(19) RÉPUBLIQUE FRANÇAISE

INSTITUT NATIONAL
DE LA PROPRIÉTÉ INDUSTRIELLE

COURBEVOIE

(11) **N° de publication :** **3 101 980**
(à n'utiliser que pour les commandes de reproduction)

(21) **N° d'enregistrement national :** **19 11348**

(51) Int Cl[8] : ***G 06 F 21/72*** *(2019.12), G 06 F 9/30, G 06 F 21/64, G 06 F 15/00, H 04 L 9/00*

(12) **BREVET D'INVENTION** **B1**

(54) Processeur.

(22) **Date de dépôt :** 11.10.19.

(30) **Priorité :**

(43) **Date de mise à la disposition du public de la demande :** 16.04.21 Bulletin 21/15.

(45) **Date de la mise à disposition du public du brevet d'invention :** 10.12.21 Bulletin 21/49.

(56) **Liste des documents cités dans le rapport de recherche :**

Se reporter à la fin du présent fascicule

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

◯ **Demande(s) d'extension :**

(71) **Demandeur(s) :** *STMICROELECTRONICS (GRENOBLE 2) SAS Société par actions simplifiée* — FR et *STMICROELECTRONICS (ROUSSET) SAS Société par actions simplifiée* — FR.

(72) **Inventeur(s) :** PEYRARD Rene, ROMAIN Fabrice, DERIEN Jean-Michel et EICHWALD Christophe.

(73) **Titulaire(s) :** STMICROELECTRONICS (GRENOBLE 2) SAS Société par actions simplifiée, STMICROELECTRONICS (ROUSSET) SAS Société par actions simplifiée.

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

FR 3 101 980 - B1

Description

Titre de l'invention : *Processeur*

Domaine technique

[0001] La présente description concerne de façon générale les systèmes et circuits électroniques, et, de façon plus particulière, les processeurs. La présente description traite plus précisément de processeurs adaptés à traiter des données masquées.

Technique antérieure

[0002] Un processeur est un composant électronique, présent dans de nombreux systèmes et circuits électroniques, qui est adapté à traiter des données en exécutant des commandes et des instructions de programmes informatiques.

[0003] Dans certains cas, un processeur peut avoir à traiter des données secrètes. Ces données secrètes sont généralement chiffrées, par exemple par masquage.

[0004] Il serait souhaitable de pouvoir améliorer, au moins en partie, certains aspects des processeurs connus.

Résumé de l'invention

[0005] Il existe un besoin dans la technique pour des processeurs plus fiables.

[0006] Il existe un besoin dans la technique pour des processeurs adaptés à traiter des données masquées.

[0007] Il existe un besoin dans la technique pour des processeurs adaptés à traiter des données masquées sans mettre en oeuvre d'opérations de démasquage de ces données masquées.

[0008] Un mode de réalisation pallie tout ou partie des inconvénients des processeurs connus.

[0009] Un mode de réalisation prévoit un procédé de traitement de données masquées par un processeur comprenant une unité arithmétique et logique, dans lequel lesdites données masquées restent masquées pendant leur traitement dans ladite unité arithmétique et logique.

[0010] Selon un mode de réalisation, le processeur comprend, en outre, une unité de génération d'adresses, dans laquelle lesdites données masquées restent masquées pendant leur traitement dans ladite unité de génération d'adresses.

[0011] Selon un mode de réalisation, le processeur comprend, en outre, des banques de registres, dans lesquelles lesdites données masquées sont stockées et restent masquées pendant tout leur stockage.

[0012] Selon un mode de réalisation, les banques de registres comprennent au moins une banque de registres pour lesdites données et au moins une banque de registres pour des masques desdites données.

[0013] Selon un mode de réalisation, le processeur comprend, en outre, des registres stockant des pointeurs masqués.

[0014] Selon un mode de réalisation, lesdits registres stockent au moins un pointeur de registre masqué et au moins un pointeur de programme masqué.

[0015] Selon un mode de réalisation, lesdits registres comprennent des pointeurs référençant des données, et des masques de pointeurs.

[0016] Selon un mode de réalisation, les données masquées peuvent être masquées selon un premier et un deuxième type de masquage.

[0017] Selon un mode de réalisation, le premier type de masquage n'utilise que des opérations arithmétiques.

[0018] Selon un mode de réalisation, une donnée masquée par le premier type de masquage est égale à l'addition d'une donnée à masquer et d'un masque.

[0019] Selon un mode de réalisation, le deuxième type de masquage n'utilise que des opérations logiques.

[0020] Selon un mode de réalisation, une donnée masquée par le deuxième type de masquage est égale à l'application de l'opération OU EXCLUSIF bit à bit entre la donnée à masquer et le masque.

[0021] Selon un mode de réalisation, l'unité arithmétique et logique comprend un circuit de masquage adapté à modifier le type de masquage desdites données masquées.

[0022] Un autre mode de réalisation prévoit un processeur adapté à mettre en oeuvre le procédé décrit précédemment.

Brève description des dessins

[0023] Ces caractéristiques et avantages, ainsi que d'autres, seront exposés en détail dans la description suivante de modes de réalisation particuliers faite à titre non limitatif en relation avec les figures jointes parmi lesquelles :

[0024] [fig.1] la figure 1 représente, schématiquement et sous forme de blocs, un mode de réalisation d'un processeur ;

[0025] [fig.2] la figure 2 représente, schématiquement et sous forme de blocs, un mode de réalisation d'une unité arithmétique et logique et de génération d'adresses du processeur de la figure 1 ;

[0026] [fig.3] la figure 3 représente un organigramme illustrant une opération de l'unité de la figure 2 ;

[0027] [fig.4] la figure 4 représente un organigramme illustrant une autre opération de l'unité de la figure 2 ;

[0028] [fig.5] la figure 5 représente, schématiquement et sous forme de blocs, un mode de réalisation d'un circuit logique d'addition ;

[0029] [fig.6] la figure 6 représente, schématiquement et sous forme de blocs, un autre mode

de réalisation d'un circuit logique du calcul d'un complémentaire ;

[0030] [fig.7] la figure 7 représente, schématiquement et sous forme de blocs, un autre mode de réalisation d'un circuit logique de décalage de bit dans un mot binaire ;

[0031] [fig.8] la figure 8 représente, schématiquement et sous forme de blocs, un mode de réalisation d'un circuit logique de multiplication ;

[0032] [fig.9] la figure 9 représente, schématiquement et sous forme de blocs, un autre mode de réalisation d'un circuit logique de multiplication ;

[0033] [fig.10] la figure 10 représente, schématiquement et sous forme de blocs, un autre mode de réalisation d'un circuit logique de multiplication ;

[0034] [fig.11] la figure 11 représente, schématiquement et sous forme de blocs, un mode de réalisation d'un circuit logique de calcul d'un ET logique bit à bit ;

[0035] [fig.12] la figure 12 représente, schématiquement et sous forme de blocs, un mode de réalisation d'un circuit logique de calcul d'un OU logique bit à bit ; et

[0036] [fig.13] la figure 13 représente, schématiquement et sous forme de blocs, un mode de réalisation d'un circuit logique de calcul d'un OU EXCLUSIF logique bit à bit.

Description des modes de réalisation

[0037] De mêmes éléments ont été désignés par de mêmes références dans les différentes figures. En particulier, les éléments structurels et/ou fonctionnels communs aux différents modes de réalisation peuvent présenter les mêmes références et peuvent disposer de propriétés structurelles, dimensionnelles et matérielles identiques.

[0038] Par souci de clarté, seuls les étapes et éléments utiles à la compréhension des modes de réalisation décrits ont été représentés et sont détaillés. Le fonctionnement complet d'un processeur ne sera pas décrit ici, les modes de réalisation décrits ci-après étant compatibles avec la plupart des procédés de fonctionnement usuels d'un processeur.

[0039] Sauf précision contraire, lorsque l'on fait référence à deux éléments connectés entre eux, cela signifie directement connectés sans éléments intermédiaires autres que des conducteurs, et lorsque l'on fait référence à deux éléments reliés ou couplés entre eux, cela signifie que ces deux éléments peuvent être connectés ou être reliés ou couplés par l'intermédiaire d'un ou plusieurs autres éléments.

[0040] Dans la description qui suit, lorsque l'on fait référence à des qualificatifs de position absolue, tels que les termes "avant", "arrière", "haut", "bas", "gauche", "droite", etc., ou relative, tels que les termes "dessus", "dessous", "supérieur", "inférieur", etc., ou à des qualificatifs d'orientation, tels que les termes "horizontal", "vertical", etc., il est fait référence sauf précision contraire à l'orientation des figures.

[0041] Sauf précision contraire, les expressions "environ", "approximativement", "sensiblement", et "de l'ordre de" signifient à 10 % près, de préférence à 5 % près.

[0042] La figure 1 représente, schématiquement et sous forme de blocs, l'architecture d'un

mode de réalisation d'un processeur 100 (CPU).

[0043] Le processeur 100 est un processeur adapté à traiter des données masquées et des instructions masquées. Plus particulièrement, le processeur 100 reçoit, en entrée, des données masquées Data_M, et leurs masques Mask_D, et des instructions masquées Instr_M, et leurs masques Mask_I. Le processeur fournit en sortie, des adresses masquées Add_M, et leurs masques Mask_A, des données de résultat masquées Result_M, et leur masques Mask_R, et, peut fournir également des données masquées Data_S qu'il stocke, et leurs masques Mask_S. Selon une variante de réalisation, le processeur 100 peut fournir, en sortie, des adresses directement démasquées.

[0044] Le processeur 100 est constitué de plusieurs circuits électroniques parmi les suivants :

- une banque de registres 101 (DATA REG) adaptée à stocker des données masquées ;
- une banque de registres 102 (MASK REG) adaptée à stocker des masques des données masquées stockées dans la banque de registre 101 ;
- un circuit 103 (INSTRUCTION DECODE EXECUTE) de décodage et d'exécution d'instructions ;
- une unité 104 (ALU AGU) arithmétique et logique et de génération d'adresses ;
- des registres 105 (SP/PC DATA) adaptés à stocker des pointeurs de données et de programmes ; et
- des registres 106 (SP/PC MASK) adaptés à stocker des masques de pointeurs.

[0045] La banque de registres 101 reçoit, en entrée, les données masquées Data_M ou les résultats masqués Result_M, et fournit, en sortie, les données masquées Data_S. La banque de registres 101 est adaptée à stocker les données masquées Data_M et les résultats masqués Result_M pendant leur traitement par le processeur 100.

[0046] La banque de registres 102 reçoit, en entrée, les masques de données Mask_D et les masques de résultat Mask_R, et fournit, en sortie, les masques Mask_S. La banque de registres 102 est adaptée à stocker les masques de données Mask_D et les masques de résultat Mask_R pendant leur traitement par le processeur 100.

[0047] Le circuit 103 de décodage et d'exécution d'instructions reçoit, en entrée, les instructions masquées Instr_M et les masques d'instruction Mask_I, et fournit, en sortie, un code opératoire Opcode. Le code opératoire Opcode n'est pas masqué. Le circuit 103 démasque les instructions Instr_M pour déterminer le code opératoire Opcode.

[0048] L'unité 104 arithmétique et logique et de génération d'adresses reçoit, en entrée, les données masquées Data_S, les masques Mask_S, des pointeurs de données Point_D masqués, des masques de pointeurs Mask_P, et le code opératoire Opcode. L'unité 104 fournit, en sortie, les adresses masquées Add_M, les masques d'adresse Mask_A, les données de résultat masquées Result_M, et les masques de résultat Mask_R. L'unité

104 a deux fonctions. Une première fonction est celle d'exécuter le code opératoire Opcode en appliquant des opérations arithmétiques et/ou logiques aux données masquées Data_S et aux masques Mask_S pour fournir les données de résultat masquées Result_M et leurs masques Mask_R. Une deuxième fonction est celle d'exécuter le code opératoire Opcode pour calculer les adresses Add_M et leurs masques Mask_A. Selon une variante, l'unité 104 peut être constituée d'une unité arithmétique et logique et d'une unité de génération d'adresses.

[0049] Les registres 105 reçoivent, en entrée, les données de résultat masquées Result_M, et fournissent, en sortie les pointeurs de données Point_D masqués. Les registres 105 sont, par exemple, composés d'un registre stockant un pointeur de registre masqué, ou pointeur de pile (Stack Pointer, SP) masqué, et d'un autre registre stockant un pointeur de programme (Program pointer, PC) masqué. Le pointeur de pile est un pointeur qui référence l'adresse de la dernière donnée stockée dans un registre ou l'adresse de la prochaine donnée à stocker dans un registre. Le pointeur de programme est un pointeur référençant l'adresse du dernier code opératoire utilisé ou l'adresse du prochain code opératoire à utiliser. Dans le cas décrit ici, les pointeurs de données Point_D masqués sont composés d'un pointeur de registre masqué référençant l'adresse de la dernière donnée écrite en mémoire, et d'un pointeur de programme masqué référençant l'adresse du dernier code opératoire Opcode utilisé par l'unité 104.

[0050] Les registres 106 reçoivent, en entrée, les masques de résultat Mask_R, et fournissent, en sortie les masques de pointeurs Mask_P. Les registres 106 sont similaires aux registres 105 mais sont associés aux masques. Ainsi, les registres 106 sont, par exemple, composés d'un registre stockant un masque du pointeur de registre, et d'un autre registre stockant un masque de pointeur de programme. Dans le cas décrit ici, les masques de pointeurs Mask_P sont les masques des pointeurs de données Point_D masqués stockés dans les registres 105.

[0051] Un mode de fonctionnement du processeur 100 est le suivant. Le processeur 100 reçoit, en entrée, les instructions Instr_M et les données masquées à traiter Data_M, ainsi que leur masques Mask_I et Mask_M. Les instructions Instr_M sont démasquées puis converties pour fournir le code opératoire Opcode par le circuit 103. Les données masquées Data_M et leurs masques Mask_M sont stockés, respectivement dans les banques de registres 101 et 102. L'unité 104 reçoit le code opératoire Opcode et les données masquées à traiter Data_S, et leurs masques Mask_S, ainsi que les pointeurs de données Point_D masqués et leurs masques Mask_P. L'unité 104 effectue différentes opérations arithmétiques et/ou logiques pour fournir les données de résultat masquées Result_M, et leurs masques Mask_R, et les adresses Add_M, et leurs masques Mask_A. L'unité 104 effectue toutes ses opérations sans jamais effectuer d'opération de démasquage des données masquées qu'elles traitent. Autrement dit, les

données masquées restent masquées pendant tout leur traitement par l'unité 104. Une unité 104 sera décrite plus en détails en relation avec la figure 2. Les données de résultat masquées Result_M peuvent être stockées dans la banque de registres de données 101, et sont utilisées pour obtenir les pointeurs de données Point_D masqués. Les masques de résultat Mask_R peuvent être stockés dans la banque de registres 102, et sont utilisés pour obtenir les masques de pointeurs Mask_P.

[0052] La figure 2 représente, schématiquement et sous forme de blocs, plus en détails un exemple d'une partie de l'architecture de l'unité 104 (ALU AGU) décrite en relation avec la figure 1. La partie décrite ici est adaptée à appliquer des opérations arithmétiques et/ou logiques aux données masquées Data_S et aux masques Mask_S fournis à l'unité 104.

[0053] L'unité 104 comprend :

- deux circuits de masquage 1041 et 1044 (TRANSMASKING) ;
- des circuits 1042 (ARITH OPERATIONS) adaptés à exécuter des opérations arithmétiques ; et
- des circuits 1043 (LOGIC OPERATIONS) adaptés à exécuter des opérations logiques.

[0054] Le circuit de masquage 1041 est un circuit adapté à modifier le type de masquage utilisé pour masquer les données masquées Data_S. En effet, les données masquées Data_S peuvent être masquées de deux façons différentes. Une première façon de masquer les données masquées Data_S est d'utiliser une ou plusieurs opérations arithmétiques pour combiner la donnée et son masque, et ainsi obtenir une donnée masquée. A titre d'exemple, les données masquées Data_S sont masquées en utilisant une opération d'addition, et plus particulièrement, en ajoutant, le masque à la donnée que l'on veut masquer. Une deuxième façon de masquer les données masquées Data_S est d'utiliser une ou plusieurs opérations logiques pour combiner la donnée et son masque, et ainsi obtenir une donnée masquée. A titre d'exemple, les données masquées Data_S sont masquées en utilisant une opération OU EXCLUSIF bit à bit, et plus particulièrement, en effectuant l'opération OU EXCLUSIF bit à bit entre la donnée et son masque.

[0055] Selon un mode de réalisation préféré, les données masquées stockées dans la banque de registres 101 présentent, de préférence, un même type de masquage. Selon un mode de réalisation, le type de masquage utilisé est le masquage arithmétique. A titre de variante, ces données masquées peuvent présenter des types de masquage différents.

[0056] Le circuit de masquage 1041 est tout particulièrement adapté à modifier le type de masquage utilisé pour adapter le type de masquage la donnée masquée Data_S en fonction des opérations qui lui seront appliquées par la suite. Plus précisément, le circuit 1041 est adapté à mettre en oeuvre deux opérations de conversion de masquage,

une première opération permettant de convertir un masquage de type logique en un masquage de type arithmétique, et une deuxième opération permettant de convertir un masquage de type arithmétique en un masquage de type logique. Les deux opérations de conversion de masquage ne comprennent pas d'opération de démasquage des données masquées. Dans l'exemple décrit ici, les deux opérations de conversion de masquage ne modifient pas le masque utilisé, elles permettent seulement de modifier la donnée masquée. A titre de variante, les opérations de conversion de masquage pourraient modifier les masques associés aux données masquées. Des exemples d'opérations de conversion de masquage sont décrits en relation avec les figures 3 et 4. Ainsi, le circuit 1041 reçoit, en entrée, les données masquées Data_S et les masques Mask_S, et fournit en sortie des données masquées Data_A et leurs masques Mask_S, et des données masquées Data_L et leurs masques Mask_S. Les données masquées Data_A sont masquées par un masquage de type arithmétique. Les données masquées Data_L sont masquées par un masquage de type logique.

[0057] Le circuit de masquage 1044 est un circuit ayant des fonctions similaires aux circuits de 1041. Le circuit de masquage 1044 est différent du circuit 1041 en ce qu'il reçoit en entrée les données masquées, et leurs masques, en sortie des circuits 1042 et 1043. Le circuit de masquage 1044 permet d'adapter le type de masquage des données masquées qu'il reçoit pour le stockage futur des données masquées dans les registres de la banque de registres 101. Le circuit 1044 fournit en sortie, les données de résultat masquées Result_M et leurs masques Mask_R.

[0058] Il est à noter que si les données arrivant au niveau du circuit 1041, ou du circuit 1044, ont déjà le type de masquage correspondant aux opérations qui sont requises ensuite, les circuits 1041 et 1044 ne modifient par leur type de masquage.

[0059] Les circuits 1042 reçoivent, en entrée, les données masquées Data_A et leurs masques Mask_S, et fournissent, en sortie, des données de résultat masquées ResultA_M et leurs masques Mask_RA. Les circuits 1042 sont des circuits mettant en oeuvre des opérations arithmétiques telles que l'addition, le calcul d'un complémentaire, la multiplication, une opération de décalage (shift) de bit, etc. Des exemples de circuits permettant de mettre en oeuvre ces opérations sont décrits en relation avec les figures 5 à 10.

[0060] Les circuits 1043 reçoivent, en entrée, les données masquées Data_L et leurs masques Mask_S, et fournissent, en sortie, des données de résultat masquées ResultL_M et leurs masques Mask_RL. Les circuits 1043 sont des circuits mettant en oeuvre des opérations logiques telles que le ET bit à bit, le OU bit à bit, le OU EXCLUSIF (XOR) bit à bit, etc. Des exemples de circuits permettant de mettre en oeuvre ces opérations sont décrits en relation avec les figures 11 à 13.

[0061] Dans certains cas, des opérations logiques et des opérations arithmétiques peuvent

être appliquées pour réaliser une même instruction, ainsi les données de résultat masquées ResultA_M et ResultL_M peuvent être réutilisées pendant un même cycle en étant renvoyées au circuit de masquage 1041 ou au circuit de masquage 1044 pour adapter leur type de masquage en fonction des opérations qui leur seront appliquées.

[0062] Les figures 3 et 4 sont des organigrammes illustrant des exemples d'opérations de conversion de masquage utilisées dans le circuit 1041 décrit en relation dans la figure 2.

[0063] Une donnée masquée A_M représente une donnée A masquée par le masque MA. Les données A et A_M et le masque MA sont tous des mots binaires comprenant n bits, n étant un entier naturel. A titre de variante, les données A, A_M et le masque MA sont des mots binaires de nombre de bits différents.

[0064] Dans les exemples des figures 3 et 4, le masquage arithmétique est un masquage de type additif dans lequel le masque est ajouté à la donnée à masquer. Plus particulièrement, une donnée masquée A_M est donnée par la formule suivante :

[Math.1]

$$A\_M = (A + MA) \bmod 2^n$$

dans lequel :

- "+" représente l'opération d'addition ; et
- "mod" représente l'opération modulo.

[0065] Le masquage logique est un masquage utilisant l'opération OU EXCLUSIF bit à bit. Plus particulièrement, la donnée masquée A_M est donnée par la formule suivante :

[Math.2]

$$A\_M = A \text{ xor } MA$$

dans lequel "xor" représente l'opération OU EXCLUSIF bit à bit.

[0066] La figure 3 illustre une opération ArithToLogic de conversion de masquage d'une donnée masquée résultat d'une opération de masquage arithmétique en une donnée masquée résultant d'une opération de masquage logique.

[0067] Une donnée B_M représente une donnée B masquée par un masque MB avec une opération de masquage arithmétique. Les données B et B_M et le masque MB sont tous des mots binaires comprenant n bits, n étant un entier naturel.

[0068] L'opération ArithToLogic prend, en entrée, la donnée masquée B_M et son masque MB, et fournit, en sortie, la nouvelle donnée masquée B_M'. La donnée B_M représente la donnée B masquée par le masque MB avec une opération de masquage logique.

[0069] A une étape 21 (Comp), on calcule le complémentaire !B_M de la donnée masquée B_M. Cette étape ne nécessite pas l'utilisation de la donnée B, un exemple de circuit mettant en oeuvre cette opération est décrit en relation avec la figure 6.

[0070] A une étape 22 (Carry), on calcule une retenue CarryB à partir de la donnée masquée

B_M et du masque MB. Plus particulièrement, la retenue CarryB est donnée par la formule suivante :

[Math.3]

$$\begin{cases} CarryB[i] = B\_M[i].\,!MB[i] + CarryB[i-1].(B\_M[\,i\,]\ xor\ !MB[i]) \\ CarryB[\,0\,] = 1 \end{cases}$$

dans lequel :

- "i" est un entier naturel variant de 0 à n-1 ;
- "x[i]" représente le bit de rang i du mot binaire x ;
- "." représente l'opération logique ET bit à bit ; et
- "!x" représente le complémentaire du mot binaire x.

[0071] L'étape 22 ne nécessite pas l'utilisation de la donnée démasquée B. Des exemples de circuits adaptés à mettre en oeuvre les opérations utilisées par cette formule sont décrits en relation avec les figures 5 à 13. Les étapes 21 et 22 peuvent, à titre d'exemple, être réalisées simultanément.

[0072] A une étape 23 (XOR), on calcule la nouvelle donnée masquée B_M' en réalisant l'opération suivante :

[Math.4]

$$B\_M' = !B\_M \text{ xor } CarryB$$

[0073] Un exemple de circuit mettant en oeuvre l'opération OU EXCLUSIF bit à bit est décrit en relation avec la figure 13.

[0074] Un avantage de l'opération ArithToLogic est qu'elle n'utilise pas la donnée démasquée B pour convertir la donnée B_M en la donnée B_M'.

[0075] La figure 4 illustre une opération LogicToArith de conversion de masquage d'une donnée masquée avec un masquage logique en une donnée masquée avec un masquage arithmétique.

[0076] Une donnée C_M représente une donnée C masquée par le masque MC par une opération de masquage logique. Les données C et C_M et le masque MC sont tous des mots binaires comprenant n bits.

[0077] L'opération LogicToArith prend, en entrée, la donnée masquée C_M et un masque MC, et fournit, en sortie, la nouvelle donnée masquée C_M'. La donnée C_M' représente une donnée C masquée par le masque MC par une opération de masquage arithmétique.

[0078] A une étape 31 (Carry), on calcule une retenue CarryC à partir de la donnée masquée C_M et du masque MC. Plus particulièrement, la retenue CarryC est donnée par la formule suivante :

[Math.5]

$$\begin{cases} CarryC[\,i\,] = !C\_M[i].MC[i] + CarryC[i-1].C\_M[\,i\,] \\ CarryC[\,0\,] = 0 \end{cases}$$

[0079] A une étape 32 (XOR), on calcule la nouvelle donnée masquée C_M' en réalisant, bit à bit, l'opération suivante :

[Math.6]

$$C\_M' = C\_M \text{ xor } CarryC$$

[0080] Un avantage de l'opération LogicToArith est qu'elle n'utilise pas la donnée démasquée C pour convertir la donnée C_M en la donnée C_M'.

[0081] Les figures 5 à 10 représentent des exemples de circuits adaptés à mettre en oeuvre des opérations arithmétiques des circuits 1042 décrits en relation avec la figure 2.

[0082] Dans les exemples décrits dans ces figures, l'opération de masquage arithmétique est identique à celle décrite en relation avec les figures 3 et 4.

[0083] La figure 5 représente, schématiquement et sous forme de blocs, un circuit logique adapté à mettre en oeuvre une opération d'addition symbolisée par un bloc 41 (+).

[0084] Dans l'exemple représenté en figure 5, l'opération d'addition prend, en entrée, deux données masquées D_M et E_M, et leurs masques MD et ME, et fournit, en sortie, une donnée masquée F_M, et son masque MF. La donnée D_M, respectivement E_M, F_M, est le résultat du masquage arithmétique d'une donnée D, respectivement E, F, par le masque MD respectivement ME, MF. La donnée masquée F_M et le masque MF sont donnés par les relations suivantes :

[Math.7]

$$\begin{cases} F\_M = D\_M + E\_M \\ MF = MD + ME \end{cases}$$

[0085] A titre de variante, l'opération d'addition pourrait comprendre plus de données en entrée.

[0086] Un avantage de cette opération est que les données D et E ne sont pas nécessaires pour le calcul de la donnée masquée F_M.

[0087] La figure 6 représente, schématiquement et sous forme de blocs, un circuit logique 42 (Comp) adapté à mettre en oeuvre une opération de calcul d'un complémentaire à 1.

[0088] Dans l'exemple représenté en figure 6, l'opération de calcul de complémentaire à 1 comprend, en entrée, une donnée masquée G_M, et son masque MG, et fournit, en sortie, une donnée masquée H_M, et son masque MH. La donnée G_M, respectivement H_M, est le résultat du masquage arithmétique d'une donnée G, respectivement H, par le masque MG, respectivement MH. La donnée masquée H_M et le masque MH sont données par les relations suivantes

[Math.8]

$$\begin{cases} H\_M = !G\_M \\ MH = !MG + 1 \end{cases}$$

[0089] Un avantage de cette opération est que la donnée G n'est pas nécessaire pour le calcul de la donnée masquée H_M.

[0090] Selon une variante de réalisation, la donnée masquée H_M et le masque MH sont donnés par les relations suivantes

[Math.9]

$$\begin{cases} H\_M = !G\_M - 1 \\ MH = !MG \end{cases}$$

dans lequel – représente l'opération soustraction.

[0091] La figure 7 représente, schématiquement et sous forme de blocs un circuit logique 43 (Shift) adapté à mettre en oeuvre une opération de décalage de bits.

[0092] Dans l'exemple représenté en figure 7, l'opération de décalage de bits prend, en entrée, une donnée masquée I_M, et son masque MI, et fournit, en sortie, une donnée masquée J_M, et son masque MJ. La donnée I_M, respectivement J_M, est le résultat du masquage arithmétique d'une donnée I, respectivement J, par le masque MI, respectivement MJ. Les mots binaires I_M, J_M, I, J, MI, MJ sont, par exemple, des mots binaires de n bits, n étant un entier naturel. L'opération de décalage de bits prend, en outre, en entrée, un mot binaire SelectLR lui indiquant s'il l'opération effectuer est un décalage de bit vers la droite ou vers la gauche. La donnée masquée J_M et le masque MJ sont, dans le cas d'un décalage de bits à gauche, donnés par les relations suivantes :

[Math.10]

$$\begin{cases} J\_M = (I\_M * 2^m) mod 2^n \\ MJ = (MI * 2^m) mod 2^n \end{cases}$$

dans lequel "*" représente l'opération de multiplication.

[0093] dans lesquelles, m est le décalage souhaité des bits de la donnée masquée en entrée, m étant un entier naturel inférieur ou égal à n.

[0094] Un avantage de cette opération est que la donnée I n'est pas nécessaire pour le calcul de la donnée masquée J_M.

[0095] Les figures 8 et 9 représentent, schématiquement et sous forme de blocs, des modes de réalisation de circuits logiques adaptés à mettre en oeuvre une opération de multiplication.

[0096] L'opération de multiplication prend en entrée, deux données masquées K_M et L_M et leurs masques respectifs MK et ML, et fournit en sortie une donnée masquée M_M et son masque MM. La donnée K_M, respectivement L_M, M_M est le résultat du masquage arithmétique d'une donnée K, respectivement L, M par le masque MK, respectivement ML, MM. Les mots binaires K_M, L_M, M_M, K, L, M, MK, ML, et MM sont des mots binaires de n bits, n étant un entier naturel. La donnée masquée M_M et le masque MM sont, par exemple, donnés par les formules suivantes :

[Math.11]

$$\begin{cases} M\_M = (\mathrm{K\_M}*\mathrm{L\_M} + \mathrm{MK}*\mathrm{ML}) \bmod 2^n \\ MM = (K\_M*ML + L\_M*MK) \bmod 2^n \end{cases}$$

[0097] D'autres expressions de la donnée masquée M_M et du masque MM peuvent être envisagées.

[0098] La figure 8 représente un circuit 51 mettant en oeuvre l'opération de multiplication. Le circuit 51 comprend :

- quatre circuits de multiplication 511, 512, 513 et 514 (X) ; et
- deux circuits d'addition 515 et 516 (+).

[0099] Le circuit de multiplication 511 reçoit, en entrée, la donnée masquée K_M et la donnée masquée L_M, et fournit en sortie la multiplication de ces données K_M*L_M.

[0100] Le circuit de multiplication 512 reçoit, en entrée, la donnée masquée K_M et le masque ML, et fournit, en sortie, le résultat K_M*ML de cette multiplication.

[0101] Le circuit de multiplication 513 reçoit, en entrée, la donnée masquée L_M et le masque MK, et fournit, en sortie, le résultat L_M*MK de cette multiplication.

[0102] Le circuit de multiplication 514 reçoit, en entrée, le masque MK et le masque ML, et fournit, en sortie, le résultat MK*ML de cette multiplication.

[0103] Le circuit d'addition 515 reçoit, en entrée, le résultat K_M*L_M de la multiplication du circuit 511 et le résultat MK*ML de la multiplication du circuit 514, et fournit, en sortie, la donnée masquée M_M.

[0104] Le circuit d'addition 516 reçoit, en entrée, le résultat K_M*ML de la multiplication du circuit 512 et le résultat L_M*MK de la multiplication du circuit 513, et fournit, en sortie, le masque MM.

[0105] Un avantage du circuit 51 est qu'il n'utilise pas les données démasquées K et L pour calculer la donnée masquée M_M et son masque MM.

[0106] Un autre avantage du circuit 51 est qu'il permet de mettre en oeuvre une opération de multiplication de deux données masquées en un seul cycle d'horloge.

[0107] La figure 9 représente un circuit 52 mettant en oeuvre l'opération de multiplication. Le circuit 52 comprend :

- trois sélecteurs 521, 522 et 523 ;
- un circuit de multiplication 524 (X) ;
- un circuit d'addition 525 (+) ;
- un circuit aiguilleur 526 ; et
- un registre 527 (REG).

[0108] Le sélecteur 521 reçoit, en entrée, la donnée masquée K_M et le masque MK, et fournit, en sortie, un premier mot binaire select1.

[0109] Le sélecteur 522 reçoit, en entrée, la donnée masquée L_M et le masque ML, et

fournit, en sortie, un deuxième mot binaire select2.

[0110] Le circuit de multiplication 524 reçoit, en entrée, les deux mots binaires select1 et select2, et fournit, en sortie, le résultat Prod de cette multiplication.

[0111] Le sélecteur 523 reçoit, en entrée, le mot binaire correspondant à zéro (0), appelée ensuite donnée nulle, et une donnée de temporisation Temp, et fournit, en sortie, un troisième mot binaire select3.

[0112] Le circuit d'addition 525 reçoit, en entrée, le résultat Prod de la multiplication et le mot binaire select3, et fournit, en sortie, le résultat Sum de cette addition.

[0113] Le circuit aiguilleur 526 reçoit, en entrée, le résultat Sum du circuit d'addition 525, et utilise, en sortie, ce résultat Sum pour former soit la donnée masquée M_M, le masque MM ou la donnée de temporisation Temp.

[0114] Le registre 527 reçoit, en entrée, la donnée de temporisation Temp, et fournit, en sortie, cette même donnée de temporisation Temp. Le registre 527 permet de stocker la donnée de temporisation Temp entre les différents cycles de l'opération.

[0115] Le circuit 52 permet de réaliser l'opération de multiplication en utilisant quatre cycles d'horloge. Des exemples d'étapes de calculs utilisés pendant les quatre cycles d'horloge sont les suivants.

[0116] A un premier cycle d'horloge, le sélecteur 521 sélectionne la donnée masquée K_M, et le mot binaire select1 est alors égal à la donnée masquée K_M. Le sélecteur 522 sélectionne la donnée masquée L_M, et le mot binaire select2 est alors égal à la donnée masquée L_M. Le circuit de multiplication 524 fournit le résultat Prod du produit des mots binaires select1 et select2. Le résultat Prod est alors égal au produit de la donnée masquée K_M par la donnée masquée L_M. Le sélecteur 523 sélectionne la donnée nulle, et le mot binaire select3 est alors égal à la donnée nulle. Le circuit d'addition 525 fournit le résultat Sum de la somme du produit Prod et du mot binaire select3. Le résultat Sum est alors égal au produit de la donnée masquée K_M par la donnée masquée L_M. Le circuit aiguilleur 526 fournit le résultat Sum à la donnée de temporisation Temp pour la stocker dans le registre 527.

[0117] A un deuxième cycle d'horloge, le sélecteur 521 sélectionne le masque MK, et le mot binaire select1 est alors égal au masque MK. Le sélecteur 522 sélectionne le masque ML, et le mot binaire select2 est alors égal au masque ML. Le circuit de multiplication 524 fournit le résultat Prod du produit des mots binaires select1 et select2. Le résultat Prod est alors égal au produit du masque MK et du masque ML. Le sélecteur 523 sélectionne la donnée de temporisation Temp à partir du registre 527, et le mot binaire select3 est alors égal à la donnée de temporisation Temp. Le circuit d'addition 525 fournit le résultat Sum de la somme du produit Prod et du mot binaire select3. Le résultat Sum est alors égal à :

[Math.12]

$$Sum = K\_M*L\_M + MK*ML$$

[0118] Le circuit aiguilleur 526 fournit le résultat Sum à la donnée masquée M_M.

[0119] A un troisième cycle d'horloge, le sélecteur 521 sélectionne la donnée masquée K_M, et le mot binaire select1 est alors égal à la donnée masquée K_M. Le sélecteur 522 sélectionne le masque ML, et le mot binaire select2 est alors égal au masque ML. Le circuit de multiplication 524 fournit le résultat Prod du produit des mots binaires select1 et select2. Le résultat Prod est alors égal au produit de la donnée masquée K_M par le masque ML. Le sélecteur 523 sélectionne la donnée nulle, et le mot binaire select3 est alors égal à la donnée nulle. Le circuit d'addition 525 fournit le résultat Sum de la somme du produit Prod et du mot binaire select3. Le résultat Sum est alors égal au produit de la donnée masquée K_M par le masque ML. Le circuit aiguilleur 526 fournit le résultat Sum à la donnée de temporisation Temp pour la stocker dans le registre 527.

[0120] A un quatrième cycle d'horloge, le sélecteur 521 sélectionne la donnée masquée L_M, et le mot binaire select1 est alors égal à la donnée masquée L_M. Le sélecteur 522 sélectionne le masque MK, et le mot binaire select2 est alors égal au masque MK. Le circuit de multiplication 524 fournit le résultat Prod du produit des mots binaires select1 et select2. Le résultat Prod est alors égal au produit de la donnée masquée L_M et du masque MK. Le sélecteur 523 sélectionne la donnée de temporisation Temp, et le mot binaire select3 est alors égal à la donnée de temporisation Temp. Le circuit d'addition 525 fournit le résultat Sum de la somme du produit Prod et du mot binaire select3. Le résultat Sum est alors égal à :

[Math.13]

$$Sum = K\_M*ML + L\_M*MK$$

[0121] Le circuit aiguilleur 526 fournit le résultat Sum au masque MM.

[0122] Un avantage du circuit 52 est qu'il permet de mettre en oeuvre une opération de multiplication en utilisant un seul circuit de multiplication et un seul circuit d'addition. Le circuit 52 peut alors présenter un encombrement moindre.

[0123] La figure 10 représente, schématiquement et sous forme de blocs, un mode de réalisation d'un circuit logique adapté à mettre en oeuvre une opération de multiplication avec accumulation (Multiply ACcumulate operation, MAC). La figure 10 représente plus particulièrement un circuit 53.

[0124] L'opération de multiplication avec accumulation prend en entrée, les deux données masquées K_M et L_M et leurs masques respectifs MK et ML, et une donnée masquée N_M et son masque MN. L'opération fournit en sortie la donnée masquée M_M et son masque MM. La donnée N_M est le résultat du masquage arithmétique d'une donnée N, par le masque MN. Les mots binaires N_M, N et MN sont des mots binaires de n

bits, n étant un entier naturel. L'opération de multiplication avec accumulation est définie par la formule suivante :

[Math.14]

$$M = K*L + N$$

[0125] La donnée M_M et le masque MM sont donnés par les formules suivantes :

[Math.15]

$$\begin{cases} M\_M = (K\_M*L\_M + MK*ML + N\_M) \bmod 2^n \\ MM = (K\_M*ML + L\_M*MK + MN) \bmod 2^n \end{cases}$$

[0126] D'autres expressions de la donnée masquée M_M et du masque MM pourront être envisagées.

[0127] Le circuit 53 de la figure 10 comprend des éléments communs avec le circuit 52 de la figure 9. Dans la suite, ces éléments communs ne seront pas détaillés à nouveau, et seules les différences entre les deux circuits 52 et 53 seront mises en exergue. Le circuit 52 comprend :

- quatre sélecteurs 521, 522, 523, et 531 ;
- le circuit de multiplication 524 ;
- le circuit d'addition 525 ;
- le circuit aiguilleur 526 ; et
- le registre 527.

[0128] La différence entre le circuit 53 et le circuit 52, est que, dans le circuit 53, le sélecteur 523 reçoit, non pas la donnée nulle, mais un mot binaire select4 et la donnée de temporisation Temp.

[0129] Le mot binaire select4 est une sortie du sélecteur 531. Le sélecteur 531 reçoit, en entrée, la donnée masquée N_M et son masque MN.

[0130] Le circuit 53 permet de réaliser l'opération de multiplication avec accumulation en utilisant quatre cycles d'horloge. Des exemples d'étapes de calculs utilisés pendant les quatre cycles d'horloge sont les suivants.

[0131] A un premier cycle d'horloge, le sélecteur 521 sélectionne la donnée masquée K_M, et le mot binaire select1 est alors égal à la donnée masquée K_M. Le sélecteur 522 sélectionne la donnée masquée L_M, et le mot binaire select2 est alors égal à la donnée masquée L_M. Le circuit de multiplication 524 fournit le résultat Prod du produit des mots binaires select1 et select2. Le résultat Prod est alors égal au produit de la donnée masquée K_M par la donnée masquée L_M. Le sélecteur 531 sélectionne la donnée masquée N_M, et le mot binaire select4 est alors égal à la donnée masquée N_M. Le sélecteur 523 sélectionne le mot binaire select4, et le mot binaire select3 est alors égal à la donnée masquée N_M. Le circuit d'addition 525 fournit le résultat Sum de la somme du produit Prod et du mot binaire select3. Le résultat Sum est alors donnée par

la formule suivante :

[Math.16]

$$Sum = K\_M*L\_M + N\_M$$

[0132] Le circuit aiguilleur 526 fournit le résultat Sum à la donnée de temporisation Temp en le stockant dans le registre 527.

[0133] A un deuxième cycle d'horloge, le sélecteur 521 sélectionne le masque MK, et le mot binaire select1 est alors égal au masque MK. Le sélecteur 522 sélectionne le masque ML, et le mot binaire select2 est alors égal au masque ML. Le circuit de multiplication 524 fournit le résultat Prod du produit des mots binaires select1 et select2. Le résultat Prod est alors égal au produit du masque MK et du masque ML. Le sélecteur 523 sélectionne la donnée de temporisation Temp stockée dans le registre 527, et le mot binaire select3 est alors égal à la donnée de temporisation Temp. Le circuit d'addition 525 fournit le résultat Sum de la somme du produit Prod et du mot binaire select3. Le résultat Sum est alors égal à :

[Math.17]

$$Sum = K\_M*L\_M + MK*ML + N\_M$$

[0134] Le circuit aiguilleur 526 fournit le résultat Sum à la donnée masquée M_M.

[0135] A un troisième cycle d'horloge, le sélecteur 521 sélectionne la donnée masquée K_M, et le mot binaire select1 est alors égal à la donnée masquée K_M. Le sélecteur 522 sélectionne le masque ML, et le mot binaire select2 est alors égal au masque ML. Le circuit de multiplication 524 fournit le résultat Prod du produit des mots binaires select1 et select2. Le résultat Prod est alors égal au produit de la donnée masquée K_M par le masque ML. Le sélecteur 531 sélectionne le masque MN, et le mot binaire select4 est alors égal au masque MN. Le sélecteur 523 sélectionne le mot binaire select4, et le mot binaire select3 est alors égal au masque MN. Le circuit d'addition 525 fournit le résultat Sum de la somme du produit Prod et du mot binaire select3. Le résultat Sum est alors égal à :

[Math.18]

$$Sum = K\_M*ML + MN$$

[0136] Le circuit aiguilleur 526 fournit le résultat Sum à la donnée de temporisation Temp.

[0137] A un quatrième cycle d'horloge, le sélecteur 521 sélectionne la donnée masquée L_M, et le mot binaire select1 est alors égal à la donnée masquée L_M. Le sélecteur 522 sélectionne le masque MK, et le mot binaire select2 est alors égal au masque MK. Le circuit de multiplication 524 fournit le résultat Prod du produit des mots binaires select1 et select2. Le résultat Prod est alors égal au produit de la donnée masquée L_M et du masque MK. Le sélecteur 523 sélectionne la donnée de temporisation Temp, et le mot binaire select3 est alors égal à la donnée de temporisation Temp. Le circuit d'addition 525 fournit le résultat Sum de la somme du produit Prod et du mot binaire

select3. Le résultat Sum est alors égal à :

[Math.19]

$$Sum = K\_M*ML + L\_M*MK + MN$$

[0138] Le circuit aiguilleur 526 fournit le résultat Sum au masque MM.

[0139] A titre de variante, le sélecteur 523 pourrait recevoir, en outre, la donnée nulle, pour pouvoir réaliser une opération de multiplication comme celle décrite en relation avec la figure 9.

[0140] Un avantage du circuit 53 est qu'il permet de mettre en oeuvre une opération de multiplication avec accumulation en utilisant un seul circuit de multiplication et un seul circuit d'addition. Le circuit 53 peut alors présenter un encombrement moindre.

[0141] Un autre avantage du circuit 53 est qu'il permet de réaliser une opération de multiplication avec accumulation sans utiliser les données démasquées K, L et N.

[0142] Les figures 11 à 13 représentent des exemples de circuits adaptés à mettre en oeuvre des opérations logiques des circuits 1043 décrits en relation avec la figure 2.

[0143] Dans les exemples décrits dans ces figures, l'opération de masquage logique est identique à celle décrite en relation avec les figures 3 et 4.

[0144] La figure 11 représente, schématiquement et sous forme de blocs, un circuit logique 61 (OR) adapté à mettre en oeuvre une opération logique OU bit à bit.

[0145] Dans l'exemple représenté en figure 11, l'opération logique OU bit à bit prend, en entrée, deux données masquées O_M et P_M, et leurs masques respectifs MO et MP, et fournit, en sortie, une donnée masquée Q_M et son masque MQ. La donnée masquée O_M, respectivement P_M, Q_M, est le résultat du masquage logique d'une donnée O, respectivement P, Q, par un masque MO, respectivement MP, MQ. La donnée masquée Q_M et son masque MQ sont donnés par les relations, exécutés bit à bit, suivantes :

[Math.20]

$$\begin{cases} Q\_M = \ !O\_M.P\_M.MP \ or \ !P\_M.MO.MP \ or \ P\_M.!MO.!MP \ or \ O\_M.!P\_M.!MP \\ MQ = MO \ xor \ MP \end{cases}$$

dans lequel "or" désigne l'opération logique OU bit à bit.

[0146] Le circuit 61 peut être mis en oeuvre en utilisant, par exemple, une porte OU à quatre entrées, et quatre portes ET à trois entrées.

[0147] Un avantage du circuit 61 est qu'il n'utilise pas les données démasquées O et P pour calculer la donnée masquée Q_M.

[0148] La figure 12 représente, schématiquement et sous forme de blocs, un circuit logique 62 (AND) adapté à mettre en oeuvre une opération logique ET bit à bit.

[0149] Dans l'exemple représenté en figure 12, l'opération logique ET bit à bit prend, en entrée, deux données masquées R_M et S_M, et leurs masques respectifs MR et MS, et fournit, en sortie, une donnée masquée T_M et son masque MT. La donnée masquée

R_M, respectivement S_M, T_M, est le résultat du masquage logique d'une donnée R, respectivement S, T, par un masque MR, respectivement MS, MT. La donnée masquée T_M et son masque MT sont donnés par les relations suivantes :

[Math.21]

$$\begin{cases} T\_M = \ !R\_M.!R\_M.MS \ or \ !S\_M.MR.!MS \ or \ S\_M.!MR.MS \ or \ R\_M.S\_M.!MS \\ MT = MR \ xor \ MS \end{cases}$$

[0150] Le circuit 62 peut être mis en oeuvre en utilisant, par exemple, une porte OU à quatre entrées, et quatre portes ET à trois entrées. A titre d'exemple, les circuits 61 et 62 des figures 11 et 12 peuvent être mis en oeuvre par des circuits similaires.

[0151] Un avantage du circuit 62 est qu'il n'utilise pas les données démasquées R et S pour calculer la donnée masquée T_M.

[0152] La figure 13 représente, schématiquement et sous forme de blocs, un circuit logique 63 (XOR) adapté à mettre en oeuvre une opération logique OU EXCLUSIF bit à bit.

[0153] Dans l'exemple représenté en figure 13, l'opération logique OU EXCLUSIF bit à bit prend, en entrée, deux données masquées U_M et V_M, et leurs masques MU et MV et fournit, en sortie, une donnée masquée W_M, et son masque MW. La donnée masquée U_M, respectivement V_M, W_M, est le résultat du masquage logique d'une donnée U, respectivement V, W, par un masque MU, respectivement MV, MW. La donnée masquée W_M et son masque MW sont donnés par les relations suivantes :

[Math.22]

$$\begin{cases} W\_M = U\_M \ xor \ V\_M \\ MW = MU \ xor \ MV \end{cases}$$

[0154] Un avantage du circuit 62 est qu'il n'utilise pas les données démasquées U et V pour calculer la donnée masquée W_M.

[0155] Divers modes de réalisation et variantes ont été décrits. L'homme de l'art comprendra que certaines caractéristiques de ces divers modes de réalisation et variantes pourraient être combinées, et d'autres variantes apparaitront à l'homme de l'art. En particulier, d'autres opérations arithmétiques et logiques pourront être envisagées. De plus, les exemples de masquage logique et arithmétique évoqués dans la description ne sont que des exemples, d'autres opérations de masquage pourront être envisagées.

[0156] De plus, un autre exemple d'une partie de l'architecture de l'unité arithmétique et logique 104 pourrait être de comprendre un premier circuit de masquage devant les circuits mettant en oeuvre des opérations logiques et un deuxième circuit de masquage devant les circuits mettant en oeuvre les opérations arithmétiques, chaque circuit de masquage permettant d'adapter le type de masquage avant l'opération demandée.

[0157] Enfin, la mise en oeuvre pratique des modes de réalisation et variantes décrits est à la portée de l'homme du métier à partir des indications fonctionnelles données ci-dessus.

Revendications

[Revendication 1] Procédé de traitement de données masquées (Data_M, Data_S) avec un premier type de masquage ou un deuxième type de masquage, mis en oeuvre par un processeur (100) comportant une unité (104) arithmétique et logique comprenant un circuit de masquage (1041, 1044) adapté à :
- modifier le type de masquage d'une donnée masquée en le premier type de masquage avant l'application à ladite donnée masquée d'une opération arithmétique ; et
- modifier le type de masquage d'une donnée masquée en le deuxième type de masquage avant l'application à ladite donnée masquée d'une une opération logique.

[Revendication 2] Procédé selon la revendication 1, dans lequel le premier type de masquage n'utilise que des opérations arithmétiques.

[Revendication 3] Procédé selon la revendication 2, dans lequel une donnée masquée par le premier type de masquage est égale à l'addition d'une donnée à masquer et d'un masque.

[Revendication 4] Procédé selon l'une quelconque des revendications 1 à 3, dans lequel le deuxième type de masquage n'utilise que des opérations logiques.

[Revendication 5] Procédé selon la revendication 4, dans lequel une donnée masquée par le deuxième type de masquage est égale à l'application de l'opération OU EXCLUSIF bit à bit entre la donnée à masquer et le masque.

[Revendication 6] Procédé selon l'une quelconque des revendications 1 à 5, dans lequel le processeur (100) comprend, en outre, une unité (104) de génération d'adresses, dans laquelle lesdites données masquées (Data_M, Data_S) restent masquées pendant leur traitement dans ladite unité de génération d'adresses.

[Revendication 7] Procédé selon l'une quelconque des revendications 1 à 6, dans lequel le processeur (100) comprend, en outre, des banques de registres (101, 102), dans lesquelles lesdites données masquées (Data_M, Data_S) sont stockées et restent masquées pendant tout leur stockage.

[Revendication 8] Procédé selon la revendication 7, dans lequel les banques de registres (101, 102) comprennent au moins une banque de registres (101) pour lesdites données (Data_M, Data_S) et au moins une banque de registres (102) pour des masques desdites données (Mask_D, Mask_S).

[Revendication 9] Procédé selon l'une quelconque des revendications 1 à 8, dans lequel le processeur (100) comprend, en outre, des registres (105, 106) stockant des pointeurs masqués.

[Revendication 10] Procédé selon la revendication 9, dans lequel lesdits registres (105, 106) stockent au moins un pointeur de registre masqué et au moins un pointeur de programme masqué.

[Revendication 11] Procédé selon la revendication 9 ou 10, dans lequel lesdits registres (105, 106) comprennent des pointeurs (Point_D) référençant des données, et des masques de pointeurs (Mask_P).

[Revendication 12] Processeur (100) adapté à mettre en oeuvre le procédé selon l'une quelconque des revendications 1 à 11.

[Fig. 1]

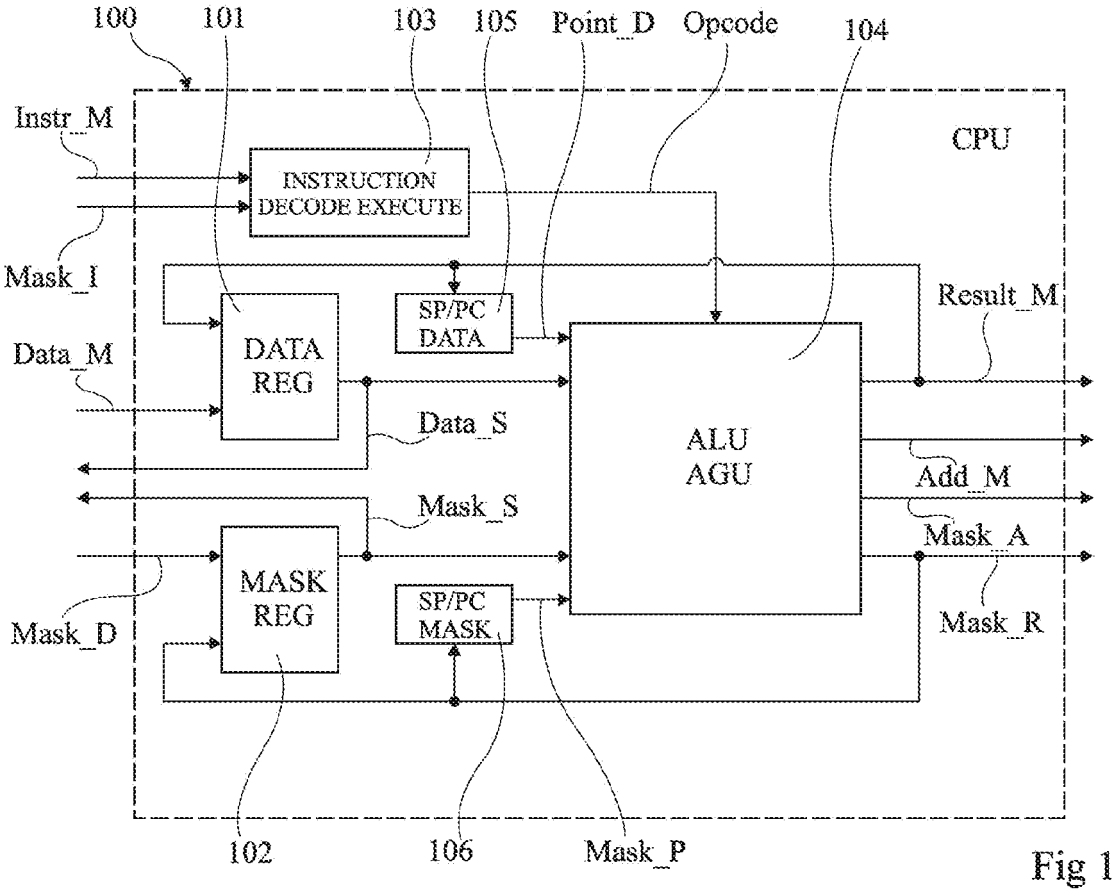

Fig 1

[Fig. 2]

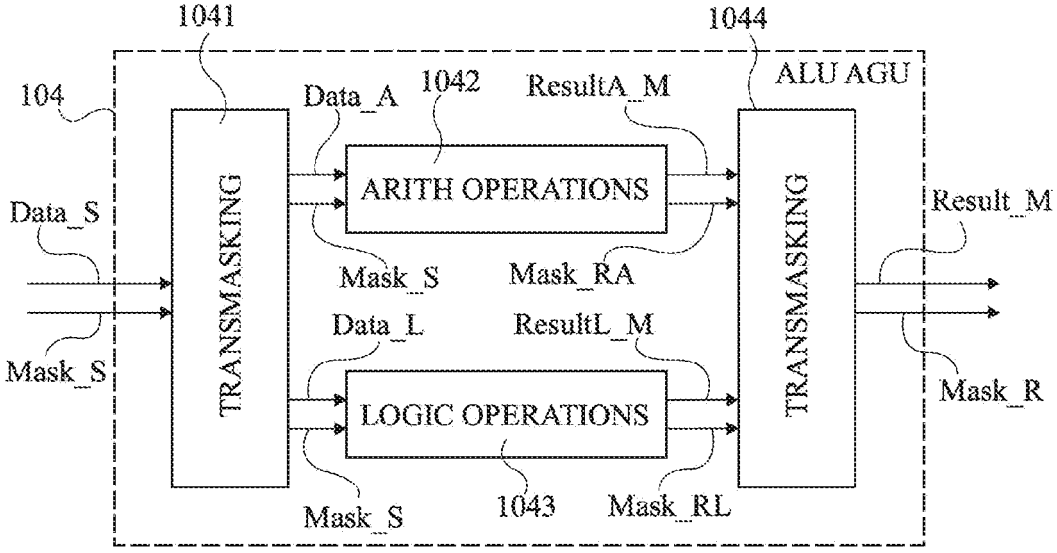

Fig 2

[Fig. 3]

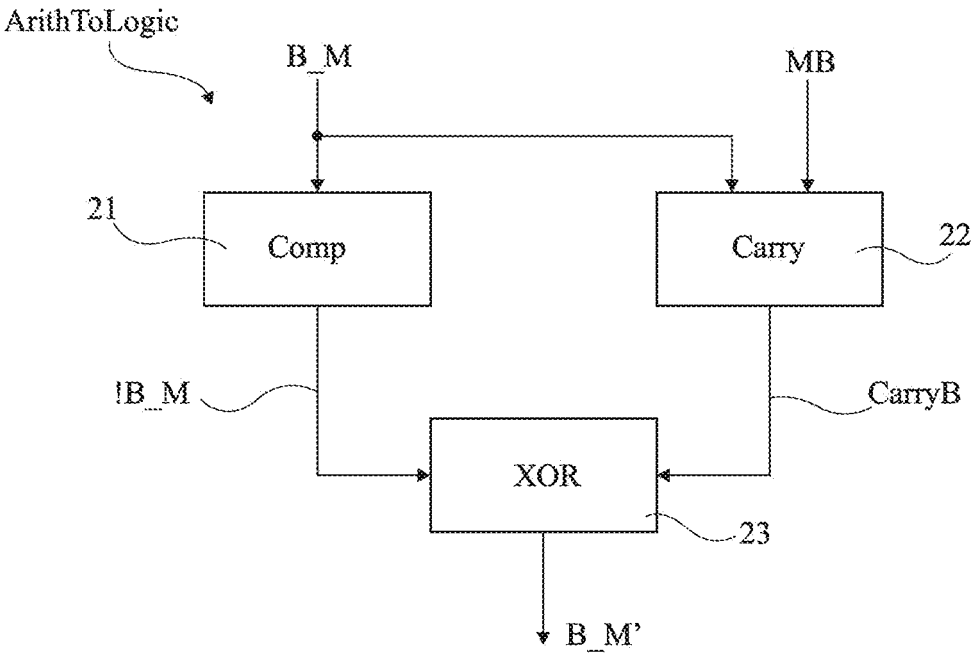

Fig 3

[Fig. 4]

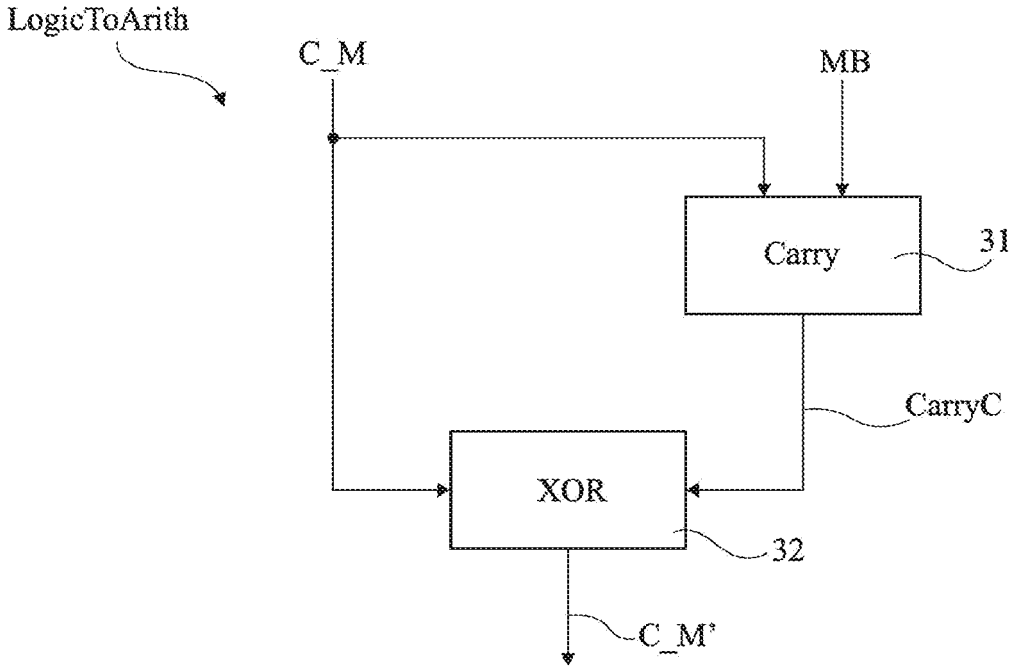

Fig 4

[Fig. 5]

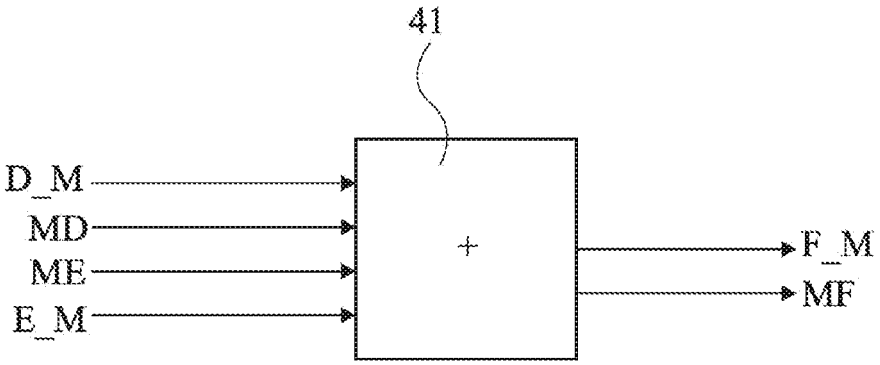

Fig 5

[Fig. 6]

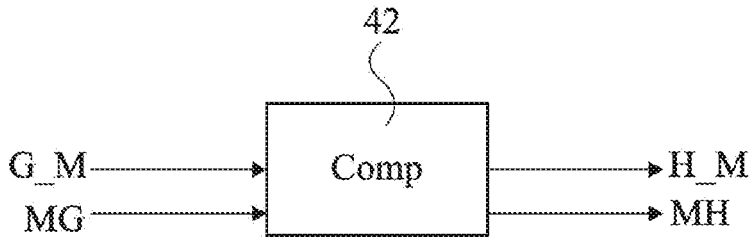

Fig 6

[Fig. 7]

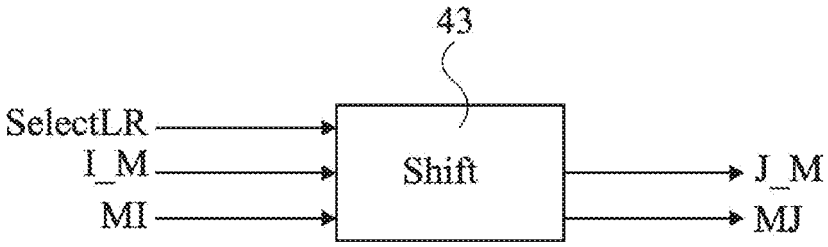

Fig 7

[Fig. 8]

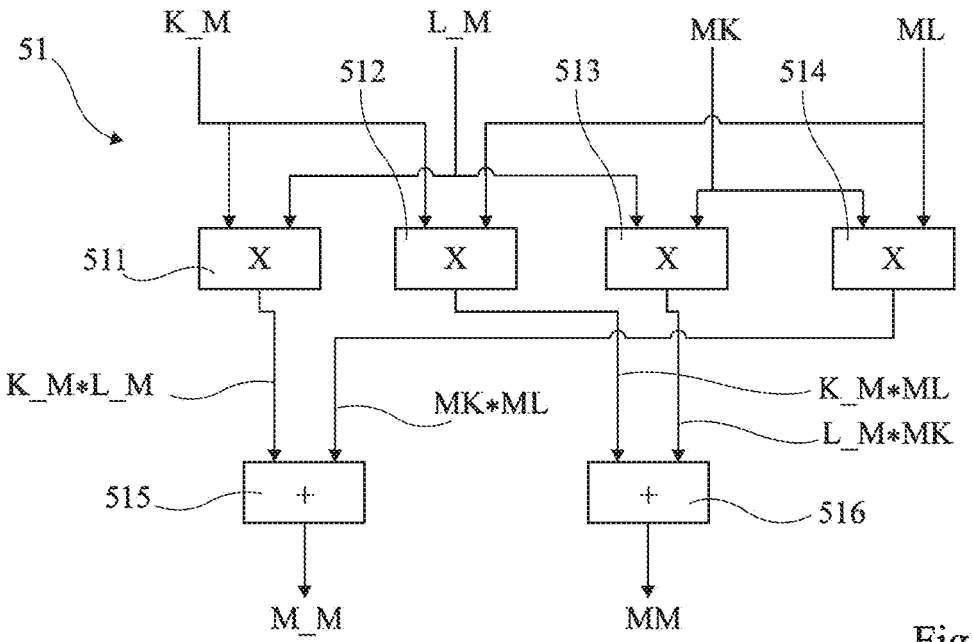

Fig 8

[Fig. 9]

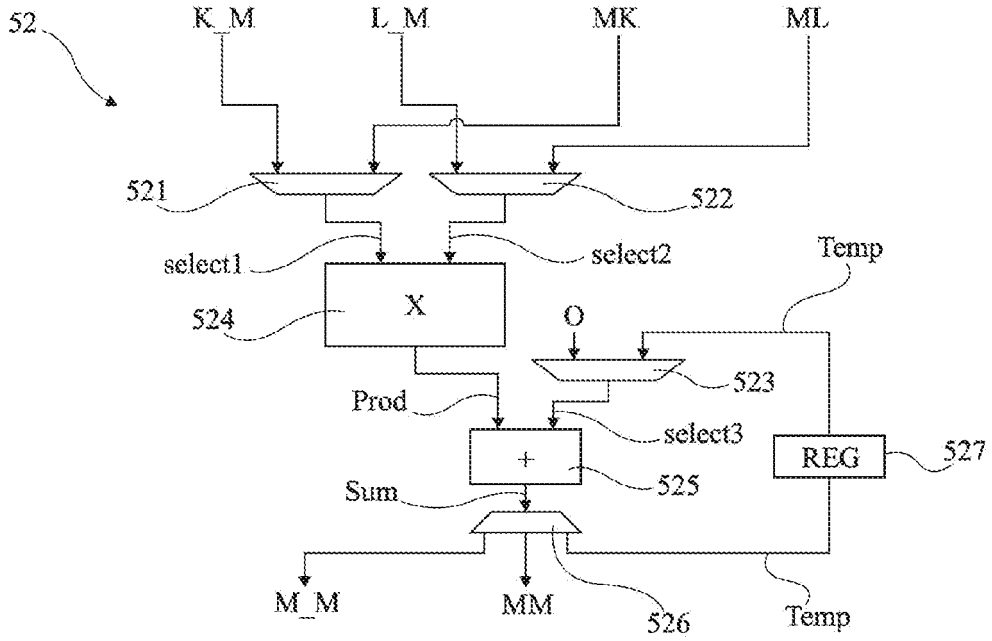

Fig 9

[Fig. 10]

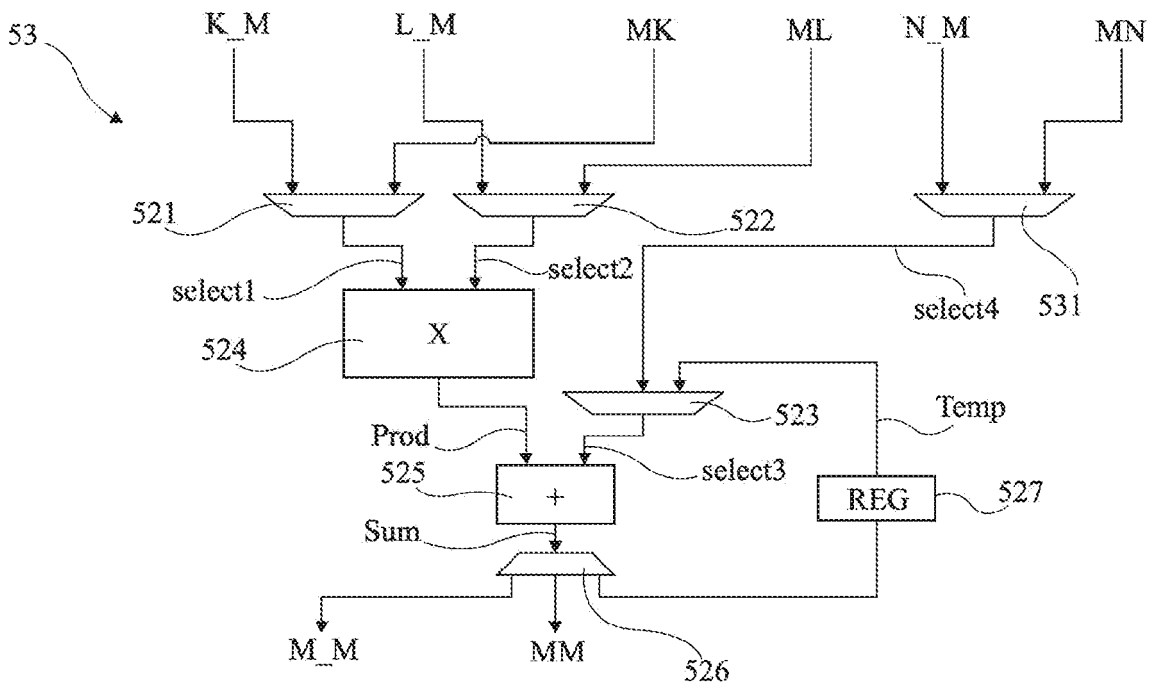

Fig 10

[Fig. 11]

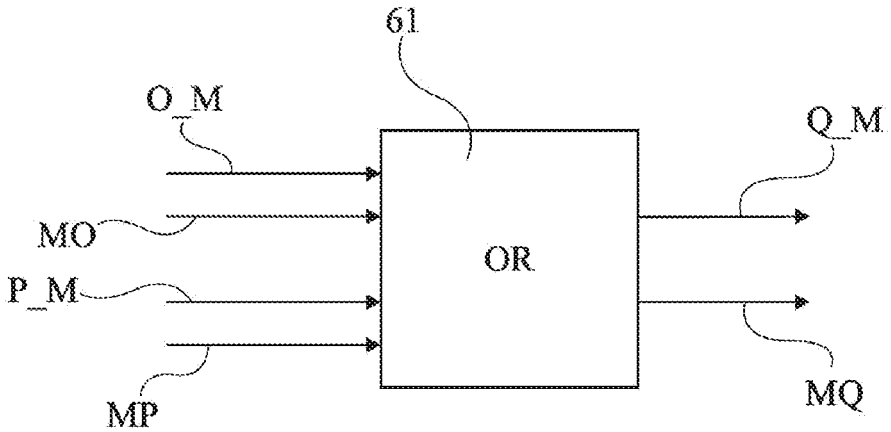

Fig 11

[Fig. 12]

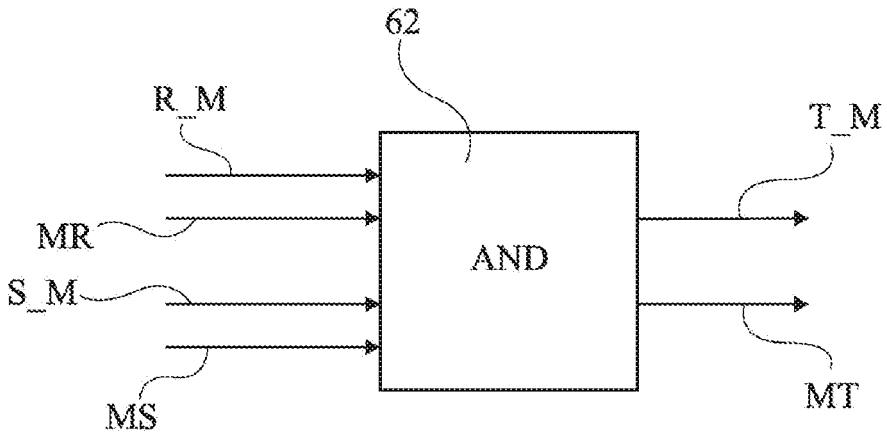

Fig 12

[Fig. 13]

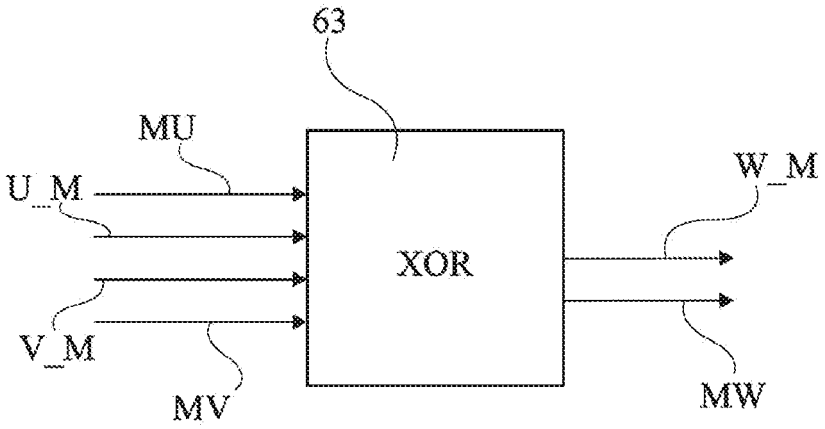

Fig 13

RAPPORT DE RECHERCHE

articles L.612-14, L.612-53 à 69 du code de la propriété intellectuelle

OBJET DU RAPPORT DE RECHERCHE

L'I.N.P.I. annexe à chaque brevet un "RAPPORT DE RECHERCHE" citant les éléments de l'état de la technique qui peuvent être pris en considération pour apprécier la brevetabilité de l'invention, au sens des articles L. 611-11 (nouveauté) et L. 611-14 (activité inventive) du code de la propriété intellectuelle. Ce rapport porte sur les revendications du brevet qui définissent l'objet de l'invention et délimitent l'étendue de la protection.
Après délivrance, l'I.N.P.I. peut, à la requête de toute personne intéressée, formuler un "AVIS DOCUMENTAIRE" sur la base des documents cités dans ce rapport de recherche et de tout autre document que le requérant souhaite voir prendre en considération.

CONDITIONS D'ETABLISSEMENT DU PRESENT RAPPORT DE RECHERCHE

[x] Le demandeur a présenté des observations en réponse au rapport de recherche préliminaire.

☐ Le demandeur a maintenu les revendications.

[x] Le demandeur a modifié les revendications.

☐ Le demandeur a modifié la description pour en éliminer les éléments qui n'étaient plus en concordance avec les nouvelles revendications.

☐ Les tiers ont présenté des observations après publication du rapport de recherche préliminaire.

☐ Un rapport de recherche préliminaire complémentaire a été établi.

DOCUMENTS CITES DANS LE PRESENT RAPPORT DE RECHERCHE

La répartition des documents entre les rubriques 1, 2 et 3 tient compte, le cas échéant, des revendications déposées en dernier lieu et/ou des observations présentées.

[x] Les documents énumérés à la rubrique 1 ci-après sont susceptibles d'être pris en considération pour apprécier la brevetabilité de l'invention.

[x] Les documents énumérés à la rubrique 2 ci-après illustrent l'arrière-plan technologique général.

☐ Les documents énumérés à la rubrique 3 ci-après ont été cités en cours de procédure, mais leur pertinence dépend de la validité des priorités revendiquées.

☐ Aucun document n'a été cité en cours de procédure.

1. ELEMENTS DE L'ETAT DE LA TECHNIQUE SUSCEPTIBLES D'ETRE PRIS EN CONSIDERATION POUR APPRECIER LA BREVETABILITE DE L'INVENTION

GOUBIN L ED - KOC C K ET AL: "A SOUND METHOD FOR SWITCHING BETWEEN BOOLEAN AND ARITHMETIC MASKING",
CRYPTOGRAPHIC HARDWARE AND EMBEDDED SYSTEMS. 3RD INTERNATIONAL WORKSHOP, CHES 2001, PARIS, FRANCCE, MAY 14 - 16, 2001 PROCEEDINGS; [LECTURE NOTES IN COMPUTER SCIENCE], BERLIN : SPRINGER, DE,
vol. VOL. 2162, 14 mai 2001 (2001-05-14),
pages 3-15, XP001061156,
ISBN: 978-3-540-42521-2

US 2015/278555 A1 (MELZANI FILIPPO [IT])
1 octobre 2015 (2015-10-01)

US 2008/040414 A1 (KUENEMUND THOMAS [DE])
14 février 2008 (2008-02-14)

2. ELEMENTS DE L'ETAT DE LA TECHNIQUE ILLUSTRANT L'ARRIERE-PLAN TECHNOLOGIQUE GENERAL

NAKATSU DAISUKE ET AL: "Combination of SW Countermeasure and CPU Modification on FPGA against Power Analysis",
24 août 2010 (2010-08-24), INTERNATIONAL CONFERENCE ON FINANCIAL CRYPTOGRAPHY AND DATA SECURITY; [LECTURE NOTES IN COMPUTER SCIENCE; LECT.NOTES COMPUTER], SPRINGER, BERLIN, HEIDELBERG, PAGE(S) 258 - 272,
XP047428751,
ISBN: 978-3-642-17318-9

ELKE DE MULDER ET AL: "Protecting RISC-V against Side-Channel Attacks",
20190602; 20190602 - 20190606,
2 juin 2019 (2019-06-02), pages 1-4,
XP058435200,
DOI: 10.1145/3316781.3323485
ISBN: 978-1-4503-6725-7

3. ELEMENTS DE L'ETAT DE LA TECHNIQUE DONT LA PERTINENCE DEPEND DE LA VALIDITE DES PRIORITES

NEANT