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

(19) Organisation Mondiale de la Propriété Intellectuelle
Bureau international

(43) Date de la publication internationale
23 septembre 2010 (23.09.2010)

PCT

(10) Numéro de publication internationale
WO 2010/105889 A1

(51) Classification internationale des brevets :
G06F 9/50 (2006.01) *G06F 9/46* (2006.01)

(21) Numéro de la demande internationale :
PCT/EP2010/052215

(22) Date de dépôt international :
22 février 2010 (22.02.2010)

(25) Langue de dépôt : français

(26) Langue de publication : français

(30) Données relatives à la priorité :
09/00833 24 février 2009 (24.02.2009) FR

(71) Déposant *(pour tous les États désignés sauf US)* : **COMMISSARIAT A L'ENERGIE ATOMIQUE ET AUX ENERGIES ALTERNATIVES** [FR/FR]; 25 rue Leblanc, Bâtiment "Le Ponant D", F-75015 Paris (FR).

(72) Inventeurs; et

(75) Inventeurs/Déposants *(pour US seulement)* : **LOUISE, Stéphane** [FR/FR]; 24 avenue des Pierrots, F-91400 Orsay (FR). **DAVID, Vincent** [FR/FR]; 2 Chemin du Collège, F-91460 Marcoussis (FR). **DAVID, Raphaël** [FR/FR]; 17 Rue Charles de Gaulle, F-91440 Bures Sur Yvette (FR).

(74) Mandataires : **LUCAS, Laurent** et al.; Immeuble Visium, 22, Avenue Aristide, F-94117 Arcueil (FR).

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

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

Publiée :

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

[Suite sur la page suivante]

(54) **Title** : ALLOCATION AND MONITORING UNIT

(54) **Titre** : UNITÉ D'ALLOCATION ET DE CONTRÔLE

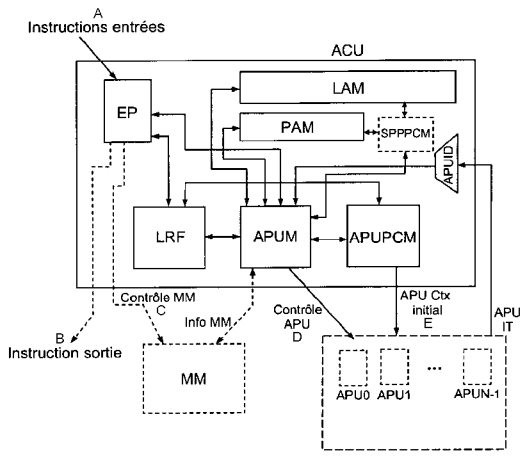

FIG.2

A Instructions in
B Instructions out
C MM control
D APU Control
E APU Starting Ctx

(57) **Abstract** : The present invention relates to an allocation and monitoring unit for allocating threads of a task to a plurality of auxiliary processing units and for monitoring the parallel execution of said threads by said auxiliary processing units, the task being executed in a sequential manner by a main processing unit. The allocation and monitoring unit comprises: a means for managing virtual auxiliary processing units; a means for managing physical auxiliary processing units, each physical auxiliary processing unit corresponding to an auxiliary processing unit; and a means for managing the auxiliary processing units. The means for managing the auxiliary processing units comprises: a means for allocating a virtual auxiliary processing unit to a thread to be executed; a means for managing the correlation between the virtual auxiliary processing units and the physical auxiliary processing units, such that the auxiliary processing units execute the threads of the task in parallel by means of the virtual auxiliary processing units, which are allocated as late as possible and released as soon as possible. The invention is useful for onboard systems with high processing power.

(57) **Abrégé** : La présente invention concerne une unité d'allocation et de contrôle pour allouer des fils d'exécution d'une tâche à une pluralité d'unités auxiliaires de traitement et pour contrôler l'exécution en parallèle desdits fils d'exécution par lesdites unités auxiliaires de traitement,

[Suite sur la page suivante]

WO 2010/105889 A1

la tâche étant exécutée de manière séquentielle par une unité principale de traitement. L'unité d'allocation et de contrôle comporte: des moyens pour gérer des unités auxiliaires de traitement logiques, des moyens pour gérer des unités auxiliaires de traitement physiques, chaque unité auxiliaire de traitement physique correspondant à une unité auxiliaire de traitement, des moyens pour gérer les unités auxiliaires de traitement. Les moyens pour gérer les unités auxiliaires de traitement comportent: des moyens pour allouer une unité auxiliaire de traitement logique à un fil d'exécution à exécuter, des moyens pour gérer la correspondance entre les unités auxiliaires de traitement logiques et les unités auxiliaires de traitement physiques. De sorte que les unités auxiliaires de traitement exécutent en parallèle les fils d'exécution de la tâche par l'intermédiaire des unités auxiliaires de traitement logiques, qui sont allouées au plus tard et libérées au plus tôt. Application: systèmes embarqués à forte puissance de calcul.

Unité d'allocation et de contrôle

La présente invention concerne une unité d'allocation et de contrôle pour allouer des fils d'exécution d'une tâche à une pluralité d'unités auxiliaires de traitement et pour contrôler l'exécution en parallèle desdits fils d'exécution par lesdites unités auxiliaires de traitement, la tâche étant exécutée de manière séquentielle par une unité principale de traitement. Elle s'applique notamment dans le domaine des systèmes embarqués à forte puissance de calcul.

Les limites des processeurs superscalaires ne sont plus technologiques: ce sont les limites du microparallélisme d'instructions. En effet, les dépendances de données forment une barrière infranchissable dans le cadre d'une exécution avec démarrage des tâches dans l'ordre et terminaison dans l'ordre. Plus on autorise à faire de l'exécution intermédiaire dans le désordre plus on doit mettre en place une logique de calcul de dépendance importante. D'un autre côté, le nombre moyen d'instructions exécutées par cycle (IPC moyen) progresse assez peu lorsqu'on augmente la fenêtre d'instructions au-delà de ce qui se fait actuellement.

De ce fait, l'architecture des systèmes de calcul subit actuellement des mutations importantes. En effet, même si à l'heure actuelle il n'y a pas de remise en question fondamentale de la fameuse « Loi de Moore », qui prédit la croissance exponentielle du nombre de transistors que l'on peut mettre en œuvre sur une puce de silicium à un instant donné, l'industrie des semi-conducteur se retrouve néanmoins devant un constat d'échec : il n'y a plus de pistes crédibles pour augmenter les performances des processeurs individuels de façon significative.

On sait néanmoins depuis les travaux de base en la matière, dans les années 1960, que le rapport entre la puissance de calcul et l'efficacité des systèmes de calculs est potentiellement beaucoup plus élevée pour les systèmes parallèles que pour les processeurs séquentiels. C'est pourquoi, à tous les niveaux, on se dirige de plus en plus vers des systèmes parallèles sur puce. Ils permettent en théorie d'utiliser de façon plus efficace les

transistors supplémentaires que l'on peut intégrer sur une même puce, du fait des progrès effectués dans les techniques de gravure.

Or, s'il est connu depuis longtemps que les systèmes parallèles sont plus efficaces que les systèmes séquentiels classiques, on pourrait se poser la question de savoir pourquoi cela ne s'est pas banalisé plus tôt, surtout dans le domaine des systèmes embarqués, qui est par principe très centré sur l'optimisation des différentes efficacités. Mais d'une part, la technologie ne permettait pas l'intégration de structures massivement parallèles sur un même composant, à l'exception des structures SIMD (« Single Instruction, Multiple Data ») facilement programmables. D'autre part, les systèmes parallèles sont beaucoup plus difficiles à programmer et à mettre au point de façon générale, surtout les systèmes symétriques basés sur la réplication du même élément de traitement et possédant des interfaces d'accès et de communication identiques et homogènes.

Dans le domaine des systèmes embarqués, notamment celui de la téléphonie mobile, des « multicores » sur une seule puce sont apparus, qui peuvent contenir des DSP (« Digital Signal Processor ») pour le traitement du signal, des GPP (« General Purpose Processor ») pour les traitements ordinaires, ainsi que des blocs analogiques d'entrées/sorties. Dans le domaine des baladeurs ou des lecteurs multimédias, des cœurs de décodage dédiés à l'audio (« MPEG Audio Layer », « Dolby D », « DTS ») ou à la vidéo (« MPEG », « H264 ») sont apparus en plus du processeur généraliste.

Ainsi, il existe d'ores et déjà dans l'art antérieur des modèles d'interaction entre un processeur généraliste et des coprocesseurs ou, d'une façon plus générique, entre un processeur principal et des unités auxiliaires de traitement. Par exemple, il existe depuis les années 1970 des unités d'accélération de traitement, en particulier pour les calculs mathématiques. Dans un certain nombre de cas, ces unités, nommées « coprocesseurs », sont distinctes du processeur dit « principal ». C'était le cas sur les processeurs pour micro-ordinateurs et stations de travail jusqu'à la fin des années 1980. Mais c'est encore le cas pour des systèmes embarqués, que ce soit pour augmenter le parallélisme par découplage potentiel des deux unités, ou que ce soit pour réduire les coûts. En effet, un processeur générique de faible coût est alors employé en conjonction avec une unité de

traitement spécifique séparée et généralement conçue « in house », comme c'est le cas du coprocesseur vectoriel « Fire » de Thomson.

Par exemple, le brevet américain US 6249858 montre un des aspects les plus récents concernant les capacités de couplage entre un processeur standard et un coprocesseur, en permettant une exécution parallèle des traitements sur les deux entités. Le couplage est assez étroit : le processeur principal envoie les ordres de calculs sur le coprocesseur en fournissant des opérandes et une adresse de programme en ROM. Cela nécessite cependant un logiciel de support dédié, car une interruption doit être prise sur le processeur principal pour gérer convenablement l'appel aux fonctionnalités du coprocesseur, et une autre interruption est générée par le coprocesseur à la fin du calcul. Il montre ainsi comment coupler faiblement le processeur principal et son accélérateur de calcul. Néanmoins la méthode n'est pas généralisable à une pluralité d'éléments d'accélération. De plus, elle ne permet pas de se passer d'un support système pour la commande et l'obtention des résultats des calculs du coprocesseur. Elle ne permet pas non plus d'assurer facilement la cohérence dans les dépendances de calculs. Ceux-ci sont a priori du ressort du programmeur, ce qui est en général difficile sur un système parallèle où les traitements peuvent être fortement hétérogènes. Cela rend également le passage à l'échelle extrêmement difficile et réservé aux spécialistes de la programmation parallèle.

Autre exemple, les GPU (Graphics Processing Unit) des cartes graphiques modernes décrites dans le brevet américain US 6987517 peuvent être considérés comme des ensembles d'unités auxiliaires spécialisées pour le calcul vectoriel à programme unique et données multiples (SPMD: « Single Program, Multiple Data »). Dans ce cas, il y a un couplage faible entre cette multitude d'unités et le processeur de contrôle, car le problème traité est massivement parallèle. En effet, il s'agit d'effectuer un même traitement sur des ensembles de données distinctes, pour calculer des pixels dans un tampon de mémoire. Mais il n'est pas important qu'il y ait une erreur à un moment donné, car du moment que le taux d'erreur reste faible, l'utilisateur n'est pas gêné. De plus, il n'y a pas de moyens de synchronisation simplement accessible, puisque le problème est intrinsèquement parallèle. La seule synchronisation importante se situe en fin

de traitement d'une image, afin d'ajouter des étages de post-traitements ou simplement d'afficher à l'écran les pixels calculés.

Autre exemple, la demande de brevet américain publiée sous le numéro US2008140989 (A1) décrit des procédés de distribution de traitements sur des unités auxiliaires. Mais les procédés décrits dans cette demande n'offrent pas de moyen simple pour gérer le parallélisme à plusieurs niveaux de façon automatique. Ils ne permettent notamment pas une cohabitation entre un parallélisme au niveau des tâches sur le processeur principal, dit parallélisme « à gros grain », et un parallélisme au niveau des « threads » sur les processeurs auxiliaires, dit parallélisme « à grain fin ». De plus, la gestion du déterminisme d'exécution repose sur une gestion du parallélisme par le programmeur, ce qui est en général difficile pour les applications typiques des systèmes embarqués.

De manière générale, lors de l'exécution d'une tâche, les solutions précitées de l'art antérieur laissent peu d'autonomie dans la gestion des unités auxiliaires de traitement, le logiciel système ayant souvent à intervenir pour l'exécution de la tâche. A contrario, si une tâche est bloquée, ces solutions de l'art antérieur n'autorisent pas la mise en œuvre du logiciel système pour la commutation de tâche, limitant par conséquent l'utilisation des différents parallélismes. Par conséquent, elles ne confèrent qu'un déterminisme d'exécution global bien éloigné de celui conféré par une architecture Von Neumann classique.

Ainsi, il apparaît clairement que les architectes de systèmes de calcul sont dans une relative impasse technologique entre un paradigme de monoprocesseur, qui montre ses limites, et les multiprocesseurs sur puce, plus connus sous l'acronyme anglo-saxon « MPSoC » signifiant « MultiProcessor System on Chip » ou sous l'acronyme anglo-saxon « CMP » signifiant « Chip MultiProcessing », qui sont difficiles à programmer. La plupart des architectures actuelles exploitent soit un parallélisme de tâches dit « parallélisme de traitements », soit un parallélisme d'instructions dit « micro-parallélisme d'instructions », ou alors une combinaison des deux pour les MPSoC.

Le parallélisme de traitements est le parallélisme d'applications ou de tâches. Bien que des systèmes de développement permettent de

programmer de tels systèmes au niveau applicatif, il est difficile d'utiliser plus d'une dizaine de processeurs pour une application standard. Il est bien entendu possible d'envisager un cadre multi-application, mais il reste alors le problème de gérer efficacement l'exécution au-delà de 8 ou 16 processeurs en configuration SMP (« Symmetric MultiProcessing »). De plus, les applications ordinaires nécessitent une réécriture et une reconception partielle pour en tirer partie, comme par exemple la mise en œuvre de « fils d'exécution », ou « threads » selon la terminologie anglo-saxonne, au standard POSIX (« Portable Operating System Interface »).

Le micro-parallélisme d'instructions est celui qui est utilisé dans les processeurs superscalaires pour exécuter plus d'une instruction par cycle. Mais comme explicité précédemment, on arrive aux limites de cette technologie en termes d'efficacité.

Les limitations évoquées précédemment peuvent conduire à explorer un nouveau niveau de parallélisme intermédiaire, qui pourrait être appelé « méso-parallélisme » ou parallélisme « à grain moyen ». Il s'agirait d'un parallélisme intermédiaire entre le parallélisme de traitement et le micro-parallélisme d'instructions. Différentes unités de calcul coopèreraient toujours pour exécuter en parallèle des séquences de code d'une même tâche. Mais cette fois, le programme principal et les problèmes de synchronisation seraient mis en œuvre sur un processeur de contrôle, alors que les sections de calculs intensifs seraient mises en œuvre sur des processeurs spécialisés.

Mais implémenter un tel méso-parallélisme n'est pas sans poser de nombreuses difficultés. Car d'un côté, pour sortir des limitations du micro-parallélisme d'instruction, il est nécessaire de s'affranchir autant que possible d'une exécution dans l'ordre du programme. Alors que d'un autre côté, il faut que le code applicatif soit aussi proche que possible d'un code de type séquentiel connu du programmeur.

S'appuyant sur une architecture faisant l'objet du brevet français numéro FR2893156 (B1) dont la demanderesse est titulaire, la présente invention a notamment pour but de pallier les inconvénients précités, en exploitant des points de synchronisation mis en place à la compilation et en

utilisant autant que possible les ressources mises à disposition par le matériel et par le logiciel système de base. A cet effet, l'invention a pour objet une unité d'allocation et de contrôle pour allouer des fils d'exécution d'une tâche à une pluralité d'unités auxiliaires de traitement et pour contrôler l'exécution en parallèle desdits fils d'exécution par lesdites unités auxiliaires de traitement, la tâche étant exécutée de manière séquentielle par une unité principale de traitement. L'unité d'allocation et de contrôle selon l'invention comporte des moyens pour gérer des unités auxiliaires de traitement logiques, des moyens pour gérer des unités auxiliaires de traitement physiques, chaque unité auxiliaire de traitement physique correspondant à une unité auxiliaire de traitement et des moyens pour gérer les unités auxiliaires de traitement. Les moyens pour gérer les unités auxiliaires de traitement comportent des moyens pour allouer une unité auxiliaire de traitement logique à un fil d'exécution à exécuter et des moyens pour gérer la correspondance entre les unités auxiliaires de traitement logiques et les unités auxiliaires de traitement physiques. Ainsi, les unités auxiliaires de traitement exécutent en parallèle les fils d'exécution de la tâche par l'intermédiaire des unités auxiliaires de traitement logiques, qui sont allouées au plus tard et libérées au plus tôt.

Dans un mode de réalisation préférentiel, l'unité peut comporter des moyens pour exécuter des instructions insérées dans la tâche, ces instructions fournissant des directives de gestion des fils d'exécution exécutables par les unités auxiliaires de traitement logiques. Ces instructions insérées peuvent inclure une instruction permettant d'allouer une unité auxiliaire de traitement logique donnée à la tâche. Ces instructions insérées peuvent également inclure une instruction permettant d'exécuter un fil d'exécution de la tâche sur l'unité auxiliaire de traitement logique donnée. Cette instruction prend alors en paramètres d'entrée un contexte d'exécution sur l'unité auxiliaire de traitement logique donnée. Le contexte d'exécution permet d'identifier le fil d'exécution à exécuter, les données d'entrées pour l'exécuter et les données de sortie.

Dans un mode de réalisation préférentiel, l'instruction permettant d'exécuter un fil d'exécution de la tâche sur l'unité auxiliaire de traitement logique donnée peut être exécutée soit avec demande de libération, soit avec demande de synchronisation. Avec demande de libération, l'unité

auxiliaire de traitement logique donnée est libérée dès lors que l'exécution du fil d'exécution est terminée. Avec demande de synchronisation, l'unité auxiliaire de traitement logique donnée n'est pas libérée tant qu'une instruction de synchronisation n'est pas rencontrée dans le flot d'instructions de la tâche, une instruction de synchronisation rencontrée dans le flot d'instructions de la tâche permettant avantageusement de libérer toutes ou partie des unités auxiliaires de traitement logiques ayant fait l'objet d'une demande de synchronisation par la tâche.

Par exemple, les moyens pour exécuter les instructions insérées dans la tâche, ces instructions fournissant des directives de gestion des fils d'exécution exécutables par les unités auxiliaires de traitement logiques, peuvent être implémentés sous la forme d'un pipeline d'exécution ou d'un séquenceur microprogrammé.

Dans un mode de réalisation préférentiel, les moyens pour gérer des unités auxiliaires de traitement logiques peuvent comporter des moyens pour fournir un identifiant d'unité auxiliaire de traitement logique libre et/ou des moyens pour libérer une unité auxiliaire de traitement logique et/ou des moyens pour associer une unité auxiliaire de traitement logique avec une unité auxiliaire de traitement physique.

Par exemple, les moyens pour fournir un identifiant d'unité auxiliaire de traitement logique peuvent fournir l'identifiant du premier élément d'une liste d'unités auxiliaires de traitement logiques libres.

Dans un mode de réalisation préférentiel, les moyens pour gérer des unités auxiliaires de traitement physiques peuvent comporter des moyens pour fournir un identifiant d'unité auxiliaire de traitement physique libre, et/ou des moyens pour associer une unité auxiliaire de traitement physique avec une unité auxiliaire de traitement logique, et/ou des moyens pour fournir l'identifiant de l'unité auxiliaire de traitement logique associée à une unité auxiliaire de traitement physique, et/ou des moyens pour libérer une unité auxiliaire de traitement physique.

Dans un mode de réalisation préférentiel, les moyens pour allouer une unité auxiliaire de traitement logique à un fil d'exécution à exécuter peuvent comporter des moyens pour rechercher une unité auxiliaire de traitement logique libre, et/ou des moyens pour allouer l'unité de traitement

logique libre à un fil d'exécution, et/ou des moyens pour fournir l'identifiant de l'unité auxiliaire de traitement logique allouée à un fil d'exécution.

Dans un mode de réalisation préférentiel, l'unité peut comporter des moyens pour gérer des contextes d'exécution sur les unités auxiliaires de traitement logiques, et/ou des moyens pour décoder des interruptions venant des unités auxiliaires de traitement.

Dans un mode de réalisation préférentiel, l'unité peut comporter des moyens pour gérer des contextes d'exécution sur l'unité principale de traitement, un contexte d'exécution sur l'unité principale de traitement permettant d'identifier une tâche exécutable par l'unité principale de traitement, les données d'entrées pour l'exécuter et les données de sortie, de sorte que plusieurs tâches peuvent être exécutées sur l'unité principale de traitement.

Dans un mode de réalisation préférentiel, l'unité peut comporter un banc de registres locaux incluant un registre de masquage des exceptions et/ou des interruptions venant des unités auxiliaire de traitement, et/ou un registre indiquant les unités auxiliaires de traitement physiques en cours d'exécution, et/ou un registre indiquant les unités auxiliaires de traitement logiques en cours d'exécution, et/ou un registre indiquant les unités auxiliaires de traitement logiques n'ayant pas fait l'objet d'une demande de synchronisation par la tâche.

L'invention a encore pour principaux avantages qu'elle ne nécessite pas de préemption partielle des unités auxiliaires de traitement. Par ailleurs, l'invention ne nécessite pas une synchronisation forte de l'ensemble des unités de traitement, seulement une synchronisation faible au niveau des unités auxiliaires de traitement, éventuellement même une synchronisation par groupes d'unités auxiliaires de traitement. L'invention permet également de libérer le logiciel système de la gestion d'une partie des interruptions.

D'autres caractéristiques et avantages de l'invention apparaîtront à l'aide de la description qui suit faite en regard de dessins annexés qui représentent :

- la figure 1, par un diagramme, un exemple d'architecture selon le brevet français numéro FR2893156 (B1) alliant un processeur généraliste et plusieurs unités de traitements spécialisées dans les calculs intensifs;
- la figure 2, par un diagramme, un exemple d'architecture d'unité d'allocation et de contrôle selon l'invention;
- la figure 3, par un chronogramme, un exemple d'enchaînement de traitements.

La figure 1 illustre, par un diagramme, un exemple d'architecture MPSoC selon le brevet français numéro FR2893156 (B1). Cette architecture allie un processeur généraliste SPU (« Standard Processing Unit ») situé dans un sous-ensemble SPP de l'architecture (« Standard Processing Part ») et N unités de traitements spécialisées dans les calculs intensifs APU0, APU2,..., APUN-1 (« Auxiliary Processing Unit ») situées dans un sous-ensemble APP de l'architecture (« Auxiliary Processing Part »). Par la suite, les APU0, APU2,..., APUN-1 seront désignées par « le(les) processeur(s) APU » ou même « l'(les) APU ». Les processeurs APU peuvent communiquer par l'intermédiaire d'une mémoire partagée SMS (« Shared Memory Space »), cette mémoire SMS pouvant par ailleurs disposer de son propre contrôleur MSC (« Memory Space Controller »). Des bancs de registres SRF (« Shared Register Files ») peuvent également être partagés par les processeurs APU. Le SPP reçoit des données par l'intermédiaire d'un bus système SB (« System Bus ») et d'un contrôleur de bus SBA (« System Bus Arbiter »).

Les deux sous-ensembles SPP et APP ont des propriétés et des fonctionnalités radicalement différentes, mais pour autant participent à un même objectif, à savoir l'exécution simultanée de plusieurs tâches. D'une manière générale, le SPP se charge de l'exécution des tâches. L'exécution des tâches inclut d'une part le traitement des instructions constituant le programme à traiter. Pour cela, le SPP comporte une unité de contrôle

ESCU (« Extended Standard Control unit »), qui se charge de la lecture et du décodage des instructions. Le SPP comporte également une unité de mémorisation comprenant deux mémoires caches de premier niveau L1D-Cache et L1I-Cache, une mémoire cache de deuxième niveau L2-Cache, ainsi qu'une unité de chargement LSU (« Load Store Unit »). L'exécution des tâches inclut d'autre part le logiciel système. A la différence d'un processeur conventionnel, le SPP est capable de faire appel aux unités d'exécution auxiliaires que sont les processeurs APU, pour traiter certaines portions applicatives nécessitant de très fortes puissances de calcul. La présente invention se situe dans la façon de faire appel aux unités de calcul auxiliaires que sont les processeurs APU.

En effet, pour implémenter le méso-parallélisme, des blocs d'instructions sont assignés aux processeurs APU par une unité d'allocation et de contrôle ACU (« Allocation and Control Unit ») située dans l'APP. Leur allocation physique aux processeurs APU, ainsi que la gestion de leur exécution et leur synchronisation, sont à la charge de l'ACU. L'APP peut comporter une mémoire de masse MM (« Main Memory »), afin de stocker l'ensemble des données et des programmes manipulés par les processeurs APU. Cette mémoire MM dispose par ailleurs de son propre contrôleur MMC (« Main Memory Controller »), sur lequel l'ACU alloue des blocs de transfert de données entre le système et les blocs de calcul intensif. Ainsi, quoique fortement couplés, les sous-ensembles SPP et APP sont capables, grâce à l'ACU, de fonctionner de manière désynchronisée. L'ACU permet de gérer au mieux l'utilisation faiblement synchronisée des APU, de manière à optimiser, au niveau du logiciel système, l'utilisation du parallélisme, les performances et la consommation.

Un principe à la base de l'invention est de partir de codes séquentiels habituels auxquels sont adjoints quelques indices laissés par le programmeur d'application, afin de mieux séparer les parties de code de type « calcul/traitement » de données (ou « code de calcul » ou « code de traitement ») des parties de code de type "contrôle" d'une application (ou « code de contrôle »). En effet, deux grands types de code exécutable peuvent être distingués. Le code de contrôle est caractérisé par un IPC potentiel assez faible, car aléas de contrôle forts qui dépendent autant des données produites et consultées que de l'histoire d'exécution. La prédictibilité

des branchements et la prédictibilité des accès mémoire sont faibles. Les processeurs généralistes sont bien adaptés aux tâches de type contrôle. Le code de calcul est quant à lui caractérisé par un IPC potentiel important, car peu d'aléas de contrôle ou d'accès mémoire non prédictibles. Le parallélisme d'instructions est important et facile à optimiser, notamment sur des processeurs spécialisés de type DSP/SIMD. Un déroulage de boucle est utile pour réduire les aléas de contrôle perçus.

Afin de séparer les parties de code de calcul des parties de code de contrôle, une première phase de compilation est chargée d'isoler le code de contrôle d'un côté, mais en gardant toute les informations nécessaires sur les flots de données qui auront disparu, et d'un autre côté d'avoir les extraits de code et les fonctions de type calculs et traitements de données. Il s'agit ensuite de compiler, si ce n'est déjà fait dans des bibliothèques, les codes de traitement qui doivent s'exécuter sur les unités de calcul auxiliaires que sont les processeurs APU. Dans une deuxième phase de compilation, le graphe de flot de données est traité pour le code de contrôle à exécuter sur le SPP. Puis les données d'allocation, de chargement et d'exécution pour les processeurs APU sont insérées dans le programme de contrôle, ainsi que les données de synchronisations entre les processeurs APU. La première phase de compilation peut être illustrée à l'aide de l'exemple d'algorithme 1 suivant:

```
[algorithme 1] exemple de séquence de code

spect_dat1= fft(dat1);
spect_dat2= fft(dat2);
my_det= det(spect_dat2)
if(Re(my_det)<Im(my_det))
   spect_dat_f2= mult(spec_dat2, i_value);
else
   spect_dat_f2= spect_dat2;
conv= mult(spect_dat1,spec_dat_f2);
ret= fft(conv);
```

La première phase correspond à l'extraction du code à positionner sur les processeurs APU et du flot de données associé. Les fonctions `fft`, `det` et `mult` sont marquées par le programmeur pour l'exécution sur les processeurs APU, le graphe de dépendance de données étant analysé. La deuxième phase de compilation peut être illustrée à l'aide de l'exemple d'algorithme 2 suivant:

```
[algoritme 2] code de contrôle schématique obtenu
(pseudo assembleur + code non changé)

Allouer processeur de calcul APP#1
Allouer processeur de calcul APP#2
charger_prog fft sur APP#2
charger données dat2 pour APP#2
lancer APP#2
charger_prog fft sur APP#1
charger données dat1 pour APP#1
lance&libère APP#1
Synchroniser sur APP#2
copier données spec_dat2
charger_prog det sur APP#2
lancer APP#2
Synchroniser APP#2
charger_prog mult sur APP#2
if(Re(my_det)<Im(my_det))
{  charger i_val sur APP#2
   lancer APP#2
}
else
{  spect_dat_f2= spect_dat2;
}
Synchroniser APP#1 et APP#2
charger spect_dat1 sur APP#2
lance&libère APP#2
Allouer processeur APP#3
charger_prog fft sur APP#3
Synchroniser APP#2
charger conv sur APP#3
lance&libère APP#3
Synchroniser APP#3
```

La deuxième phase met en place le code de contrôle, sur la base du graphe de dépendance, du parallélisme potentiel et de son exécution sur les APP. Les pseudo-instructions qui commencent par une majuscule sont des opérations qui nécessitent une synchronisation du parallélisme. Elles mettent en place des marqueurs au niveau du processeur afin que le logiciel système de base puisse savoir si la tâche est en attente de synchronisation ou pas.

Ainsi, après compilation, le modèle vu par le SPP jouant le rôle de processeur de contrôle et par le logiciel système de base est de type séquentiel ordinaire, mais avec des points de synchronisation sous forme d'instructions assembleur qui apparaissent dans le code. Une séquence typique est donnée ci-après en exemple :

- une tâche en exécution peut s'allouer un processeur APU via une instruction assembleur. L'exécution de cette instruction peut-être bloquante, par exemple s'il ne reste plus de processeur disponible dans l'APP pour l'application;

- un processeur de l'APP est à la suite de cela alloué à la tâche jusqu'à ce que ledit processeur ait fini l'exécution. Il revient alors implicitement dans la liste des processeurs disponibles pour d'autres traitements;
- l'autre instruction bloquante pour une tâche est l'instruction de synchronisation. Elle est associée à une liste de processeurs de l'APP dont on attend la fin d'exécution.

Sur le SPP, les interruptions et exceptions sont gérées comme sur un processeur standard. Par défaut, elles n'ont pas d'impact sur les processeurs de l'APP. Les conséquences pour la mise en œuvre sont :

- pour une interruption ou une exception non fatale du type « underflow », il n'est en général pas nécessaire de forcer l'arrêt des APU, qui pourront ainsi poursuivre leur calcul alors même que l'interruption ou l'exception est traitée, voire tandis que la tâche en cours d'exécution est commutée au profit d'une autre. Lorsque la tâche interrompue sera réactivée, les calculs sur les APU auront avancé et on aura perdu le moins de temps possible;
- pour une interruption qui conduit à un réordonnancement qui met en avant une tâche qui est forcée en attente d'APU par d'autres moins urgentes qui se sont allouées des APU, il est également de la responsabilité du logiciel système de base de choisir le processeur APU à l'arrêt pour réallocation à la tâche plus urgente.

Mais il faut bien comprendre que ceci ne décrit que de façon très simplifiée ce que peut être le modèle d'exécution mis en œuvre sur l'exemple d'architecture de la figure 1. Il est néanmoins clair que, grâce à l'ACU, le SPP doit pouvoir mettre en œuvre :

- l'allocation des processeurs APU à la tâche en cours d'exécution. Pour des raisons d'efficacité, les désallocations devront dans la mesure du possible être implicites ou au moins semi-implicites;
- la gestion de la mémoire partagée sur puce. Cette mémoire est nécessaire pour des raisons de performances dans tous les MPSoC. Cela inclut :

- l'allocation et la désallocation de mémoire à un processeur APU particulier;
- le chargement et l'invalidation de mémoire avec des données ou des programmes.

- le lancement de traitements sur les processeurs APU.

Par ailleurs, l'ACU doit également pouvoir gérer de façon autonome, c'est-à-dire indépendamment du SPP ou de tout autre élément matériel, l'essentiel des procédures de désallocation des processeurs APU, des procédures de terminaison de traitements, ainsi qu'une partie de la gestion de la mémoire embarquée sur la puce.

La figure 2 illustre par un diagramme un exemple d'architecture selon l'invention pour l'ACU de la figure 1, en charge d'accélérer les opérations de gestion du méso-parallélisme sur les APU0, APU2,...APUN-1. Les connexions internes sont en traits pleins alors que les interfaces externes sont en traits pointillés. L'ACU est également chargée de gérer une partie de la synchronisation faible, sans passer par le cadre d'une gestion explicite par un noyau dédié. Cela est possible dès lors que les opérations à réaliser sont univoques, qu'elles sont simples à implémenter dans le matériel et qu'elles sont susceptibles d'améliorer notablement les performances de l'ensemble. Par essence, l'ACU a donc une dimension système très importante. Un objectif est de s'abstraire des particularités du système d'exploitation, lorsqu'il y en a un, et de rendre le système aussi efficace que possible. Bien entendu, un tel système ne fournit un niveau de performance optimum que lorsqu'un noyau dédié permet de profiter des temps morts du processeur de contrôle pour faire fonctionner d'autres tâches. Globalement, l'ACU offre une virtualisation de l'utilisation des processeurs APU, de telle sorte que la mise en œuvre d'un modèle d'exécution de type synchronisation faible est simple et efficace, les outils de compilation et le logiciel système n'intervenant que lorsque cela s'avère nécessaire. L'ACU sert à exécuter des instructions particulières de gestion du parallélisme, qui sont insérées dans le programme qui est en cours d'exécution sur le SPP. L'ACU sert également à gérer les APU et à gérer l'interface des APU avec le SPP.

LE PIPELINE D'EXECUTION (EP):

Avantageusement, l'ACU peut comporter un pipeline d'exécution EP (« Execution Pipeline »), qui est chargé de l'exécution des instructions spécifiques à la gestion du parallélisme sur les APU ou de la MM à partir des instructions insérées dans le flot d'instruction du SPP. C'est une unité d'exécution classique, qui peut être implémentée de différentes manières comme un pipeline effectif ou un séquenceur microprogrammé. Mais dans le présent mode de réalisation préférentiel, il s'agit d'un pipeline.

La description qui suit est faite sur une base où les processeurs APU sont homogènes avec une gestion unifiée. La généralisation à des processeurs APU hétérogènes est simple: il suffit de séparer les descripteurs dédiés pour chaque type d'APU. L'impact au niveau des instructions du pipeline d'exécution est qu'il faut ajouter un identifiant de type d'APU au niveau de l'instruction d'allocation. Par type d'APU, il y a alors une gestion d'APU logiques par un gestionnaire d'APU logiques LAM (« Logical APU Management ») et une gestion d'APU physiques par un gestionnaire d'APU physiques PAM (« Physical APU Management »). Un gestionnaire d'APU APUM (« APU Manager ») est alors en charge d'allouer les APU logiques au contexte en exécution, ainsi que de gérer la correspondance entre APU logiques et APU physiques.

Le pipeline d'exécution EP peut avantageusement permettre de traiter les instructions de base suivantes :

- allocation d'un processeur APU : il s'agit d'associer un APU logique à la tâche en cours d'exécution sur le SPP. L'instruction renvoie un numéro d'APU logique, obtenu par le gestionnaire d'APU APUM dans un registre global ou une mémoire indexée par l'instruction. L'instruction peut prendre des paramètres supplémentaires qui peuvent être à comparer avec un banc de registres locaux LRF (« Local Register File »), qui sera détaillé par la suite, par exemple pour limiter le nombre d'APU allouables dans une section particulière du code SPP. La mise en œuvre est assurée par le gestionnaire d'APU APUM, comme explicité par la suite;

- exécution sur un processeur APU : cette instruction nécessite un contexte spécifique, qui peut être fourni soit par un identifiant de contexte dans l'hypothèse de fonctionnement avec des contextes élaborés, soit par un simple triplet (identifiant de programme, identifiant de données en entrée, identifiant de données en sortie). Ce deuxième cas est un cas courant des traitements et permet un traitement générique: il vaut mieux l'implémenter dans tous les cas. Le premier cas permet une plus grande souplesse dans la définition des traitements, mais nécessite des instructions particulières de génération de contexte de traitements APU sur le SPP, ce qui nécessite éventuellement d'ajouter un lien interne entre le pipeline d'exécution EP et un gestionnaire de contexte partiel d'APU APUPCM (« APU Partial Context Manager ») qui sera détaillé par la suite. Il y a deux possibilités de mise en œuvre pour les instructions d'exécution, et il est possible d'implémenter au choix l'une des deux ou les deux. La solution où les deux sont implémentées est une implémentation préférée :
    - exécution sur un processeur APU avec libération : un APU physique est alloué pour l'exécution dès que possible. L'APU logique associé est libérée dès lors que l'APU physique a terminé son exécution, sans nécessiter de synchronisation;
    - exécution sur un processeur APU avec synchronisation : un APU physique est alloué pour l'exécution dès que possible. L'APU logique n'est pas libéré pour le contexte (mais seulement pour ce contexte d'exécution du SPP) tant qu'une instruction de synchronisation n'est pas rencontrée. Par contre l'APU logique peut être réaffecté dans un contexte SPP différent dès que l'APU physique a terminé son exécution.
- synchronisation d'exécution de processeurs APU : il s'agit de vérifier qu'un ou plusieurs APU logiques spécifiés dans l'instruction ont fini leur exécution. Le gestionnaire d'APU est directement en charge de l'exécution proprement dite de cette synchronisation, qui peut provoquer une exception. Une fois l'exécution de l'instruction

terminée, les APU logiques en demande de synchronisation sont tous libérés pour le contexte courant. Dans un mode d'implémentation préférentielle, plusieurs APU peuvent être synchronisés sur la même instruction. Une façon simple de le faire est d'utiliser un masque d'APU logique à synchroniser. L'information peut être passée par un ou plusieurs registres en une ou plusieurs instructions de synchronisation, même s'il faut préférer le cas où une seule instruction est nécessaire. L'information peut également être passée par une structure en mémoire dont on passe l'adresse. Par la suite, on appellera « masque de synchronisation » l'ensemble des APU logiques dont on a demandé la synchronisation, même s'il s'agit d'un cas particulier d'implémentation.

Le pipeline d'exécution EP peut également permettre de traiter les instructions suivantes pour faciliter la mise en œuvre du système :

- ajout d'une entrée TLB (« Translation Lookaside Buffer ») ou d'une valeur de registre de démarrage associé à un APU logique pour la gestion des contextes partiels d'APU ;
- définition d'une adresse de correspondance en mémoire pour les masques d'exécution et de synchronisation des APU logiques. Ceci permet d'éviter de passer par le logiciel système pour gérer les libérations d'APU dans des contextes de tâche SPP différents de celui en cours d'exécution comme détaillé par la suite ;
- instructions de modification de registres locaux, en particulier les registres locaux optionnels indiquant le nombre d'APU physiques allouables dans le contexte de la tâche, le masque d'allocation d'APU logiques dans le contexte de la tâche ou encore les masques d'interruptions et d'exceptions détaillés par la suite. Le cas échéant, il faut également modifier les tables de gestion internes des APU, comme détaillé par la suite.

LE BANC DE REGISTRES LOCAUX (LRF):

En ce qui concerne le banc de registres locaux LRF, qui est optionnel, il peut comporter dans une configuration de base les registres suivants :

- un indicateur de masquage des exceptions et interruptions spécifiques de l'ACU comme l'exception d'allocation ou de synchronisation ;
- un indicateur des APU physiques en cours d'exécution ;
- un indicateur des APU logiques en cours d'exécution dans le contexte ;
- un indicateur des APU logiques non synchronisés dans le contexte courant.

En plus de ces registres de base, le banc de registres locaux LRF peut également comporter divers autres registres, pour contrôler le nombre d'APU allouables dans un contexte donné par exemple, dont :

- un nombre maximum d'APU physiques allouables dans le contexte courant ;
- un masque d'allocation des APU logiques dans le contexte courant ;
- une adresse du masque de synchronisation des APU logiques dans le contexte courant.

Le banc de registres locaux LRF assure une partie de l'interface avec le logiciel système. A ce titre, il peut aussi comporter des moyens pour gérer l'énergie sur la puce, en fournissant par exemple des configurations de basse consommation pour les APU, avec probablement des performances moindres, voire des moyens de mettre en veille certains APU. L'autre partie de l'interface avec le logiciel système se fait par l'intermédiaire du gestionnaire d'APU APUM comme détaillé ci-après.

LE GESTIONNAIRE D'APU (APUM):

Dans une certaine mesure, le gestionnaire d'APU APUM peut être considéré comme le cœur de l'ACU. Son rôle est notamment d'allouer les APU logiques au contexte en exécution, de gérer la correspondance entre APU logiques et APU physiques, ainsi que d'accélérer certaines opérations

habituellement dévolues à un logiciel système, voire de se substituer à l'intervention du logiciel système dans certains cas. Le gestionnaire d'APU APUM peut être assez simple, car il s'agit principalement d'un système réactif. Le tableau qui suit décrit les traitements qu'il réalise en fonction des signaux qu'il reçoit du pipeline d'exécution EP.

Signal d'entrée	Sorties	Action
Attribution d'un nouvel APU logique	Gestion d'APU logiques. Registres globaux du SPP. Le cas échéant, la ligne d'exception du SPP.	Recherche un APU logique libre et l'attribue au contexte courant. Renvoie le numéro de l'APU alloué ou une exception s'il n'y a plus d'APU logiques libres.
Attribution d'une nouvelle entrée TLB	Gestion de contextes partiels	Ajout d'entrée TLB dans le contexte partiel associé à l'APU logique.
Exécution d'un traitement sur un APU	Gestion d'APU physique. Le cas échéant, la ligne d'exception du SPP.	Attribue un APU physique parmi les APU physiques libres. Met à jour le masque des APU non synchronisés si nécessaire ; s'il n'y a plus d'APU physique libres, une exception est générée.
Synchronisation d'exécution	Registres locaux. Le cas échéant, la ligne d'exception du SPP	Comparaison (OU logique) entre le masque des APU en cours d'exécution et les APU synchronisés. Génère une exception si des APU du masque de synchronisation sont encore en cours d'exécution.

L'ACU peut également comporter un décodeur d'interruptions APUID (« APU IT Decoder ») permettant de reformater les interruptions issues des APU sous une forme plus simple à traiter par le gestionnaire d'APU APUM. Le tableau qui suit décrit les traitements que le gestionnaire d'APU APUM

réalise en fonction des signaux qu'il reçoit du décodeur d'interruptions APUID:

Signal d'entrée	Sorties	Actions
Fin d'exécution d'un APU physique	Registres locaux. Le cas échéant, la ligne d'exception du SPP.	l'APU n'est plus considéré comme occupé et les indicateurs associés sont mis à jour (sauf s'il existe des traitements en attente). Mise à jour du masque de synchronisation si l'exécution est demandée avec synchronisation (si le contexte a changé, soit une interruption masquable est levée, soit on utilise un mécanisme de mise à jour automatique optionnel).
Erreur d'exécution	Ligne d'exception du SPP (Dispatch/ Completion/ contrôle d'exécution)	Une exception est remontée au SPP. Le numéro d'APU fautif (la traduction physique-logique est effectuée) est renvoyé au gestionnaire d'exception (via les mécanismes d'exception standards du SPP).
Timers (extension possible/ facultatif) Débug	idem	Le but ici est de pouvoir gérer des dépassements de temps d'exécution pour les traitements sur les APU, ou de l'assistance à la mise au point. Le mécanisme est le même que ci-dessus.

Il faut noter que le gestionnaire d'APU APUM n'a pas besoin de plus pour faire fonctionner le système multiprocesseur de la figure 1 dans le modèle d'exécution de la synchronisation faible selon l'invention. Il faut également noter que les exceptions associées aux instructions spécialisées exécutées sur l'APU ont un avantage à être masquées par défaut. En effet, la réponse de l'ACU à un tel masquage est d'attendre que la situation associée à l'exception se résorbe d'elle même. Cela permet de mettre en œuvre le modèle d'exécution sans avoir besoin d'adapter le logiciel système.

Cependant, un tel modèle d'exécution réduit nettement les performances potentielles, car les exceptions sont prévues et conçues pour que le SPP puisse optimiser le parallélisme exploitable en incitant intelligemment le logiciel système à faire une commutation de tâche dès que la tâche en cours d'exécution est bloquée dans son exécution, que ce soit pour des problèmes de ressources ou pour des problèmes de synchronisation. Il faut également noter que le gestionnaire d'APU APUM se fonde sur les services d'association entre APU logiques et APU physiques et réciproquement.

Le gestionnaire d'APU APUM est l'interface privilégiée avec le logiciel système. Lorsque le logiciel système est adapté et que l'option de gestion automatique des APU est mise en œuvre comme détaillé par la suite, l'intervention du logiciel système est limitée au stricte minimum. Il intervient uniquement lorsqu'il est avantageux de commuter de tâche au niveau système pour optimiser le parallélisme. A ce niveau, le logiciel système est le seul à disposer de l'information complète pour mettre le SPP en mode économie d'énergie (« idle »), qui pourrait avantageusement être couplé avec une gestion d'énergie du SPP lui-même. Par contre, le gestionnaire d'APU APUM possède toutes les informations pour la gestion des APU. Grâce à sa connaissance en avance de possibles commutations de tâches, le gestionnaire d'APU APUM peut même gérer la mise en veille des APU puis leur réveil. Ainsi, lorsque la file des traitements en attente d'APU est vide et qu'il y a des APU sans traitement, celles-ci peuvent être mises en veille, quitte à en réveiller au moins un lorsque l'ACU envoie une interruption au SPP ou lorsqu'une nouvelle instruction issue du SPP est exécutée sur l'ACU, en particulier l'instruction de changement de contexte partiel SPP détaillée par la suite. Par contre, l'adaptation en fréquence requiert d'être adaptée à chaque traitement si elle est mise en œuvre. Elle est donc à la charge du code utilisateur et du logiciel système sur le SPP et elle nécessite des registres locaux particuliers pour stocker les configurations des APU.

LE GESTIONNAIRE DE CONTEXTE PARTIEL D'APU (APUPCM):

Le gestionnaire de contexte partiel d'APU, qui est optionnel, ne contient que le contexte de démarrage des traitements sur les APU0,

APU2,..., APUN-1. Dans l'implémentation préférée, il n'est pas envisagé qu'il contienne des contextes matériels complets et encore moins des contextes étendus, car cela prendrait beaucoup de place. Un principe de base est d'avoir des tâches dont la durée d'exécution est relativement faible, de l'ordre de quelques dizaines de milliers de cycles d'exécution, de sorte que la nécessité de préemption partielle des APU n'existe pas réellement. La contrepartie de cette hypothèse est que le démarrage des traitements sur les APU doit être très rapide. Cependant, les simplifications opérées dans ce brevet rendent cela possible. Il s'agit notamment de stocker des TLB d'accès à la mémoire de masse MM. Le reste du contexte, en particulier les registres, ne sont pas stockés. En effet, les traitements sur les APU sont préférentiellement non préemptibles car en principe relativement courts. Si les APU sont optimisés pour un démarrage simple et un arrêt simple sur des frontières de code bien définies, alors leur démarrage quasiment « à froid » sur un nouveau traitement devrait pouvoir être effectué en quelques cycles. Le code de traitement peut être générique, mais alors les adresses de traitement logiques pour les APU sont constantes dans le code, les adresses physiques sont données par les quelques entrées TLB du contexte partiel fourni au démarrage. De cette façon, un traitement SPDM (Single Program Multiple Data) peut être effectué facilement sur la base des APU, même dans le cadre des traitements cascadés ou en pipeline.

LE GESTIONNAIRE D'APU PHYSIQUES (PAM):

La gestion des APU physiques utilise essentiellement des structures de stockage pour associer un APU physique alloué à un traitement sur le SPP. Dans le présent exemple, la gestion est organisée en double queue : une queue pour les APU physiques libres et une autre pour les APU physiques en cours de traitement. Dans un autre mode de réalisation, une structure de donnée unique et non mutable peut être utilisée pour les deux. Les fonctionnalités de gestion d'APU physiques peuvent alors être implémentées à l'aide d'encodeurs de priorité ou de mémoires associatives. Lorsque l'instruction de démarrage est rencontrée dans le flot du programme sur le SPP, le gestionnaire d'APU logiques cherche à associer un APU physique à l'APU logique dont la demande d'exécution d'un traitement est

faite. La procédure est donc simple : la gestion des APU physiques est constituée en queue d'APU physiques libres. Sur instruction de lancement au niveau du gestionnaire d'APU, le premier élément de la queue des APU physiques libres est pris. Selon l'implémentation, les traitements non attribués à un APU physique peuvent être mis dans une file d'attente. Dans ce cas, un tampon de traitements en attente peut se mettre en route, indépendamment des traitements en cours sur le SPP. Il est simple de constater que la file d'attente des traitements est limitée à une profondeur maximale de NAPUlogiques-NAPUphysiques où NAPUlogiques est le nombre d'APU logiques offerts et NAPUphysiques est le nombre d'APU physiques disponibles au total. Lorsqu'un APU physique termine son exécution et que le signal de terminaison atteint le gestionnaire d'APU, celui-ci remet l'APU physique dans la liste des APU physiques libres. Si une file d'attente de traitements en attente est implémentée, la phase de remise de l'APU dans la liste des APU libres peut être court-circuitée, en retirant le premier traitement en attente dans la liste et en l'affectant à l'APU libéré. Dans le cas de l'utilisation d'encodeurs de priorité, la mise à 0 ou a 1 d'un bit de donnée associé à l'APU physique suffit à le marquer comme occupé ou libre. La gestion de queue n'existe alors plus, mais le reste de l'implémentation ne varie pas.

La gestion des APU physiques aboutit à deux ensembles°: l'ensemble des APU physiques libres et l'ensemble des APU physiques attribués. Optionnellement, il peut aussi y avoir une file d'attente pour les processus en attente d'un APU physique. Lors du processus d'allocation, les APU physiques se voient associés avec un numéro d'APU logique. Ils peuvent également se voir associés avec un numéro dans une table facultative de contextes, cette table étant détaillée par la suite. Enfin, ils peuvent se voir associés avec un bit optionnel de validité, un APU ayant par exemple un indicateur de validité positionné à « vrai » (1) lorsqu'il est alloué, à « faux » (0) sinon.

Les services fournis par le gestionnaire d'APU physiques PAM peuvent avantageusement être les suivants°:

- allocation d'un APU physique libre et association avec un numéro d'APU logique ;

- fourniture sur demande du numéro d'APU logique associé à un numéro d'APU physique, ceci étant utile au gestionnaire d'APU pour libérer l'APU logique lorsque le signal de terminaison de traitement sur l'APU physique est reçu par ledit gestionnaire;
- libération d'un APU physique ;
- fourniture d'une entrée dans une table de gestion de contexte associé à un APU physique ;
- gestion d'une queue de traitements en attente d'APU physiques.

LE GESTIONNAIRE D'APU LOGIQUES (LAM):

Comme la gestion d'APU physiques, la gestion d'APU logiques utilise essentiellement une structure de stockage. Il est à noter que la gestion d'APU logiques tend à s'effacer en partie lorsque le nombre d'APU logiques s'approche du nombre d'APU physiques. Lorsque le gestionnaire d'APU APUM demande une APU logique, le gestionnaire d'APU logiques LAM peut renvoyer le premier APU logique libre disponible. Si aucun APU logique n'est disponible, le gestionnaire d'APU logiques LAM renvoie un signal au gestionnaire d'APU APUM, qui émet alors une exception ou attend qu'un APU logique se libère, comme explicité précédemment. Lorsqu'une demande de libération d'un APU logique est faite par le gestionnaire d'APU logiques LAM, l'APU du numéro transmis est remis dans l'ensemble des APU libres. Selon l'implémentation, une gestion de contextes partiels du SPP peut être mise en œuvre. Cette dernière peut être mise en relation avec la gestion de contextes SPP pour les APU physiques comme explicité précédemment.

L'organisation des structures de données permet de distinguer un ensemble d'APU logiques libres et un ensemble d'APU logiques alloués. A chaque entrée est associé un numéro d'APU physique et un bit de validité. Le bit de validité indique si l'APU physique associé au numéro est effectivement associé à cette APU logique ou pas. Optionnellement, un bit optionnel de demande d'exception peut être utilisé, ainsi qu'un champ pour un numéro de contexte SPP. Il apparaît clairement qu'il y a des informations en commun entre la structure de description des allocations des APU physiques et celle des APU logiques. Cela permet de choisir différentes

voies d'implémentation entre 2 structures mémoires à mettre à jour en parallèle ou une seule structure à mémoire associative.

Les services fournis par le gestionnaire d'APU logiques LAM peuvent avantageusement être les suivants :

- allocation d'un numéro d'APU logique, par exemple le premier élément de la liste des APU logiques libres, mais optionnellement une entrée pour un descripteur de contexte partiel SPP peut être associée ;
- libération d'un APU logique ;
- association d'un APU logique avec un numéro d'APU physique.

LE DECODEUR D'INTERRUPTIONS (APUID):

Avantageusement, les interruptions issues des APU peuvent être reformatées sous une forme plus simple à traiter par le gestionnaire d'APU APUM. Certaines interruptions, comme les erreurs d'exécution sur une APU, sont remontées directement vers l'unité de terminaison du SPP où elles sont traitées comme des exceptions globales du SPP. Les autres signaux à reformater pour le gestionnaire d'APU APUM concernent les fins d'exécution des traitements en cours sur les APU, ainsi que tous les signaux potentiellement intéressants, comme par exemple les signaux particuliers pour les opérations de mise au point des programmes sur les APU (« débug »).

Un rôle du décodeur d'interruptions APUID est de sérialiser les différents signaux d'intérêt vers le gestionnaire d'APU APUM. Une fois le signal relayé vers le gestionnaire d'APU APUM, son rôle est également de fournir les signaux d'acquittement des interruptions aux APU qui les ont émis, ceci au fur et à mesure des traitements. A noter que l'implémentation à préférer comporte un tampon d'événements, ce qui permet de libérer la ligne de signalisation entre l'APU et l'ACU au plus tôt, et ainsi une allocation au plus tôt des APU libérés pour de nouveaux travaux.

LE GESTIONNAIRE DE CONTEXTE PARTIEL DE SPP (SPPPCM):

Le gestionnaire de contexte partiel de SPP SPPPCM est optionnel, il permet simplement de libérer les APU logiques alloués avec demande de synchronisation de façon transparente et au plus tôt. Cette fonctionnalité est normalement dévolue au logiciel système, mais dans le présent exemple de réalisation l'ACU peut agir comme un accélérateur dédié pour dégager une partie du travail du logiciel système. Cela diminue de façon importante les temps de commutation de contexte et les temps de traitement. Bien entendu, même lorsqu'il est implémenté, ce mécanisme est débrayable par le logiciel système. Pour activer ce mécanisme, le logiciel système doit renseigner l'adresse du masque de synchronisation du contexte comme explicité précédemment, sinon le mécanisme est de base désactivé. A chaque fois qu'une mise à jour de cette adresse est faite par le logiciel système, l'adresse fournie est comparée avec celles déjà présentes dans la table des contextes partiels SPP. S'il n'y a pas d'entrée correspondante, le gestionnaire SPPPCM présélectionne une entrée libre de cette table. Elle n'est allouée complètement que lors de l'allocation d'un APU logique, le numéro d'entrée étant alors renseigné dans la table décrite précédemment. Le champ d'allocation des APU logiques est mis à jour dans cette même table, le bit correspondant à l'APU logique alloué étant mis à 1. Lorsqu'un APU physique est alloué, le champ de la table des APU logiques non synchronisés est mis à jour avec la valeur courante correspondant au numéro d'APU logique associé à l'APU physique alloué. De même, le champ de la table des APU physiques décrit précédemment est renseigné avec le numéro d'entrée dans la table des contextes SPP. Lorsqu'un APU physique est libéré, le champ de synchronisation de la table est mis à jour, ainsi que le champ d'allocation. Les valeurs sont écrites en mémoire à l'adresse fournie. Si les champs sont revenus à zéro, l'entrée de la table de contexte est libérée. Dans tous les cas, l'APU logique correspondant est libéré, son utilisation par une autre tâche ayant été correctement tracée.

Un avantage important du gestionnaire de contexte partiel de SPP SPPPCM est que les APU logiques peuvent être libérés à la volée, même si l'exécution est multitâche avec des utilisations indépendantes d'APU à l'intérieur des différentes tâches. Les APU logiques peuvent donc être remis à disposition du contexte courant sans passer par une commutation sur le contexte qui a alloué l'APU logique en question.

Par ailleurs, quelques interfaces peuvent être utiles à un fonctionnement optimal des APU en relation avec l'ACU, comme par exemple :

- une ligne d'interruption et d'exception APU IT peut être connectée au décodeur d'interruption APUID. Cette ligne peut être couplée à un bus ou à une ligne pour la transmission d'informations annexes, comme par exemple un pointeur d'instruction en cas d'erreur d'exécution sur un APU. L'activation de cette ligne par un APU bloque alors ledit APU jusqu'à ce que l'ACU renvoie un signal d'acquittement de l'exception. Cette ligne peut en particulier être chargée de signaler les fins de traitements sur un APU;
- une unité de chargement de contexte partiel, en particulier pour les TLB, peut avantageusement être couplée à un mécanisme de réinitialisation à zéro des autres registres d'usage de l'APU correspondant, si cela a un sens pour l'APU en question;
- des unités peuvent gérer la production/consommation des données, afin de coordonner les exécutions de différents étages de pipelines logiciels mis en œuvre par des traitements sur les APU. Le producteur utilise alors une partie de la mémoire partagée de masse MM pour écrire atomiquement des indicateurs de production de données. Au moins un consommateur réinitialise l'indicateur de façon atomique également. Cette fonctionnalité peut être réalisée aisément de façon logicielle, mais elle peut avantageusement être mise en œuvre par un accélérateur matériel dédié. Ainsi, de nombreux traitements peuvent s'enchaîner sans nécessiter de synchronisation par l'ACU. En procédant ainsi, le nombre d'opérations que doit réaliser l'ACU peut être diminué de façon importante, améliorant sensiblement l'efficacité d'exécution parallèle du système. Ce mécanisme vise à traquer une partie des mécanismes de gestion des dépendances de données, l'autre partie étant gérée par les instructions de synchronisation. Lorsque les mécanismes de gestion de consommation sont mis en œuvre, la ligne entre le gestionnaire d'APU APUM et les APU peut aussi

être utilisée aussi pour transmettre le mode de fonctionnement de l'APU.

La figure 3 illustre par un chronogramme un exemple d'enchaînement de traitements sur l'architecture des figures 1 et 2. Cet exemple vise plus particulièrement à montrer, d'une part, la coopération entre l'ACU et le logiciel système de base et, d'autre part, l'optimisation de l'exploitation des ressources de traitements auxiliaires que sont les APU. Il s'agit d'un extrait d'une application composée de trois tâches actives :

- une tâche T1 : allocation de deux APU, exécution sur les deux APU puis synchronisation des deux APU pour le contexte;
- une tâche T2 : allocation de trois APU, exécution sur les trois APU puis synchronisation des trois APU pour le contexte;
- une tâche T3 : allocation d'un APU puis exécution avec libération implicite, suivi de l'allocation d'un APU et exécution puis synchronisation.

Afin d'illustrer une majorité de comportements, l'hypothèse est faite que seulement quatre APU physiques APU1, APU2, APU3 et APU4 sont disponibles et qu'une queue de six APU logiques est utilisable.

A un instant t1/1, T1 alloue un APU et le numéro logique 1 (premier numéro de la liste des APU logiques libres) est renvoyé à T1.

A un instant t1/2, T1 alloue un nouvel APU et le numéro logique 2 (premier numéro de la liste des APU logiques libres) est renvoyé à T1.

A un instant t1/3, T1 demande l'exécution d'un traitement sur l'APU logique 1 ; l'APU physique 1 (premier numéro de la liste des APU physiques libres) est réservé par l'ACU.

A un instant t1/4, T1 demande l'exécution d'un traitement sur l'APU logique 2 ; l'APU physique 2 (premier numéro de la liste des APU physiques libres) est réservé par l'ACU.

A un instant t1/5, T1 demande de synchroniser sur la fin des traitements des APU logiques 1 et 2 ; les APU physiques correspondants n'ont pas achevé les traitements, le masque d'exécution étant à 0x3, donc l'ACU génère une exception E1 nommée « synchronisation demandée mais traitements en cours ». Le logiciel système capte cette exception E1 et

commute la tâche T1, il fait donc passer la tâche T2 au premier plan sur le SPP à la place de T1.

A un instant t2/1, T2 alloue un APU logique ; l'ACU lui alloue l'APU logique 3 (premier numéro de la liste des APU logiques libres).

A un instant t2/2, T2 alloue à nouveau un APU logique ; cette fois l'APU logique 4 lui est affecté.

A un instant t2/3, T2 alloue à nouveau un APU logique; l'APU logique 5 est affecté à la tâche T2.

A un instant t2/4, T2 demande l'exécution d'un traitement sur l'APU logique 3 (le premier alloué par T2); l'ACU alloue l'APU physique 3 (premier APU physique libre) et demande le début d'exécution.

A un instant t2/5, T2 demande l'exécution d'un traitement sur l'APU logique 4 (deuxième alloué par T2); l'ACU alloue l'APU physique 4 (premier APU physique libre) et demande le début d'exécution.

A un instant t2/6, T2 demande l'exécution d'un traitement sur l'APU logique 5 (troisième alloué par T2); il n'y a plus d'APU physiques libres (liste vide), donc l'ACU place l'APU logique 5 dans la queue des APU logiques en attente d'un APU physique pour l'exécution.

A un instant t2/7, T2 demande la synchronisation sur l'ensemble des trois APU logiques alloués, le masque de synchronisation étant à 0x1c; aucun des APU physiques n'a finit son exécution, un des APU logiques n'étant même pas alloué sur un APU physique; une exception E2 nommée « synchronisation demandée mais traitements en cours » est traitée par le logiciel système de base, qui donne ensuite la main à la tâche T3, les marqueurs d'exécution pour la tâche T1 la montrant toujours en attente de synchronisation.

A un instant t0/1, l'APU physique 1 finit son traitement; l'ACU met à jour le statut d'exécution pour la tâche T1, le masque d'exécution des APU logiques pour T1 passant de 0x3 à 0x2; comme la liste des APU logiques en attente d'APU physique n'est pas vide, l'ACU associe l'APU physique 1 à l'APU logique 5 pour T2, son masque d'exécution restant inchangé à 0x1c; la liste des APU logiques libres contient l'APU logique 1, puisqu'il a été libéré pour T1, même si l'instruction de synchronisation n'a pas encore été exécutée.

A un instant t3/1, T3 alloue un APU logique; c'est l'APU logique 1 qui lui est attribué (le seul qui était libre); puis T3 demande l'exécution du traitement sur l'APU logique 1 avec libération implicite après le traitement; comme il n'y a pas d'APU physique libre, l'ACU place l'APU logique 1 dans la file d'attente des APU logiques en attente d'APU physique.

A un instant t3/2, T3 tente d'allouer un nouvel APU logique, mais il n'y a plus d'APU logique disponible; une exception E3 nommée « plus d'APU logique disponible » est levée, elle est traitée par le logiciel système; T3 est en instance d'être commutée.

A un instant t0/2, l'APU physique 2 finit son traitement; l'APU logique 2 qui y est associé est libéré au niveau exécution pour T1 et le masque de synchronisation restant pour T1 passe de 0x2 à 0x0; T1, qui avait été commutée pour cause de synchronisation, peut être synchronisée; le logiciel système peut donc commuter l'exécution qui était en suspend pour T3 sur T1.

A un instant t1/6, l'instruction de synchronisation de T1 est exécutée au retour de commutation de tâche; comme les deux APU logiques avaient été libérés pour la tâche T1, l'instruction de synchronisation est exécutée sans provoquer d'exception, c'est la raison pour laquelle le logiciel système avait commuté sur T1; formellement, c'est le moment dans l'exécution de T1 où le programme utilisateur sur le SPP est certain d'avoir libéré les deux APU logiques.

A un instant t1/7, T1 libère le SPP, au moins pour un certain temps d'utilisation des résultats de calculs; le logiciel système reprend la main et remet au premier plan T3, qui avait été commuté par défaut d'APU logique disponible.

A un instant t3/3, T3 exécute l'instruction d'allocation d'un APU logique; cette fois, l'APU logique 2 est libre, il est donc alloué à T3.

A un instant t0/3, c'est la fin de l'exécution sur l'APU physique 3; l'APU logique associé, ici l'APU logique 3 associé à T2, est libéré par l'APU; le masque de synchronisation pour T2 passe de 0x1c à 0x18.

A un instant t3/4, T3 demande l'exécution sur l'APU logique 2 qu'elle avait alloué; le masque d'exécution local est donc 0x3; l'APU physique 3, qui est libre, est associé à l'APU logique 2 pour T3.

A un instant t0/4, c'est la fin de l'exécution sur l'APU physique 4 associé à l'APU logique 4 pour T2; l'ACU place l'APU physique 4 dans la liste des APU physique libres et l'APU logique 4 dans la liste des APU logiques libres; le masque d'exécution pour T2 est donc mis à 0x10, de même pour le masque de synchronisation résiduel.

A un instant t0/5, c'est la fin d'exécution sur l'APU physique 2 associé à l'APU logique 1 pour T3; le masque d'exécution passe de 0x3 à 0x2 pour T3 et l'APU est libéré complètement pour T3.

A un instant t3/5, T3 demande une synchronisation sur l'APU logique 2, qui est alloué sur l'APU physique 3; comme l'APU physique n'a pas finit son exécution, une exception est levée pour synchronisation, le masque de synchronisation étant à 0x2; le logiciel système de base prend la main, mais il ne reste pas de tâche allouable qui ne soit pas bloquée; le logiciel système autorise la prise d'interruption sur fin d'exécution d'APU et passe en mode économie d'énergie sur le SPP.

A un instant t0/6, c'est la fin d'exécution pour l'APU physique 3 associé à l'APU logique 5 pour T2; le masque d'exécution pour T2 passe à 0x0, de même que le masque de synchronisation; la tâche T2 devient à nouveau exécutable sur le SPP; l'interruption de fin d'exécution n'étant pas masquée au niveau SPP, le logiciel système reprend la main sur le mode économie d'énergie, masque à nouveau les interruptions de fin d'exécution d'APU et constate que la fin d'exécution permet à T2 d'être exécuté; il commute donc l'exécution pour T2 sur le SPP.

A un instant t2/8, T2 reprend son exécution sur l'instruction de synchronisation, qui ne provoque pas d'exception puisque le masque de synchronisation est à 0; elle poursuit son exécution jusqu'au moment où elle rend la main au système à un instant t2/9.

A un instant t0/7, c'est la fin d'exécution de l'APU physique 3 associé à l'APU logique 2 alloué pour T3; le masque d'exécution pour T3 passe à 0x0, de même que le masque de synchronisation; la tâche T3 est a nouveau exécutable.

A un instant t2/9, c'est la fin de T2, qui redonne la main au logiciel système; celui-ci commute sur T3 puisqu'elle est redevenue exécutable.

A un instant t3/6, une instruction de synchronisation arrive pour T3; elle ne provoque pas d'exception puisque le masque de synchronisation est à 0; T3 finit son exécution.

Cet exemple d'enchaînement de traitements montre combien la présente invention permet une excellente occupation des ressources de traitement annexes que sont les APU physiques APU1, APU2, APU3 et APU4. Ceci est dû notamment au degré d'abstraction assez élevé conféré par les APU logiques. Par ailleurs, le logiciel système s'avère très efficace, puisqu'il ne prend la main sur l'exécution des tâches sur le SPP que lorsqu'il n'est pas possible de faire autre chose que de commuter de tâche ou de passer en mode économique.

La présente invention a encore pour principaux avantages qu'elle utilise un processeur ordinaire pour gérer le parallélisme à grain moyen, ce parallélisme étant, au niveau de la dépendance des données, conforme à ce qui a été calculé par le compilateur de l'application.

Dans un cadre monotâche, l'utilisation optimale du parallélisme à grains moyens peut être atteinte par la gestion automatique des éléments de traitements auxiliaires, grâce au traitement d'un nombre très limité d'instructions spécifiques simples introduites à la compilation de la tâche. Hors de ces instructions, le reste de la gestion du parallélisme est automatique. La présente invention propose une interface haut niveau qui rend abstraite l'utilisation des unités de traitements, cette interface permettant l'allocation au plus tard et la libération au plus tôt des unités de traitements auxiliaires, la correspondance avec les unités réelles étant intégralement à la charge de la présente invention. Cependant, le logiciel système peut se voir accorder le droit de modifier cela, grâce à l'accès aux registres internes de la présente invention dans des cas particuliers où la compilation et l'analyse statique de l'exécution pour une tâche n'a pas suffi à assurer une exécution qui satisfasse à coup sûr les critères imposés pour la tâche. Le logiciel système dispose alors de la main pour prendre des décisions dynamiques qui peuvent, dans certains cas, s'avérer plus efficaces.

Dans un cadre multitâche, la présente invention offre les mécanismes nécessaires à l'activation du logiciel système, afin d'opérer une commutation de tâches dès que la tâche en cours d'exécution est bloquée dans sa gestion du parallélisme à grain moyen. L'invention offre également les mécanismes nécessaires à la mise à jour automatique des indicateurs de blocage des tâches dès la levée de ces blocages, ceci sans nécessiter l'intervention du logiciel système, de manière à améliorer la mise en œuvre des deux niveaux de parallélisme, à gros grains et à grains moyens. L'invention offre aussi les mécanismes nécessaires au logiciel système pour choisir aisément quelles sont les tâches activables. L'invention offre également, en partie au niveau monotâche (code utilisateur) et en partie au niveau multitâche (logiciel système), les mécanismes nécessaires pour gérer le nombre d'unités de traitements utilisées à un instant donné. Ainsi, le système global est plus à même de tenir des délais concernant les traitements, malgré le partage des unités de traitements entre plusieurs tâches en cours d'exécution. L'invention offre enfin les mécanismes nécessaires pour gérer des aspects avancés de l'économie d'énergie, en facilitant la mise en mode économique tant des unités de traitements auxiliaires que du processeur principal qui intègre la présente invention.

De manière générale, un système selon l'invention permet d'exécuter des instructions spécialisées dans la gestion du parallélisme dans des systèmes multi-cœurs hétérogènes. Une fois les instructions de gestion de parallélisme données, la gestion du parallélisme sur l'architecture multi-cœur devient automatique et n'a pas besoin d'assistance dans un cadre d'exécution monotâche sur le processeur qui intègre l'invention. Dans le cadre multitâche notamment, l'invention devient à la fois un assistant et un accélérateur spécifique à la gestion du parallélisme à grain moyen pour le logiciel système présent sur le processeur qui intègre l'invention.

Ainsi, lors de l'exécution d'une tâche, l'invention décrite précédemment laisse la plus grande autonomie possible dans la gestion des unités auxiliaires de traitement, ceci sans que le logiciel système ait à intervenir pour l'exécution de la tâche. En effet, le logiciel système n'intervient que dans les cas ou il n'y a pas d'autres alternatives, comme les

cas d'erreur ou les cas où il est nécessaire d'attendre une synchronisation. A contrario, si une tâche est bloquée, l'invention décrite précédemment autorise la mise en œuvre du logiciel système pour la commutation de tâche, ceci afin d'optimiser l'utilisation des différents parallélismes. Ainsi, l'invention décrite précédemment confère un déterminisme d'exécution global qui se rapproche de celui conféré par une architecture Von Neumann classique.

REVENDICATIONS

1. Unité d'allocation et de contrôle (ACU) pour allouer des fils d'exécution d'une tâche (T1) à une pluralité d'unités auxiliaires de traitement (APU0, APU2,...,APUN-1) et pour contrôler l'exécution en parallèle desdits fils d'exécution par lesdites unités auxiliaires de traitement, la tâche étant exécutée de manière séquentielle par une unité principale de traitement (SPP), l'unité d'allocation et de contrôle (ACU) étant caractérisée en ce qu'elle comporte:
   - des moyens (LAM) pour gérer des unités auxiliaires de traitement logiques ;
   - des moyens (PAM) pour gérer des unités auxiliaires de traitement physiques, chaque unité auxiliaire de traitement physique correspondant à une unité auxiliaire de traitement (APU0, APU2,...,APUN-1) ;
   - des moyens (APUM) pour gérer les unités auxiliaires de traitement (APU0, APU2,...,APUN-1), ces moyens comportant:
     - o des moyens pour allouer une unité auxiliaire de traitement logique à un fil d'exécution à exécuter ;
     - o des moyens pour gérer la correspondance entre les unités auxiliaires de traitement logiques et les unités auxiliaires de traitement physiques ;

   de sorte que les unités auxiliaires de traitement (APU0, APU2,...,APUN-1) exécutent en parallèle les fils d'exécution de la tâche (T1) par l'intermédiaire des unités auxiliaires de traitement logiques, qui sont allouées au plus tard et libérées au plus tôt.

2. Unité selon la revendication 1, caractérisée en ce qu'elle comporte des moyens pour exécuter des instructions insérées dans la tâche (T1), lesdites instructions insérées fournissant des directives de gestion des fils d'exécution exécutables par les unités auxiliaires de traitement logiques, lesdites instructions insérées incluant:
   - une instruction permettant d'allouer une unité auxiliaire de traitement logique donnée à la tâche (T1), et/ou ;
   - une instruction permettant d'exécuter un fil d'exécution de la tâche (T1) sur l'unité auxiliaire de traitement logique donnée, ladite

instruction prenant en paramètres d'entrée un contexte d'exécution sur l'unité auxiliaire de traitement logique donnée, ledit contexte d'exécution permettant d'identifier le fil d'exécution à exécuter, des données d'entrées pour l'exécuter et des données de sortie.

3. Unité selon la revendication 2, caractérisée en ce que l'instruction permettant d'exécuter un fil d'exécution de la tâche (T1) sur l'unité auxiliaire de traitement logique donnée est exécutée:
   - avec demande de libération, auquel cas l'unité auxiliaire de traitement logique donnée est libérée dès lors que l'exécution du fil d'exécution est terminée, ou;
   - avec demande de synchronisation, auquel cas l'unité auxiliaire de traitement logique donnée n'est pas libérée tant qu'une instruction de synchronisation n'est pas rencontrée dans le flot d'instructions de la tâche (T1);

   une instruction de synchronisation rencontrée dans le flot d'instructions de la tâche (T1) permettant de libérer toutes ou partie des unités auxiliaires de traitement logiques ayant fait l'objet d'une demande de synchronisation par la tâche (T1).

4. Unité selon la revendication 2, caractérisée en ce que les moyens pour exécuter les instructions insérées dans la tâche (T1) pour la gestion des fils d'exécution exécutables par les unités auxiliaires de traitement logiques sont implémentés sous la forme d'un pipeline d'exécution ou d'un séquenceur microprogrammé.

5. Unité d'allocation et de contrôle (ACU) selon la revendication 1, caractérisée en ce que les moyens (LAM) pour gérer des unités auxiliaires de traitement logiques comportent:
   - des moyens pour fournir un identifiant d'unité auxiliaire de traitement logique libre, et/ou ;
   - des moyens pour libérer une unité auxiliaire de traitement logique, et/ou ;
   - des moyens pour associer une unité auxiliaire de traitement logique avec une unité auxiliaire de traitement physique.

6. Unité d'allocation et de contrôle (ACU) selon la revendication 5, caractérisée en ce que les moyens pour fournir un identifiant d'unité auxiliaire de traitement logique fournissent l'identifiant du premier élément d'une liste d'unités auxiliaires de traitement logiques libres.

7. Unité d'allocation et de contrôle (ACU) selon la revendication 1, caractérisée en ce que les moyens (PAM) pour gérer des unités auxiliaires de traitement physiques comportent:
   - des moyens pour fournir un identifiant d'unité auxiliaire de traitement physique libre, et/ou ;
   - des moyens pour associer une unité auxiliaire de traitement physique avec une unité auxiliaire de traitement logique, et/ou ;
   - des moyens pour fournir l'identifiant de l'unité auxiliaire de traitement logique associée à une unité auxiliaire de traitement physique, et/ou ;
   - des moyens pour libérer une unité auxiliaire de traitement physique.

8. Unité d'allocation et de contrôle (ACU) selon la revendication 1, caractérisée en ce que les moyens pour allouer une unité auxiliaire de traitement logique à un fil d'exécution à exécuter comportent:
   - des moyens pour rechercher une unité auxiliaire de traitement logique libre, et/ou ;
   - des moyens pour allouer l'unité de traitement logique libre à un fil d'exécution, et/ou ;
   - des moyens pour fournir l'identifiant de l'unité auxiliaire de traitement logique allouée à un fil d'exécution.

9. Unité selon la revendication 1, caractérisée en ce qu'elle comporte des moyens (APUPCM) pour gérer des contextes d'exécution sur les unités auxiliaires de traitement logiques.

10. Unité selon la revendication 1, caractérisée en ce qu'elle comporte des moyens (APUID) pour décoder des interruptions venant des unités auxiliaires de traitement.

11. Unité selon la revendication 1, caractérisée en ce qu'elle comporte des moyens (SPPPCM) pour gérer des contextes d'exécution sur l'unité principale de traitement, un contexte d'exécution sur l'unité principale de traitement permettant d'identifier une tâche exécutable par l'unité principale de traitement, des données d'entrées pour l'exécuter et des données de sortie, de sorte que plusieurs tâches (T1, T2, T3) sont exécutables sur l'unité principale de traitement (SPP).

12. Unité selon la revendication 1, caractérisée en ce qu'elle comporte un banc de registres locaux (LRF) incluant :
    - un registre de masquage des exceptions et/ou des interruptions des venant des unités auxiliaire de traitement, et/ou ;
    - un registre indiquant les unités auxiliaires de traitement physiques en cours d'exécution, et/ou ;
    - un registre indiquant les unités auxiliaires de traitement logiques en cours d'exécution, et/ou ;
    - un registre indiquant les unités auxiliaires de traitement logiques n'ayant pas fait l'objet d'une demande de synchronisation par la tâche (T1).

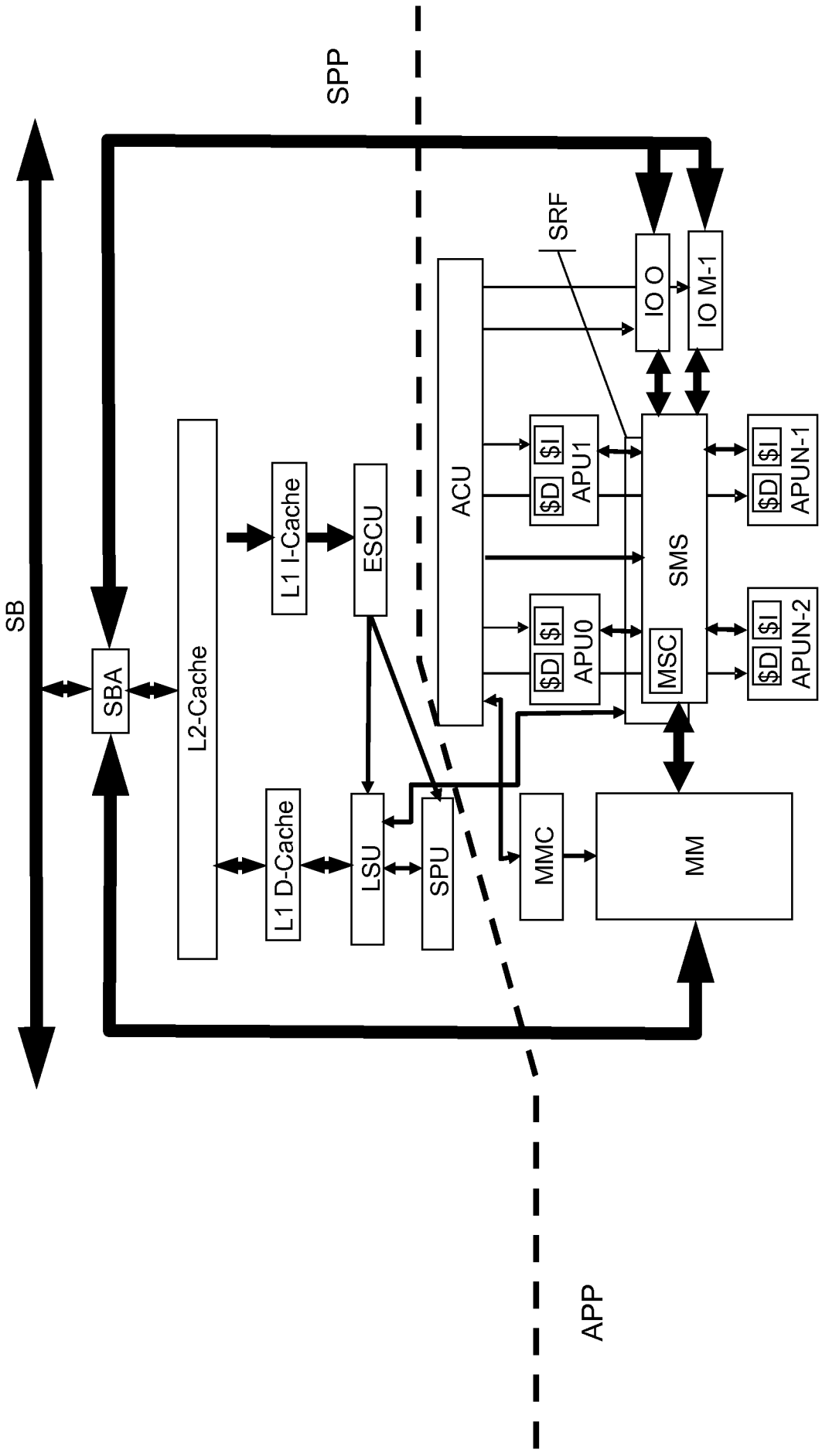
SPP
SB
SBA
L2-Cache
L1 I-Cache
ESCU
L1 D-Cache
LSU
SPU
ACU
SRF
IO O
IO M-1
$D
$I
APU1
APU0
SMS
MSC
APUN-1
APUN-2
MMC
MM
APP

FIG.1

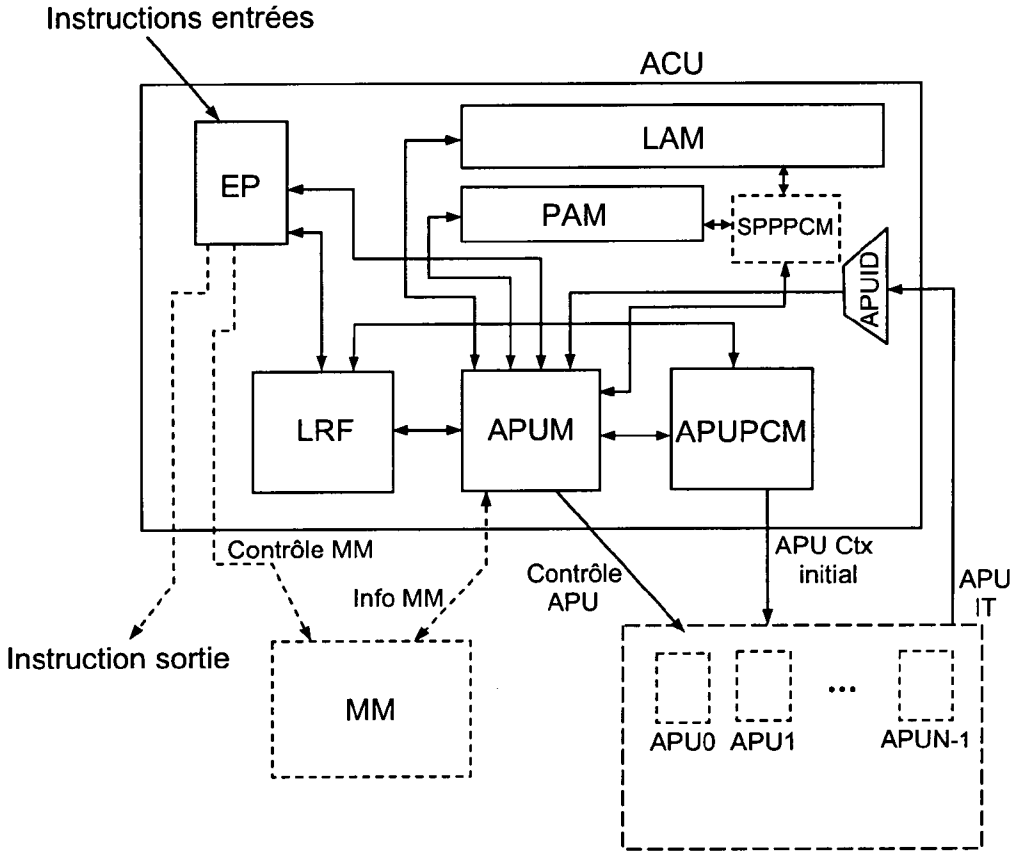
Instructions entrées
ACU
EP
LAM
PAM
SPPPCM
APUID
LRF
APUM
APUPCM
Contrôle MM
Info MM
Contrôle APU
APU Ctx initial
APU IT
Instruction sortie
MM
APU0
APU1
APUN-1

FIG.2

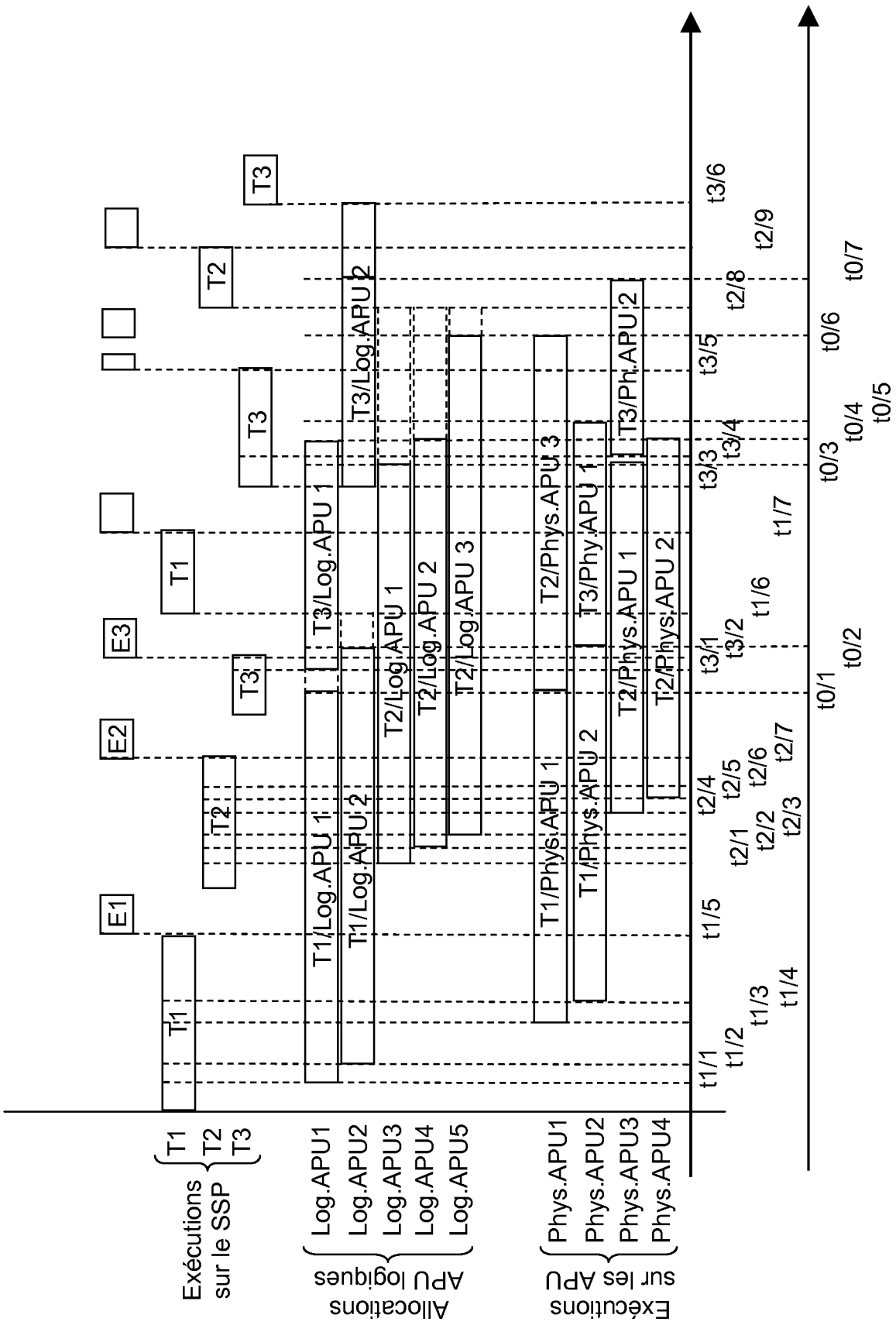
Exécutions sur le SSP
T1
T2
T3
Allocations APU logiques
Log.APU1
Log.APU2
Log.APU3
Log.APU4
Log.APU5
Exécutions sur les APU
Phys.APU1
Phys.APU2
Phys.APU3
Phys.APU4
E1
E2
E3
T1/Log.APU 1
T1/Log.APU 2
T2/Log.APU 1
T2/Log.APU 2
T2/Log.APU 3
T3/Log.APU 1
T3/Log.APU 2
T1/Phys.APU 1
T1/Phys.APU 2
T2/Phys.APU 1
T2/Phys.APU 2
T2/Phys.APU 3
T3/Phy.APU 1
T3/Ph.APU 2
t1/1
t1/2
t1/3
t1/4
t1/5
t1/6
t1/7
t2/1
t2/2
t2/3
t2/4
t2/5
t2/6
t2/7
t2/8
t2/9
t3/1
t3/2
t3/3
t3/4
t3/5
t3/6
t0/1
t0/2
t0/3
t0/4
t0/5
t0/6
t0/7

FIG.3

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2010/052215

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F9/50
ADD. G06F9/46

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F

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

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	WO 2007/051935 A (COMMISSARIAT ENERGIE ATOMIQUE [FR]; DAVID RAPHAEL [FR]; DAVID VINCENT) 10 May 2007 (2007-05-10) cited in the application the whole document	1-12
Y	US 2006/130062 A1 (BURDICK DEAN J [US] ET AL) 15 June 2006 (2006-06-15) paragraphs [0029] - [0058]; figures 1,2	1-12

-/--

[X] Further documents are listed in the continuation of Box C.

[X] See patent family annex.

* Special categories of cited documents :

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

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

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

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

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

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

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

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

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
29 June 2010	06/07/2010

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Michel, Thierry

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2010/052215

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	PHILBIN J ET AL: "VIRTUAL TOPOLOGIES : A NEW CONCURRENCY ABSTRACTION FOR HIGH-LEVEL PARALLEL LANGUAGES" LANGUAGES AND COMPILERS FOR PARALLEL COMPUTING. 8TH. INTERNATIONAL WORKSHOP, LCPC '95. COLUMBUS, AUG. 10 - 12, 1995; [ANNUAL WORKSHOP ON LANGUAGES AND COMPILERS FOR PARALLEL COMPUTING], BERLIN, SPRINGER, DE, vol. WORKSHOP 8, 10 August 1995 (1995-08-10), pages 450-464, XP000689164 ISBN: 978-3-540-60765-6 the whole document	1-12

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/EP2010/052215

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 2007051935 A	10-05-2007	EP 1949234 A1	30-07-2008
		FR 2893156 A1	11-05-2007
		JP 2009515246 T	09-04-2009
		US 2009327610 A1	31-12-2009
US 2006130062 A1	15-06-2006	NONE	

RAPPORT DE RECHERCHE INTERNATIONALE

Demande internationale n°
PCT/EP2010/052215

A. CLASSEMENT DE L'OBJET DE LA DEMANDE
INV. G06F9/50
ADD. G06F9/46

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

B. DOMAINES SUR LESQUELS LA RECHERCHE A PORTE

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

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

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

C. DOCUMENTS CONSIDERES COMME PERTINENTS

Catégorie*	Identification des documents cités, avec, le cas échéant, l'indication des passages pertinents	no. des revendications visées
Y	WO 2007/051935 A (COMMISSARIAT ENERGIE ATOMIQUE [FR]; DAVID RAPHAEL [FR]; DAVID VINCENT) 10 mai 2007 (2007-05-10) cité dans la demande le document en entier	1-12
Y	US 2006/130062 A1 (BURDICK DEAN J [US] ET AL) 15 juin 2006 (2006-06-15) alinéas [0029] - [0058]; figures 1,2	1-12

-/--

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

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

* Catégories spéciales de documents cités:

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

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

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

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

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

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

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

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

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

Date à laquelle la recherche internationale a été effectivement achevée	Date d'expédition du présent rapport de recherche internationale
29 juin 2010	06/07/2010

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

RAPPORT DE RECHERCHE INTERNATIONALE

Demande internationale n°
PCT/EP2010/052215

C(suite). DOCUMENTS CONSIDERES COMME PERTINENTS

Catégorie*	Identification des documents cités, avec, le cas échéant, l'indication des passages pertinents	no. des revendications visées
A	PHILBIN J ET AL: "VIRTUAL TOPOLOGIES : A NEW CONCURRENCY ABSTRACTION FOR HIGH-LEVEL PARALLEL LANGUAGES" LANGUAGES AND COMPILERS FOR PARALLEL COMPUTING. 8TH. INTERNATIONAL WORKSHOP, LCPC '95. COLUMBUS, AUG. 10 - 12, 1995; [ANNUAL WORKSHOP ON LANGUAGES AND COMPILERS FOR PARALLEL COMPUTING], BERLIN, SPRINGER, DE, vol. WORKSHOP 8, 10 août 1995 (1995-08-10) , pages 450-464, XP000689164 ISBN: 978-3-540-60765-6 le document en entier	1-12

RAPPORT DE RECHERCHE INTERNATIONALE

Renseignements relatifs aux membres de familles de brevets

Demande internationale n°
PCT/EP2010/052215

Document brevet cité au rapport de recherche	Date de publication	Membre(s) de la famille de brevet(s)	Date de publication
WO 2007051935 A	10-05-2007	EP 1949234 A1	30-07-2008
		FR 2893156 A1	11-05-2007
		JP 2009515246 T	09-04-2009
		US 2009327610 A1	31-12-2009
US 2006130062 A1	15-06-2006	AUCUN	