(19) RÉPUBLIQUE FRANÇAISE

INSTITUT NATIONAL
DE LA PROPRIÉTÉ INDUSTRIELLE

PARIS

(11) **N° de publication :** **2 884 005**

(à n'utiliser que pour les commandes de reproduction)

(21) **N° d'enregistrement national :** **05 03178**

(51) Int Cl[8] : **G 06 F 17/10** (2006.01), G 06 F 7/38

(12) DEMANDE DE BREVET D'INVENTION A1

(22) **Date de dépôt :** 01.04.05.

(30) **Priorité :**

(43) **Date de mise à la disposition du public de la demande :** 06.10.06 Bulletin 06/40.

(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) :** *THALES Société anonyme* — FR.

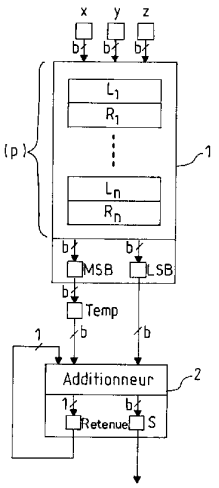
(72) **Inventeur(s) :** BERNARD FLORENT, SAUZET ALAIN et GARRIDO ERIC.

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

(74) **Mandataire(s) :** MARKS & CLERK FRANCE.

(54) METHODE D'IMPLEMENTATION DE LA MULTIPLICATION MODULAIRE DE MONTGOMERY ET SON DISPOSITIF.

(57) Dispositif de mise en oeuvre de la multiplication modulaire caractérisé en ce qu'il comporte au moins une cellule de calcul comportant un multiplieur-additionneur comportant p couples logique-registre pipeliné, recevant plusieurs chiffres à additionner et à multiplier, au moins deux sorties correspondant au poids faible et au poids fort, un additionneur recevant les deux sorties du multiplieur-additionneur, le nombre p étant choisi de façon telle que la fréquence maximale du multiplieur-additionneur soit supérieure ou égale à la fréquence maximale de l'additionneur.



FR 2 884 005 - A1

Méthode d'implémentation de la multiplication modulaire de Montgomery et son dispositif

L'invention concerne une méthode et un dispositif pour exécuter une multiplication modulaire de Montgomery.

Elle est notamment utilisée dans le domaine de la cryptologie, dans des circuits intégrés de type FPGA (Field Programmable Gate Array) ou ASIC (Application Specific Integrated Circuit). La multiplication modulaire est traitée sur un anneau modulaire (Z/NZ) ou sur un corps (Z/pZ) avec p nombre premier.

Dans la description, on définit la fréquence de fonctionnement d'un multiplieur-additionneur pipeliné à p étages donnée par l'inverse du temps de traversée d'un seuil des étages du multiplieur-additionneur.

Dans la description, on définit le terme « pipeline » comme une opération consistant à rajouter des barrières de registres entre les phases logiques et l'architecture en résultant.

La multiplication modulaire est au cœur de nombreux crypto systèmes à clé publique (RSA, ECDSA).

Les opérations de chiffrement et de signature présentent des contraintes différentes qui requièrent notamment une architecture rapide, flexible et peu coûteuse en ressources matérielles.

Deux niveaux de flexibilité sont ainsi définis :

- Le premier niveau se situe avant la synthèse de l'architecture. La conception de l'architecture doit comporter suffisamment de degrés de liberté pour pouvoir satisfaire des contraintes matérielles/temporelles données.
- Le deuxième niveau se situe après la synthèse de l'architecture.

L'architecture doit être capable de traiter différentes tailles de données afin de pouvoir effectuer aussi bien des opérations de type ECDSA

(Elliptique Curve Digital Signature Algorithm) taille 256-512 bits que des opérations de type RSA (du nom de ses inventeurs : Rivest, Shamir, Adleman) taille 1024-4096 bits.

Pour évaluer les performances d'une architecture, le produit temps*surface est utilisé. Le temps est donné en nombre de cycle horloge. La surface est estimée en terme de nombre de portes Nand équivalentes (ASIC) ou en terme de cellules élémentaires (LC (Logic Cell) ou CLB (Configuration Logic Bloc), blocs combinatoires séquentiels, pour FPGA ou field-programmable gate arrays).

L'art antérieur décrit différents exemples de dispositif et de méthode utilisant la multiplication modulaire en cryptographie asymétrique.

Les implémentations les plus fréquentes de la multiplication modulaire sont de type systolique et utilisent l'algorithme de Montgomery. De nombreuses publications traitent de ces implémentations. La référence dans ce domaine étant due à Peter Kornerup, A systolic, linear-array for a class of right-shift algorithms, IEEE Transactions on Computers 43 (1994), no. 8, 892-898.

L'implémentation de la multiplication modulaire peut aussi se faire selon l'algorithme de Barret , Johann Groβschädl, High speed RSA Hardware based on Barret's modular reduction method, CHES'00, 2000.

Ce dernier type d'implémentation, bien que performant, présente l'inconvénient de ne pas être flexible, car elle est définie sur une taille donnée et ne peut traiter des opérations de tailles supérieures.

Des représentations de nombre « exotiques » (c'est-à-dire non conventionnelles, comme par exemple la représentation RNS (Residue Number System)) sont également utilisées pour paralléliser l'implémentation de la multiplication modulaire, par exemple, L-S Didier, J-C Bajard and P. Kornerup, An RNS Montgomery multiplication Algorithm, Proceedings of ARITH13, IEEE Computer Society, 1997, pp. 234-239. Cette solution requiert toutefois d'importantes ressources matérielles et ne résout pas le problème de flexibilité.

Il est aussi connu des dispositifs présentant des architectures flexibles, ou « scalable hardware ». La première publication d'une telle architecture est due à Alexandre F. Tenca and Cetin K. Koç, A scalable architecture for Montgomery multiplication, CHES'99, 1999.

D'autres publications plus récentes, reprennent l'idée développée dans cet article et proposent des améliorations et des extensions. C'est le cas de :

- Alexandre F. Tenca and Cetin K. Koç, A scalable architecture for Modular Multiplication based on Montgomery's Algorithm, IEEE Transactions on computers 52 (2003), no 9.
- Alexandre F. Tenca, Cetin K. Koç and Erkay Savas, A scalable and unified multiplier architecture for finite field GF(p) and GF(2^n), CHES'00, 2000.
- Alexandre F. Tenca, Georgi Todorov and Cetin K. Koç, High-Radix Design of a scalable modular multiplier, CHES'01, vol. LNCS 2162, 2001, pp. 185-201.
- Gunnar Gaubatz, Versatile Montgomery Multiplier Architectures, Ph.D. thesis, Worcester Polytechnic Institute, April 2002.

Ces architectures sont basées sur le cablage de petites cellules élémentaires composées d'un additionneur et d'un multiplieur. Ces dispositifs répondent au problème de flexibilité. Toutefois, ces structures impliquent le choix d'une taille de bus interne faible. Le nombre de cycles pour réaliser une multiplication modulaire est donc élevé.

Avant d'expliciter le dispositif et les étapes de la méthode selon l'invention quelques rappels concernant l'algorithme de Montgomery sont donnés.

L'algorithme de Montgomery utilisé est le suivant :

$MM_nS(A,B,N)$

$S \leftarrow a_0B$

Pour i de 1 à n faire

$m_i \leftarrow S_0N' \bmod r$

$S \leftarrow a_iB+(m_iN+S)/r$

$m_{n+1} \leftarrow S_0N' \bmod r$

$S \leftarrow (m_{n+1}N+S)/r$

Retourne S

L'étape $S \leftarrow a_iB+(m_iN+S)/r$ est la plus coûteuse d'un point de vu temps de calcul. On peut remarquer que cette étape est obtenue en faisant deux fois la même opération : chiffre×entier long+entier long → entier long ($x_iY+Z \rightarrow Z$).

Cette étape est réalisée en faisant une boucle sur les chiffres des deux entiers longs (Y et Z).

On distingue cependant deux cas, selon que le décalage est effectué ou non.

La multiplication classique : $a_iB+T \rightarrow S$ (1)

Cette multiplication-addition est réalisée par la boucle suivante :

c=0

Pour j allant de 0 à n faire :

$P \leftarrow a_ib_j+t_j+c$

$s_j \leftarrow LSB(P)$

$c \leftarrow MSB(P)$

$s_{n+1}=c$

Pour l'itération j=n, $b_j \leq 1$, donc $P \leq 3(r-1) = 2r+r-3$, ce qui implique $MSB(P) \leq 2$. On en déduit que $s_{n+1} \leq 2$.

Remarque :

L'opération de base est x×y+z+c avec x, y, z et c compris entre 0 et r-1. Le résultat est donc compris entre 0 et r^2-1, représentable par deux chiffres.

La multiplication et décalage : $(m_iN+S/r) \rightarrow T$ (2)

Dans le cas particulier de la division par r (décalage à droite), la boucle est la même que ci-dessus, avec un décalage d'indice au niveau de la sortie T.

En effet, la détermination de m_i rend la quantité $P=m_i \times N_0+s_0$ divisible par r. Dans ce cas là, on est assuré que LSB(P) est nul. On ``perd'' donc volontairement cette première valeur (indice -1) la première valeur significative (t_0) étant obtenue à l'itération suivante (j=1).

Du fait du décalage d'indice, la dernière itération (j=n) donnera accès à la valeur t_{n-1}. La valeur t_n sera donc obtenue en ajoutant c=MSB(P) de l'itération j=n avec s_{n+1}.

La boucle est donc la suivante :

c=0

Pour j allant de 0 à n faire :

$P \leftarrow m_i N_j+s_j+c$

$t_{j-1} \leftarrow LSB(P)$

$c \leftarrow MSB(P)$

$t_n=c+s_{n+1}$

Pour l'itération j=n, $N_j=0$, donc $P \leq 2(r-1) = r+r-2$, ce qui implique $MSB(P) \leq 1$. On en déduit que $t_n \leq 3$.

Comme $r \geq 4$, on en déduit que T est bien représenté par n+1 chiffres.

La présente invention repose sur une nouvelle approche qui considère notamment, en fonctionnement normal, une cellule unique de taille plus importante que les cellules considérées dans l'art antérieur et un processus d'itération sur elle-même en utilisant un algorithme de Montgomery en base élevée. C'est-à-dire que dans la suite, la base r est supposée plus grande ou égale à 4 (condition de validité de l'algorithme). En pratique la base r correspond le plus souvent à 2^{32} ou à 2^{64}, ce qui justifie la dénomination de base élevée.

L'invention concerne un dispositif de mise en œuvre de la multiplication modulaire caractérisé en ce qu'il comporte au moins une cellule de calcul comportant un multiplieur-additionneur comportant p couples

logique-registre pipeliné, recevant plusieurs chiffres à additionner et à multiplier, au moins deux sorties correspondant au poids faible et au poids fort, un additionneur recevant les deux sorties du multiplieur-additionneur, le nombre p étant choisi de façon telle que la fréquence maximale du multiplieur-additionneur soit supérieure ou égale à la fréquence maximale de l'additionneur.

Le dispositif peut comporter un multiplieur partie basse dont les entrées correspondent à la sortie LSB de l'additionneur et à une constante précalculée, ledit multiplieur étant pipeliné sur une profondeur k.

L'invention présente notamment les avantages suivants :
L'utilisation d'une base plus élevée réduit le nombre de cycles nécessaires au calcul d'une multiplication modulaire. Une seule cellule élémentaire est alors utilisée. Le produit temps surface de cette architecture est donc amélioré.

D'autres caractéristiques et avantages de la présente invention apparaîtront mieux à la lecture de la description qui suit d'un exemple de réalisation donné à titre illustratif et nullement limitatif annexé des figures qui représentent :

- La figure 1 un exemple d'agencement des composants élémentaires d'une cellule multiplieur-additionneur selon l'invention,
- La figure 2 un exemple d'agencement des composants constituant le cœur du multiplieur modulaire de Montgomery.

L'idée mise en œuvre dans le multiplieur selon l'invention consiste notamment à scinder l'opération $x \times y+z+c$ en $(x \times y+z)+c$, en pipelinant l'opération de multiplication-addition $(x \times y+z)$ et en ajoutant, à mesure que les résultats sont obtenus, la variable c, où x,y et z sont les données, c étant le chiffre de poids fort provenant de l'itération précédente.

Pipeline

Le pipeline consiste à rajouter des barrières de registres entre les phases logiques afin de réduire le chemin critique, et ainsi d'augmenter la fréquence maximale de fonctionnement (théoriquement celle d'un additionneur en base r).

On définit la profondeur de pipeline d'un composant élémentaire par son nombre de registres internes. On ne compte pas le registre de sortie.

Implémentation de la multiplication-addition

L'exemple donné à la figure 1 suppose que l'on dispose d'un multiplieur-additionneur 1 pipeliné, de profondeur p.

Il comporte notamment un ensemble de couples logique-registre (li, ri). Le nombre p de ces couples est notamment choisi pour que la fréquence maximale F1max du multiplieur-additionneur pipeliné soit supérieure ou égale à la fréquence maximale F2max de l'additionneur, les valeurs de ces deux fréquences étant les plus proches possibles.

La fréquence maximale de fonctionnement du mulplieur-additionneur est donnée par l'inverse du temps d'exécution de l'opération de multiplication-addition, alors que la fréquence maximale de fonctionnement du multiplieur-additionneur pipeliné est donné par l'inverse du temps d'exécution d'un seul des p étages. Pour un fonctionnement optimal, on détermine la fréquence maximale de l'additionneur, ce qui donne le temps d'exécution de l'additionneur et on découpe le multiplieur-additionneur en p étages de temps de traversée inférieur ou égal, mais le plus proche possible, au temps d'exécution de l'additionneur.

Les entrées du multiplieur-additionneur 1 correspondent à trois chiffres : x,y et z et la sortie est un couple de chiffres correspondant à LSB(x×y+z) et MSB(x×y+z). La sortie tient sur deux chiffres.

Les résultats du multiplieur-additionneur sont transmis à un additionneur trois entrées, référencé 2 : chiffre+chiffre+carry → chiffre+carry, fonctionnant en 1 cycle (pipeline 0) à la fréquence F2max.

Dans la suite on désignera par b la taille du bus interne (l'hypothèse qui est faite est : $b \geq 2$ pour satisfaire $r \geq 4$), ($r=2^b$).

Le registre Temp correspond au stockage de c nécessaire au calcul suivant : addition de c avec le LSB suivant et la retenue précédente.

Les données (chiffres de $x \times Y + Z$) sortent donc en série à chaque cycle, chiffre de poids faible en tête, dans le même sens que la propagation de la retenue.

La figure 2 décrit un exemple d'agencement du composant de la figure 1, avec un dispositif 3 adapté à la partie basse de la multiplication, et un dispositif 5 composé de registres et de multiplexeurs.

Dans cet exemple, les principaux composants du circuit sont : un multiplieur-additionneur 1 pipeliné, un additionneur 3 entrées, référencé 2, un multiplieur partie basse 3 et un additionneur 2 bits + 1 bit vers 2 bits, désigné 4. Une barrière de $n_{reg\text{-}max}$-1 registres et de multiplexeurs, désigné 5.

Le nombre de multiplexeurs et de registres dépend notamment des données intrinsèques au circuit, les profondeurs de pipeline p et k, ainsi que du nombre de chiffres des données.

Dans l'exemple de la figure 2, la barrière de $n_{reg\text{-}max}$-1 registres et de multiplexeurs (composant référencé 5) a notamment pour fonction de retarder l'arrivée des données dans le multiplieur-additionneur 1.

Multiplication partie basse – composant 3

Dans la boucle principale de l'algorithme, à chaque itération, on détermine m_i, le chiffre qui rend la quantité $S+m_iN$ divisible par r.

m_i est déterminé en effectuant une multiplication partielle du chiffre de poids faible de S avec une constante N', précalculée une fois pour toute pour un module N donné. Dans cette multiplication, seule la partie basse nous intéresse : on effectue cette opération modulo r.

Cette opération étant plus lente qu'une addition, elle est également pipeliné. On appelle k la profondeur de pipeline de cet opérateur et on suppose $k<p$.

Rebouclage, latence et registres supplémentaires

Deux cas sont distingués, le passage de la multiplication classique $a_iB+T \rightarrow S$, (1), à la multiplication et décalage $(m_i \times N+S)/r \rightarrow T$, (2), et le passage de (2) à (1), le retard n'étant pas le même.

Ce retard détermine notamment le nombre de registres à utiliser : n'_{reg} ou n_{regk} selon les cas. Ainsi par exemple, dans le passage de (2) à (1), et dans le cas où $p+2 \leq n$ (cas où n'_{reg} est défini), le nombre de registres à utiliser est n'_{reg}. Comme il y en a $n_{reg\text{-}max}$ -1, il faut en sauter $n_{reg\text{-}max}$ -1- n'_{reg}. Ceci est fait grâce aux multiplexeurs, placés en conséquence.

Passage de (1) à (2)

Pour pouvoir chaîner sans perte de temps (ie : sans rajouter de latence) les multiplications-additions (1) et (2), il convient de déterminer m_i avant d'avoir parcouru la totalité des chiffres de la multiplication en cours.

Il est donc souhaitable de réaliser la condition : $p+k+2 \leq n$.

En effet, le chiffre de poids faible du résultat de la multiplication-addition est disponible lorsque le chiffre d'indice p+1 se présente à l'entrée du multiplieur-additionneur.

Au cycle suivant, l'addition est réalisée. S_0 est disponible, le calcul de m_i peut donc débuter.

Après k+1 coups d'horloge supplémentaires, m_i est disponible en sortie du multiplieur partie basse. Il peut donc être utilisé comme entrée du multiplieur-additionneur au coup d'horloge suivant. Ceci explique la condition $p+k+2 \leq n$.

Rebouclage

Si on veut chaîner les multiplications-additions (1) et (2), sans perte de temps, on choisit $p+k+2 \leq n$.

Dans ce cas, la donnée m_i est disponible avant la fin de parcours des chiffres de la multiplication-addition en cours. On reboucle donc cette valeur, afin de retarder son entrée dans le multiplieur-additionneur. On définit alors $n_{rebk}=n-p-k-2$ qui correspond au nombre de rebouclage nécessaire de cette valeur.

Dans le cas particulier où $n_{rebk}=0$, la donnée m_i est synchrone avec les nouvelles entrées du multiplieur-additionneur.

Cependant, dans tous les cas on reboucle la valeur m_i n fois afin que l'entrée soit la même pour tous les chiffres de N.

Si en revanche n-p-k-2 est négatif, ceci correspond à un retard du calcul de m_i, on rajoute donc de la latence.

Latence

Lorsque la condition $p+k+2 \leq n$ n'est pas réalisée, ceci signifie que les sorties ont du retard sur les entrées, des coups d'attente (latence) pour synchroniser les données sont rajoutés.

Pendant les coups de latence, les entrées sont bloquées (au sens où on prend comme nouvelle entrée 0 (pour tenir compte de la dernière retenue de S)), le calcul se poursuivant pour les données déjà entrées.

Dans ce cas (p+k+2>n), on définit $n_{latk}=p+k+2-n$. Cette grandeur représente le nombre de coups de latence à effectuer avant de présenter de nouvelles entrées au multiplieur-additionneur.

Dès que m_i est déterminé, il est utilisé comme entrée de la multiplication-addition. S_0 quant à lui est forcément déterminé avant m_i et doit être stocké (ainsi que S_1, S_2 ...) jusqu'à ce que m_i soit calculé.

Pour cette raison, on rajoute des registres afin de retarder l'arrivée des résultats à l'entrée du multiplieur-additionneur.

Registres supplémentaires

Deux cas sont distingués. En effet, selon que p+k+2 est plus grand ou plus petit que n, le nombre et l'utilisation des registres ajoutés ne sont cependant pas les mêmes.

<u>Cas 1 : $p+k+2 \leq n$</u>

Dans ce cas, l'ensemble S_0 et m_i est déterminé avant la fin de parcours des chiffres des données.

Pour m_i, voir la section sur le rebouclage, on utilise la méthode décrite ci-dessus dans la section de rebouclage.

Pour S_0, on retarde son arrivée à l'entrée du multiplieur additionneur par le rajout de registres à décalage.

On définit alors n_{reg} par $n_{reg}=n-p-1$. Cette quantité correspond au nombre de registres à rajouter pour synchroniser l'arrivée du chiffre de poids faible du résultat d'une multiplication-addition avec les données de poids faible de la suivante.

Cas 2 : $p+k+2 > n$

Deux sous cas selon que p est ou non plus grand que n, sont distingués. En fait, quelque soit la valeur de p, m_i sera déterminé après S_0.

On retarde donc l'arrivée de S_0 à l'entrée du multiplieur par le rajout de registres. Ce nombre de registres ne dépendra donc que de k, la profondeur du pipeline de l'opérateur multiplication partie basse.

On définit donc dans ce cas, $n_{regk}=k+1$ le nombre de registres à rajouter pour retarder l'arrivée de S_0 à l'entrée du multiplieur.

Passage de (2) à (1)

On s'intéresse ici au passage de (2) à (1). Si il n'y a pas de m_i à déterminer, il faut en revanche tenir compte du décalage (division par r).

Comme précédemment, on synchronise l'arrivée des résultats avec l'entrée de nouvelles données. Ici, seule la quantité p est importante, m_i n'ayant pas besoin d'être déterminé, k n'intervient pas.

En revanche, on prend en compte le décalage (ie : considérer t_0 comme chiffre de poids faible et non pas t_{-1} qui est nul). Ceci peut être vu comme un niveau de pipeline supplémentaire.

Registres supplémentaires et latence

On distingue donc comme précédemment, deux cas, selon que p+2 est plus grand que n ou pas.

Cas 1 : $p+2 \leq n$

Dans ce cas, t_0 est disponible avant la fin de parcours des chiffres des données. On rajoute donc des registres pour prendre en compte le retard. On définit alors, $n'_{reg}=n-p-2$ qui indique le nombre de registre à rajouter pour prendre en compte le retard.

Cas 2 : p+2> n

Dans ce cas, t_0 est disponible après le parcours des chiffres des données. On retarde donc l'entrée des nouvelles données. Ceci se fait comme précédemment en rajoutant des coups d'attente (latence). On définit $n'_{lat}=p+2-n$ qui représente le nombre de coups d'attente à réaliser.

Additionneur supplémentaire et rebouclage

La détermination de t_n se fait par une addition de s_{n+1} avec c, $s_{n+1} \leq 2$ et $c \leq 1$. Pour cela on prévoit un additionneur (de la logique) 2 bits + 1 bits vers 2 bits ($t_n \leq 3$) (composant désigné 4).

Dans le calcul de T, le décalage permet d'économiser l'utilisation d'un registre. Celui-ci est utilisé pour le stockage de s_{n+1}. Cette valeur est stockée jusqu'à ce que c soit déterminé, puis l'addition des deux est réalisée libérant ainsi le registre de stockage de s_{n+1}.

On définit donc $n_{reb}=n+n_{latk}-1$ qui est le nombre de rebouclage nécessaire de s_{n+1}.

Paramètres de correction

Le dessin final du composant dépend notamment des profondeurs de pipeline p et k ainsi que du nombre de chiffres n des entiers longs pour lequel il est initialement prévu. En particulier, le nombre de registres à rajouter est un point délicat puisqu'il n'est pas le même suivant que l'on se trouve dans le cas du passage de (1) à (2) ou dans le passage de (2) à (1).

Le tableau 1 de synthèse qui suit lie les quantités p, k et n aux paramètres de correction précédemment définies.

Tableau 1

	$p+k+2 \leq n$	$p+k+2 > n$	
	$(\Rightarrow p+2 \leq n)$	$p+2 \leq n$	$p+2 > n$
n'_{reg}	n-p-2	n-p-2	0
n'_{lat}	0	0	p+2-n
n_{reg}	n-p-1	k+1	
n_{latk}	0	p+k+2-n	
n_{rebk}	n-p-k-2	0	
n_{reb}	$n+n_{latk}-1=n-1$	$n+n_{latk}-1=p+k+1$	

En pratique le nombre de registres à rajouter est défini par $n_{reg\text{-}max}=\max(n_{regk},n_{reg})$ et est égal à n-p-1 si $p+k+2 \leq n$ et k+1 sinon. En particulier, $n_{reg\text{-}max} \geq 1$.

Les étapes nécessitant moins de registres sont effectuées en raccourcissant la chaîne des registres par le rajout de multiplexeurs.

Séquencement décrit à la figure 2

Un exemple de séquencement des opérations est décrit en relation avec la figure 2. On adopte comme convention de fixer les états des multiplexeurs à la fin du coup d'horloge en cours pour commander le coup d'horloge suivant.

Comportement général des multiplexeurs

La latence intervient ici. En effet, selon qu'il y a de la latence ou non, les changements d'état ne se font pas aux même moments.

Il est cependant possible d'utiliser les paramètres de correction de latence du circuit (n'_{lat} et n_{latk}) pour définir un comportement général des multiplexeurs.

En effet, on peut supposer que $|B|=n+1+ n'_{lat}$ avec les n'_{lat} premiers chiffres de B nuls. (Sauf évidemment pour le calcul de a_0B).

De même, on peut supposer que $|N|=n+1+ n_{latk}$ avec les n_{latk} premiers chiffres de N nuls.

De plus, on isole le cas particulier du premier calcul a_0B pour lequel on ne prend pas en compte la latence (les n'_{lat} premiers chiffres nuls de B).

Enfin, passé ce cas particulier, on remarque que les données présentées successivement en entrée du multiplieur-additionneur peuvent être regroupées par tranches de $2n+2+ n'_{lat} + n_{latk}$ données. Les $n_{latk} +n+1$ premières correspondent aux données m_i et N_j. Les $n'_{lat} +n+1$ dernières correspondent aux données a_i et b_j.

On aura donc un comportement cyclique des multiplexeurs, de période $2n+2+n'_{lat}+n_{latk}$

mux 1

mux1 est un multiplexeur à deux états symbolisant quel type d'entrées doit être pris en compte au niveau du multiplieur-additionneur. Les deux états sont :

- 0 : $x=a_i$ et $y=b_j$ sont pris comme entrées du multiplieur-additionneur.
- 1 : $x=m_i$ et $y=N_j$ sont pris comme entrées du multiplieur-additionneur.

L'utilisation des constantes n'_{lat} et n_{latk} permet notamment de vérifier que le changement d'état intervient lorsque tous les chiffres de B (ou de N) ont été parcourus.

Pour a_0 on ne prend pas en compte les n'_{lat} premiers chiffres nuls. Le calcul commence directement par les données $a_0b_{n'lat}$.

Ainsi mux1, initialement à 0 (reset), reste dans cet état pendant les n premiers coups d'horloge, et passe dans l'état 1 au $n+1^{ème}$ coup.

A la fin de ce $n+1^{ème}$ coup, les données présentées en entrée du multiplieur-additionneur sont a_0 et b_n, tous les chiffres de B auront donc été parcourus.

mux1 est à 1 à la fin de ce même coup d'horloge, donc à la fin du coup d'horloge suivant, m_1 et N_0 seront présentés en entrée du multiplieur-additionneur.

Le comportement général de mux1 en fonction du coup d'horloge clock peut être résumé par les étapes suivantes :

Si clock<n+1, alors mux1=0

Sinon :

Si (clock-(n+1) mod (2n+2+ n_{latk} + n'_{lat}))<n+1+ n_{latk}, alors mux1=1

Sinon mux1=0

mux2

mux2 est un multiplexeur à deux états symbolisant à quel moment doit être fait l'addition s_{n+2}+c. On rappelle que s_{n+2} est stocké dans $Stab_1$.

De plus, lorsque cette addition est réalisée, la retenue de l'additionneur trois entrées doit être initialisée à 0 puisque une nouvelle addition commence.

Les deux états sont :

- 0 : L'addition s_{n+1}+c ne peut être effectuée, c n'est pas encore déterminé.
- 1 : Les entrées s_{n+1} et c sont positionnées pour être ajoutées au coup d'horloge suivant, la retenue de l'additionneur trois entrées est initialisée à 0.

Cette addition est réalisée une seule fois par itération principale (boucle sur i), elle se situe dans la deuxième boucle sur les chiffres de N.

Ceci signifie que mux2 n'est jamais deux fois de suite dans l'état 1.

De plus, cette addition porte sur les valeurs du registre Stab1, donc la profondeur de pipeline p intervient pour déterminer le comportement de mux2.

Le chiffre de poids faible (s_0) du produit $a_0 \times b_0$ est dans le registre de sortie de l'additionneur à clock p+3 (=1(load)+(p+1)(s_0 dans LSB)+1(s_0 dans registre de sortie de l'additionneur)).

s_n est donc dans ce même registre à clock p+3+n. Comme s_n correspond au chiffre de poids faible de $a_0 \times b_n$, les entrées suivantes sont donc les chiffres de N et m_1.

Or l'addition doit être réalisée lorsque t_{n-1} se trouve dans le registre de sortie de l'additionneur puisqu'à ce moment on dispose de la bonne valeur de carry à ajouter à s_{n+1} pour déterminer t_n.

mux2 doit donc être dans l'état 1 lorsque t_{n-1} se trouve dans le registre de sortie de l'additionneur, c'est à dire au coup d'horloge $p+3+n+n_{latk}+n+1=p+2n+n_{latk}+4$. (Il ne faut pas oublier qu'avec le décalage, t_{n-1} correspond au calcul de $m_1 \times N_n$).

Par périodicité, on peut décrire le comportement général de mux2.

Si clock = $p+2n+4+n_{latk}$ mod $2n+2+ n_{latk} + n'_{lat}$, alors mux2=1

Sinon mux2=0

mux3

Lors de la boucle sur les chiffres de N, un décalage à droite doit être effectué sur les chiffres de la sortie pour tenir compte de la division par r. mux3 est un multiplexeur à deux états symbolisant à quel moment on effectue le décalage (modélisé par un saut de registre). Les deux états sont :

- 0 : Le décalage n'est pas effectué.
- 1 : Le décalage est effectué.

Ce décalage d'indice est réalisé en hardware par le saut d'un registre.

Le décalage intervient lorsque la donnée s_{n+1} est présente dans le registre Stab1. En effet, à cet instant, le registre S de l'additionneur contient, suivant la valeur de n_{latk}, soit 0 (résultat d'un coup de latence), soit la valeur de t_0. t_0 ayant un temps de retard par rapport à la multiplication classique (s_0), il doit alors sauter un registre pour rattraper ce temps de retard. Ce décalage doit donc être effectué jusqu'à ce que tous les chiffres (latence comprise) de T, jusqu'à t_{n-1} , soient déterminés. En effet, lorsque t_{n-1} est déterminé (ie : dans le registre S de l'additionneur), au coup d'horloge suivant, t_n est déterminé dans Stab1 par addition de c avec s_{n+1}, et t_{n-1} se trouve dans le registre Stab2. Le décalage prend alors fin, les données t_{n-1} et t_n étant de nouveau dans deux registres successifs. mux3 reste donc dans l'état 1 durant $n+ n_{latk} = n_{reb} +1$ coups.

Comme nous l'avons dit précédemment, le décalage correspond au saut du registre Stab1 qui est alors utilisé pour le rebouclage de s_{n+1}. Le rebouclage de s_{n+1} intervient donc au même moment que le décalage. Donc le changement d'état de mux3 de 0 à 1 indique également qu'il faut reboucler la valeur s_{n+1} dans le registre Stab1. Ce rebouclage à lieu n_{reb} fois.

Initialement, mux3 est dans l'état 0 (multiplication classique). Il passe à 1 lorsque s_{n+1} est dans le registre Stab1. Or s_n est dans S au coup d'horloge p+3+n (cf : comportement de mux2). Donc s_{n+1} est dans ce même registre au coup p+n+4, et dans Stab1 au coup suivant : p+n+5.

Par périodicité, on déduit le comportement général de mux3 :

Si clock<p+n+5, alors mux3=0

Sinon :

Si (clock-(p+n+5) mod (2n+2+ n_{latk} + n'_{lat}))<n_{reb} +1, alors mux3=1

Sinon mux3=0

La remarque précédente permet de définir et de décrire la commande reb.

Commande reb

Cette commande représente les instants pour lesquels s_{n+1} doit être rebouclé dans le registre Stab1.

Les deux états sont :

- 0 : pas de rebouclage
- 1 : rebouclage

Le comportement de reb est décrit en même temps que celui de mux3. On en déduit donc :

Si clock<p+n+5, alors reb=0

Sinon :

Si (clock-(p+n+5) mod (2n+2+ n_{latk} + n'_{latk}))<n_{reb}, alors reb=1

Sinon reb=0

mux4

mux4 est un multiplexeur à deux états qui fait parti de la barrière de registres.

Dans les cas où ce multiplexeur est présent, il indique s'il faut utiliser n'_{reg} ou n_{regk} registres. Les deux états sont :

- 0 : Utilisation de la totalité des registres (correspond à la multiplication (2)).
- 1 : Utilisation de n'_{reg} registres (correspond à la multiplication (1)).

mux4 doit être dans l'état 1 lorsque t_0 est déterminé dans S. On a vu (cf : mux3) que s_{n+1} est présent dans S à clock=p+n+4, donc n_{latk} +1 coups d'horloge après, c'est à dire à clock=p+n+5+ n_{latk} , t_0 est dans S.

mux4 doit rester à 1 jusqu'à ce que t_n se trouve dans Stab1 soit durant n+1 coups d'horloge.

Par périodicité, on déduit un comportement général de mux4 :

Si clock<p+n+5+ n_{latk}, alors mux4=0

Sinon :

Si (clock-(p+n+5+ n_{latk}) mod (2n+2+ n_{latk} +n'_{lat}))<n+1, alors mux4=1

Sinon mux4=0

Commande reb_k

La commande reb_k indique à quel moment il faut reboucler la valeur de m_i dans le registre de sortie du multiplieur partie basse.

Les deux états sont :

- 0 : pas de rebouclage
- 1 : rebouclage

Initialement, reb_k =0.

m_1 est déterminé (ie : présent dans le registre de sortie du multiplieur partie basse) à la fin du coup d'horloge clock=p+k+4. En effet, m_1 est déterminé à l'aide de s_0 qui est lui même présent dans le registre S à la fin du coup d'horloge clock=p+3 (cf : mux2). Il sert donc d'entrée au multiplieur partie basse qui donne le résultat k+1 coups d'horloge après, soit à clock=p+k+4.

Il faut donc reboucler cette valeur à partir de ce moment là, au moins n+1 fois afin que cette entrée soit la même pour tous les chiffres de N. Il faut également tenir compte de la valeur de n_{rebk} qui est le nombre de

rebouclages nécessaires de m_1 dans le cas où m_1 est déterminé avant le parcours de tous les chiffres de B.

Le nombre de rebouclage total de m_1 est donc : $n+1+n_{rebk}$.

Par périodicité, on en déduit le comportement général de reb_k :

Si clock<p+n+4, alors $reb_k=0$

Sinon :

Si (clock-(p+n+4) mod (2n+2+ n_{latk} + n'_{lat}))<n+1+n_{rebk}, alors reb_k =1

Sinon $reb_k=0$

REVENDICATIONS

1 - Dispositif de mise en œuvre de la multiplication modulaire caractérisé en ce qu'il comporte au moins une cellule de calcul comportant un multiplieur-additionneur (1) comportant p couples logique-registre pipeliné, recevant plusieurs chiffres à additionner et à multiplier, au moins deux sorties correspondant au poids faible et au poids fort, un additionneur (2) recevant les deux sorties du multiplieur-additionneur, le nombre p étant choisi de façon telle que la fréquence maximale F1max du multiplieur-additionneur soit supérieure ou égale à la fréquence maximale F2max de l'additionneur.

2 – Dispositif selon la revendication 1 caractérisé en ce que la fréquence maximale F1max du multiplieur-additionneur est sensiblement égale à la fréquence maximale F2max de l'additionneur.

3 - Dispositif selon la revendication 1 caractérisé en ce qu'il comporte un multiplieur (3) partie basse dont les entrées correspondent à la sortie LSB de l'additionneur et à une constante précalculée, ledit multiplieur étant pipeliné sur une profondeur k.

3 - Dispositif selon l'une des revendications 1 ou 2 caractérisé en ce qu'il comporte un ensemble de registres et de multiplexeurs adapté à retarder l'arrivée des données dans le multiplieur-additionneur.

4 - Dispositif selon l'une des revendications 1 à 2 caractérisé en ce qu'il comporte un additionneur complémentaire 2 bits + 1 bit vers 2 bits.

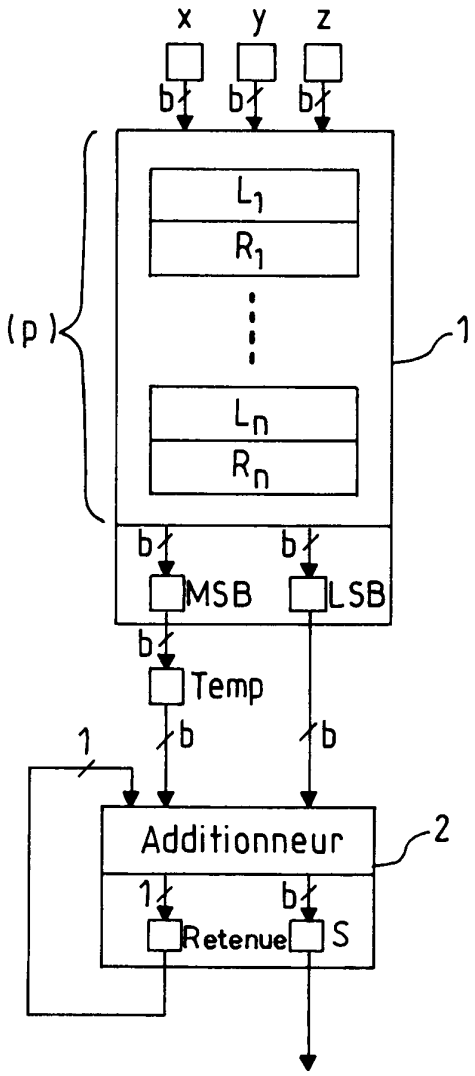

x
y
z
b
b
b
L_1
R_1
L_n
R_n
(p)
1
b
b
MSB
LSB
b
Temp
1
b
b
Additionneur
2
1
b
Retenue
S

FIG.1

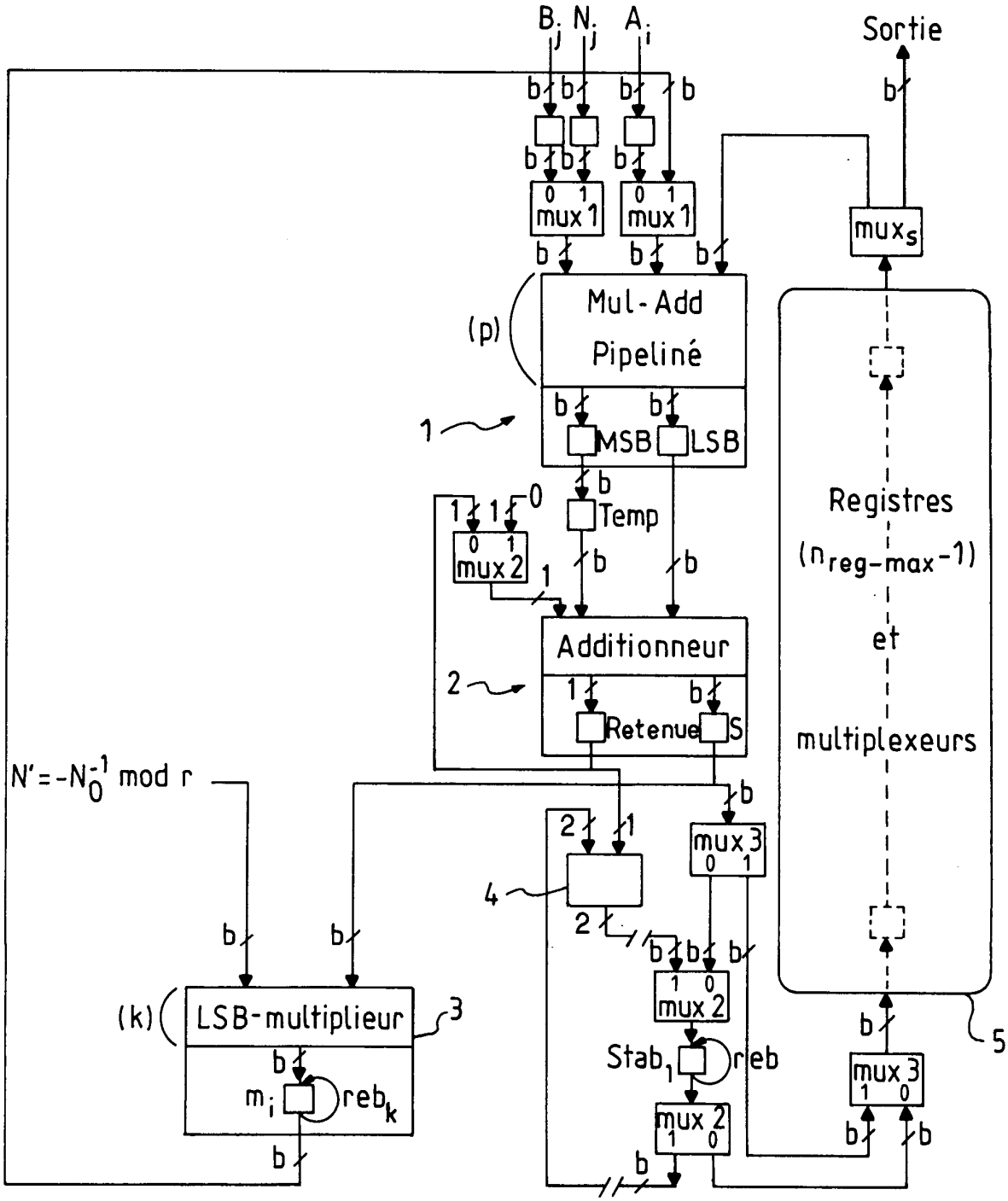

B. N. A.
j j i
Sortie
mux 1
mux 1
mux s
Mul - Add
Pipeliné
(p)
1
MSB
LSB
Temp
mux 2
Registres
(n reg-max -1)
et
multiplexeurs
Additionneur
2
Retenue
S
N' = -N0^-1 mod r
mux 3
4
mux 2
Stab1
reb
mux 2
mux 3
(k)
LSB-multiplieur
3
mi
rebk
5

FIG.2

RÉPUBLIQUE FRANÇAISE

2884005

INPI
INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE

RAPPORT DE RECHERCHE PRÉLIMINAIRE

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

N° d'enregistrement national

FA 666423
FR 0503178

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		
Y	US 2005/033790 A1 (HUBERT GERARDUS TARCISIUS MARIA) 10 février 2005 (2005-02-10) * abrégé * * alinéas [0091] - [0101], [0115] - [0118]; figure 1 *	1-4	G06F17/10 G06F7/38
Y	US 5 133 069 A (ASATO ET AL) 21 juillet 1992 (1992-07-21) * colonne 1, ligne 47 - ligne 61 * * colonne 5, ligne 46 - ligne 60; figures 1,4 *	1-4	
A	US 2002/194237 A1 (TAKAHASHI RICHARD J ET AL) 19 décembre 2002 (2002-12-19) * alinéas [0042], [0060] - [0062] *	1-4	

DOMAINES TECHNIQUES RECHERCHÉS (Int.CL.7)

G06F

Date d'achèvement de la recherche	Examinateur
4 novembre 2005	Cohen, B

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 0503178 FA 666423

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 04-11-2005
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 2005033790 A1	10-02-2005	AU 2002353282 A1	30-06-2003
		CN 1605059 A	06-04-2005
		WO 03052584 A2	26-06-2003
		JP 2005513532 T	12-05-2005
US 5133069 A	21-07-1992	AUCUN	
US 2002194237 A1	19-12-2002	WO 02101539 A2	19-12-2002

EPO FORM P0465