H(101)=011 c'est à dire h_3(101)=1, h_2(101)=1, h_1(111)=0
H(110)=110 c'est à dire h_3(110)=0, h_2(110)=1, h_1(110)=1
H(111)=110 c'est à dire h_3(111)=0, h_2(110)=1, h_1(110)=1

Finalement l'application E obtenue par F_1 puis F_2 puis H puis G_2 est obtenue comme une séquence de 4K-2 assignations, et le registre K étant modifié deux fois consécutives au milieu de la table on peut simplifier en 4K-3 étapes.

Il est également possible de réduire le nombre d'assignations à 4K lorsque, avec S={0,1....M-1} on a 2^(k-1)<M<2^k+1. Dans ce cas, il est possible de regrouper k bits successifs. En effet, les trois simplifications effectuées lorsqu'un bit est modifié deux fois d'affilée sont supprimées car c'est tout le bloc de k bits qui doit être modifié. Il en résulte un gain de trois opérations

aboutissant à 4k assignations.

Bien entendu, divers modes de réalisation sont envisageables. Notamment, la segmentation et l'implémentation peuvent être faits en temps réels dans un programme exécuté par un ordinateur ou un micro composant afin d'obtenir la séquence d'assignations.

Le procédé peut également être utilisé pour créer des tables de référence permettant d'exprimer des fonctions sous la forme d'une séquence d'assignation. Un composant vient ensuite consulter ces tables lors de l'utilisation des fonctions.

Ainsi, l'invention peut être mise en œuvre par un compilateur ou encore lors de la conception de microprocesseurs.

Les principes de l'invention peuvent être aussi appliqués à des ensembles de valeurs importants en combinant les suites d'assignation construites pour agir sur des bits successifs en les regroupant pour déterminer une assignation unique pour tous les bits du registre. Ainsi, le calcul est construit en agissant bit à bit puis, est regroupé par bloc d'assignation de bits en assignation de registres.

Par exemple, la fonction d'échange du contenu de deux registres binaires A et B sur N bits peut être réalisée bit à bit et on obtient la séquence de calcul suivante sur les registres:

1 le registre A prend la valeur A XOR B ;

2 le registre B prend la valeur A XOR B ;

3 le registre A prend la valeur A XOR B ;

où XOR est l'addition binaire bit à bit. A l'issue de ces trois étapes, les valeurs des registres A et B sont échangées.

REVENDICATIONS

1. Procédé d'implémentation d'une opération (E) sur un nombre quelconque K, strictement supérieur à 1, de mémoires élémentaires (A, B, C) pouvant chacune prendre M valeurs, caractérisé en ce qu'il comprend les étapes suivantes :
   - au moins une segmentation (10 ; 40, 44 ; 60, 70) de l'opération en une série de bijections ;
   - une combinaison (16 ; 46 ; 90) desdites bijections en fonction des mémoires élémentaires qu'elles affectent ;
   - une détermination (18, 48) d'une séquence d'assignation des mémoires élémentaires à partir desdites bijections combinées pour effectuer le calcul de ladite opération (E).

2. Procédé selon la revendication 1, caractérisé en ce que ladite opération est bijective et en ce que ladite étape de segmentation comprend une segmentation (12, 14) en M fonctions bijectives portant chacune sur K-1 mémoires.

3. Procédé selon la revendication 2, caractérisé en ce que ladite étape de segmentation de l'opération en bijections est renouvelée de manière itérative jusqu'à ce que les bijections obtenues soient directement calculables par des assignations des mémoires élémentaires sur lesquelles elles portent.

4. Procédé selon la revendication 3, caractérisé en ce que ladite étape de segmentation comprend au moins une coloration en M couleurs à chaque itération.

5. Procédé selon l'une quelconque des revendications 2 à 4, caractérisé en ce que ladite étape de détermination d'une série d'assignations délivre une séquence de 2K-1 assignations réalisées sur les K mémoires élémentaires.

6. Procédé selon la revendication 1, caractérisé en ce qu'il comprend une étape (30) préalable de décomposition de l'opération en en deux bijections (F et G), et une application (P) portant chacune sur K variables et en ce que ladite étape de segmentation comprend une segmentation (40, 44) de chacune de ces bijections (F, G) en M fonctions bijectives.

7. Procédé selon la revendication 6, caractérisé en ce que ladite étape préalable (30) de décomposition comporte la détermination (32) d'une première bijection (F) à partir de l'opération à transformer afin d'attribuer une image différente aux antécédents ayant la même image par l'opération à transformer, la détermination (34) d'une séquence de K assignation de calcul d'une application (P) qui identifie par paquets les images obtenues par la première bijection, et la détermination (36) d'une deuxième bijection (G) transformant les produits obtenus par ladite première bijection et ladite application pour former les termes de l'opération à calculer.

8. Procédé selon l'une quelconque des revendications 6 et 7, caractérisé en ce que lesdites deux bijections sont calculées par des opérations réalisées sur les mémoires élémentaires dans l'ordre décroissant puis croissant pour l'obtention de ladite première bijection et de manière symétrique dans l'ordre croissant puis décroissant pour l'obtention de ladite deuxième bijection.

9. Procédé selon l'une quelconque des revendications 6 à 8, caractérisé en ce que ladite étape de détermination d'une série d'assignations délivre une séquence de 5K-4 assignations réalisées sur les mémoires élémentaires.

10. Procédé selon la revendication 1, caractérisé en ce qu'il comporte une étape préalable (50) d'ordonnancement des antécédents ayant une même image par ladite opération (E), et une étape de décomposition de l'opération en en deux bijections (F et G), et une application (P) portant chacune sur K variables et en ce que ladite étape de segmentation comprend une segmentation (60, 70) de chacune de ces bijections (F, G) en M fonctions bijectives.

11. Procédé selon la revendication 6, caractérisé en ce que ladite étape préalable (50) d'ordonnancement comporte le calcul (52) du nombre d'ensembles d'antécédents ayant une même image, la numération (54) de ces ensembles et leur mémorisation (56).

12. Procédé selon l'une quelconque des revendications 10 et 11, caractérisé en ce que lesdites deux bijections sont calculées par des opérations réalisées sur les mémoires élémentaires dans l'ordre décroissant puis croissant pour l'obtention de ladite première bijection et de manière symétrique dans l'ordre croissant puis décroissant pour l'obtention de ladite deuxième bijection.

13. Procédé selon l'une quelconque des revendications 10 à 12, caractérisé en ce que ladite étape de détermination d'une série d'assignations délivre une séquence de 4K-3 assignations réalisées sur les mémoires élémentaires.

14. Procédé selon l'une quelconque des revendications 10 à 13, caractérisé en ce que les paramètres K et M sont tels que 2^(k-1)<M<2^k+1 et en ce que ladite étape de détermination d'une série d'assignation délivre une séquence de 4Kassignations réalisées sur les mémoires élémentaires

15. procédé selon l'une quelconque des revendications précédentes, caractérisé en ce que ladite étape de combinaison comprend le regroupement et la simplification (18 ; 48) des assignations portant sur les mêmes mémoires élémentaires.

16. Programme d'ordinateur comportant des instructions de codes instructions pour mettre en œuvre les étapes d'un procédé selon l'une quelconque des revendications précédentes, lors d'une exécution par un calculateur dudit ordinateur.

17. Processeur d'ordinateur comportant, dans une mémoire, un programme selon la revendication 16.

18. Compilateur de programmes d'ordinateur comportant des moyens de mise en œuvre du procédé selon l'une quelconque des revendications 1 à 15 pour l'implémentation d'un moins une opération.

5

19. Procédé de conception d'un microprocesseur comportant au moins une étape d'implémentation d'une opération selon le procédé selon l'une quelconque des revendications 1 à 15.

FIG. 1

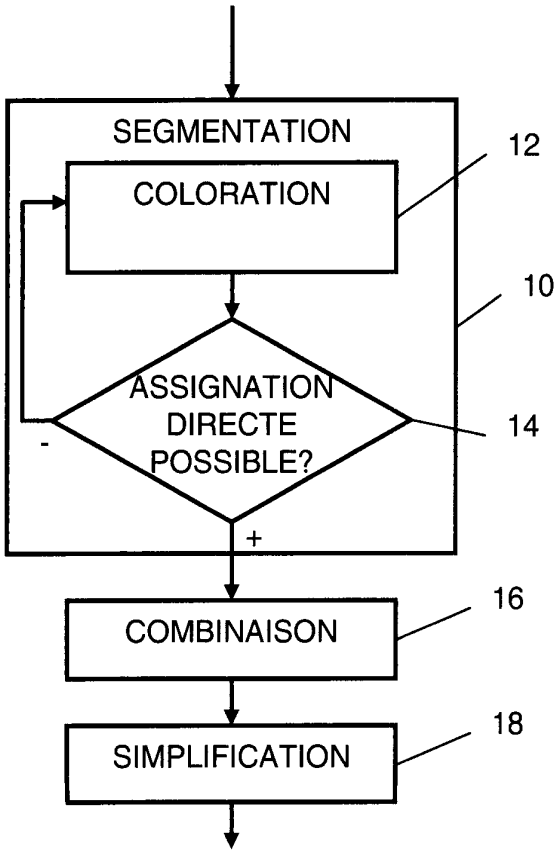

SEGMENTATION
COLORATION
12
10
ASSIGNATION DIRECTE POSSIBLE?
14
-
+
COMBINAISON
16
SIMPLIFICATION
18

FIG. 2

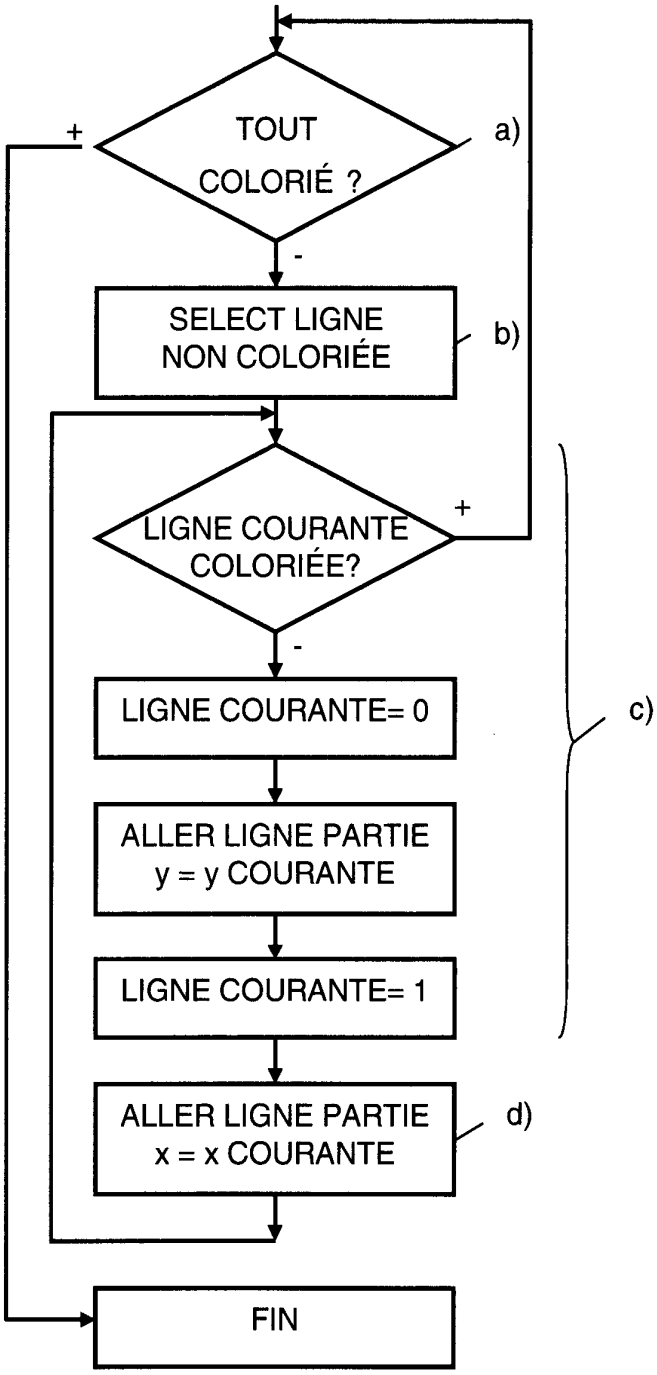

TOUT COLORIÉ ?
a)
+
-
SELECT LIGNE NON COLORIÉE
b)
LIGNE COURANTE COLORIÉE?
+
-
LIGNE COURANTE= 0
ALLER LIGNE PARTIE y = y COURANTE
LIGNE COURANTE= 1
c)
ALLER LIGNE PARTIE x = x COURANTE
d)
FIN

FIG. 3

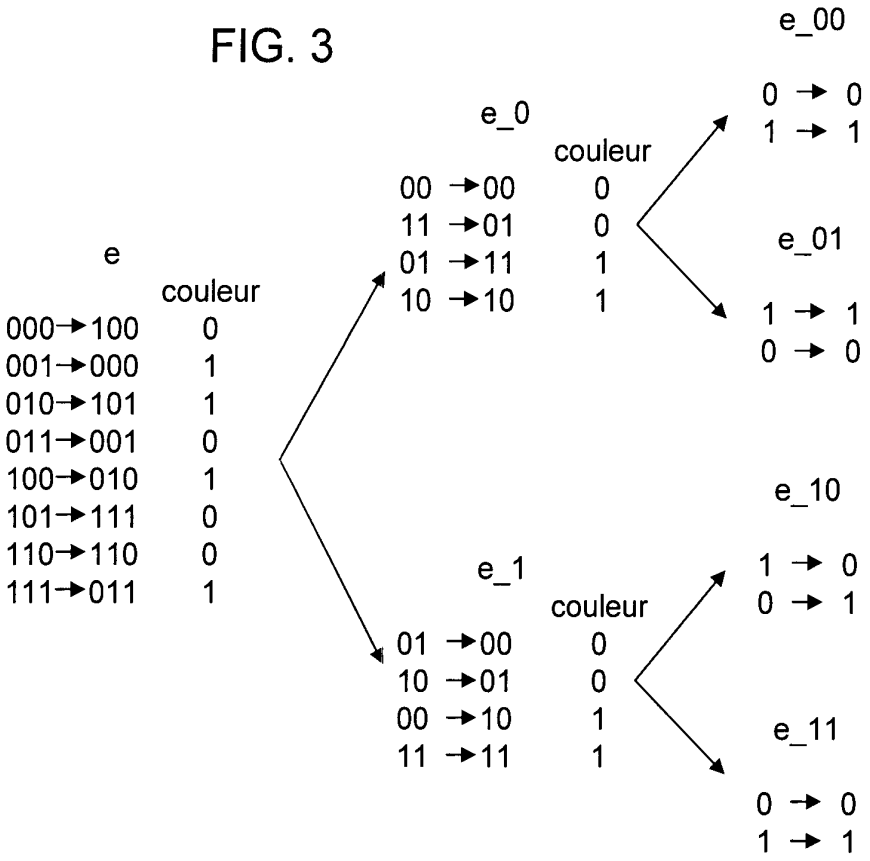

e
couleur
000→100 0
001→000 1
010→101 1
011→001 0
100→010 1
101→111 0
110→110 0
111→011 1
e_0
couleur
00 →00 0
11 →01 0
01 →11 1
10 →10 1
e_1
couleur
01 →00 0
10 →01 0
00 →10 1
11 →11 1
e_00
0 → 0
1 → 1
e_01
1 → 1
0 → 0
e_10
1 → 0
0 → 1
e_11
0 → 0
1 → 1

FIG. 4

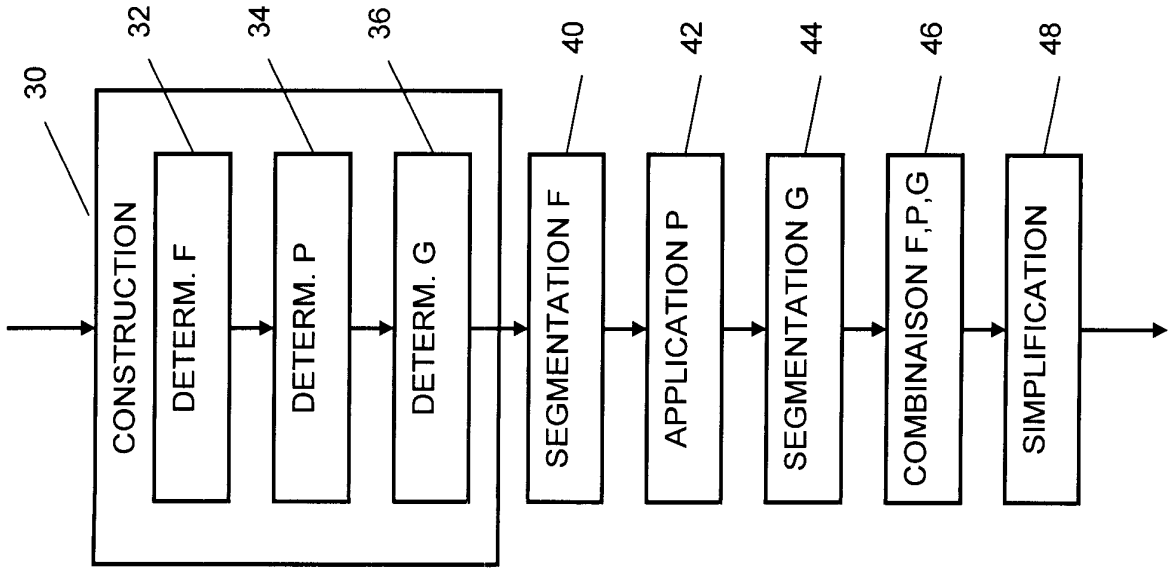
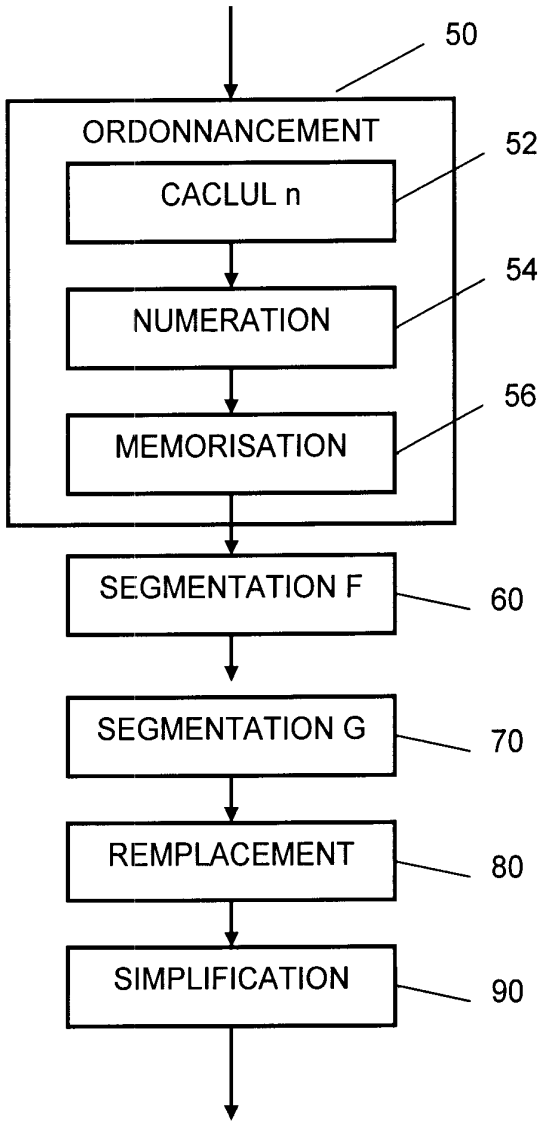

30
32
34
36
40
42
44
46
48
CONSTRUCTION
DETERM. F
DETERM. P
DETERM. G
SEGMENTATION F
APPLICATION P
SEGMENTATION G
COMBINAISON F,P,G
SIMPLIFICATION

FIG. 5


50
ORDONNANCEMENT
CACLUL n
52
NUMERATION
54
MEMORISATION
56
SEGMENTATION F
60
SEGMENTATION G
70
REMPLACEMENT
80
SIMPLIFICATION
90

RÉPUBLIQUE FRANÇAISE

2919081

INSTITUT NATIONAL DE LA PROPRIÉTÉ INDUSTRIELLE

RAPPORT DE RECHERCHE PRÉLIMINAIRE

établi sur la base des dernières revendications déposées avant le commencement de la recherche

N° d'enregistrement national

FA 697319
FR 0705152

DOCUMENTS CONSIDÉRÉS COMME PERTINENTS		Revendication(s) concernée(s)	Classement attribué à l'invention par l'INPI
Catégorie	Citation du document avec indication, en cas de besoin, des parties pertinentes		
X	US 3 398 402 A (SERGE DELAIGUE ET AL) 20 août 1968 (1968-08-20) * abrégé * * figure 3 * -----	1-19	G06F7/38
X	US 5 812 845 A (MACHIDA HIROHISA [JP]) 22 septembre 1998 (1998-09-22) * colonne 3, ligne 66 - colonne 4, ligne 13; figures 2-4 * * colonne 11, ligne 14 - ligne 20 * * colonne 14, ligne 36 - ligne 48 * -----	1-19	
			DOMAINES TECHNIQUES RECHERCHÉS (IPC)
			G06F

Date d'achèvement de la recherche	Examinateur
17 mars 2008	Verhoof, Paul

CATÉGORIE DES DOCUMENTS CITÉS

X : particulièrement pertinent à lui seul
Y : particulièrement pertinent en combinaison avec un autre document de la même catégorie
A : arrière-plan technologique
O : divulgation non-écrite
P : document intercalaire

T : théorie ou principe à la base de l'invention
E : document de brevet bénéficiant d'une date antérieure à la date de dépôt et qui n'a été publié qu'à cette date de dépôt ou qu'à une date postérieure.
D : cité dans la demande
L : cité pour d'autres raisons

.....

& : membre de la même famille, document correspondant

EPO FORM 1503 12.99 (P04C14)

ANNEXE AU RAPPORT DE RECHERCHE PRÉLIMINAIRE RELATIF A LA DEMANDE DE BREVET FRANÇAIS NO. FR 0705152 FA 697319

La présente annexe indique les membres de la famille de brevets relatifs aux documents brevets cités dans le rapport de recherche préliminaire visé ci-dessus.
Les dits membres sont contenus au fichier informatique de l'Office européen des brevets à la date du 17-03-2008
Les renseignements fournis sont donnés à titre indicatif et n'engagent pas la responsabilité de l'Office européen des brevets, ni de l'Administration française

Document brevet cité au rapport de recherche	Date de publication	Membre(s) de la famille de brevet(s)	Date de publication
US 3398402 A	20-08-1968	BE 670446 A	04-04-1966
		CH 443733 A	15-09-1967
		DE 1268672 B	22-05-1968
		FR 1421389 A	17-12-1965
		GB 1127551 A	18-09-1968
		NL 6512866 A	04-04-1966
US 5812845 A	22-09-1998	JP 9305401 A	28-11-1997

EPO FORM P0465